Izrada aplikacije za učenje programiranja

Prpić, Nikola

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Izrada aplikacije za učenje programiranja

Završni rad

Nikola Prpić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 25.08.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Nikola Prpić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4415, 22.07.2019.
OIB Pristupnika:	87477569584
Mentor:	Prof. dr. sc. Krešimir Nenadić
Sumentor:	Dr. sc. Krešimir Romić
Sumentor iz tvrtke:	
Naslov završnog rada:	Izrada aplikacije za učenje programiranja
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Izraditi aplikaciju koja bi korisnicima pomogla u učenju programiranja. Aplikacija je namijenjena programerima početnicima i omogućava im da na interaktivan način vježbaju znanje na jednostavnim zadacima. Bazirati se na učenju jednog programskog jezika te implementirati nekoliko razina zahtjevnosti zadataka za vježbu. Implementirati više tipova zadataka u aplikaciji te omogućiti dodavanje novih zadataka. Tema rezervirana: Nikola Prpić Sumentor: dr. sc. Krešimir Romić
Prijedlog ocjene završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 1 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	25.08.2022.
Datum potvrde ocjene od strane Odbora:	07.09.2022.

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

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 10.09.2022.

Ime i prezime studenta:	Nikola Prpić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R 4415, 22.07.2019.
Turnitin podudaranje [%]:	14

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada aplikacije za učenje programiranja**

izrađen pod vodstvom mentora Prof. dr. sc. Krešimir Nenadić

i sumentora Dr. sc. Krešimir Romić

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

Potpis studenta:

SADRŽAJ

1. UVOD

Tema ovog završnog rada je izrada aplikacije koja bi na interaktivni način omogućila korisnicima svladavanje i ponavljanje gradiva osnovnih pojmova iz programiranja. Bazirana je na gradivu koje studenti FERIT-a uče na prvoj godini preddiplomskog studija. Rješavanjem zadataka, korisnik dobiva osnovna saznanja iz područja programskog jezika C.

Pri izradi ove web aplikacije, znatno su pomogla znanja stečena kroz dosadašnje obrazovanje i kolegije koji se slušaju na fakultetu kao što su „Osnove razvoja web i mobilnih aplikacija“, „Objektno orijentirano programiranje“ te razni drugi kolegiji vezani za samo programiranje. Ova web aplikacija namijenjena je početnicima koji tek ulaze u svijet programiranja te moraju upoznati neke od osnovnih stvari poput inicijalizacije, strukture određenih naredbi, petlji i slično. Za temeljitu izradu ove web aplikacije korištene su tehnologije kao što je biblioteka programskog jezika „JavaScript“ React, *Firebase*, JSON format za skladištenje pitanja te opisni i stilski jezici kao što su HTML (engl. *Hypertext Markup Language*) i CSS (engl. *Cascading Style Sheets*).

Struktura završnog rada je zamišljena da nakon uvoda dolazi područje pregleda rada, odnosno istraživanje aplikacija koje se nude na tržištu slične ovoj te navedemo njihove značajke, prednosti i nedostatke. Nakon toga slijedi opis korištenih tehnologija, osnovne stvari o njima, gdje se najviše koriste i slično. U nastavku rada, govori se o implementaciji i testiranju aplikacije. Na kraju, u zaključku, ocjenjuje se aplikacija i navode se mogućnosti unapređenja.

1.1. Zadatak završnog rada

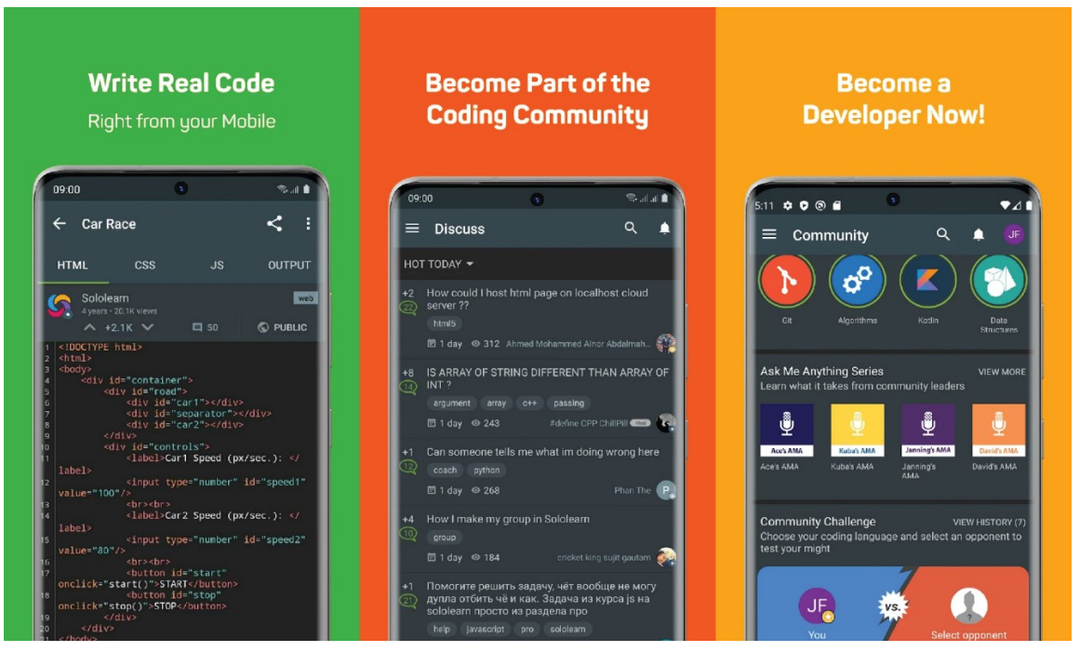
Zadatak završnog rada je napraviti React web aplikaciju za programere početnike u obliku kviza. Aplikacija ima mogućnost prijave ili registracije korisnika unoseći email i lozinku. Nakon prijave, korisnik može pristupiti kvizu. Ukoliko nije siguran kako se igra, aplikacija nudi upute koje objašnjavaju što je potrebno kako bi se provjerilo njihovo znanje. Ulaskom u kviz, započinje odbrojavanje te ukoliko se u tom roku ne odgovore sva pitanja, automatski će biti završen. U određenim zadacima, korisnik ima mogućnost korištenja pomoći koji mu eliminiraju jedan netočan odgovor u pitanju. Završetkom kviza, korisnik dobiva sažetak o svom uspjehu. Sažetak sadrži ukupan broj pitanja, broj odgovorenih pitanja, broj točnih i netočnih odgovora te broj korištenja pomoći. Također, na vrhu sažetka nalazi se vaš rezultat u obliku postotka uz odgovarajuću poruku.

2. PREGLED PODRUČJA TEME

U današnje vrijeme, ukoliko znate bilo koji programski jezik, vrlo ste traženi na tržištu rada. Programeri su potrebni gotovo svugdje, od velikih i malih poduzeća, banaka, za izradu raznih aplikacija i video igrica i drugo. Zbog toga ne čudi činjenica da sve više ljudi ide upravo u tome smjeru, jer osim praktički sigurnog pronalaska posla, programeri su izuzetno dobro plaćeni. Ovakve aplikacije dolaze do izražaja za one koji studiraju u području računarstva, ali i za one koji samostalno uče programirati. To su, uglavnom, besplatne aplikacije koje nude mogućnost usavršavanja znanja iz raznih jezika rješavanjem zadataka i samostalnim vježbanjem. Aplikacije mogu biti web, desktop ili mobilne. Web aplikacije su najrašireniје, najbrojnije te one predstavljaju računalni programski kod koji se pokreće u nekom internetskom pregledniku. S druge strane, desktop aplikacije se moraju preuzeti te spremiti lokalno na odgovarajući operacijski sustav. Mobilne aplikacije su također izrazito popularne u zadnje vrijeme, zbog sve boljih i pametnijih mobilnih uređaja. U nastavku će biti objašnjeno nekoliko najpoznatijih aplikacija za učenje programiranja koje se danas nude.

2.1. Sololearn

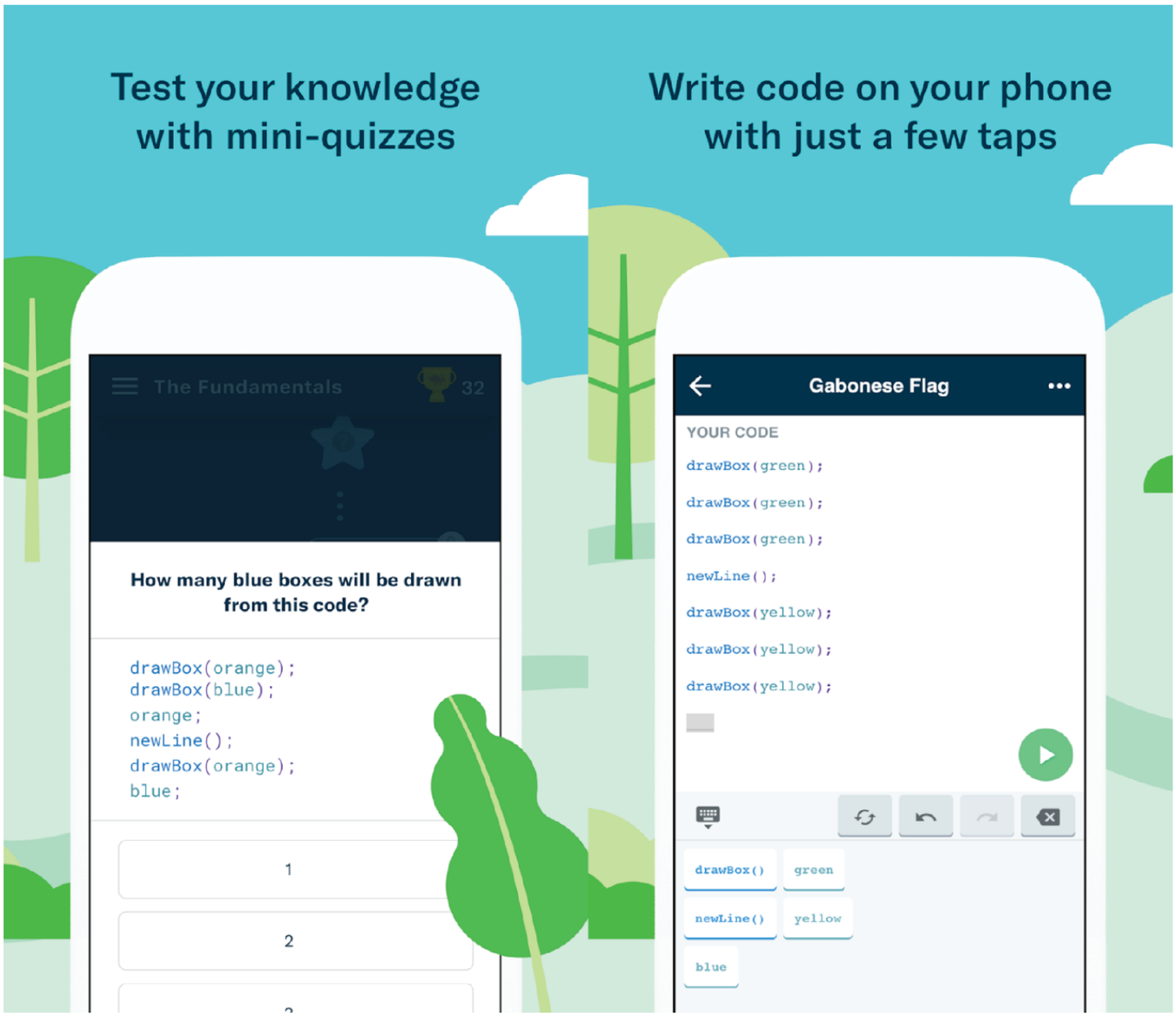
Sololearn je najraširenija, najkorištenija aplikacija te najveća svjetska zajednica programera. Pruža brojne mogućnosti kao što je prikazano na slici 2.1. Aplikacija nudi preko dvadeset različitih tečaja za brojne jezike. Broji oko pedeset milijuna korisnika. Sastoji se od više manjih aplikacija te se kroz razne načine, od nadopunjavanja linija koda, kvizova pa sve do pisanja vlastitih kodova, laganim koracima usavršava odabrani programski jezik. Tipova zadataka ima za sve, i početnike i profesionalce. Aplikacija je besplatna za mobitele, ali verzija bez oglasa s naprednim alatima se naplaćuje. Najpopularniji jezici su *Python, Java* i *C++*.



Slika 2.1. Mogućnosti unutar aplikacije Sololearn [1]

2.2. *Grasshopper*

Ova aplikacija je bazirana za početnike. Nastoji se učiti na kratak, zabavan način te se pokušavaju riješiti određene zagonetke koje zadatak nudi kao što je prikazano na slici 2.2. Ono što je zanimljivo kod ove aplikacije jest to što nudi određene razine te samim time možete vidjeti svoj napredak. Prolaskom svih razina kroz ovu aplikaciju, spremni ste za veće i složenije stvari. Jedini nedostatak je taj što aplikacija nudi tečaj za samo jedan jezik, a to je *Java Script.*

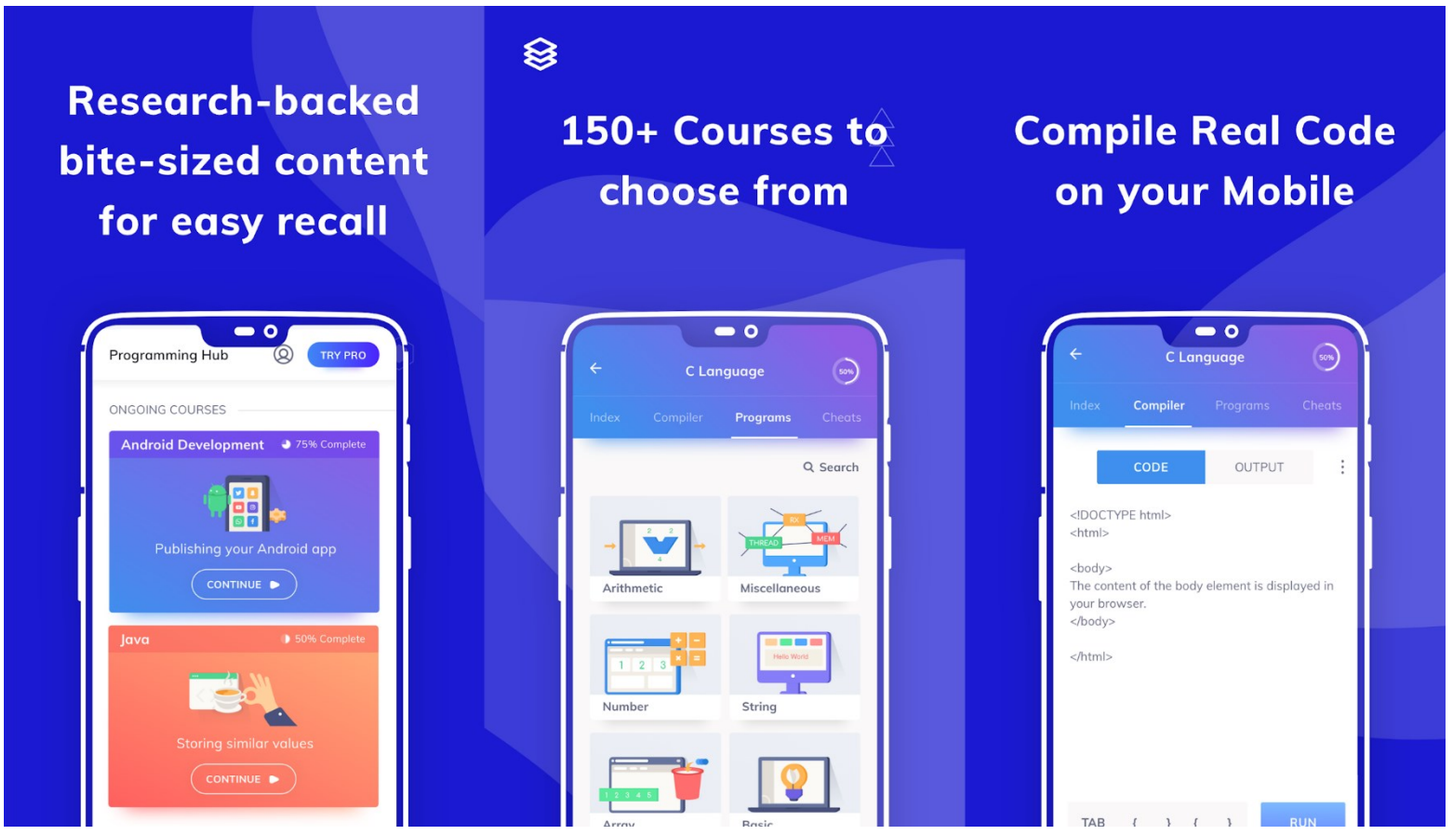


Slika 2.2. Primjer unutar aplikacije Grasshopper [2]

2.3. *Programming Hub*

Poput drugih aplikacija, tako i ova nastoji korisniku pružiti što zabavniji te u isto vrijeme što poučniji sadržaj za učenje programiranja. Svakako prednost ove aplikacije je što nudi preko sedamnaest popularnih programskih jezika te tisuće

linija gotovog koda za svoje korisnike. Isto tako, određeni jezici se mogu koristiti i bez interneta. Aplikacija ima sistem sličan onom školskom, kroz bodovanje vam vraća rezultat i vaš napredak. U početku vam je sve dostupno besplatno, no nakon nekoliko odslušanih predavanja, morate kupiti *premium* pretplatu koja je prikazana u kutu na slici 2.4.



Slika 2.3. *Programming Hub* - Izgled korisničkog sučelja [3]

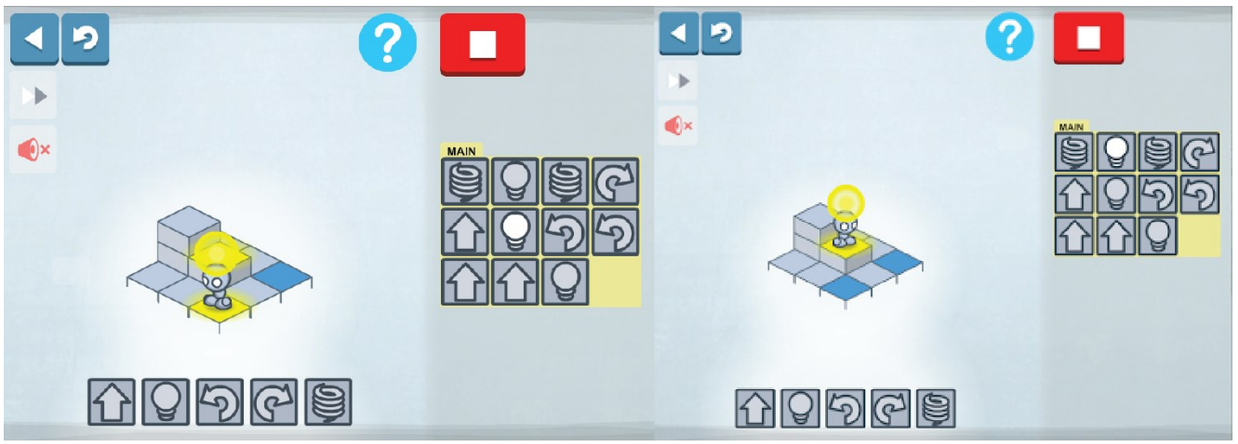
2.4. C Programming

Programski jezik C je preteča svih danas poznatijih programskih jezika te zbog toga možemo reći da je on osnova za daljnje učenje programiranja. Upravo iz tog razloga, nije loše proučiti ovaj jezik i vidjeti njegovu strukturu te kako on radi. Ova aplikacija vam nudi upravo to, prolazak kroz osnovne elemente ovog jezika kroz razne *tutoriale,* pitanja i odgovore i slično kao što je prikazano korisničkim sučeljem aplikacije slikom 2.4.

Slika 2.4. Izgled korisničkog sučelja

2.5. Lightbot

Lightbot je zapravo video igra koja osim zabave, pruža poučni i edukativni dio. Igrajući, razvijate svoje programerske vještine i logiku. Namijenjena je prvenstveno djeci te je igrana preko dvadeset milijuna puta. Aplikacija je dostupna za mobilne uređaje. Primjer iz igre prikazan je slikom 2.5.



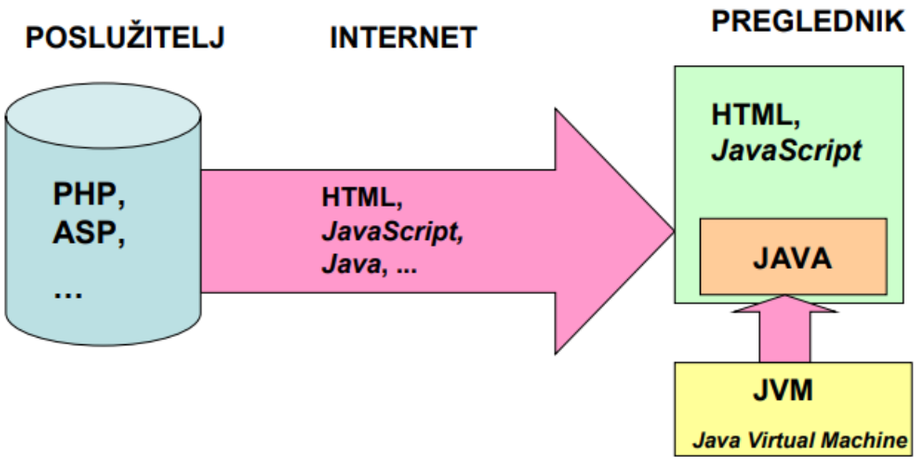
Slika 2.5. *Lightbot* - Primjer rješavanja zadatka [5]

3. KORIŠTENE TEHNOLOGIJE

3.1. JavaScript

JavaScript je izuzetno popularan programski jezik koji ponajviše služi izradi interaktivnih stranica. Jezgra ovog jezika uključena je u gotovo sve internetske preglednike. Iako se zbog slične sintakse i imena često uspoređuje s programskim jezikom *Java, JavaScript* nije pojednostavljena inačica najpopularnijeg programskog jezika za razvoj mobilnih aplikacija. U samome početku nazvan je *LiveScript,* no zbog želje da se potakne korištenje ovog jezika, preimenovan je po uzoru na tada popularnu, već spomenutu, *Javu* [6].

Kada se govori o povijesti jezika, *JavaScript* se razvija od 1995. godine kada izlazi nekoliko inačica jezika [6]. Nedugo nakon toga Microsoft je objavio jezik sličan JavaScriptu pod nazivom JScript [6]. Danas je za standardizaciju skriptnih jezika, pa tako i JavaScripta, zadužena organizacija ECMA (eng. *European Computer Manufacturers Association*) [6]. HTML, opisni jezik kojeg će se upoznati detaljnije u nastavku, koristi se samo za opis elemenata te nema dinamičkih elemenata. Međutim, kako se ukazala potreba za takvim elementima, danas razlikujemo više oblika interaktivnog sadržaja u HTML dokumentima. Jedan od oblika su tehnologije za dinamičko stvaranje HTML-a odnosno stranice su napisane pomoću nekog skriptnog jezika poput PHP, *Python, Java.* Drugi oblik je *Java, Flash i Shockwave* za koje je nužan vanjski program koji zna pokrenuti navedene programe [6]. I treću skupinu, u koju pripada sama *JavaScript,* čine klijentski jezici jer se njihov kod pokreće u pregledniku. Primjer toka podataka od web-poslužitelja do web-preglednika dan je slikom 3.1. [6].



Slika 3.1. Tok podataka [6]

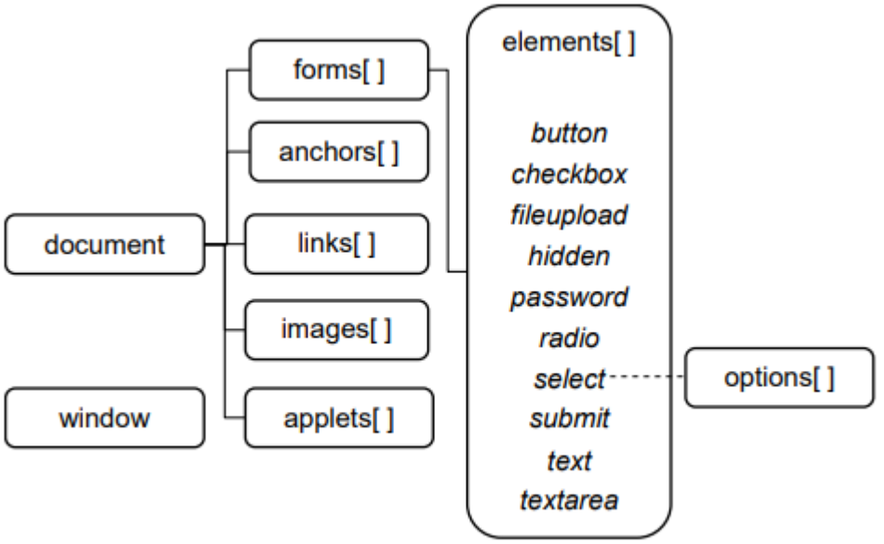
Kao što je već spomenuto, *JavaScript* je programski jezik koji se pokreće u pregledniku stoga ne čudi da je vrlo popularan i korišten jer mu je podrška odlična u gotovo svim preglednicima. Zbog toga se preglednici danas natječu u brzini

izvođenja algoritama [6]. No ne natječu se samo u brzini, već i u jednostavnosti razvoja. Što jednostavniji razvoj, više korisnika će koristiti taj preglednik. Najpopularniji odnosno najčešći preglednici koji se koriste su *Internet Explorer, Google Chrome* i *Mozilla Firefox.*

JS je dosta fleksibilan oko toga u kojem uređivaču teksta ga se može pisati jer podržava sve ukoliko uređivač podržava ASCII standard. Kako bi programiranje bilo uspješno, preporuka je koristiti uređivač teksta koji ima mogućnost otvaranja više dokumenata u jednom prozoru te da prepoznaje dijelove kode i oboja ih drugačijim bojama. Potrebno je imati podršku za UTF- 8 (eng. *Unicode Transformation Format 8*) te automatsko poravnanje.

Program koji obrađuje i izvršava skripte zove se *interpreter* [6]. *Interpreter* čita kod i prevodi ga u strojni jezik svakog puta kada se pokrene skripta. Svaki jezik koji se interpretira, tj. koji izvršava *interpreter*, naziva se skriptni jezik [6]. *Interpreter* za JavaScript ugrađen je u većinu današnjih preglednika [6]. Glavni razlog korištenja skriptnih jezika je taj što je razvijanje programa znatno jednostavnije.

Document Object Model (DOM) je model za prikaz i interakciju s objektima u HTML-dokumentu [6]. *JavaScript* je jezik koji samostalno kreira svoj DOM u obliku prikazanom na slici 3.2. te se svakom objektu pristupa kroz taj model [6]:



Slika 3.2. Hijerarhijska struktura DOM-a [6]

Programi napisani u *JavaScript*-u ne mogu brisati datoteke ili pristupati drugima jer ne mogu pristupati lokalnim datotekama. Također, ne mogu obavljati mrežne radnje.

3.1.1. React

React se službeno vodi kao *JavaScript* biblioteka. Originalno je napravljen od strane Facebook inženjera s ciljem razvoja složenih korisničkih sučelja [7]. Pod korisnička sučelja ubrajamo trake za pretraživanje, razne gumbove i slične stvari s čime netko koristi web aplikaciju. Takve biblioteke su skup funkcija napisane u *JavaScriptu* koje olakšavaju izradu raznih programa. Može ih svatko napisati te često mijenjaju dio koda koji se ponavlja. *React* nema posebne strukture te je on samo velika biblioteka koja se može koristiti u kombinaciji s drugim bibliotekama. Za razliku od ostalih jezika, *React* koristi virtualni DOM umjesto stvarnog, koji je možda i najbitniji dio *React*-a [7].

Kada se govori o *React* aplikacijama, temelj svake od njih su komponente te su ujedno i najveći dio razvoja. Komponente su srž *React*-a i pogled na vašu aplikaciju [7]. Možemo ih usporediti s funkcijama odnosno metodama jer primaju određeni parametar (*props*) te vraćaju određeni element. Najispravniji oblik korištenja komponenti je taj da svaka komponenta služi za samo jednu funkcionalnost što znači da će se složenije aplikacije sastojati od mnogobrojnih komponenti. Primjer jednostavne komponente prikazan je na slici 3.3. :

```
1  import React from "react";
2
3  const High = () => {
4    return <h1>HELLO</h1>;
5  };
6
7  export default High;
8
```

Slika 3.3. Struktura React komponente

High je ime komponente te je ono proizvoljno. Same komponente je najbolje imenovati u skladu s onim što rade radi lakšeg snalaženja i čitanja koda. Uz već spomenute parametre koji se mogu predavati i vraćanju elemenata, jako važna je zadnja linija koda koja omogućava korištenje te komponente bilo gdje u kodu.

Uz komponente, važna karakteristika *React-a* je JSX. Dolazi od skraćenice *JavaScript Syntax Extenstion* te predstavlja transformacijski sloj koji pretvara XML sintaksu za pisanje komponenti u sintaksu koju React koristi za prikaz elemenata u *JavaScript*-u [7]. Korištenje JSX-a nije obavezno, ali znatno olakšava kreiranje elemenata te se svakako preporučuje [7]. U slučaju ako ga se ne koristi, za stvaranje React elemenata se koristi metoda *React.createElement()* te prima tri argumenta gdje prvi argument predstavlja tip elementa, drugi svojstvo elementa te treći djecu elementa.

3.2. HTML

Iako je dosta sličan programskom jeziku, HTML (engl. *Hyper Text Markup Language*) je standardni opisni jezik za kreiranje raznih WEB stranica. Pomoću njega opisujemo strukturu sadržaja neke stranice ili dokumenta uporabom posebnih oznaka, specifičnih za HTML, poznatih kao *tagovi.* Korištenjem oznaka kreiramo elemente u HTML-u.

Osnovna struktura svakog dokumenta u HTML-u prikazana je slikom 3.4. [8]:

```
<!DOCTYPE html>

<html>
      <head>
            <title>Page Title</title>
            <meta charset="utf-8">
      </head>

      <body>

      </body>
</html>
```

Slika 3.4. Osnovna struktura HTML koda [8]

U počecima, *doctype* je predstavljao veze na skup pravila koje je neka stranica rađena u HTML-u trebala poštivati. Pružalo je automatsko prepravljanje grešaka i druge korisne stvari. Navodi ga se na početku samo jednom, a ovakvom oznakom kao na primjeru govori da se radi o petoj verziji HTML-a. Element koji okružuje sav sadržaj i poznat je kao korijenski element stranice je html.

<head> </head> element služi kao kontejner te se u njemu navode sve stvari koje žele biti uključene za neku stranicu ili dokument. Ne predstavljaju sadržaj koji će biti prikazan, već nekakvi CSS stilovi za ukrašavanje stranice, znakovni skupovi, ključne riječi i slično. U elementu <title></title> navodi se ime stranice koje će biti prikazano u kratici nekog web preglednika. Koristi se i za opis stranice te je obavezan u svim HTML dokumentima. Sadržaj koji korisnik vidi kada posjeti web stranicu nalazi se unutar <body></body> elementa [8].

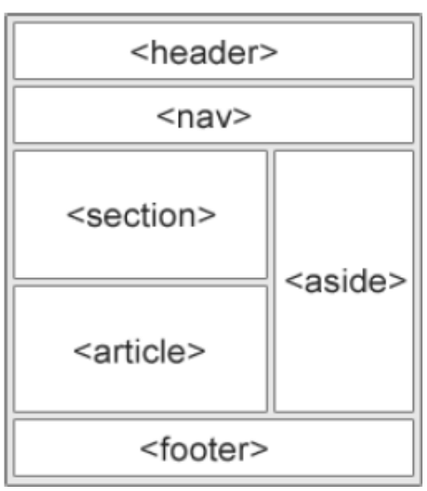
HTML elementi se u većini slučaja nalaze unutar početnih i završnih oznaka, a sadržaj se nalazi između njih. Primjer korištenja oznaka: <ime_oznake> Sadržaj </ime_oznake>. Ne razlikuje mala i velika slova, stoga kako god napisali određenu oznaku, preglednik će razumjeti. Svaki element može imati i dodatne atribute koji nude nekakvu dodatnu informaciju i uvijek se navode u početnoj oznaci.

Razlikujemo dvije vrste elemenata u HTML-u, a to su blok i linijski elementi. Blok element uvijek počinje na novoj liniji i zauzima punu dostupnu širinu (rastegnuto je od lijeve prema desnoj strani koliko je moguće). Blok elementi znaju biti strukturni elementi koji predstavljaju paragraf, listu i slično. S druge strane, linijski elementi ne započinju u novoj liniji i zauzimaju onoliko koliko im je potrebno širine. Neće prouzročiti prelazak u novi red i najčešće se koriste za isticanje teksta [8].

Jedna od glavnih zadaća HTML-a je davanje strukture i značenje tekstu. Većina takvog teksta se sastoji od naslova i odlomaka. Postoji šest definiranih naslova unutar jezika, a navedeni su po veličini od najvećeg od najmanjeg brojevima jedan do šest (od h1 do h6). Naslovni su izuzetno važni jer korisnici preko njih mogu brže pretraživati sadržaj. Ispod svakog naslova, dolazi tekst koji se piše unutar odlomka. Element koji definira paragraf označava se s . Uz naslov i paragraf, vrlo bitan i jako korišten element su liste. Razlikujemo nenumerirane (<ul>) i numerirane (<ol>) liste. Elementi unutar obadvije vrste lista kreću s oznakom <li> .

Sve složenije web stranice imaju sljedeću strukturu: zaglavlje, navigacijska traka, glavni sadržaj, bočna traka i podnožje. Zaglavlje predstavlja vrh stranice gdje se obično nalazi naslov ili logo stranice. Ondje se nalaze glavne stvari vezanu uz stranicu. Ispod se nalazi navigacijska traka koja se koristi kako bi ostvarili vezu s ostalim dijelovima stranice. Glavni sadržaj obuhvaća najveći dio prostora te se u njemu nalazi većina sadržaja poput teksta, slika, video zapisa i slično. Pojedine stranice imaju bočnu traku u kojoj se nalaze razne reklame, veze i slične stvari. Osim toga, može predstavljati i sekundarnu navigaciju. Na kraju svake stranice nalazi se podnožje te sadrži informacije vezanu uz autora, informacije vezane uz kontakt. Često sadržavaju i veze za društvene veze poput *Facebooka* [9]. U HTML-u, oznake za navedene elemente redom glase te su grafički prikazani slikom 3.5. [9]:

- <header>
- <nav>
- <main> s kojim se kombinira <article> ,<section> i
- <aside>
- <footer>



Slika 3.5. Primjer rasporeda [9]

Article i *section* su slični elementi koji se sastoje od blok sadržaja, međutim najlakši način za shvatiti razliku je da se *article* može podijeliti na više *section* elemenata, odnosno to je kompletan blok elemenata.

Primjer strukture web stranice korištenjem svih navedenih elemenata prikazan je slikom 3.6. [9]:

```
<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <header>
        <h1></h1>
    </header>
    <nav>
        <ul>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
        </ul>
    </nav>
    <main>
        <article>
            <h2></h2>
            <p></p>
            <section>
                <h3></h3>
                <p></p>
            </section>
            <section>
                <h3></h3>
                <p></p>
            </section>
        </article>
        <aside>
            <h2></h2>
            <ul>
                <li></li>
                <li></li>
                <li></li>
                <li></li>
            </ul>
        </aside>
    </main>
    <footer></footer>
</body>
</html>
```

Slika 3.6. Struktura WEB stranice [9]

3.3. CSS

Za izradu stranica najčešće se kombinira već spomenuti *JavaScript* i HTML te CSS (engl. *Cascading Style Sheets*). To je stilski jezik te jednostavan mehanizam za dodavanje stilova poput fontova, boja, uređivanja tablica i slično [10]. Korištenjem CSS-a postalo je moguće odvojiti prezentaciju podataka i dizajn od same strukture podataka. HTML kod postaje pregledniji i manji što znači da ga je puno lakše kontrolirati, a također je moguće jednostavnom primjenom parametara promijeniti izgled stranice [10].

Struktura stilskih obrazaca sastoji se od stilskih pravila koje se sastoji od dva dijela, selektora i deklaracije. Selektor određuje element na koji se stilsko pravilo odnosi, a deklaracija određuje kako izgleda sadržaj opisan CSS-om. Sintaksa opisanog glasi: selektor{deklaracija;} [10]. Unutar deklaracije imamo podjelu na *properties*, koji govori kako računalo treba prikazati tekst i grafiku, i *values,* podaci koji govore kako trebaju izgledati tekst i slike na stranici. Za jedan selektor može se uvesti više pravila. Možemo svaku deklaraciju pisati posebno ili kombiniranjem više deklaraciju odjednom u jedno stilsko pravilo.

Što se tiče primjene CSS-a na HTML dokumente, razlikujemo tri načina:

- Stil unutar linije (*engl. Inline style*)
- Unutarnja lista stilova (engl. *Internal stylesheet*)
- Vanjska primjena liste stilova (engl. *External stylesheet*)

Stil unutar linije je prikazan primjerom na slici 3.7., a on predstavlja CSS deklaraciju koja ima utjecaj na samo jedan element i upisujemo ju unutar *style* atributa [10]. Ovaj način se ne preporuča i ne koristi se često ukoliko ne postoji opravdan razlog za to jer je iznimno teško za održavanje, čitanje i razumijevanje.

```
<p>
    Koristit ćemo:
    <span style="border: 1px solid black; background-color:
lime;">Kaskadne</span>
    <span>Liste</span>
    <span>Stilova</span>
</p>
```

Slika 3.7. Stil unutar linije [10]

Unutarnja lista stilova se koristi unutar HTML dokumenta tako da se unutar <head> elementa primjeni <style> element. Ovakav način može biti praktičan, no u ovom slučaju bi se CSS morao ponavljati na svakoj stranici. Korištenje ovog načina predstavljeno je slikom 3.8. [10].

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Primjena CSS stilova</title>
    <style>
        span {
            border: 1px solid black;
            background-color: lime;
        }
    </style>
</head>
```

Slika 3.8. Unutarnja lista stilova [10]

Zadnji način, ujedno i najbolji, je vanjska primjena stilova. On predstavlja primjenu CSS-a koji se nalazi u posebnoj datoteci s ekstenzijom .css, a u HTML ju uključujemo pomoću elementa <link> kojeg postavljamo u <head> elementu. S ovom metodom, jednom listom stilova možemo stilizirati više HTML dokumenata. Kod koji pišemo u posebnu .css datoteku prikazan je slikom 3.9. [10].

```
span {
    border: 1px solid black;
    background-color: lime;
     }
```

Slika 3.9. Vanjska primjena stilova [10]

Kada se govori o CSS tehnikama za raspoređivanje HTML elemenata kojim se kontrolira njihova pozicija, razlikujemo više metoda pozicioniranja. Najpoznatije i najkorištenije tehnike su normalni tok, *float, flexbox* i *grid*. Svaka od navedenih tehnika ima svoju prednost, ali i manu. Upravo zbog toga nikada se ne koristi samo jedna metoda, već se koristi više tehnika za određenu zadaću [11].

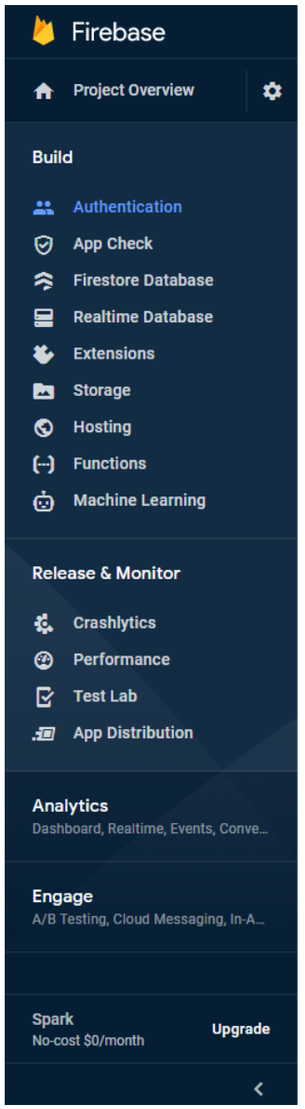
Normalni tok predstavlja način kako preglednik, bez utjecaja CSS-a, raspoređuje po zadanom HTML elemente po stranici. Elementi u ovoj metodi su složeni jedan na drugi. Svojstvo *float* je namijenjeno pozicioniranju slika, ali također s ovim svojstvom možemo dobiti stupčasti raspored web stranice. Vrijednosti koje svojstvo može poprimiti su *right, left* i *none. Flexbox* i *grid* su slični elementi, ali razlika je u tome što je *flexbox* kreiran za jednu dimenziju, a *grid* za dvije. U svim složenijim stranicama, gotovo pa je nemoguće ne koristiti barem jedan od ta dva elementa [11].

3.4. Firebase

Firebase je platforma koja pomaže mobilnim i web *developerima* razvijati aplikacije koje koriste baze podataka. Platforma koja je preuzeta od strane Google-a 2014.godine te je korištena od više milijuna korisnika [12]. Pruža API usluge

korisnicima za aplikacije. Korištenjem *firebase*-a, ne moramo sami pisati i stvarati servere, vlastiti API i slično. Možemo reći da je on cijela pozadina aplikacije, odnosno ono što korisnik ne vidi.

Neki od servisa koje nudi su: *Authentication, Realtime Database, Storage, Hosting*. U projektu, najzanimljiviji je bio *Authentication,* koji omogućuje prijavu preko email-a i lozinke. Ovaj servis nudi i drugačije oblike prijave, poput prijave putem Google-a, društvenih mreža kao što su Facebook i Twitter. Nudi i prijavu putem broja mobitela, ali i mnoge druge. Dodavanje servisa u projekt je dosta jednostavno te je upravo to najveći razlog zbog čega je *Firebase* popularan. Izgled korisničkog sučelja te servisa koje nudi prikazano je slikom 3.9.[12].



Slika 3.9. Servisi *Firebase-a* [12]

3.5. JSON

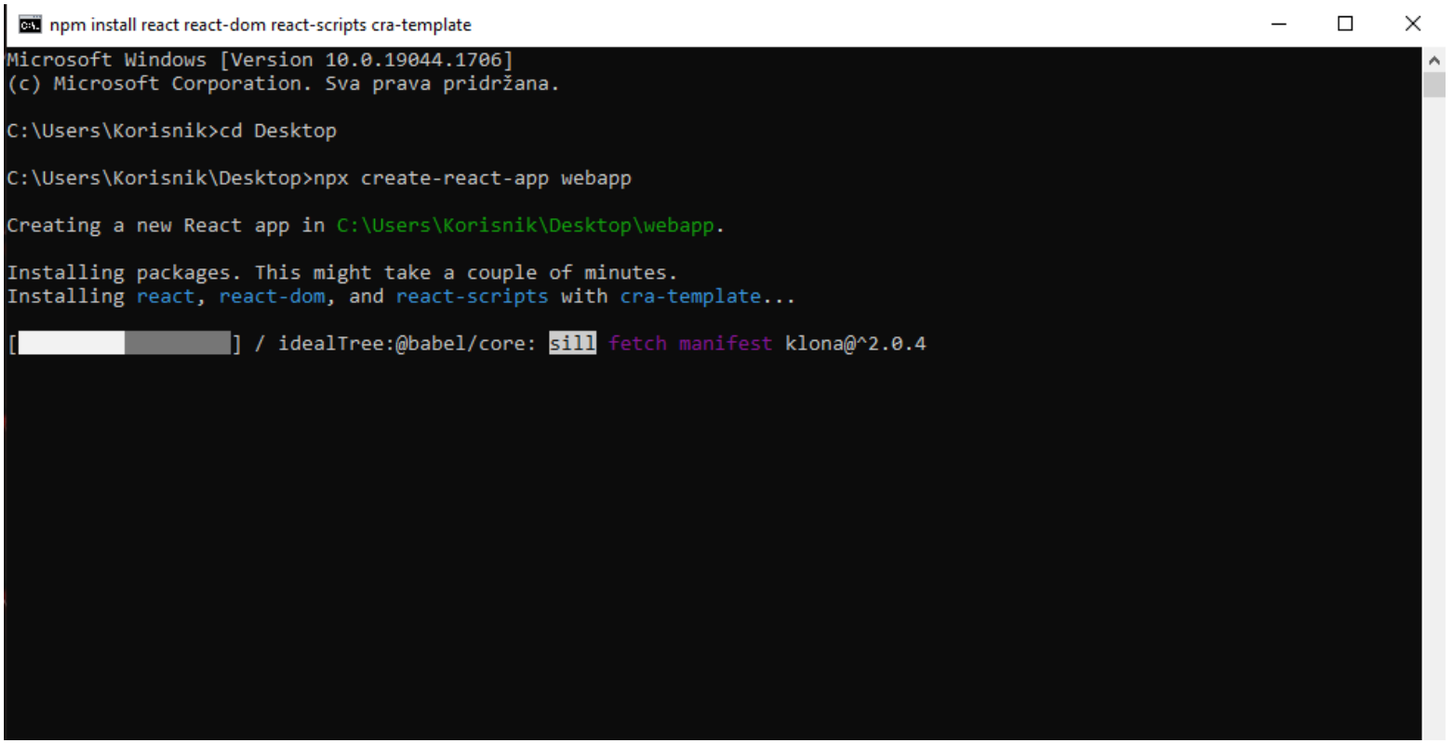
JavaScript Object Notation (JSON) je izveden iz literala programskog jezika JavaScript te ga čini podskupom tog jezika [13]. Iako je podskup programskog jezika, JSON nije programski jezik već format razmjene podataka [13]. JSON se može koristiti kao baza podataka pitanja koja se koristi u aplikaciji. Iako vrlo lagan za korištenje, najsitnije greške poput izostanka zareza ili krivo postavljenih navodnika dovode do pada JSON-a, stoga treba biti vrlo oprezan u radu s njime. Za korištenje ovog formata potrebno je nakon proizvoljnog imena navesti datoteku .json.

3.6. Visual Studio Code

Visual Studio Code je razvojno okruženje napravljeno od strane Microsofta koje se može koristiti na raznim sustavima poput Windowsa, Linuxa i MacOS [14]. Trenutno je najpopularnije razvojno okruženje, a uključuje podršku za ispravljanje pogrešaka, isticanje sintakse, inteligentno dovršavanje koda i još mnoge druge funkcije koje uvelike pomažu pri izradi projekata [14]. U njemu možemo pisati kod u raznim programskim jezicima, poput nabrojanih u ovom radu. Visual Studio Code je vrlo lagan za korištenje te je izuzetno pregledan.

4. IMPLEMENTACIJA APLIKACIJE

Prije samog početka izrade aplikacije u React-u, potrebno je stvoriti projekt u naredbenom retku naredbom (*cmd*) kao što je prikazano primjerom na slici 4.1. *cd Desktop* naredba koja je ubačena prije govori samo gdje će se spremiti projekt, a u ovom slučaju to je radna površina. *npx create-react-app* je obavezan dio te ide za izradu bilo koje aplikacije, dok je *webapp* dio proizvoljan i označava ime projekta.



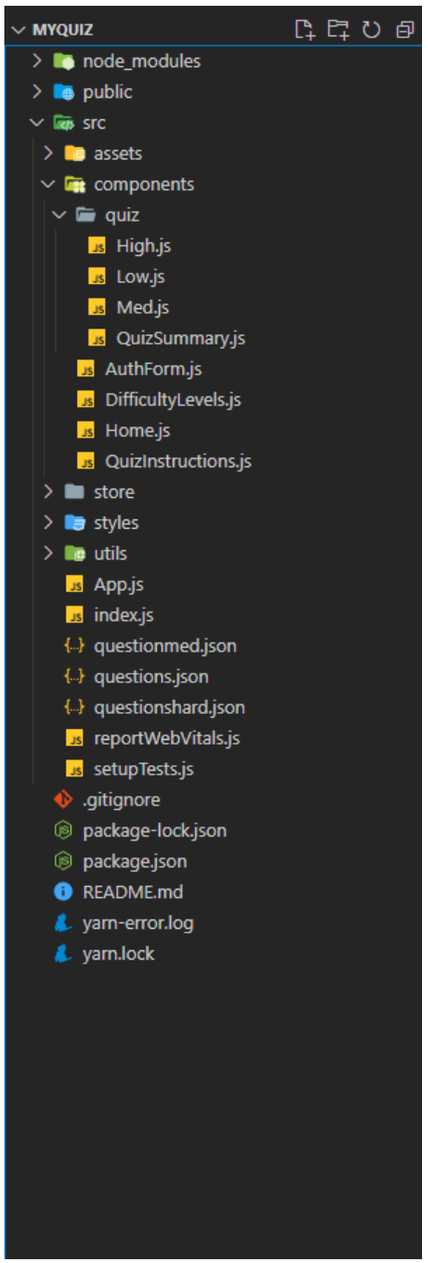
Slika 4.1. Stvaranje React aplikacije

Nakon završetka instalacije paketa pokrenutih naredbom koji su potrebni za React aplikaciju, ponekad je potrebno pokrenuti dodatne naredbe kako bi sve ispravno radilo. Najčešće su to naredbe poput *npm audit fix* ili *npm audit fix –force*. Kada se sve pokrene, možemo započeti s izradom aplikacije.

U početku imamo samo jednu značajnu datoteku zvanu App.js te zapravo jedino vidljivu pokretanjem lokalnog servera naredbom *npm start*. Od ostalih datoteka koje su vrijedne spomena možemo istaknuti index.js, iako se s njime ne radi gotovo ništa.

Kao što je rečeno, komponente su glavni dio svake React aplikacije. Upravo su one s čime se započinje izrada svake aplikacije. U projektu imamo osam glavnih, različitih komponenti i svaka obavlja svoj dio zadatka. Na slici 4.2. se nalaze komponente izrađene za ovu aplikaciju. Imena su na engleskom jeziku kako bi njihovu namjenu znali prepoznati i strani programeri. Uz njih, u datoteci *styles* se nalaze stilovi koji se koriste kako bi ukrasili svaku od komponenti. Osim navedenih, u datoteci *assets* su spremljene fotografije korištene u aplikaciji, *utils* sadrži komponentu koja provjerava je li

trenutno pitanje u kvizu ujedno i zadnje te datoteka *store* koja je bitna u prijavi korisnika. Datoteke s ekstenzijom .json će biti objašnjene kasnije, a ostale datoteke nisu toliko važne.



Slika 4.2. Struktura aplikacije

4.1. Front-end

Svaka složenija aplikacija se sastoji od *front-end* i *back-end* dijela. *Front-end* dio služi za razvoj grafičkog (korisničkog) sučelja kako bi korisnik mogao biti u interakciji s web stranicom. To postižemo kombinacijom HTML-a, CSS-a te JavaScript-a. Velika većina ove aplikacije je izrađena upravo na taj način. Elementi koji su kreirani, poput gumbova,

navigacije, fotografija i slično, namijenjeni su korisniku da ih vidi i njima se služi. Pri izradi *front-end* dijela, uvijek se razmišlja o korisniku, kako bi njemu bilo pruženo što bolje iskustvo. Zbog toga se još naziva i „client-side“.

Prije početne stranice aplikacije, korisnik mora imati svoj račun ukoliko želi nastaviti s korištenjem aplikacije. To je prvi dio gdje se od korisnika traži unos, odnosno njegova aktivnost na stranici. Sve funkcionalnosti oko prijave ili registracije korisnika nalaze se u komponenti AuthForm.js.

Kada se korisnik uspješno prijavio, aplikacija ga prebacuje na početnu stranicu. Ona se sastoji od tri mogućnosti, a to su: igraj, upute i odjava. Klikom na odjavu, aplikacija vraća nazad na dio s prijavom i registracijom te se možemo prijaviti ponovno koristeći isti račun ili neki drugi. Stranica upute sadrži osnovna pravila koje trebamo znati kako bi uspješno pristupili i rješavali zadatke. Sadržaj i struktura stranice opisani su u komponenti QuizInstructions.js. Najvažniji i najkompleksniji dio aplikacije se krije iza mogućnosti igraj. Aplikacija nudi tri razine težine rješavanja zadataka, poredanih redom od najlakše do najteže razine. Za takvu mogućnost odabira težine napravljena je nova komponenta DifficultyLevels.js. Svaka razina je jedinstvena i sadrži drugačija pitanja. Struktura i izgled navigacije koji se koristi na početnoj stranici, koristi se i u drugim dijelovima aplikacije, a prikazan je slikom 4.3.

```
<div id="home">
  <section className="sectionhome">
    <div style={{ textAlign: "center" }}>
      <span className="mdi mdi-cube-outline cube "></span>
    </div>
    <h1>Osnove Programiranja</h1>

    <div className="play-button-container">
      <ul>
        <li>
          <Link className="play-button" to="/home/play">
            IGRAJ
          </Link>
        </li>
      </ul>
    </div>
    <div className="play-button-container">
      <ul>
        <li>
          <Link className="instruction-button" to="/home/instructions">
            UPUTE
          </Link>
        </li>
      </ul>
    </div>
    <div className="play-button-container">
      <ul>
        <li>
          <Link className="logoff-button" to="/">
            ODJAVA
          </Link>
        </li>
      </ul>
    </div>
  </section>
</div>
```

Slika 4.3. Navigacija unutar aplikacije

Glavni dio *front-end* dijela je izrada kviza za učenje programiranja. Komponente koje sadrže kod i logiku za kvizove svih težina sadrže brojne metode od kojih je svaka odgovorna za svoj dio posla. Funkcije za različite razine su jako slične, uz nekoliko manjih izmjena. Za laganu težinu je najveća karakteristika korištenje pomoći, u kodu nazvanih „hintovi“, dok u teškoj razini metoda koja omogućuje unos odgovora od strane korisnika na postavljeno pitanje. Opisana metoda za tešku razinu prikazana je slikom 4.4.

```
checkAnswer = () => {
  var inputVal = document.getElementById("myInput").value;
  if (inputVal === this.state.answer) {
    this.correctAnswer();
    document.getElementById("myInput").value = "";
  } else {
    this.wrongAnswer();
    document.getElementById("myInput").value = "";
  }
};
```

Slika 4.4. Samostalan unos odgovora

Od ostalih metoda, metode *correctAnswer* i *wrongAnswer* se pojavljuju u svim razinama te je njihov zadatak javiti korisniku je li odgovor koji su odabrali ili upisali točan ili netočan. Sve razine kviza su vremenski ograničene, a metoda koja odbrojava vrijeme trajanja kviza je *startTimer.* Vrijeme se razlikuje za svaku težinu te se postavlja u prvom redu u milisekundama, kao što je prikazano slikom 4.5.

```
startTimer = () => {
  const countDownTime = Date.now() + 180000;
  this.interval = setInterval(() => {
    const now = new Date();
    const distance = countDownTime - now;

    const minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
    const seconds = Math.floor((distance % (1000 * 60)) / 1000);

    if (distance < 0) {
      clearInterval(this.interval);
      this.setState(
        {
          time: {
            minutes: 0,
            seconds: 0,
          },
        },
        () => {
          this.endGame();
        }
      );
    } else {
      this.setState({
        time: {
          minutes,
          seconds,
        },
      });
    }
  }, 1000);
};
```

Slika 4.5. Metoda za vrijeme trajanje kviza

Kviz završava kada korisnik riješi sve dane zadatke ili kada istekne vrijeme trajanja. Ove, ali i mnoge druge metode nalaze se u Low.js, Med.js i High.js komponentama.

Po završetku kviza bilo koje razine, korisnik je prebačen na stranicu koja sadrži rezultat rješavanja. Dobiva povratnu informaciju o tome koliko je pitanja bilo u kvizu, na koliko je odgovorio te koliko je odgovorenih pitanja bilo točno, a koliko netočno. Za laganu razinu imamo mogućnost korištenja pomoći pa u taj sažetak ulazi i broj korištenja spomenutog sredstva. Osim toga, aplikacija će izbaciti rezultat u obliku postotka te ovisno o postotku riješenosti ispisati odgovarajuću poruku. Sažetak po završetku kviza nalazi se u komponenti QuizSummary.js, a poruke za određeni interval vrijednosti prikazane su slikom 4.6.

```
if (userScore <= 30) {
  remark = "Trebate još vježbanja";
} else if (userScore > 30 && userScore <= 50) {
  remark = "Više sreće drugi put";
} else if (userScore > 50 && userScore <= 70) {
  remark = "Možes ti još bolje!";
} else if (userScore > 70 && userScore <= 89) {
  remark = "Vrlo dobro!";
} else {
  remark = "Genijalac!!!";
}
```

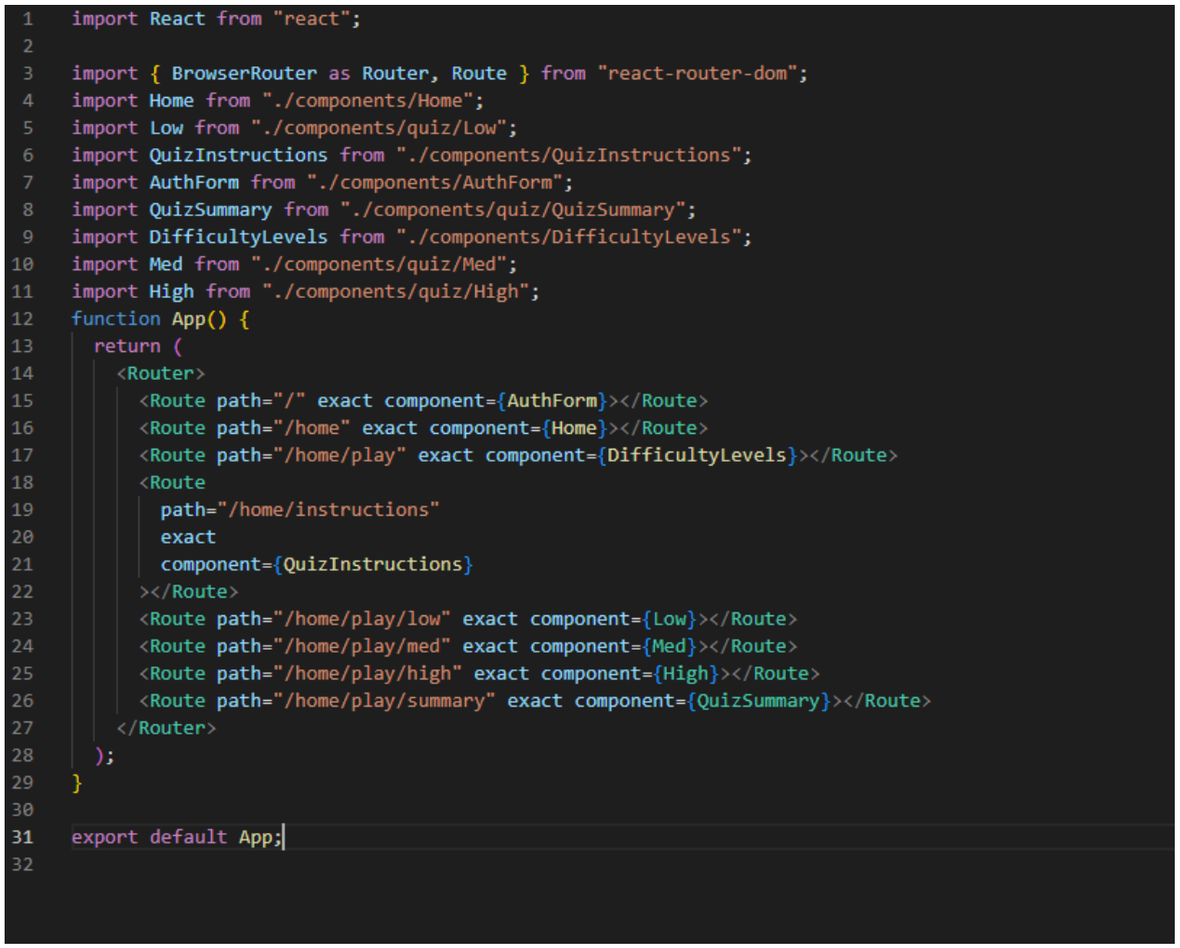
Slika 4.6. Postotak riješenosti

Izmjenu i povezivanje komponenti omogućuje *React Router*. Iako je u trenutku izrade aplikacije najnovija verzija bila v6, u ovom završnom radu koristila se verzija v5. Možemo reći da je to nužan element kojeg moramo imati u projektu kako bi aplikacija bila funkcionalna. U projekt ga uključujemo na sličan način kao što stvaramo svaku React aplikaciju. U naredbenom retku ili terminalu, upisujemo naredbu *npm install react-router-dom@5* te nakon završetka instalacije možemo koristiti *React Router*. Osim instalacije, ako ga se želi koristiti, moramo ga uključiti na način prikazan na slici 4.7.

```
import { BrowserRouter as Router, Route } from "react-router-dom";
```

Slika 4.7. Uključivanje *React Routera*

Iz slike se može vidjeti uključivanje dva elementa, *Router*-a i *Route*-a. *Router* možemo promatrati kao omotač unutar kojeg se nalaze *Route* elementi te svaki od njih predstavlja putanju do određene komponente. Unutar njih moramo navesti atribut *path* koji predstavlja proizvoljnu url adresu svake komponente te atribut *component* u kojem se navodi, unutar vitičastih zagrada, imena komponenti. *Exact* se koristi kako bi nas navedena adresa odvela do točno očekivane komponente. *React Router* u aplikaciji se koristi u već spomenutoj datoteci App.js, a sve opisano je prikazano slikom 4.8.



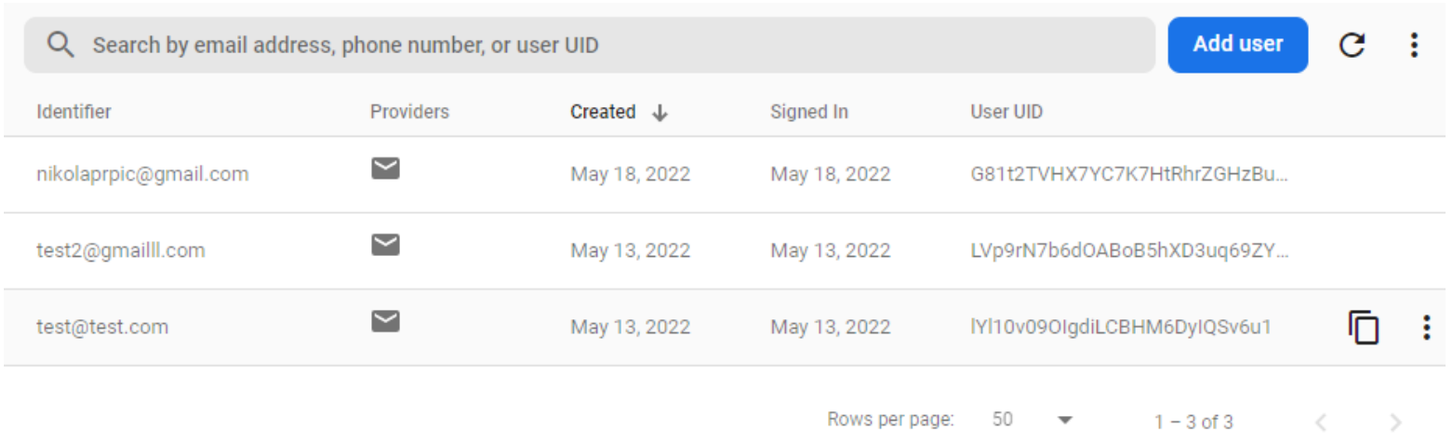
```
import React from "react";

import { BrowserRouter as Router, Route } from "react-router-dom";
import Home from "./components/Home";
import Low from "./components/quiz/Low";
import QuizInstructions from "./components/QuizInstructions";
import AuthForm from "./components/AuthForm";
import QuizSummary from "./components/quiz/QuizSummary";
import DifficultyLevels from "./components/DifficultyLevels";
import Med from "./components/quiz/Med";
import High from "./components/quiz/High";
function App() {
  return (
    <Router>
      <Route path="/" exact component={AuthForm}></Route>
      <Route path="/home" exact component={Home}></Route>
      <Route path="/home/play" exact component={DifficultyLevels}></Route>
      <Route
        path="/home/instructions"
        exact
        component={QuizInstructions}
      ></Route>
      <Route path="/home/play/low" exact component={Low}></Route>
      <Route path="/home/play/med" exact component={Med}></Route>
      <Route path="/home/play/high" exact component={High}></Route>
      <Route path="/home/play/summary" exact component={QuizSummary}></Route>
    </Router>
  );
}

export default App;

```

Slika 4.8. App.js

4.2. Back-end

Ono što korisnik ne vidi, a potrebno je imati kako bi stranica bila interaktivna događa se u *back-end* dijelu aplikacije. U aplikaciji se koristi za spremanje podataka unesenih od strane korisnika pri prijavi. Iako postoje mnogi drugi načini izrade *back-end* dijela aplikacije kao što je samostalno pisanje koda, ovdje se koristila spomenuta tehnologija, već izrađeni *back-end Firebase*. Za potrebe ove aplikacije, Firebase je najjednostavnija i najučinkovitija platforma.

Kao što je već spomenuto, *back-end* dio se koristi u početku aplikacije pri prijavi ili registraciji novog korisnika. Način na koji možemo pratiti sve registrirane korisnike prikazan je slikom 4.9.



Slika 4.9. Registrirani korisnici

4.2.1. Baza podataka

Kako bi se uspješno napravio kviz, potrebno je još kreirati pitanja za određenu razinu. Sva pitanja se nalaze u bazi podataka, koja su spremljena u JSON-u. Imamo tri različite .json datoteke te svaka sadrži pitanja za odgovarajuću težinu. Pitanja za laganu i normalnu razinu se sastoje od šest dijelova: pitanja, četiri ponuđene opcije odgovora te točan odgovor. Kod teške razine imamo samo pitanje te točan odgovor. Sva pitanja se nalaze unutar uglatih zagrada, a primjer jednog pitanja lagane razine prikazan je slikom 4.10.

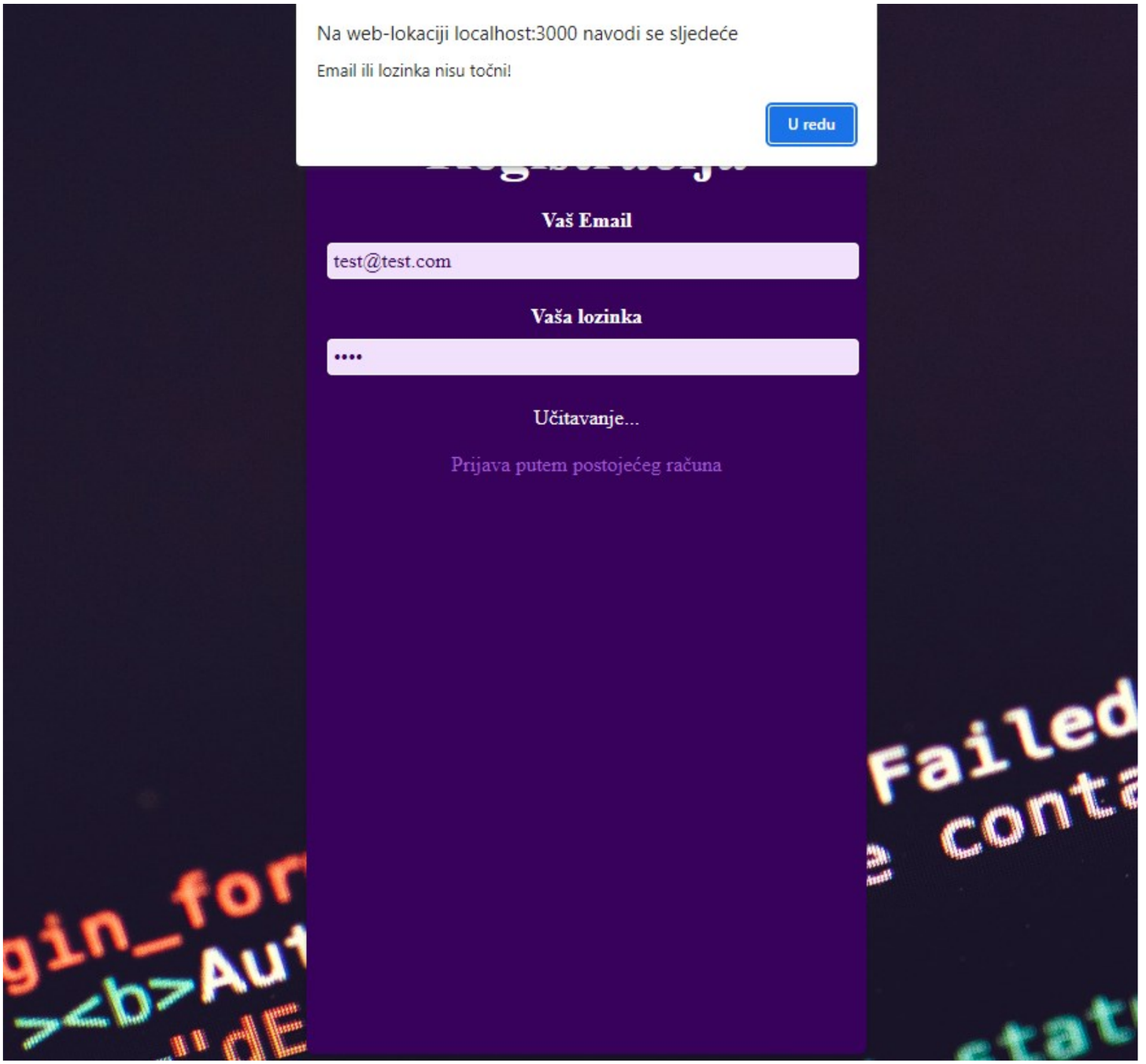
```
[
    {
        "question": "Koji tip podatka predstavlja cjelobrojni tip?",
        "optionA": "float",
        "optionB": "double",
        "optionC": "int",
        "optionD": "char",
        "answer": "int"
    },
```

Slika 4.10. Kreiranje pitanja

5. TESTIRANJE RADA APLIKACIJE

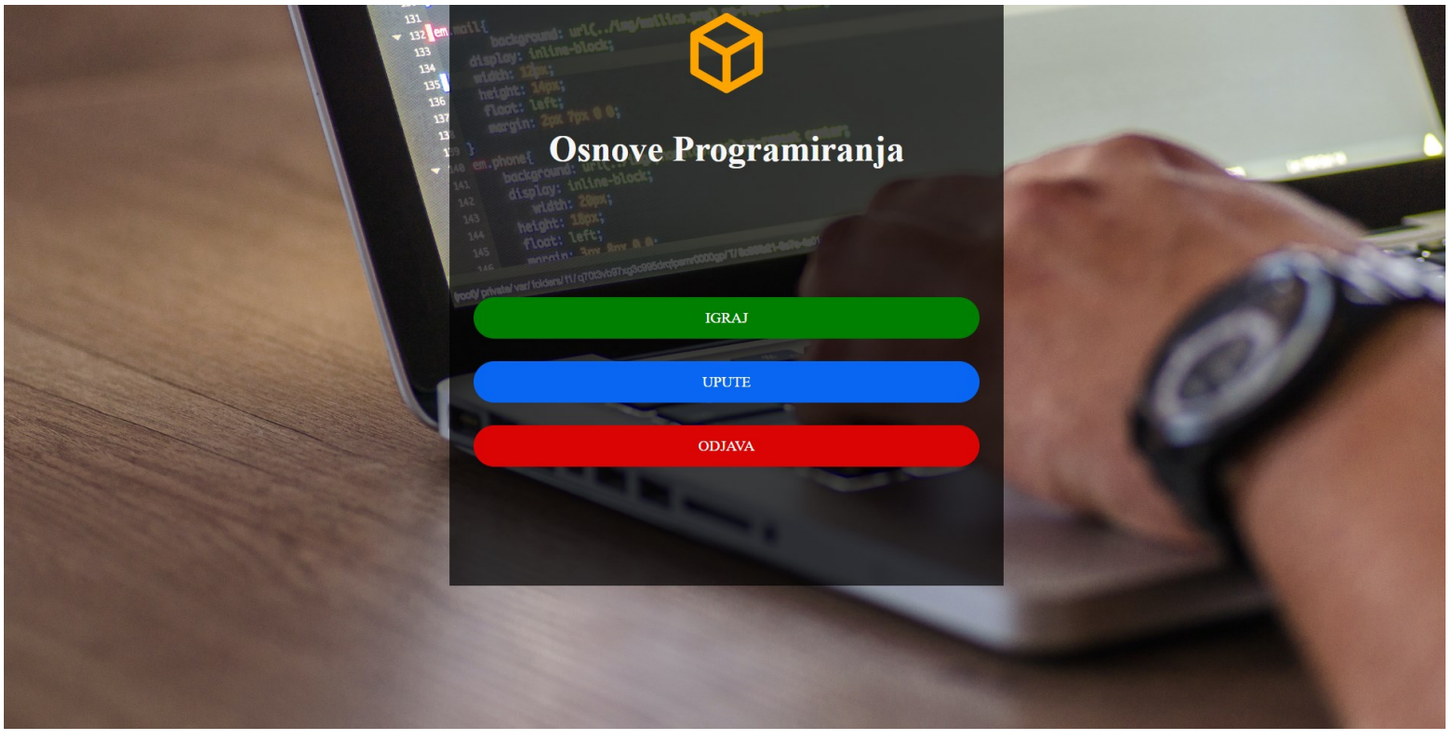
Pokretanjem aplikacije otvara se prozor prikazan slikom 5.1. Ukoliko nemamo račun, klikom na „Napravite novi račun“ korisnik se može registrirati. Svi detalji o ispravnosti prijave su navedeni. Ukoliko unesemo nešto krivo izbacit će poruku prikazanu slikom 5.2. Ako se sve točno unijelo, klikom na gumb „Prijava“ ili „Napravite račun“ aplikacija nas vodi na početnu stranicu.

Slika 5.1. Prijava/Registracija korisnika

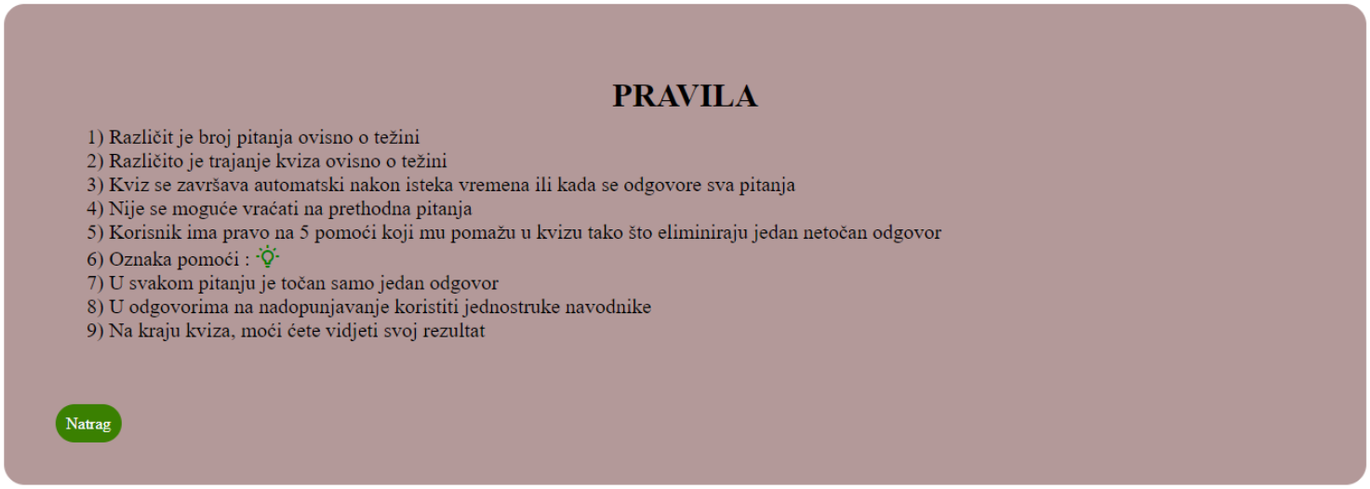


Slika 5.2. Netočan unos lozinke

Početna stranica je jednostavnog oblika te služi kao navigacijska traka za ostale funkcionalnosti koje aplikacija nudi. Sastoji se od tri gumba, a izgled je prikazan slikom 5.3. Opcija upute, odnosno njen sadržaj dan je slikom 5.4.

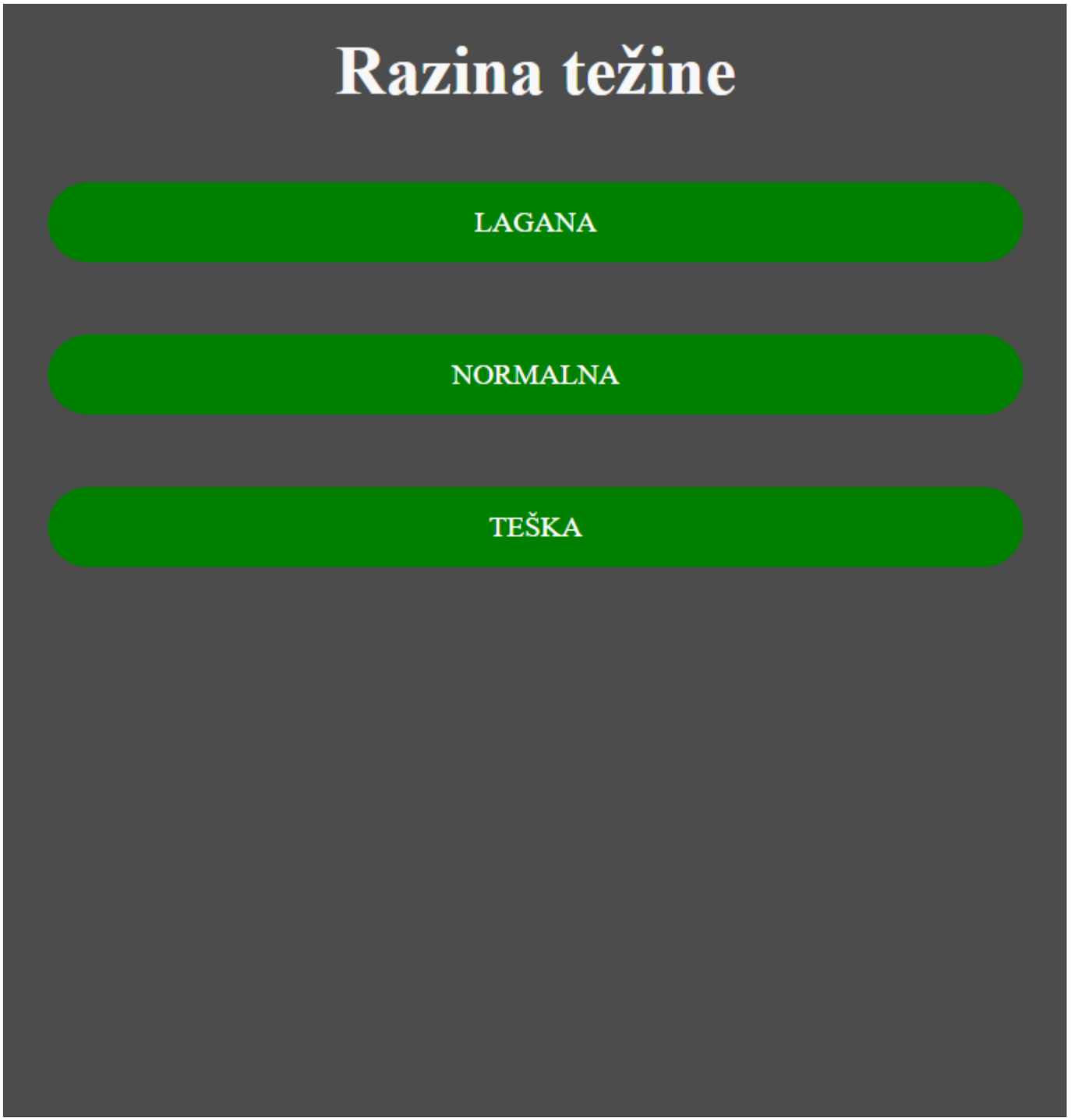


Slika 5.3. Početna stranica



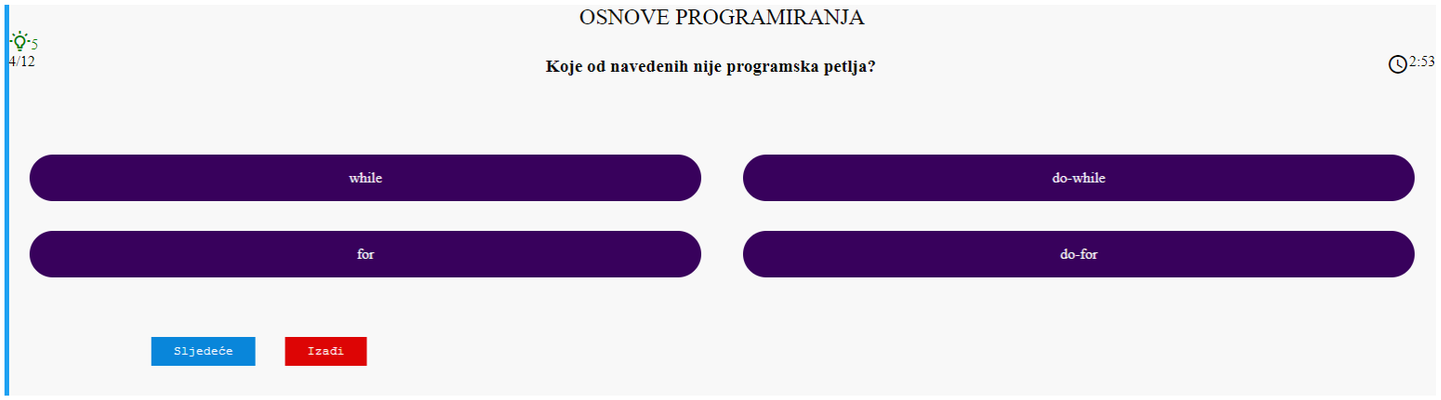
Slika 5.4. Upute

Pritiskom na gumb igraj, otvara se novi prozor s razinama težine. Pritiskom na svaku od ponuđenih opcija, otvara se vrsta kviza jedinstvena za tu razinu. Slikom 5.5. prikazan je izgled odabira težine.

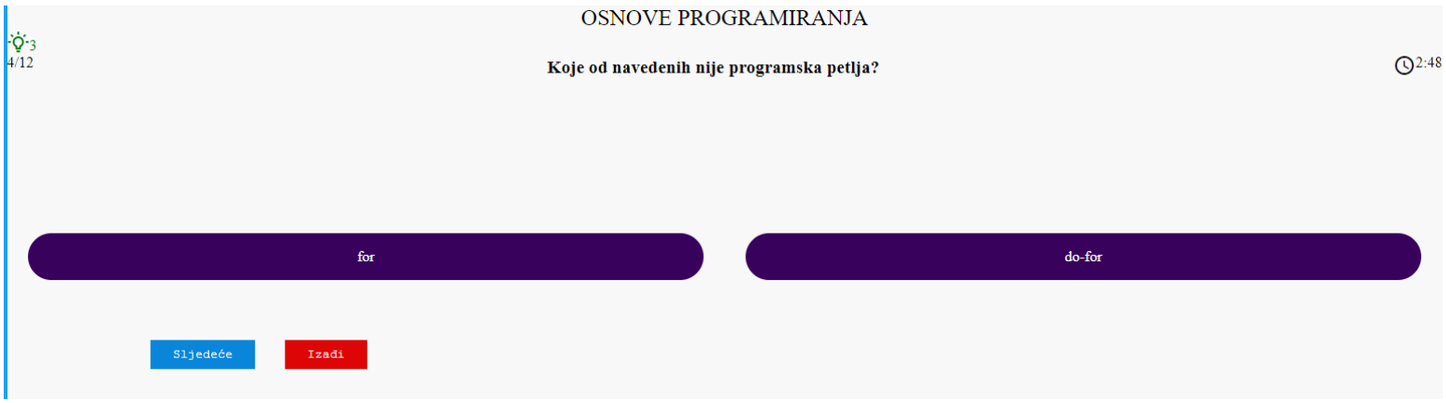


Slika 5.5. Odabir težine kviza

Lagana težina je razina s najviše mogućnosti. Lampica u gornjem lijevom kutu predstavlja već spomenutu pomoć. Ona eliminira jedan netočan odgovor iz pitanja te maksimalno možemo iskoristiti tri pomoći po pitanju. Sveukupno, kroz cijeli kviz imamo maksimalno pet pomoći, a kako će ih rasporediti, korisnik odlučuje samostalno. Slikom 5.6. prikazan je početni oblik pitanja, a slikom 5.7. oblik pitanja kada se iskoriste dvije pomoći na pitanju.



Slika 5.6. Lagana razina – oblik pitanja

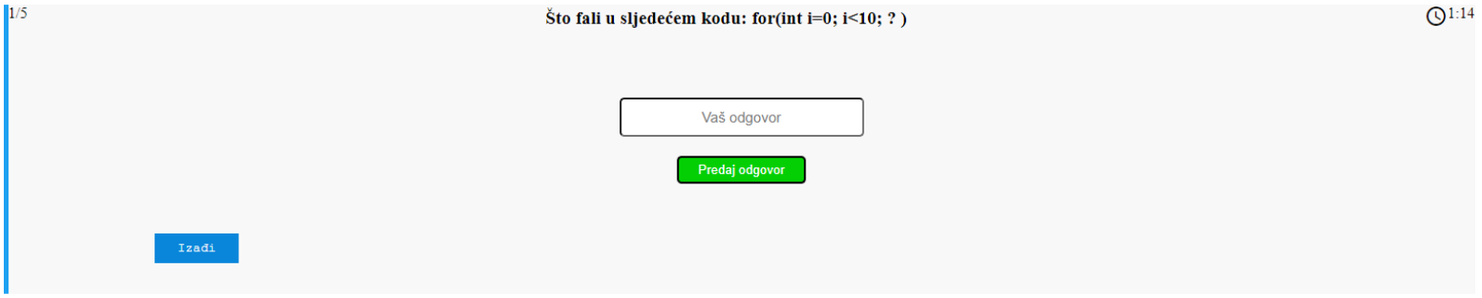


Slika 5.7. Lagana razina – korištenje pomoći

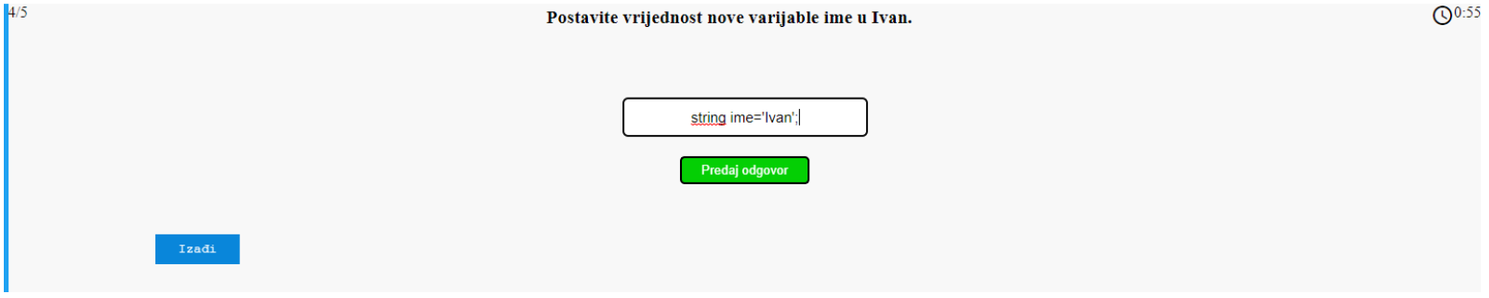
U laganoj i normalnoj razini možemo preskočiti pitanje gumbom „Sljedeće“, ali se ne možemo vratiti na preskočena pitanja. Osim toga, možemo napustiti kviz gumbom „Izađi“ koji će nas odvesti na početnu stranicu. Normalna razina ima isti oblik pitanja kao i lagana razina. Razlika je samo u tome što je vrijeme trajanja kviza u toj razini kraće te nemamo mogućnost korištenja pomoći.

Teška razina ne nudi odgovore na pitanja, već se dio koda ili cijela linija koda mora napisati. Također, nudi zadatke gdje je napisan nekakav kod, a korisnik mora nešto izračunati. Primjeri zadataka iz teške razine prikazani su slikama 5.8. i 5.9.

Nakon završetka kviza, korisnik je prebačen na stranicu koja prikazuje njegov rezultat rješavanja. Na slici 5.10. prikazan je sažetak nakon rješavanja lagane razine.



Slika 5.8. Teška razina – nadopunjavanje koda



Slika 5.9. Teška razina – samostalno pisanje linije koda

Slika 5.10. Rezultat

6. ZAKLJUČAK

Cilj ovog završnog rada bio je izraditi React web aplikaciju za učenje programiranja. Aplikacija je bazirana na učenju osnovnih pojmova i strukture programskog jezika C. Kroz više razina težine, implementirano je i više oblika rješavanja zadataka. Aplikacija je izuzetno jednostavna za koristiti te bi joj se svaki novi korisnik vrlo brzo prilagodio. Prolaskom kroz sve razine, trebao bi dobiti osnovna znanja iz programiranja u tom jeziku.

Moguća su brojna unaprjeđenja ove aplikacije, poput dodavanja mogućnosti proširivanja baze podataka pitanja ili novi oblik rješavanja zadataka. Sve je veći broj aplikacija s istom tematikom, stoga korisniku treba pružiti što bolji i što je mogući jednostavniji oblik aplikacije kako bi se odlučio baš za našu.

Za izradu ove aplikacije bilo je potrebno proučiti sve popularniji React, biblioteku JavaScript u kombinaciji s HTML-om i CSS-om. Osim toga, bilo je potrebno uključiti u projekt i Firebase, koji je znatno pomogao kao gotovi oblik *back-end* dijela aplikacije. Za pohranu pitanja korišten je JSON koji se pokazao vrlo jednostavnim za koristiti s obzirom na potrebe ove aplikacije. Sve ove tehnologije, osim Firebase-a, korištene su u Visual Studio Code-u koji je poslužio kao vrlo praktično i pregledno razvojno okruženje.

LITERATURA

[1] *Sololearn*, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.sololearn

[Datum zadnjeg pregleda: 1.lipnja 2022.]

[2] *Grasshopper*, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.area120.grasshopper [Datum zadnjeg pregleda: 1. lipnja 2022.]

[3] *Programming Hub*, Trgovina Google Play, dosutpno na: https://play.google.com/store/apps/details?id=com.freeit.java [Datum zadnjeg pregleda: 1. lipnja 2022.]

[4] *C Programming*, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.spdroid.c [Datum zadnjeg pregleda: 1.lipnja 2022.]

[5] *Lightbot*, Trgovina Google Play, dostupno na: https://play.google.com/store/apps/details?id=com.lightbot.lightbothoc [Datum zadnjeg pregleda: 1.lipnja 2022.]

[6] D. Stančer, Osnove JavaScripta, Creative Commons, Zagreb, 2015.

[7] C. Gackenheimer, Introduction to React, Apress, New York, 2015.

[8] Osnove razvoja web i mobilnih aplikacija – Laboratorijska vježba 1, dipl.ing. Marina Peko, dostupno na: https://moodle.srce.hr/2021-2022/mod/folder/view.php?id=2153170 [Datum zadnjeg pregleda: 3. lipnja 2022.]

[9] Osnove razvoja web i mobilnih aplikacija – Laboratorijska vježba 2, dipl.ing. Marina Peko, dostupno na: https://moodle.srce.hr/2021-2022/mod/folder/view.php?id=2153171 [Datum zadnjeg pregleda: 3.lipnja 2022.]

[10] Osnove razvoja web i mobilnih aplikacija – Laboratorijska vježba 3, dipl.ing. Marina Peko, dostupno na: https://moodle.srce.hr/2021-2022/mod/folder/view.php?id=2153172 [Datum zadnjeg pregleda: 4.lipnja 2022.]

[11] Osnove razvoja web i mobilnih aplikacija – Laboratorijska vježba 4, dipl.ing. Marina Peko, dostupno na: https://moodle.srce.hr/2021-2022/mod/folder/view.php?id=2153174 [Datum zadnjeg pregleda: 4.lipnja 2022.]

[12] Službena web stranica *Firebase* platforme dostupno na: https://firebase.google.com/ [Datum zadnjeg pregleda: 6.lipnja 2022.]

[13] B. Smith, Beginning JSON, Apress, New York, 2015.

[14] Službena Microsoft stranica o Visual Studio Code-u, dostupno na: https://docs.microsoft.com/hr-hr/power-apps/maker/portals/vs-code-extension [Datum zadnjeg pregleda: 2.srpnja 2022.]

SAŽETAK

U ovom završnom radu bilo je potrebno izraditi aplikaciju koja bi korisnicima pomogla u učenju programiranja. Aplikacija je namijenjena programerima početnicima i omogućava im da na interaktivan način vježbaju znanje na jednostavnim zadacima. Bazirana je na učenju jednog programskog jezika te nudi nekoliko razina zahtjevnosti zadataka za vježbu. Za izradu ove aplikacije korišten je React, biblioteka programskog jezika JavaScript, HTML, CSS te Firebase. Navedene su i druge, poznatije aplikacije s istom namjenom u radu. Rješavanjem zadataka koje aplikacija nudi, korisnik dobiva osnovna znanja iz programiranja.

Ključne riječi: aplikacija, kviz, programiranje, React

ABSTRACT

Application development for programming learning

This graduation thesis required designing an application which would help users learn programming. The application is intended for beginner level programmers, allowing them to practice their programming skills interactively through simple tasks. It is based on learning one of the programming languages, and offers several levels of difficulty. To design this application, we have used React, HTML, CSS, Firebase and a Javascript library. Other, more popular applications with the same purpose are also listed. The user gains fundamental programming skills by completing the given tasks.

Keywords: application, quiz, programming, React