**ZAVRŠNI RAD**

**INTERAKTIVNO WEB PROGRAMSKO RJEŠENJE ZA PROCJENU VJEŠTINA KANDIDATA**

**Učenik: Rei Krstić** Mentorica:

Razred: g4p Nikolina Smilović

Sadržaj

[UVOD 2](#_Toc97734127)

[1. Osnovno o projektu 3](#_Toc97734128)

[1.1 Cilj projekta 3](#_Toc97734129)

[1.2 Korištene tehnologije 3](#_Toc97734130)

[2. Sadržaj projekta 4](#_Toc97734131)

[2.1 Index stranica 4](#_Toc97734132)

[2.2 Baza podataka 11](#_Toc97734133)

[2.3 Prijava i registracija 12](#_Toc97734134)

[2.4 Test brzine kliktanja 16](#_Toc97734135)

[2.5 Test – Trener Ciljanja 19](#_Toc97734136)

[2.6 Test punjenja ljestvice 23](#_Toc97734137)

[2.7 Test sreće 26](#_Toc97734138)

[2.8 Test pogađanja niza brojeva 28](#_Toc97734139)

[2.9 Test pogađanja velikog broja 30](#_Toc97734140)

[2.10 Test Reakcije 32](#_Toc97734141)

[2.11 Rezultati 34](#_Toc97734142)

[2.12 Odjava 36](#_Toc97734143)

[LITERATURA 38](#_Toc97734144)

# UVOD

U današnje vrijeme za naći posao u dosta slučajeva biti će potreban samo životopis, no isto tako, u dosta slučajeva se također ispostavi da taj sami životopis nije točno i u cijelosti prikazao sposobnosti, te puni potencijal zaposlenika. Zbog ovoga se često može dogoditi da taj zaposlenik bude postavljen na neprikladnu radnu jedinicu(na kojoj će raditi posao koji mu je ili previše lagan ili pretjerano težak).Priloženi projekt bavi se interaktivnim web programskim rješenjem za procjenu vještina kandidata(ili bolje rečeno zaposlenika)!

Cilj ovog interaktivnog rješenja bi bio pomoći pri odabiranju pravog posla za odabranog zaposlenika kroz serije testova za provjeru njihovih fizičkih i psihičkih sposobnosti(npr. test reakcije, test pamćenja…).

# Osnovno o projektu

## Cilj projekta

Ovo interaktivno web programsko rješenje procjene vještine kandidata imalo bi upravo taj cilj, procijeniti vještine kandidata(bile one psihičke ili fizičke). To je omogućeno kroz seriju testova koji procjenjuju razne vještine. Projekt uključeno ima sedam testova(procjena).

U njih su uključeni testovi:

* Test brzine kliktanja
* Test – trener ciljanja
* Test punjenja ljestvice
* Test sreće
* Test pogađanja niza brojeva
* Test pogađanja velikog broja
* Test reakcije

## Korištene tehnologije

Cijeli projekt isprogramiran je u programu Visual Studio Code, a potreban je i program XAMPP kako bi se projekt i baza podataka mogli pokrenuti na lokalnom serveru, samo se treba cijeli projekt staviti u mapu na putanji „C:\xampp\htdocs“. Visual Studio Code je *code editor* redefiniran i optimiziran za *debugging* i izgradnju modernih web aplikacija.

U programu se koriste programski jezici i *frameworkovi*:

1. **HTML**(Hypertext Markup Language, standardizirani sustav za označavanje tekstualnih datoteka kako bi se postigli efekti fonta, boje, grafike i linka na World Wide Web stranicama)

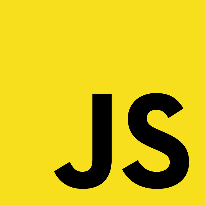


1. **CSS**(Cascading Style Sheets, programski jezik dizajna koji uključuje sve relevantne informacije koje se odnose na prikaz web stranice)



[Ta fotografija](https://www.multitek.no/innlegg/hvordan-virker-nettsider-2/) korisnika Nepoznat autor: licenca [CC BY-SA-NC](https://creativecommons.org/licenses/by-nc-sa/3.0/)

1. **JavaScript**(programski jezik koji vam omogućuje implementaciju složenih komponenti na web-stranicama)



1. **PHP**(Hypertext Pre-processor, poslužiteljski ili serverski programski jezik i moćan alat za izradu dinamičnih i interaktivnih web-stranica)



[Ta fotografija](https://pngimg.com/download/60275) korisnika Nepoznat autor: licenca [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

1. **SQL**(Structured Query Language, standardni programski jezik za pristup i interakciju s bazama podataka)



[Ta fotografija](https://pngimg.com/download/60311) korisnika Nepoznat autor: licenca [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

1. **Bootstrap**(*open-source* CSS *framework* usmjeren na, čak i za mobilne uređaje, responzivan front-end web razvoj)



[Ta fotografija](https://getbootstrap.com/) korisnika Nepoznat autor: licenca [CC BY](https://creativecommons.org/licenses/by/3.0/)

# Sadržaj projekta

## Index stranica

**Index stranica**(početna stranica) je stranica koja se prva pokrene kad se uđe u direktorij od projekta.

Prvo se radi html *tag* ili oznaka(<html>) koja se također, kao mnoge druge oznake trebaju zatvoriti(</html>). Cijeli kod za stranicu piše se unutar ovih oznaka. Zatim se cijeli html dijeli na dva dijela, head i body oznake. U body oznaci se radi primarno struktura stranice(dijelovi koji će se zapravo renderirati na stranici). Head oznaka je većinom zaslužna za ono što se događa u pozadini(npr. učitavanje skripti, dizajn…). Ova se struktura koristi za sve html dokumente u projektu.

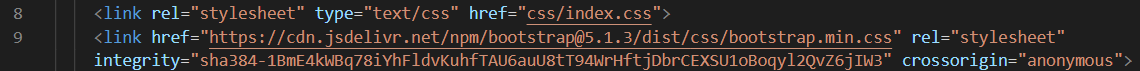
Ta html struktura bi tipično trebala nekako ovako izgledati(u Visual Studio Code-u ova cijela struktura se može kreirati samo pomoću tipke za uskličnik):

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Gore u priloženoj slici osim samo html, head i body oznaka tu se nalaze još neke automatski dodane linije koda. U prvoj liniji koda kraj ključne riječi html dodano je još !DOCTYPE. Ova će ključna riječ samo reći pregledniku kojeg trenutno koristimo o kojoj se verziji HTML-a radi na web stranici. Na drugoj liniji uz već poznatu html oznaku može se vidjeti lang=“hr“. Ovo će samo postaviti primarni jezik na stranici na hrvatski. Kroz linije koda četiri, pet i šest se može vidjeti nova oznaka – meta(valja također primijetiti kako je ovo jedna od oznaka koje nije potrebno zatvarati). Pomoću ove oznake se mogu dohvatiti i postaviti određeni podatci u tom html dokumentu. Na primjer pomoću charset=“UTF-8“ možemo koristiti više različitih simbola koje inače nismo u mogućnosti koristiti, kod pod linijom pet pomaže podržati određene internetske preglednike i linija koda pod linijom šest pomaže postaviti rezoluciju te stranice da odgovara rezoluciji uređaja na kojoj je stranica trenutno pregledavana. Title oznaka pod linijom sedam postavlja naslov na kartici(*tabu*) prozora preglednika.

Unutar head oznake sam stavio još ove dvije linije koda:



Preko osme linije koda(kroz link oznaku) *linkam* se na css datoteku(datoteka u mapi css u kojem se nalazi dizajn za stranicu, dizajn se još mogao kroz stavljanje style oznake direktno u ovu datoteku mogao odmah ovdje koristiti). Taj css je „ručno“ napisan dok pomoću koda sa devete linije se povezuje sa **Bootstrapom** radi lakšeg samog dizajniranja i strukturiranja stranice.

Ovi slika i naslov su prvi dio index stranice projekta(prva stvar koju korisnik vidi):

Slika na kojoj se prikazuje objekt na otvorenom, web

Opis je automatski generiran

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Prvo otvaramo **div** oznaku(html element u koji možemo stavljati druge html elemente i kojim možemo manipulirati JavaScriptom, te dizajnirati) i zatvaramo ga pri kraju body oznake da „zatvorimo“ sve druge html elemente unutar njega kako bi njima lakše manipulirali. Zato ovom divu dajemo **id**(id se može postaviti na neki element kako bi ga se poslije lakše lociralo pomoću npr. CSS-a ili JS-a). Kraj ostalih divova su postavljene **klase**(imaju sličnu funkciju kao id-ovi samo što isti id se ne može staviti na više elemenata dok se jedna klasa može) koje se u ovom slučaju zapravo koriste kao dio Bootstrapa(u ovom slučaju svaka klasa s ovog primjera ima svoj style ili „dizajn“ već kodiran, bolje rečeno pri pozivanju jedne postojeće Bootstrap klase, onda se style te klase pozove na element na koji je ta klasa stavljena, npr. uz pomoć klase text-center mi ne trebamo centrirati tekst u ovom div-u sami, već klasa to napravi za nas). Tekst na stranici izgleda kao naslov jer je napisan unutar h1 oznake(**oznake h6 do h1** su naslovi, h6 je najmanji naslov, a h1 najveći). Uz klasu na div-u kod linije broj trinaest postavljen je i id „header“. Pomoću njega se na taj element poziva style koji je kodiran kroz linkanu index.css datoteku:

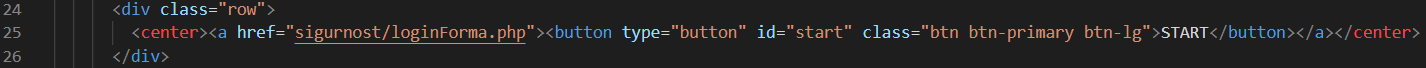
Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Kako bismo označili da se radi o id-u, a ne o klasi ili općenito nekoj drugoj oznaci stavljamo znak *hashtag(*#), a stilove navodimo unutar vitičastih zagrada. Kako bismo stvorili ovu naslovnu sliku da se pojavi imamo ju dostupnu lokalno(u folderu „slike“). Na liniji četiri u url-u od background-imagea definiramo putanju kako bi se našla ta lokalna slika i prikazala se(sa dvije točke se izlazi iz trenutnog foldera, zatim se ulazi u folder slike, te se napokon odabire sama slika). Na liniji dva definiramo height(osim heighta ili visine ima i width ili širina koju također možemo definirati) div-a na 100vh(100vh je točno veličina prozora). Background repeat se stavlja na „no-repeat“ kako slika ne bi, ako je manja, nego što bi trebala biti, se ponavljala ili *popločavala.* Background-size „coverom“ potvrđuje da će se pozadinska slika prikazati preko cijelog div-a.

Sljedeći dio stranice:





Kao prvo, center oznakom je lakše centrirati elemente, button „START“ ćemo koristiti za prebačaj na login nakon čega će korisnik moći pristupiti testovima. To će se napraviti na način da se button oznaka za pravljenje gumba stavi u *a* oznaku, te u *href* stavljamo putanju za login kako bismo se linkali na tu stranicu, no više o tom kasnije. Na stil buttona i njegove okoline djeluje se u CSS-u kroz id:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Koristi se **background-color**, te se postavlja boja u **rgba** formatu(rgb znače udio crvene zelene i plave boje, a zadnji parametar predstavlja a, jačina transparentnosti ili prozirnosti, stavlja se a na dvjesto pedeset pet koji je maksimalan broj u formatu – element neće biti uopće biti proziran).Font-sizeom mijenjamo veličinu fonta. **Margin** djeluje između mjesta van elementa i bordera od elementa po pikselima(na ovom primjeru se mijenja samo gornja margina).

Slika na kojoj se prikazuje tekst, monitor, na zatvorenom, snimka zaslona

Opis je automatski generiran

Iduće komponenete su zaobljeni divovi sa slikama unutra, te tekstom ispod. Ovdje će se koristititi važna komponenta vezana za Bootstrap. Kad se stavi na div klasa row sve će unutra biti dio retka. Ako stavimo unutar tog rowa dva div-a sa klasama col(kao što smo ovdje) oni će stvoriti dva stupca(div-a) kojim se može upravljati grid strukturom od dvanaest dijelova. Unutar tih *stupaca* se nalaz div-ovi s klasama „card-body“ jer imaju specifičnu predefiniranu Bootstrap strukturu. Umjesto background-imagea ovaj put se stavlja **img** oznaka pomoću koje se stavi slika preko src-a. Pomoću margina se kartice odmaknu od okoline.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Na gore priloženoj slici prikazan je stil za okrugle div-ove sa klasom ikone, te za slike u div-ovima sa klasama ikone-img(klase se u CSS-u označavaju sa točkom). Div je okrugao jer je border-radius postavljen na sto posto(pomoću njega se potpuno zaoble svi kutovi običnog div-a). Kako bi se slike centrirale u sredinu diva koristi se position:“absolute“ na slici, te onda možemo manipulirati sliku lijevo(left), desno(right), gore(top) ili desno(bottom), no prije toga trebamo staviti position:“relative“ na parent element(okrugli div) kako bi o njemu *ovisio* child element(slika).

Sljedeći dio index stranice je:

Slika na kojoj se prikazuje tekst

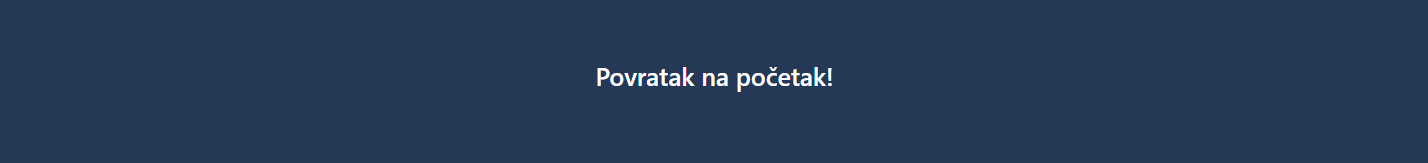
Opis je automatski generiran

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

U p oznaku se stavljaju paragrafi i tekst. Pomoću br oznake tekst prelazi u novi red. Hr oznaka stavlja vodoravnu liniju. Na nju je odmah stavljen style(izvan posebne datoteke), brži način i *specifičniji.*

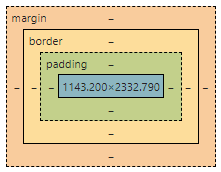
Ovo je posljednji dio index stranice(na samom dnu):



Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Color će promijeniti boju teksta u bijelo. **Padding** označava veličinu prostora koju možemo mijenjati između bordera(ruba elementa) i sadržaja elementa. Primjer paddinga gore u prilogu je napisan u skraćenom obliku(inače se može specifično pisati npr. padding top, padding-bottom…). Nakon dvotočke od paddinga pedeset piksela od bordera na prvom i trećem mjestu između razmaka vrijedi za gornji i donji padding, dok ovi s nula piksela čine padding s desne i lijeve strane. Primjer rada(sveukupno) margina i paddinga:



Na primjeru crvena boja označava mjesto gdje djeluju margine, narančasta predstavlja border od elementa, zelena predstavlja mjesto gdje djeluje padding, a plava boja označava gdje se nalazi element. Ovo je kod u HTML-u za priloženi dio stranice gore:

Slika na kojoj se prikazuje tekst, zaslon, snimka zaslona

Opis je automatski generiran

Od šezdeset pete do šezdeset osme linije koda sve bi trebalo biti poznato, no na sedamdesetoj liniji se već počinje koristiti JavaScript. Kao i s CSS datotekom script oznaka se može pisati direktno u ovoj datoteci, no ipak je pisan u zasebnoj kako bi sve bilo efikasnije. Tu JS(JavaScript) datoteku pozivamo na samom kraju body oznake kako bi se poboljšao *loadtime* učitavanje svega na stranici. Ovo je sve što se nalazi u toj datoteci:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Cilj ove skripte je da nakon što mi kliknemo na naslov h5(„Povratak na početak!“) da nas stranica odbaci skroz do početne naslovne slike s naslovom(„Psihoevaluacijski testovi“). Na prvoj liniji koda u prvom dijelu pomoću **document.getElementById**(„scroll“) program će tražiti kroz HTML datoteku neki jedan element koji ima id „scroll“ u ovom slučaju element h5(s tekstom „Povratak na početak!“). Nadalje u kodu piše .**addEventListener**. Pomoću addEventListenera će se na onaj element s id-om „scroll“ postaviti *slušač* – kad se neki određeni događaj dogodi izvrši **funkciju**(blok koda koji se izvrši kad se pozove). Iza addEventListenera stoje zagrade koje primaju ta dva parametra, događaj koji je u ovom slučaju klik mišem i funkciju koja je u ovom slučaju **anonimna funkcija**(funkcija koja nema ime, stoga se ne može pozvati imenom). Kad se klikom izvrši ta funkcija prvo što će se dogoditi je console.log(„aa“). Ovim će se u konzoli ispisati „aa“. Ovaj se dio većinski koristio za **debugging**(proces rješavanja grešaka u programskom kodu) i nije nužno potreban. Pomoću koda treće linije će se na id „header“ postaviti ugrađena funkcija scrollIntoView(). Ovom funkcijom će se browser automatski „scrollati“ do tog id-a koji se u ovom slučaju nalazi na naslovnoj slici na vrhu stranice.

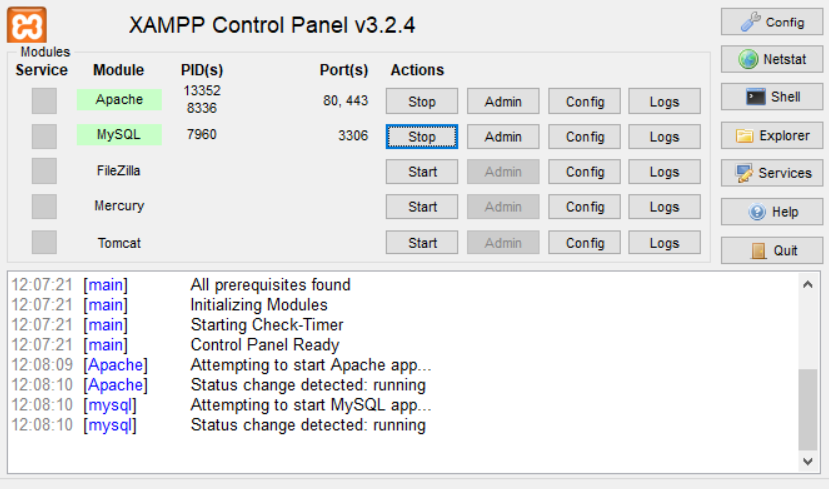
Slika na kojoj se prikazuje tekst

Opis je automatski generiran

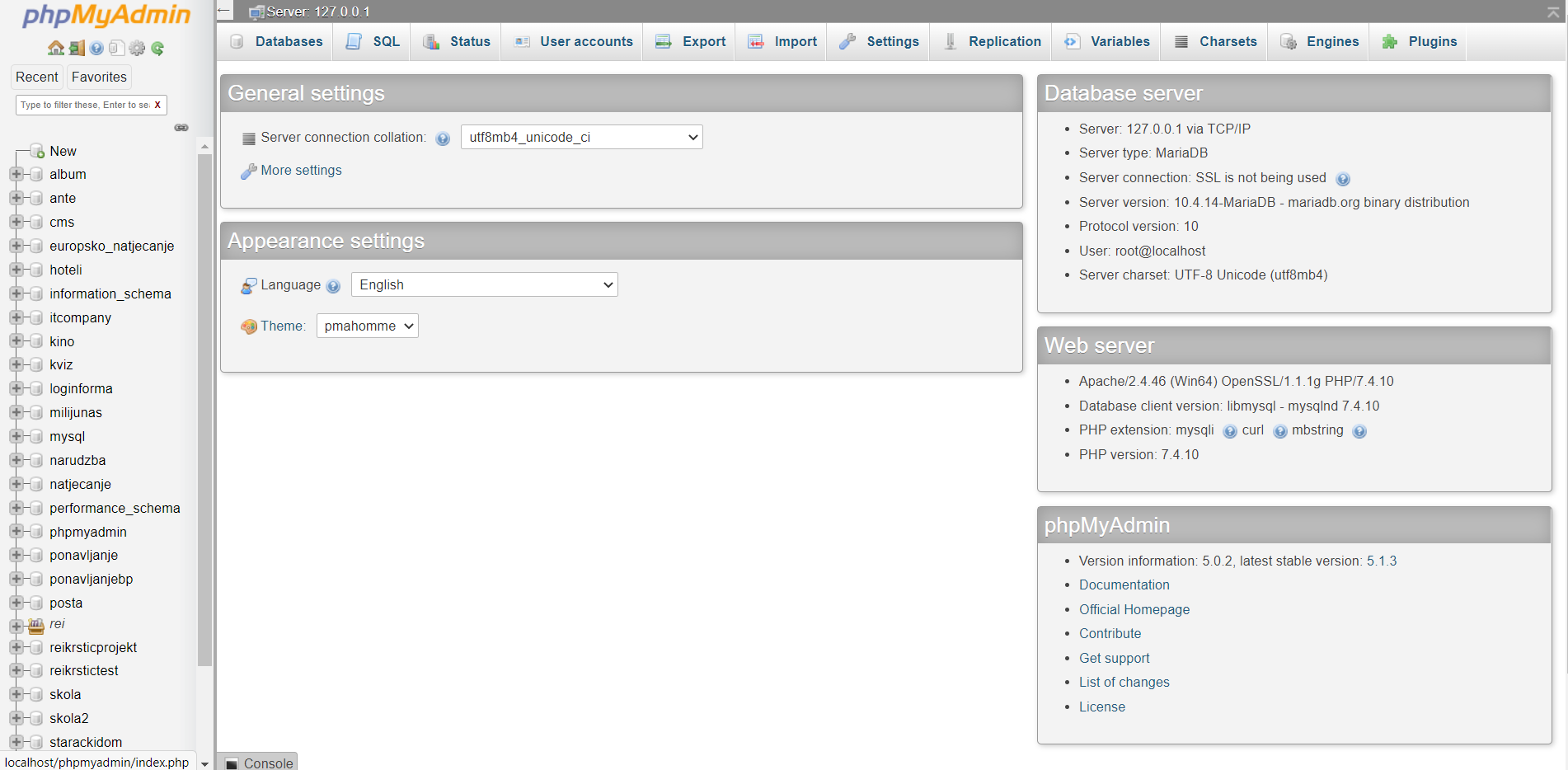
Nakon analiziranja cijele stranice može se vratiti na ovaj button gore. Klikom na ovaj button ide se do *logina* ili sustava prijavljivanja. Nakon kojeg će se napokon moći krenuti sa testovima.

## Baza podataka

Prvo se treba napraviti baza podataka. Prvo startamo Apache i MySQL u XAMPP-u:



Zatim na URL-u od nekog preglednika upišemo „localhost/phpmyadmin/“.

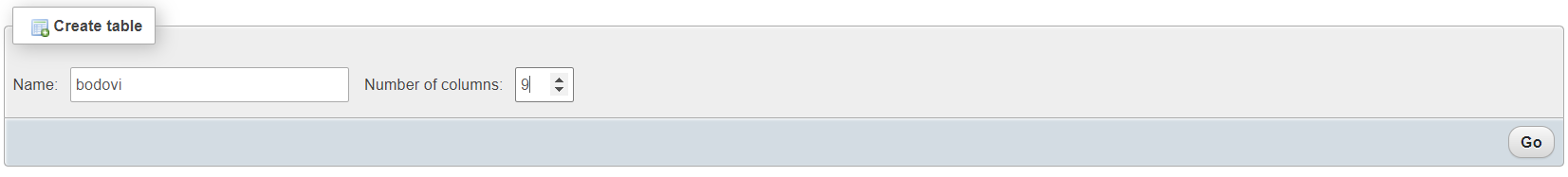


Ide se prvo na „New“ sa lijeve strane.

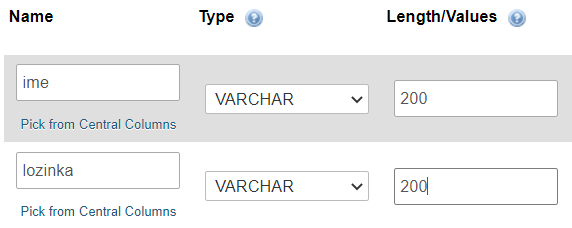
Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Zatim se zapravo kreira baza podataka.



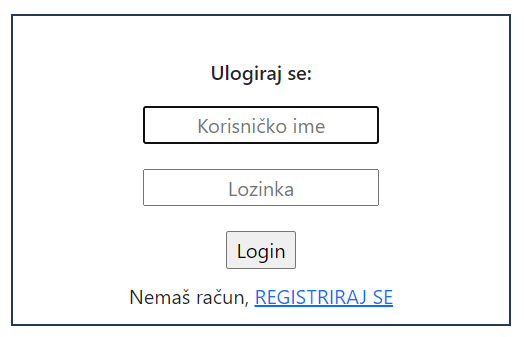
Pa se kreira tablica.



Na stupcima tablice ima više atributa, no najvažniji su Name(ime stupca), Type(tip vrijednosti stupca npr. varchar ili tekst, int ili brojevi…) i Length/Values(koliko svaka vrijednost može biti duga). Osim imena i lozinke se treba dodati jedan stupac za svaki test.

## Prijava i registracija

Nakon potpunog kreiranja baze podataka može se napokon kliknuti button start. Taj će nas button odvesti na prijavu:



Korisnik treba upisati korisničko ime i lozinku kako bi se zapamtio svaki pojedini korisnik, te da se svaki rezultat poveže sa svojim odgovarajućim korisnikom. Ukoliko korisnik nema registriranih korisničkih imena i lozinki treba se ispod registrirati.

Slika na kojoj se prikazuje tekst, monitor, zaslon, snimka zaslona

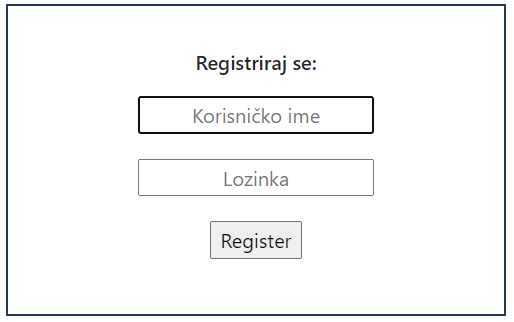
Opis je automatski generiran

Prvo se na body stavlja onload(kad se učita stranica)=“window.document.prijava.korisnik.focus()“. To će **input** (dio forme, može biti različitih vrsta, u nju korisnik sam upisuje podatke) s nameom (imenom) korisnik *focusati*(to je prvi element u koji će korisnik moći upisivati). Pomoću namea će se također poslije s PHP-om *manipulirati* podaci. Na formi(šesnaesta linija) stavlja se name, **method**(POST ili GET, jedna od razlika bi bila npr. nakon što se pošalju podatci oni će biti prikazani u URL-u sa GET metodom dok sa POST metodom neće) i action(govori na koju se stranicu s PHP-om treba linkati kako bi se podatci obradili nakon što se zapravo pošalju podatci, u ovom slučaju pomoću ovog koda unutar actiona će se linkati na istu datoteku jer ima PHP koda ispod ove forme). Zatim se postavljaju inputi, s placeholderom unutar njih se na input postavi tekst koji je tu sve dok nema ništa upisano u input, s typeom se zapravo određuje o kojem će se inputu raditi(npr text za obični tekst, password za lozinku – crne točke umjesto slova, submit koji radi na sličan način kao button – kad se klikne šalju se podatci…).Podatci se šalju ovdje:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Prvo se otvara php oznaka, zatim se provjerava sa isset je li onaj submit iz forme kliknut. Ako je kliknut onda se u varijable stavljaju korisnik i zaporka(iz forme, ispred njih se piše $\_POST jer se radi o POST metodi). Zatim stvaramo konekciju na bazu podataka sa **mysqli\_connect**. U nju se stavljaju četiri parametra, prvo localhostom određujemo na kojem se hostu može naći baza podataka, zatim drugim i trećim parametrom određujemo korisničko ime i lozinku za pristupiti toj bazi podataka(u ovom slučaju se stavlja root i prazno kako se od nas ne bi tražili nikakvo korisničko ime ni lozinka), zadnjim parametrom napišemo ime te baze. Odmah ispod konekcije nalazi se linija SQL koda. Ovom će se linijom *izvaditi* ime i lozinka iz tablice „bodovi“, ali samo one kojima su ime i lozinka isti kao u formi. Na četrdeset prvoj liniji sa **mysqli\_query** naredbomse povezuju baza i ta linija koda. Pomoću linije koda na četrdeset prvoj liniji dohvaća se ta rezultat nakon mysqli\_querya i sprema se kao lista. Zatim sa ifom provjerava jesu li postavljene te vrijednosti, te jesu li jednake null(null nije isto što i nula). Ako nisu onda započinje **sesija**(način spremanja varijabli koje se mogu koristiti kroz više različitih stranica) sa session\_start. Zatim otvaramo SESSION varijable i u njih spremamo to korisničko ime i lozinku iz baze kako bismo ih *zapamtili* za poslije. Kod na pedeset četvrtoj liniji samo služi da se dohvat id sesije. Pomoću headera se kreće na prvi test. Ako neki od uvjeta u ifu nije zadovoljen onda se iz elsea ispisuje da su neispravni korisničko ime ili lozinka i izvrši se exit(kompletno zaustavlja trenutnu skriptu) naredba. Ako korisnik nema svoje korisničko ime i lozinku u bazi podataka onda kad se prijavljuju nemaju s čime usporedit i potvrdit svoje podatke. Zato(ako već nije) se korisnik treba registrirati:



Sam izgled forme je dosta sličan prijavi.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Što se PHP dijela tiče, također je dosta sličan PHP dijelu od prijave samo što se ovaj put za SQL liniju koristi kod koji ubacuje korisničko ime i zaporku u tablicu „bodovi“. Nakon što se ovdje upišu podatci ovako bi trebali izgledati ti podatci u tablici da je za korisničko ime upisano „mate“, a za zaporku „5678“(zadnja linija na dolje priloženoj slici):

Slika na kojoj se prikazuje stol

Opis je automatski generiran

Onda se vraća na index stranicu nakon čega se može napokon prijaviti i početi s testovima.

## 2.4 Test brzine kliktanja

Kreće se s prvim testom:

Slika na kojoj se prikazuje trg

Opis je automatski generiran

Cilj ovog testa je testirati brzinu kliktanja mišem. Radi na način da kad se počne kliktati na plavi kvadrat donji broj sa slike će se smanjivati – to su sekunde, a gornji broj će se povećavati – to je broj klikova. Na zadnji klik kad sekunde dođu do nula ispisat će se naš broj klikova po sekundi. Ovaj container oko testa je stavljen oko svakog testa.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

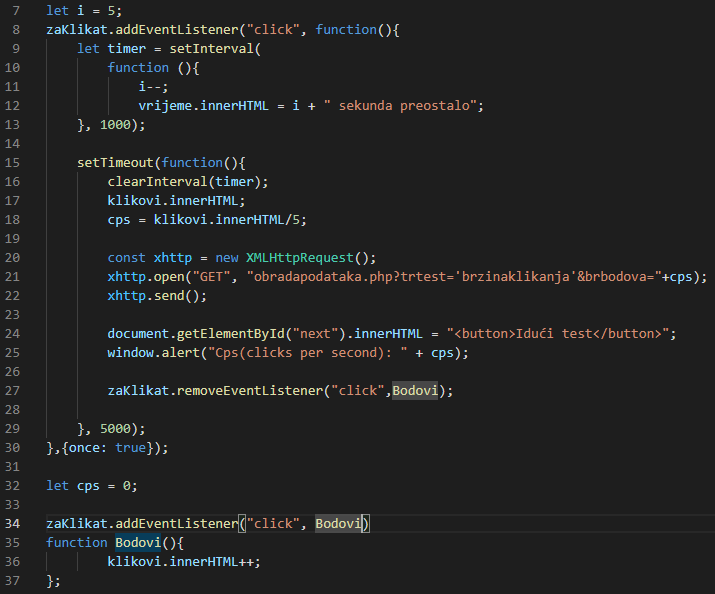
Ovo gore je početak JS datoteke ove provjere. Prvo se stvaraju **varijable**(poput *kutija* u koje se stavljaju neke vrijednosti kako bi se programiralo efikasnije i brže) u koje se stavljaju elementi s određenim id-ovima. Window.alertom(možemo koristiti i samo alert) će se ispisati uputstva napočetku svakog testa. Na ovaj će se način ispisati tekst unutar alertovih zagrada:



Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Ostatak koda:



Stvara se varijabla i, te joj se dodjeljuje vrijednost pet. Postavlja se EventListener s funkcijom do tridesete linije koda. Na toj liniji dodan je dodatan parametat, once:true(taj EventListener je moguće samo jedan put kliknuti prije nego što bude onemogućen). Unutar te funkcije prvo se radi o timeru **setInterval**(prima dva parametra funkciju i milisekunde, u ovom slušaju tisuću). Svakih tisuću milsekundi i će se smanjivati za jedan. Linijom dvanaest i **innerHTML-om**(iskoristi se na elementu, te se zatim mogu pročitati njegove vrijednosti ili znakom jednako i plus jednako se nešto može mijenjati i nadodavati) će biti prikazano smanjivanje sekundi ispod kvadrata. Dok je pokrenut ovaj interval na liniji trideset četiri taj addEventListener broji klikove i povećava ih na stranici sa klikovi.innerHTML++(to je skraćeno od klikovi.innerHTML = klikovi.innerHTML + 1). Ovom će se Listeneru dodijeliti obična funkcija s imenom jer će se ovaj Listener poslije trebati *izgasiti*. Osim toga također ima i **setTimeout**(sličan setIntervalu samo što stane nakon što se jedan put izvrši) koji djeluje nakon pet tisuća milisekundi. Na taj će način djelovati test samo pet sekundi - tj. kad removamoEventListener(linija dvadeset sedam). Osim toga clearamoInterval(zaustavimo interval od ponavljanja na liniji šesnaest), te na liniji osamnaest izračunamo klikove po sekundi. Na linijama dvadeset četiri i dvadeset pet se oko a linka ispod samog testa stavi button oznaka s tekstom unutra „Idući test“ kako bi se mogao kliknuti za preći na sljedeći test. Također se i prije toga prikazuje alert s našim prikazanim rezultatom. Pomoću linija koda od dvadesete do dvadeset druge linije pravi se AJAX(Asynchronous JavaScript And XML, koristi se za komuniciranje servera) request. Na dvadeset prvoj liniji prvo se određuje koja se metoda koristi. Zatim se napiše na koju se datoteku šalju podatci – datoteka „obradapodataka.php“ i nakon upitnika se napišu varijable koje se šalju na tu datoteku(varijable odvajamo znakom „&“).Šalje se test o kojem se trenutno radi i rezultat tog testa. Ovaj se dio stavlja na svaki test. Ovako izgleda „obradapodataka.php“:

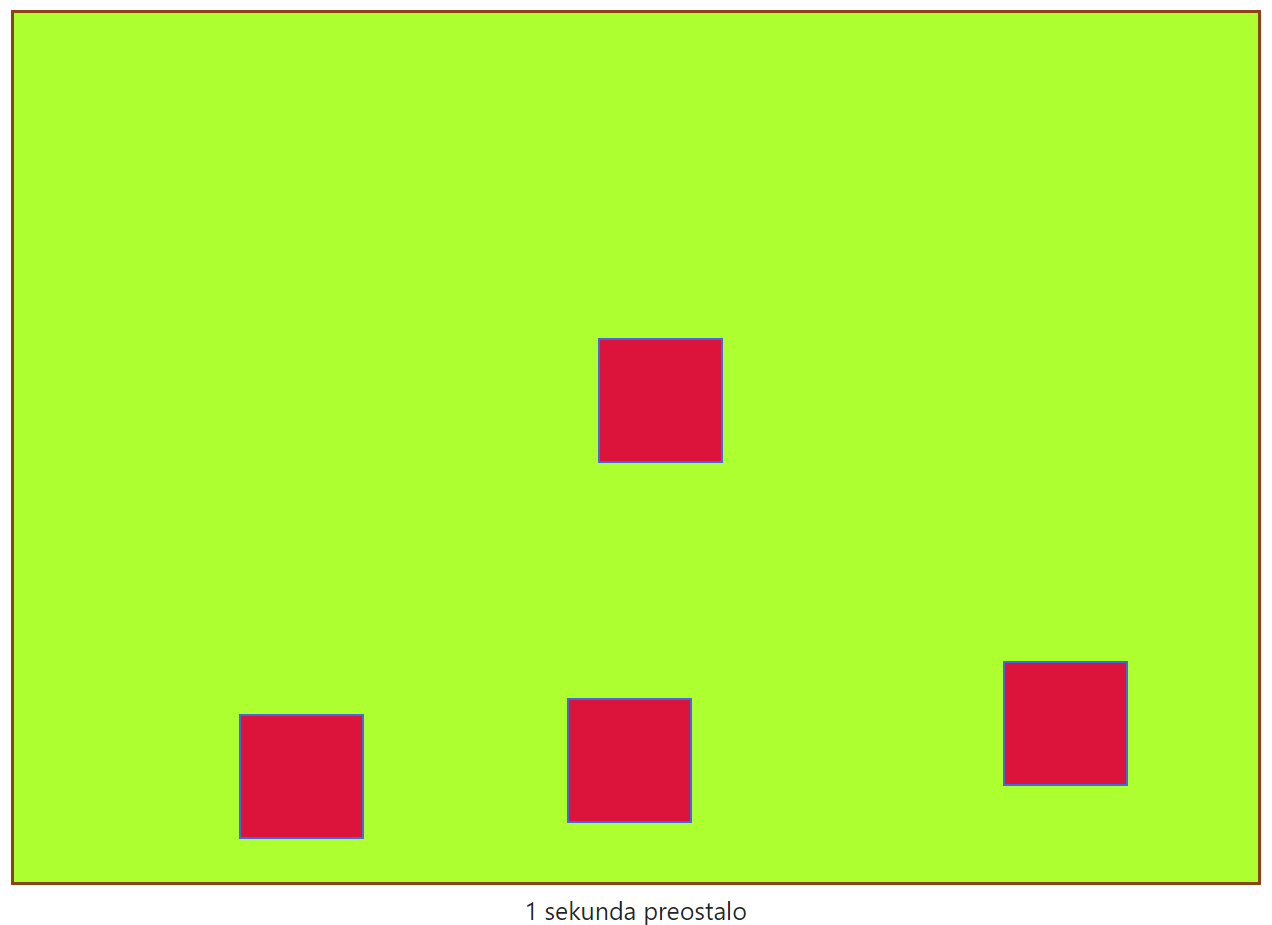
Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Ovdje se opet otvara sesija kako bismo mogli koristiti varijable iz prijave. Zatim se iz trenutnog testa koji je sada u obliku teksta miču navodnici kako bi ga prepoznala baza podataka. Ovdje se koristi **echo**(način ispisivanja) samo u svrhu debugginga. Linija SQL koda koja će se ubaciti u bazu mijenja rezultat koji se nalazi na trenutačnom testu(naravno na pravom korisniku).

## 2.5 Test – Trener Ciljanja

Idući test:



Cilj testa je odrediti koliko korisnik puta može kliknuti crvene kvadrate u pet sekundi s tim da svaki put kad neki crveni kvadrat je kliknut on promjeni poziciju. Ispod testa piše koliko je sekunda preostalo do isteka vremena. Pogotkom(klikom na jedan od malih crvenih kvadrata) dodaje se jedan bod finalnom rezultatu, promašajem(klikom na veliki zeleni kvadrat) oduzima se jedan bod od finalnog rezultata.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Ovo je konstruktor funkcija, pomoću nje se kreiraju **objekti**(poput varijable koja kao vrijednost sadrži još varijabli), u ovom slučaju crveni kvadrati. Linije koda od dvadeset pet do dvadeset osam su ti objekti(iako su *prazni*), no njima upravljamo funkcijom od pete do dvadeset treće linije koda. Prvo se napravi div od njih(linija sedam). Zatim se na njih doda klasa „meta“ kako bi se na njih primijenio ovaj CSS:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Zato i izgledaj tako kako izgledaju. Pomoću linija koda jedanaest i dvanaest početna pozicija crvenih kvadrata bude nasumična. Sa EventListenerom odmah ispod toga se dogodi da ti kvadrati promjene poziciju na svaki klik(na ovim linijama valja još samo spomenuti da **Math.random** vraća nasumičan broj od nula do jedan, a **Math.floor** zaokruži neki broj na najmanju vrijednost, u ovom se slučaju uzima nasumična pozicija između dimenzija zelenog containera – 100 jer se ubraja i veličina crvenih kvadrata). Na samom kraju konstruktora se appendaju(naljepljuju) ti objekti unutar containera(zelenog kvadrata).

Idući dio koda:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

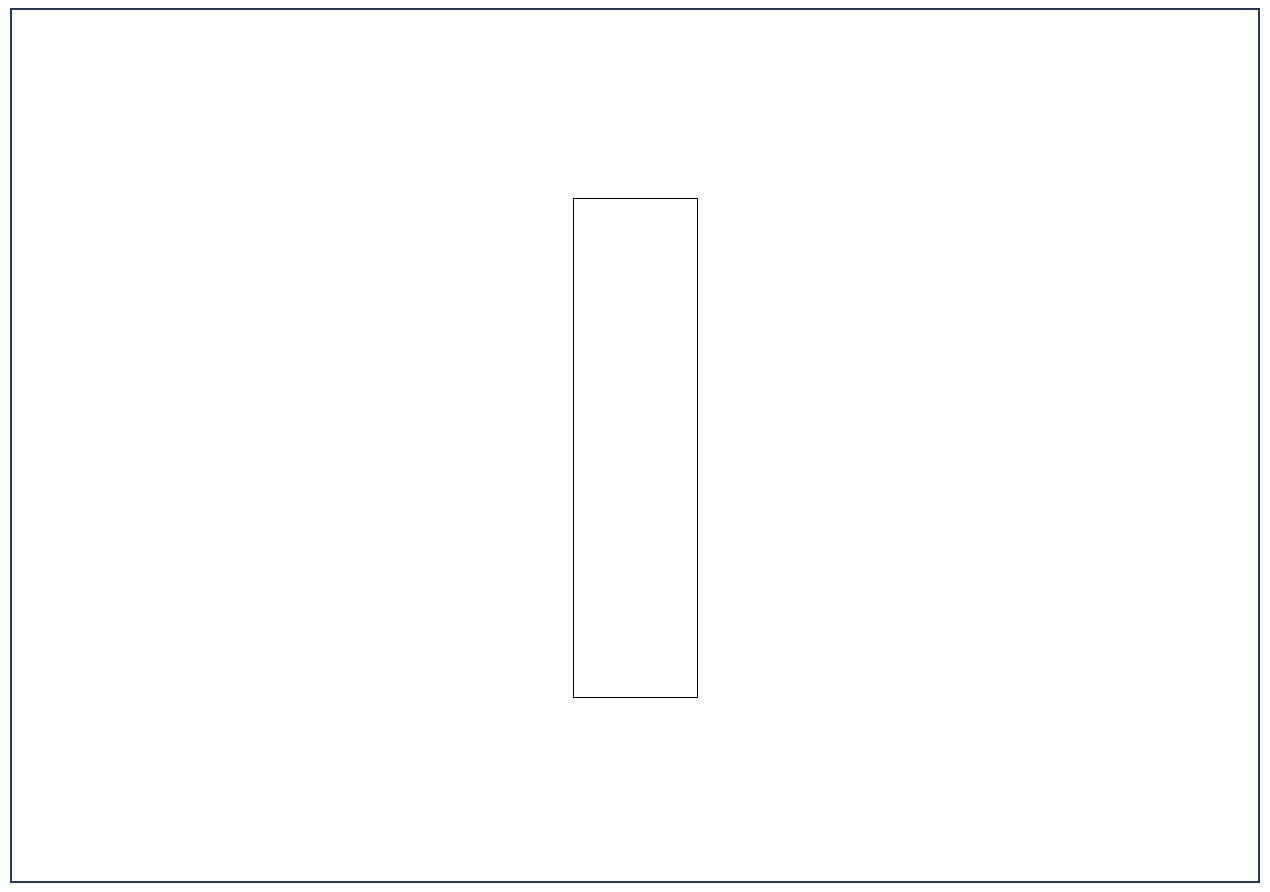
U trideset trećoj liniji dohvaćamo, ovaj put, klasu, samo što, pošto istu klasu može imati više elemenata onda se treba specificirati o kojem se elementu s tom klasom radi pomoću indexa(npr. index [0] vratiti će prvi element s tom klasom), no te ćemo indexe poslije dodati. Sljedeći dio koda je **for petlja** koja ponavlja isti blok koda onoliko puta koliko je specificirano(prima tri parametra, varijabla i njena vrijednost, dokle treba broj *ići* i za koliko će se svaki put povećati), u ovom slučaju se kroz petlju prolazi četiri puta(onoliko koliko ima „meta“). Zatim će se u petlji na onu klasu postavljati indexi i(svi – od nula do tri) kako bi se dohvatili svi crveni kvadrati i na njih se postavili EventListeneri pomoću kojih će se na klik povećati brojač(rezultat). Na četrdeset prvoj liniji na container(promašaj mete) stavljamo EventListener. Ovaj put anonimnoj funkciji prosljeđujemo parametar, zatim sa **if elsom/if statementom**(if - ako je uvjet ispunjen izvrši se taj blok koda, else if - ako je ovaj uvjet ispunjen izvrši se ovaj blok koda i ako nije prošli uvjet ispunjen, else – ako nije nikoji drugi uvjet ispunjen onda neka se izvrši ovaj blok koda) provjeravamo ako je element na kojeg je kliknuto(e.target) container(trebalo bi spomenuti kako == provjerava je li nešto jednako, a = postavlja novu vrijednost). Ako je smanji brojač(rezultat) za jedan bod.

Zadnji dio koda(za timer) je jako sličan timeru iz prošle provjere(brzine klikanja):

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

## 2.6 Test punjenja ljestvice



Cilj je ovu ljestvicu napuniti do kraja(od dolje prema gore) brzim kliktanjem tipke spacebar. Pri stiskanju tipke će se korisniku mjeriti vrijeme(rezultat).

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Postavlja se EventListener na ljestvicu u sredini(ovaj EventListener obuhvaća skoro cijeli program, ovaj put se koristi keyup – kad se otpusti neka tipka). Iako se za prošli test koristio e.target, ovaj put se koristi e.which koji provjerava koja je tipka na tipkovnici pritisnuta, te je uspoređuje sa trideset dva(ASCII vrijednost tipke spacebar). Svi ASCII kodovi:

Slika na kojoj se prikazuje stol

Opis je automatski generiran

Ako je stisnuta tipka spacebar onda će se removati ovaj EventListener(jer se koristio da se samo započne test) i stavlja se interval u kojem se varijabla i poveća za jedan svakih deset milisekundi(to će biti potrebno poslije za izračunavanje vremena).

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Pri testiranju, kako bi se stvorio izazov, kad god je ljestvica napunjena za više od trideset piksela taj dio ljestvice koji je napunjen počet će se smanjivati za trideset piksela svako dvjesto pedeset milisekundi.

Slika na kojoj se prikazuje tekst

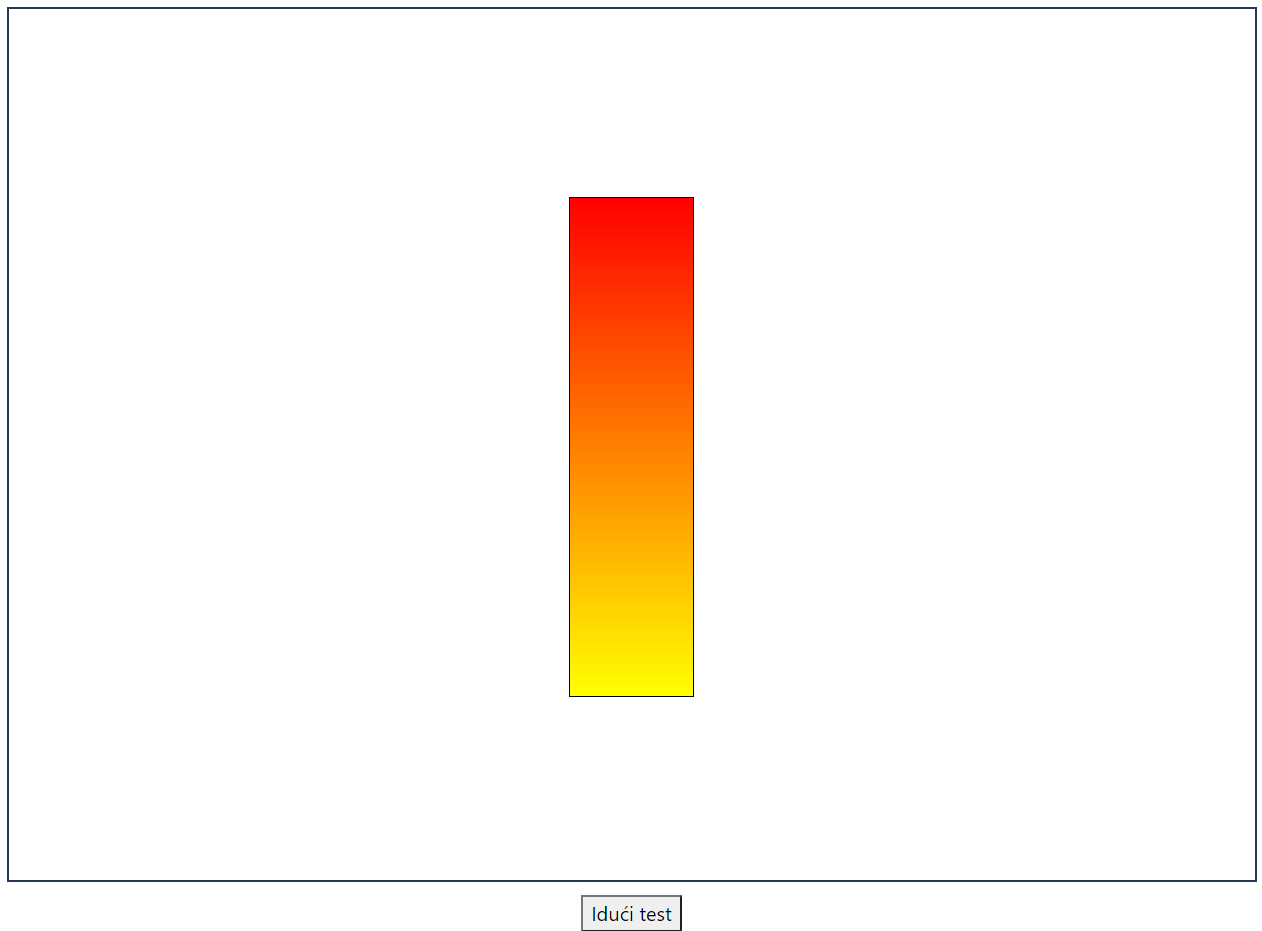
Opis je automatski generiran

Na ovaj način gore će se zapravo puniti ljestvica(za pedeset piksela svaki stisak tipke).

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

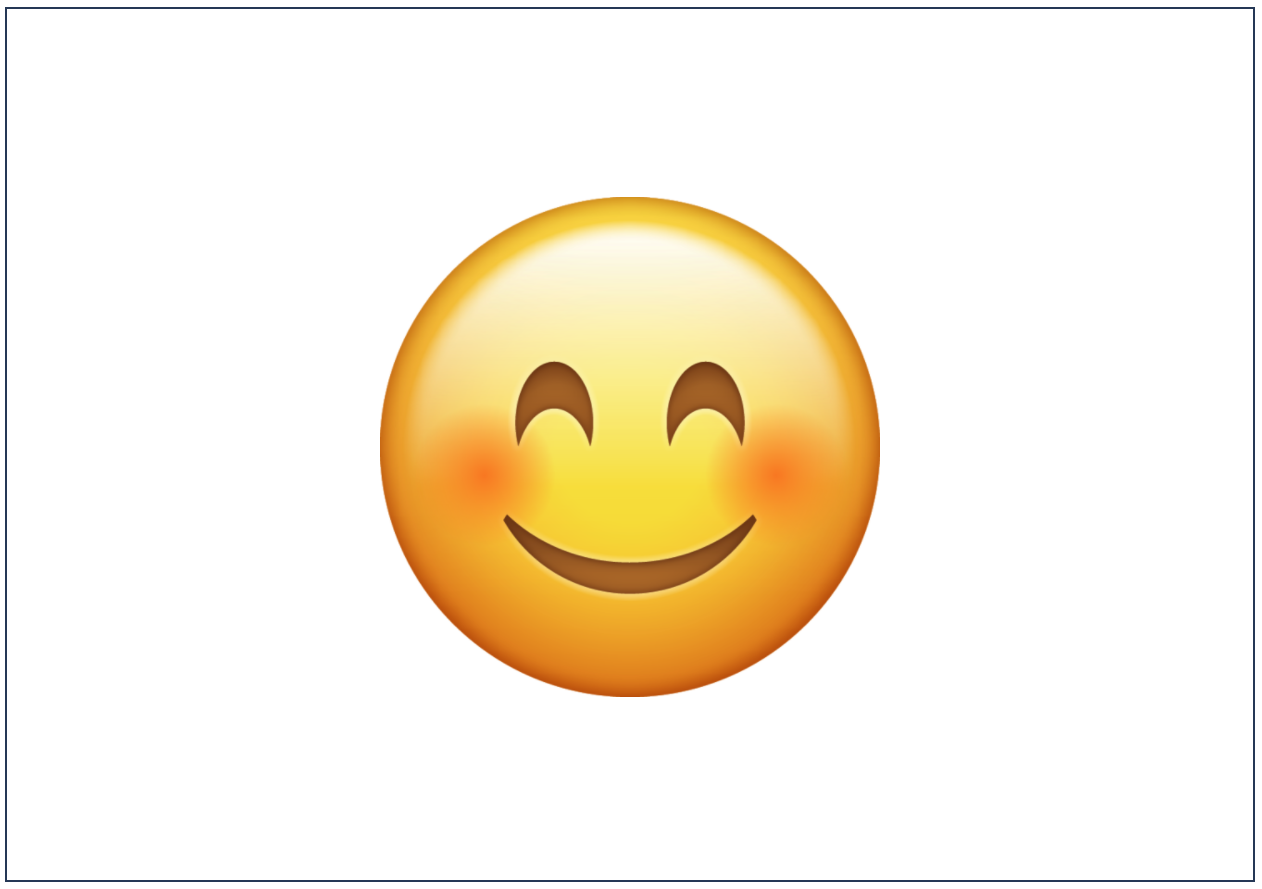
Ovaj zadnji dio provjerava ako je ljestvica puna više ili jednako četiristo piksela. Ako je onda se pravi varijabla vrijeme u kojoj se pravi decimalni zapis vremena ili rezultata(pomoću ugrađene funkcije toFixed) kako bismo ga prikazali. Zatim se gase svi intervali i EventListeneri.



Ovako to izgleda nakon što je test gotov i ljestvica je napunjena. Ovu boju punjenja smo postigli uz pomoć CSS-a:



## 2.7 Test sreće



Za ovaj test je na korisniku da klikne lijevu strelicu na tipkovnici ako misli da će idući *emoji* biti sretan, a desnu strelicu ako misli da će biti tužan. Cilj testa je imati što veći lucky streak(pogoditi što više puta zaredom).

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Na početku stavljamo nekoliko varijabli, no na drugoj liniji se kao vrijednost varijabli dodjeljuje **lista**(niz vrijednosti). Kao vrijednosti u nju postavljamo putanje za slike sretnog i tužnog emojia.



Ovdje se u varijabli nasumično *izabire* broj od nula do jedan(dužina liste). Zatim se na polje iz stranice kao putanja postavi jedna od one dvije iz liste na način da onaj nasumičan broj iz varijable se koristi kako index.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

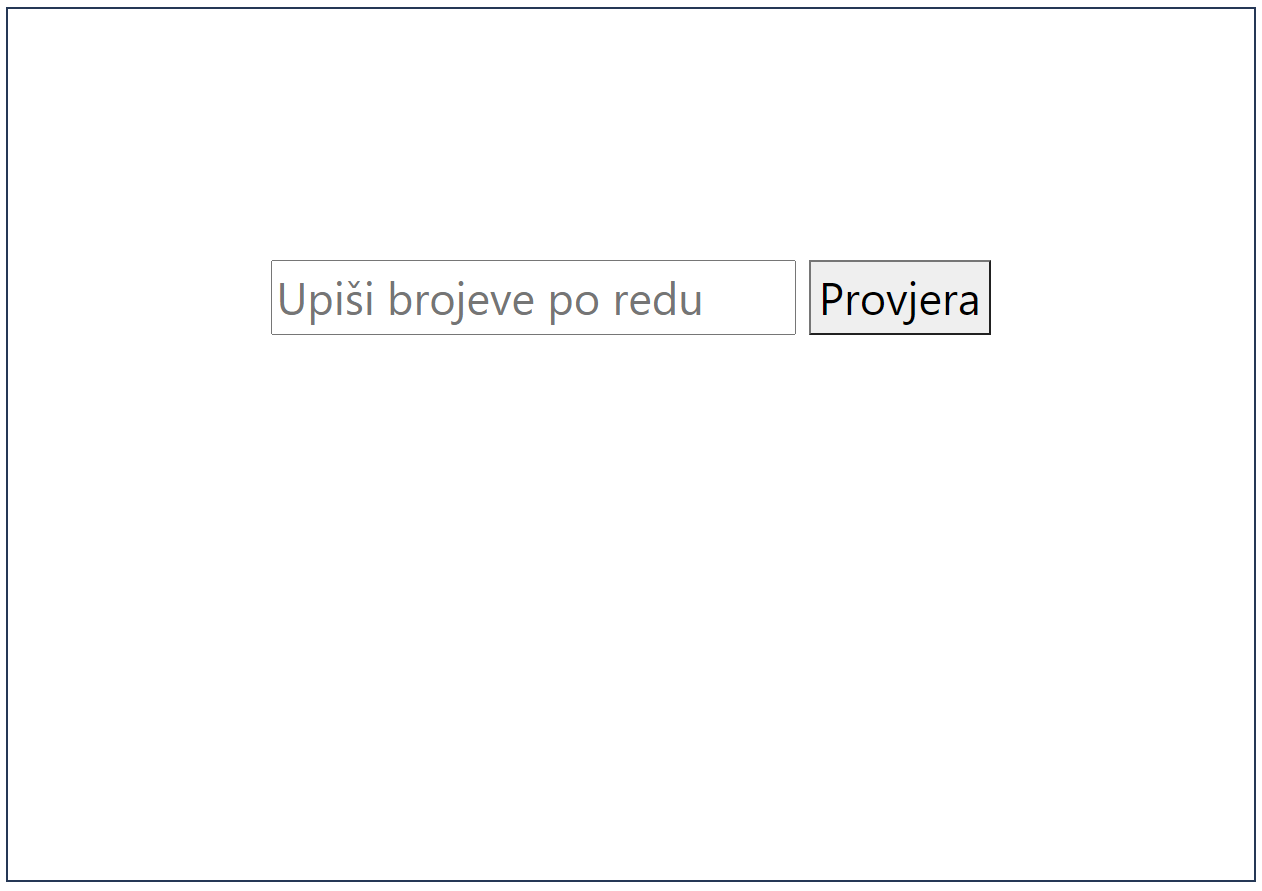
U ovom EventListeneru prvo se već određuje koja će biti slika u pitanju, zatim se postavlja pitanje. Ako je stisnuta lijeva strelica i sretan je emoji u pitanju onda se baš postavlja na stranicu ta slika i rezultat se povećava za jedan, Ako je stisnuta lijeva strelica i sretan emoji nije u pitanju onda se remova EventListener.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Else if je jako sličan, samo što se provjerava ako je stisnuta desna strelica kad je u pitanju tužan emoji(tad bodovi idu gore). Na samom kraju je else samo kako bi se provjerilo da korisnik ne tipka po krivim tipkama.

## 2.8 Test pogađanja niza brojeva



Prije ovog inputa pojavio se nasumični jednoznamenkasti broj. Od korisnika se traži da nakon što se broj pojavi i upiše taj broj. Težina testa je u tome što nakon što se taj jednoznamenkasti broj napiše idući broj koji će se pojaviti bit će nasumični dvoznamenkasti broj, onaj nakon toga troznamenkasti… Cilj provjere je provjeriti koliko dobro korisnik može memorizirati.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Većinom se cijeli program stavlja u funkciju kako bi se mogla poslije opet pozvati. Od četrnaeste do sedamnaeste linije ima **do while petlja**(slična je for petlji samo što se kod for petlje odmah odredi kad treba stati, a do while petlja prvo izvrši neki kod pa onda treba ispuniti neki uvjet kako bi se taj kod nastavio vrtjeti) u kojoj se postavlja i pamti nasumični broj koji korisnik treba upisati. Zatim se umjesto broja postavlja sami input, te onda provjerava je li korisnik točan broj upisao(petlja se ponavlja samo sa većim brojevima i povećan je rezultat) ili ako nije.

## 2.9 Test pogađanja velikog broja

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Ovaj test je sličan prošlom testu, ne samo po izgledu, nego i po cilju(testiranje memorije). Samo što ovaj test prikaže jedan devetoznamenkasti broj, ali znamenku po znamenku, te se onda traži od korisnika da što bolje moguće napiše taj broj u input(po redu kako su znamenke prikazane). Što više znamenki bude na odgovarajućem mjestu toliko će biti bolji rezultat.



Prvo se stvara prazna lista.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

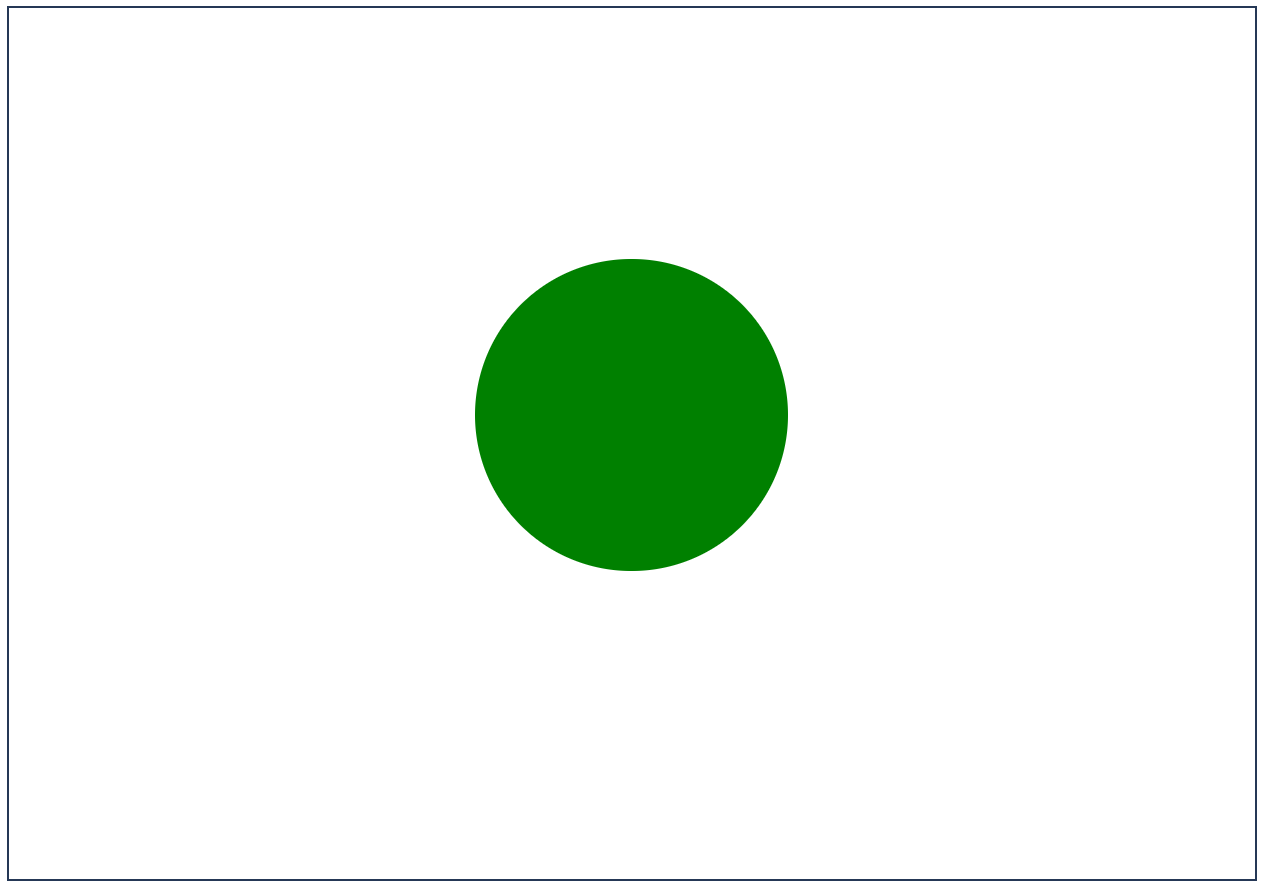
Zatim se u else uđe ako ima manje od devet vrijednosti u listi. Tu se *stvori* broj koji se brzo stavi i makne sa stranice, te se pusha(naredba za *guranje* neke vrijednosti u listu) u listu. Pošto je ovo interval ovaj bi se else trebao ponavljati sve dok ne dođe do desetog puta. Kad dođe do desetog puta, napokon će doći do ovog ifa:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Prvo se dodaje input s maxlengthom od devet kako bi korisnik mogao upisati maksimalno devet slova. Dolje u EventListeneru se splita value(vrijednost) od inputa kako bi se upisani broj uzeo i podijelio na devet jednoznamenkastih brojeva. Tad bi se stavio u listu gdje je svaki broj jedna vrijednost te liste. Onda se ti korisnikovi brojevi uspoređuju sa onim koji su prikazani bili na stranici(po znamenkama, npr. prvi se broj u listi brojeva korisnika uspoređuje s prvim brojem u listi brojeva koji su prikazani na stranici, drugi broj s drugim brojem…). Vrijedi spomenuti da smo za to koristili **for i in** **petlju**(kod se prestavje izvršavati kad se dođe do kraja određene liste).

## 2.10 Test Reakcije



Kad se pokrene test vidjeti će se crveno svijetlo. Cilj je da ga se klikne što brže kad postane zeleno kako bi se utvrdila brzina reakcije.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Kao prvotno stanje svijela se postavlja crvena. Zatim u setTimeoutu postavimo da nakon nekog nasumičnog vremena(do pet sekundi) se promjeni boja u zeleno, te se također počne štopati vrijeme. Dolje u EventListeneru će se zaustaviti „štoperica“ i provjeriti je li stisnuto svijetlo prije ili poslije nego što je pozelenilo. Ako je prije onda će se upozoriti korisnika, a ako je poslije onda se prikazuje rezultat. Nakon što su pomoću XHR-a(XMLHttpRequest-a) po zadnji put poslani podatci onda se klikće button koji korisnika ovaj put vodi na prikaz svojih rezultata.

## 2.11 Rezultati

Nakon što je na zadnjem testu pritisnut button „Rezultati“ korisnik je *odvezen* na ovu stranicu:

Slika na kojoj se prikazuje stol

Opis je automatski generiran

Ovi su rezultati povučeni iz ove tablice(linija devet):

Slika na kojoj se prikazuje tekst, snimka zaslona, na zatvorenom

Opis je automatski generiran

Taj prikaz na stranici se prikazao kroz ovaj kod:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Pokreće se, kao prvo, sesija kako bi se dohvatilo korisničko ime korisnika. Zatim se dohvaćaju rezultati svih testova tog korisnika i postavljaju se u tablicu(struktura tablice, oznaka table ju označava, oznaka tr označava redak tablice, a oznaka td označava stupac tablice). Klikom na button „Odjava“ se korisnik šalje na „odjava.php“.

## 2.12 Odjava

Kod za stranicu „odjava.php“:

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Ponovno se započinje sesija(, ali samo kako bi se uspjela *ugasiti*). Pomoću session\_unset se

*Prazne/oslobađaju* varijable. Pomoću session\_destroy *uništavaju* se sve varijable i sesija. Na ovoj se stranici skoro uopće ne ostaje jer služi samo za zaustavljanje sesije i njenih varijabli, pa se korisnik odmah linka pomoću headera na „index.html“.

# LITERATURA

<https://www.pymetrics.ai/>

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/>

<https://stackoverflow.com/>

<https://getbootstrap.com/>

<https://www.php.net/>

<https://developer.mozilla.org/en-US/>

# ZAVRŠETAK

Ovih sedam testova/provjera trebali bi dati uvid u stanje(psihičko i fizičko) korisnika koji trenutačno koristi ovaj način provjere.

Testovi se temelje na koordinaciji, brzini, refleksima i tako dalje.

Ovim testovima će poslodavac moći naći odgovarajuću poziciju zaposleniku.