Web sustav za automatsku provjeru i vrednovanje programskih rješenja

Kovačević, Dominik

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

WEB SUSTAV ZA AUTOMATSKU PROVJERU I VREDNOVANJE PROGRAMSKIH RJEŠENJA

Rijeka, rujan 2024.

Dominik Kovačević
0069093302

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

WEB SUSTAV ZA AUTOMATSKU PROVJERU I VREDNOVANJE PROGRAMSKIH RJEŠENJA

Mentor: izv. prof. dr. sc. Sandi Ljubić

Rijeka, rujan 2024.

Dominik Kovačević
0069093302

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 06.03.2024.

Zavod: Zavod za računarstvo

Predmet: Baze podataka

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Dominik Kovačević (0069093302)**

Studij: Sveučilišni prijediplomski studij računarstva (1035)

Zadatak: **Web sustav za automatsku provjeru i vrednovanje programskih rješenja / A Web System for Automatic Verification and Evaluation of Program Solutions**

Opis zadatka:

Istražiti mogućnosti postojećih (otvorenih) sustava za automatsku evaluaciju programskih rješenja. Odabrati i primijeniti postojeće rješenje u implementaciji izvornog sustava koji, pored definicije programskog problema, testnih slučajeva i vremenskog / memorijskog ograničenja, omogućava dodatne funkcionalnosti poput komentiranja zadataka / ispita, upravljanja ljestvicama poretka, uvođenje sustava virtualnog nagrađivanja, prikaz statističkih pokazatelja, upravljanje korisničkim profilima i slično. Posebnu pažnju valja usmjeriti na upotrebljivost klijentskog dijela aplikacije, s ciljem implementacije rješenja koje je responzivno, intuitivno i jednostavno za koristiti.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Predsjednik povjerenstva za
završni ispit:
prof. dr. sc. Miroslav Joler

Mentor:
izv. prof. dr. sc. Sandi Ljubić

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2024.

_____
Dominik Kovačević

Zahvala

Zahvaljujem RitehWebTeamu na podršci tijekom studiranja te znanju stečenom pohađanjem njihovih izvannastavnih aktivnosti, na kojima sam naučio dio vještina korištenih za izradu ovog rada

Sadržaj

Popis slika

Poglavlje 1

Uvod

U današnje vrijeme, s rastućom popularnošću programiranja i informatike, sve veći broj ljudi odlučuje se za učenje programskih jezika i rješavanje raznovrsnih programskih zadataka. Programiranje je postalo ključna vještina u mnogim industrijama, a sposobnost efikasnog i točnog pisanja programskog koda od iznimne je važnosti. Kako bi se osigurala kvaliteta i točnost programskih rješenja, potrebni su sustavi koji mogu automatski provjeriti i vrednovati radove korisnika. Ručno pregledavanje i ocjenjivanje programskih zadataka može biti vrlo dugotrajno i sklono pogreškama, stoga se javlja potreba za automatiziranim sustavima koji mogu pojednostaviti i ubrzati taj proces.

Ovaj rad bavi se dizajniranjem i implementacijom sustava za automatsku provjeru i vrednovanje programskih rješenja, s ciljem pojednostavljenja procesa evaluacije programskih zadataka. Takav sustav omogućava brz i pouzdan način provjere ispravnosti i kvalitete koda, analizirajući rješenja korisnika u realnom vremenu i identificirajući eventualne greške. Glavna prednost ovoga sustava leži u njegovoj sposobnosti da pruži trenutne povratne informacije korisnicima, što im omogućuje da odmah dobiju uvid u svoje pogreške i tako poboljšaju svoje vještine kroz iterativni proces učenja. Implementacija ovakvog web sustava donosi brojne prednosti kako za instruktore, tako i za studente. Instruktori mogu značajno uštedjeti vrijeme koje bi inače proveli na ručnom pregledavanju i ocjenjivanju zadataka, dok studenti imaju priliku brzo učiti iz svojih grešaka i usavršavati svoje vještine programiranja. Pored toga, sustav omogućava dosljednost u ocjenjivanju, što doprinosi pravednoj i objektivnoj evaluaciji studentskih radova.

U ovome radu bit će detaljno opisana arhitektura sustava, korištene tehnologije te procesi provjere i vrednovanja, s ciljem pružanja dubljega uvida u način na koji ovakav sustav može unaprijediti proces učenja programiranja. Na kraju, bit će razmotreni i potencijalni izazovi u razvoju i implementaciji ovakvih sustava te prijedlozi za njihovo prevladavanje, kako bi se osigurala maksimalna učinkovitost i pouzdanost.

Poglavlje 2

Postojeći sustavi za vrednovanje programskih rješenja

U sklopu završnog rada bilo je potrebno istražiti postojeća programska rješenja za vrednovanje programskih rješenja i utvrditi mogu li se implementirati u sklopu izvorno razvijenog sustava. Ključni kriterij za prihvaćanje programskog rješenja bio je da ono bude otvorenog koda, što znači da je moguć potpuni pristup i izmjena izvornog koda. Dodatno, to podrazumjeva da se programska podrška može koristiti bez naknade za pretplatu, a jedini je trošak poslužitelj na kojem će se program nalaziti. Mnoge platforme za automatsko vrednovanje koda odlučuju se koristiti već postojećim rješenjima zbog složenosti i obima posla koji je potreban za razvoj vlastitog sustava. Razvoj takvog sustava zahtijeva značajne resurse jer je potrebno riješiti niz tehničkih i sigurnosnih izazova.

Jedan od ključnih problema s kojim se sustavi za vrednovanje koda susreću jest rad s kodom koji se ne može prevesti ili koji sadrži sintaksne greške. Validacijski sustav mora biti dovoljno sofisticiran da prepozna različite vrste grešaka i pruži korisniku povratnu informaciju koja je korisna za daljnji rad na kodu. Ovo zahtijeva ne samo dobar parser za određeni programski jezik, nego i mogućnost prilagodbe različitim verzijama i dijalektima tog jezika.

Osim toga, sustav mora podržavati više programskih jezika, što dodatno komplicira njegov razvoj. Svaki jezik ima svoje specifičnosti, a sustav mora biti u mogućnosti precizno validirati kod napisan u bilo kojem od podržanih jezika. To uključuje pravilno rukovanje sa standardnim knjižnicama, različitim prevodiocima i interpre-

tatorima, kao i specifičnim alatima za testiranje. Implementacija ove raznolikosti zahtijeva duboko razumijevanje svakog jezika, što dodatno povećava složenost sustava.

Sigurnosni aspekt također je kritičan. Validacijski sustavi moraju biti zaštićeni od zlonamjernog koda koji može ugroziti stabilnost sustava ili kompromitirati podatke. Primjeri takvog koda uključuju "fork bombe", koje mogu iscrpiti resurse sustava, ili beskonačne petlje, koje mogu uzrokovati da sustav postane neodgovarajući. Razvoj zaštitnih mehanizama protiv ovakvih prijetnji zahtijeva napredne tehnike detekcije i izolacije procesa, kako bi se osiguralo da sustav ostane stabilan i funkcionalan čak i u prisutnosti zlonamjernih pokušaja.

2.1 Judge0

Judge0 [1] je platforma otvorenog koda koja je specijalizirana za izvršavanje i ocjenjivanje koda u širokom rasponu programskih jezika. Dizajniran je da bude vrlo svestran, podržava više od 40 različitih programskih jezika, uključujući one popularne kao što su Python, Java, C++ i JavaScript. Platforma je posebno korisna u scenarijima kao što su natjecanja u kodiranju, online platforme za kodiranje i obrazovna okruženja u kojima je bitno automatizirano testiranje koda. U svojoj srži, Judge0 nudi *RESTful* aplikacijsko programsko sučelje (engl. Application Programming Interface (API)) koji programerima i edukatorima omogućuje jednostavnu integraciju platforme u svoje aplikacije. Ovaj API olakšava podnošenje koda, koji Judge0 zatim prevodi i pokreće u sigurnom, izoliranom okruženju. Ovo izvođenje u zaštićenom okruženju osigurava da kod radi bez ikakvog rizika za glavni sustav ili da uzrokuje smetnje drugim korisnicima. Nakon izvršenja, Judge0 pruža detaljne povratne informacije, uključujući rezultate izvršenja, poruke o pogrešci i metriku upotrebe resursa poput potrošnje memorije i procesora.

Judge0 je dizajniran imajući na umu skalabilnost, odnosno sposobnost za rukovanje velikim brojem istodobnih predaja koda. To ga čini posebno prikladnim za aplikacije velikih razmjera, kao što su natjecanja u kodiranju ili obrazovne platforme s mnogo korisnika. Ukratko, Judge0 je moćan, fleksibilan i siguran alat koji pojednostavljuje proces pokretanja i programske validacije koda na više jezika, što ga čini vrijednim sredstvom za programere, edukatore i organizatore natjecanja.

2.2 Piston

Piston je programsko rješenje za izvršavanje koda visokih performansi koje je stvorila zajednica Engineer Man [2]. Omogućuje korisnicima sigurno pokretanje nepouzdanog ili potencijalno zlonamjernog koda u zaštićenom okruženju pomoću Dockera. Piston podržava veliki izbor programskih jezika i može se integrirati s više platformi, uključujući i automatizirano testiranje koda na platformi Discord što je jedan od čestih načina korištenja ovog specifičnog rješenja [3]. Projekt je otvorenog koda i dostupan je pod MIT-ovom licencom, što ga čini pristupačnim za osobnu ili komercijalnu upotrebu. Pistonov javni API može izvršavati kod i upravljati vremenom izvođenja, no ima ograničenja brzine radi sprječavanja zlouporabe. No, zbog limitiranih funkcija za mjerenje performance koda, iako je jeftiniji izbor od Judge0 platforme, u ovom radu nije bio korišten.

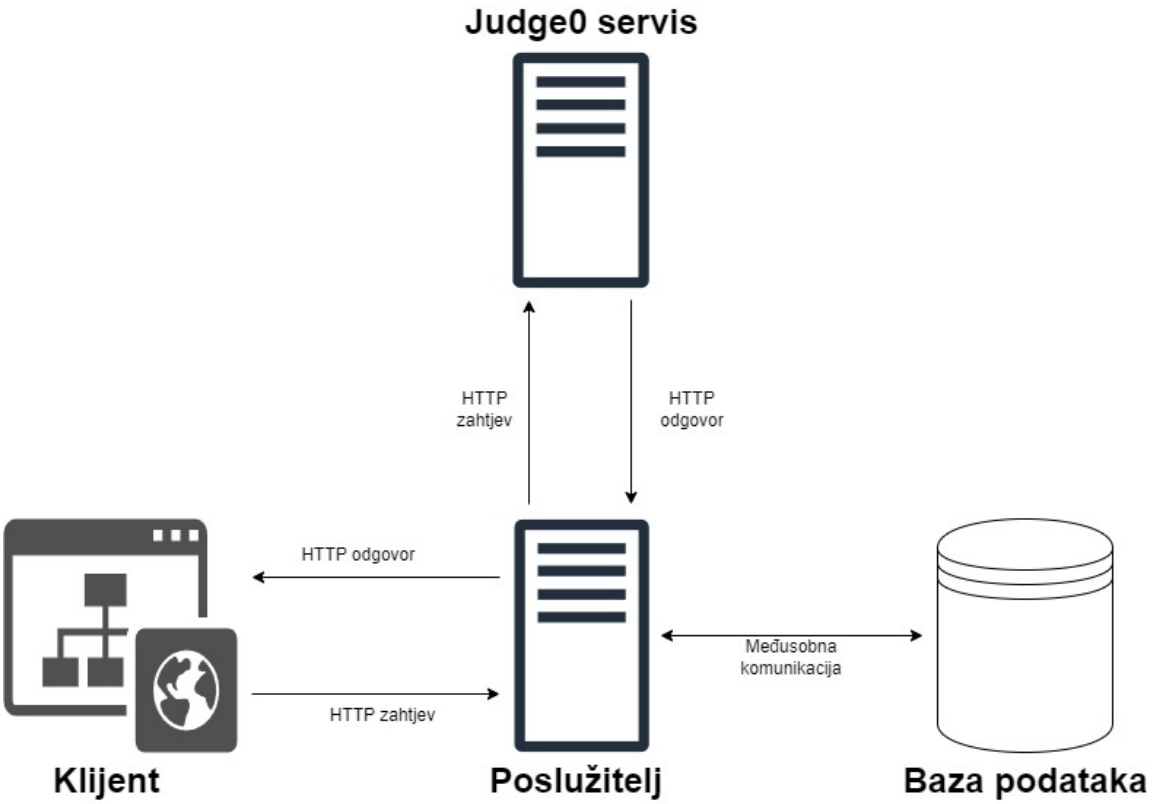
Poglavlje 3

Razvoj web sustava za automatsku provjeru i vrednovanje programskih rješenja

U ovom dijelu rada detaljno će se opisati tehnologije koje su primijenjene u razvoju klijentskog i poslužiteljskog dijela aplikacije. Također, pružit će se uvid u proces odabira odgovarajuće baze podataka, odnosno objasnit će se razlozi za izbor korištenih tehnologija te njihov utjecaj na cjelokupno programsko rješenje.

3.1 Arhitektura sustava

Na Slici 3.1 nalazi se prikaz arhitekture sustava koji se sastoji od klijenta, poslužitelja, baze podataka i Judge0 2.1 servisa/poslužitelja. Klijent vrši komunikaciju s poslužiteljem slanjem Hypertext Transfer Protocol (HTTP) zahtjeva koji u sebi sadrže podatke formatirane u JavaScript objektnoj notaciji (engl. JavaScript Object Notation (JSON)) te pomoću istih prima odgovore od poslužitelja. Osim komunikacije s klijentom, poslužitelj se brine i za spremanje podataka u samu bazu, koristeći se strukturiranim jezikom za upite (engl. Structured Query Language (SQL)) kako bi spremio podatke kao što su informacije o korisniku, zadacima, događajima i rješenjima zadataka. Za samu provjeru rješenja zadataka poslužitelj se koristi već napravljenim servisom za vrednovanje programskih rješenja Judge0. U tom je slučaju on klijent koji s pomoću HTTP zahtjeva komunicira s Judge0 servisom, koji je u ovom kontekstu poslužitelj i obrađuje zahtjev te šalje podatke natrag klijentu (našem poslužitelju) na obradu i pohranu.



Slika 3.1 Prikaz arhitekture sustava

3.2 Poslužiteljska strana sustava

Poslužiteljska strana sustava odgovorna je za upravljanje logikom na strani poslužitelja, interakciju s bazom podataka i cjelokupnu funkcionalnost koja je skrivena od korisnika. Ona obrađuje zahtjeve iz sučelja, obrađuje podatke, izvodi izračune i komunicira s bazom podataka radi pohranjivanja ili dohvaćanja informacija. Ukratko, poslužitelj se brine kako bi svi zahtjevi bili obrađeni glatko i sigurno. Za izradu poslužitelja u ovom radu izabran je Spring Boot okvir koji omogućuje komunikaciju s PostgreSQL bazom podataka.

3.2.1 Spring Boot

Spring Boot [4] je popularan okvir koji se koristi za izgradnju Java aplikacija, posebno kada se otvaraju pozadinski sustavi. Pojednostavljuje proces razvoja web aplikacija pružanjem mnogo već unaprijed izgrađenih konfiguracija, tako da programeri ne moraju početi od nule. To olakšava i ubrzava postavljanje novih projekata. Također, Spring Boot dolazi s nizom alata i značajki koje su bitne za razvoj pozadine, poput podrške za stvaranje *RESTful* web usluga, povezivanja s bazama podataka i izgradnje mikro servisa. Jedna od najvećih prednosti korištenja Spring Boota je ta što smanjuje količinu koda koji treba napisati, a to ubrzava razvojni proces. Osim toga, budući da se Spring Boot široko koristi, postoji velika zajednica programera koji doprinose njegovom stalnom poboljšanju, osiguravajući da ostane ažuriran s najnovijim tehnologijama. Ova kombinacija jednostavne upotrebe, snažnih značajki i snažne podrške zajednice čini Spring Boot odličnim izborom za izgradnju pouzdanih i skalabilnih pozadinskih sustava [5].

3.2.2 Lombok

U ovom se radu u velikoj mjeri koristila knjižnica Lombok [6] koja je Java alat koji pojednostavljuje kodiranje smanjenjem standardnog koda, koji se ponavlja i često je neophodan, ali je zamoran za pisanje. Primjerice, u Javi se često moraju ručno pisati *getter* i *setter* metode, konstruktori, metode *toString()*, *equals()* i *hashCode()* te mnoge druge. Lombok pomaže to izbjeći automatskim generiranjem ovih metoda tijekom prevođenja pomoću jednostavnih komentara. Kada se koristi Lombok, mogu se označiti razredi i polja s određenim komentarima poput *@Getter*, *@Setter*, *@AllArgsConstructor* i *@ToString*. Ove napomene upućuju Lombok da generira odgovarajuće metode ili konstruktore, čineći kod čišćim i lakšim za održavanje.

3.2.3 Autentifikacija korisnika pomoću JWT-a

U modernim web-aplikacijama zaštita korisničkih podataka i osiguranje autentificiranog pristupa resursima predstavljaju ključne aspekte koji direktno utječu na sigurnost i povjerenje korisnika. S obzirom na sve veću kompleksnost web-aplikacija i rastući broj prijetnji u digitalnom svijetu, postalo je neophodno koristiti robusne

metode za zaštitu osjetljivih podataka i osiguranje, tako da samo ovlašteni korisnici mogu pristupiti određenim resursima. Jedno je od široko prihvaćenih rješenja za upravljanje autentifikacijom i autorizacijom korištenje JSON web tokena (JSON Web Token (JWT)). JWT je otvoreni standard definiran u RFC 7519 [7] koji pruža kompaktnu, samostalnu i sigurnu metodu za prijenos informacija između različitih strana u web-aplikacijama. Ovaj standard koristi JSON format, koji je lagan i jednostavan za korištenje, što omogućava efikasan prijenos podataka. Jedna je od ključnih prednosti JWT-a njegova sposobnost da osigura integritet i autentičnost podataka koje prenosi. Unutar JWT-a informacije su digitalno potpisane, što omogućava primatelju da provjeri autentičnost podataka i osigura da nisu izmijenjeni nakon što su generirani. Ova je karakteristika izuzetno važna u scenarijima gdje je povjerenje između strana od kritične važnosti. JWT-ovi se često koriste u scenarijima autentifikacije, posebno u modernim aplikacijama koje se oslanjaju na *RESTful* arhitekturu ili slične pristupe. U ovim slučajevima poslužitelj generira JWT nakon što korisnik uspješno prođe proces prijave. Ovaj token se potom šalje klijentu, koji ga uključuje u sve naredne zahtjeve prema poslužitelju. Na taj način poslužitelj može provjeriti identitet korisnika i odobriti pristup različitim resursima, a da pritom ne mora pohranjivati informacije o sjednici (engl. session na samom poslužitelju. Ova metoda eliminira potrebu za tradicionalnim sjednicama koje se čuvaju na poslužitelju, smanjujući opterećenje na resurse poslužitelja i omogućavajući bolju skalabilnost aplikacija. Osim toga, JWT-ovi su posebno korisni u distribuiranim sustavima i mikroservisnim arhitekturama, gdje više servisa može provjeravati autentičnost zahtjeva koristeći isti token, bez potrebe za centraliziranom kontrolom sjednica.

3.2.4 Pokretanje poslužitelja

Za pokretanje poslužitelja na računalu potrebno je instalirati Java razvojni alat (engl. Java Development Kit (JDK)), koji omogućava izvršavanje Java koda te Maven, alat za automatsku izgradnju koda (engl. Build Automation tool). Maven pomaže u instalaciji vanjskih knjižnica koje su korištene u poslužiteljskom kodu. Osim toga, dodatni alat koji može biti koristan je IDE (engl. Integrated Development Environment). IDE-ovi olakšavaju pisanje koda i često dolaze s unaprijed instaliranim JDK-ovima i alatima poput Mavena. Nakon što su instalirani svi nužni alati za pokretanje koda, potrebno je izmijeniti neke linije u datoteci application.properties unutar

poslužiteljskog koda ili dodati odgovarajuće okolišne varijable (engl. Environment variables) na računalu kako bi poslužitelj imao ispravnu konfiguraciju. Varijable povezane s bazom podataka uključuju Uniform Resource Locator (URL) PostgreSQL baze, kao i korisničko ime i lozinku baze podataka. Također, potrebno je nasumično generirati dvije linije teksta, svaku dugu minimalno 32 znaka, koje će se koristiti za generaciju JWT tokena za autentifikaciju korisnika. Za klijentsku stranu potrebno je poslužitelju dati URL kako bi omogućili djeljenje resursa s više izvora (engl. Cross-origin resource sharing (CORS)), koji služi za zaštitu poslužitelja od neovlaštenih zahtjeva s drugih domena. Na kraju, potrebni su URL i API ključ (engl. API key) za instancu Judge0, koja će biti korištena za validaciju koda.

Ovaj pristup osigurava ispravnu konfiguraciju poslužitelja, omogućujući nesmetano izvršavanje koda i pravilnu autentifikaciju korisnika. Pravilno postavljanje okolišnih varijabli i ostalih postavki ključno je za osiguravanje sigurnog i učinkovitog rada aplikacije.

3.3 Klijentska strana sustava

Najvažnija je komponenta klijentskog dijela sustava React [8] [9], vrlo poznata JavaScript biblioteka koju je razvio Facebook. React je jako popularan u svijetu razvoja korisničkih sučelja zbog svoje sposobnosti da izgradi dinamična i brza korisnička sučelja, posebno u aplikacijama koje imaju samo jednu stranicu. Reactova arhitektura, koja se temelji na komponentama, omogućuje programerima da složena korisnička sučelja razbiju na manje dijelove, odnosno komponente, koje se mogu ponovno koristiti. Ovo ne samo da olakšava održavanje koda nego i omogućuje lakše skaliranje aplikacija. Korištenje virtualnog DOM-a u Reactu znači da se ažuriraju samo oni dijelovi korisničkog sučelja koji su se promijenili, a ne cijela stranica, što poboljšava brzinu i efikasnost aplikacije. Stiliziranje korisničkog sučelja, koje je bitan dio razvoja, značajno je olakšano pomoću TailwindCSS-a [10]. Za razliku od klasičnih Cascading Style Sheets (CSS) okvira TailwindCSS ne nudi gotove komponente kao što su gumbi ili kartice, već daje niz korisnih klasa koje omogućuju programerima da sami stvaraju svoje dizajne izravno unutar HyperText Markup Language (HTML) i JavaScript XML (JSX) datoteka. Ovaj pristup omogućuje puno veću fleksibilnost i prilagodbu dizajna, što je važno kako bi se izbjegla ograničenja koja dolaze s una-

prijed definiranim stilovima. TailwindCSS sa svojim pristupom, u kojem su klase korisnih programa na prvom mjestu, potiče drukčiji način razmišljanja o stiliziranju. Tu se stilovi primjenjuju direktno u kodu, pa to rezultira boljom održivošću i skalabilnošću koda. Kada se React koristi s TailwindCSS-om, dobiva se vrlo dobar alat za razvoj korisničkih sučelja koja su brza, responzivna i estetski privlačna. React se brine za logiku i strukturu aplikacije, dok TailwindCSS osigurava da dizajn bude fleksibilan i jednostavan za održavanje.

3.4 Baza podataka

Relacijske baze podataka vrsta su sustava za upravljanje bazama podataka (DBMS) koje pohranjuju podatke u tablicama, koje su strukturirane kao redci i stupci. Svaka tablica predstavlja entitet, a svaki redak predstavlja zapis unutar tog entiteta, dok stupci predstavljaju atribute tog entiteta. Relacijski model, koji je predložio E.F. Codd 1970. godine, uveo je koncept organiziranja podataka u povezane tablice, koje se mogu pretraživati s pomoću strukturiranog upitnog jezika (SQL). Primarna prednost relacijskih baza podataka je njihova sposobnost učinkovitog rukovanja velikim količinama strukturiranih podataka, uz osiguranje integriteta podataka putem ograničenja i odnosa (kao što su primarni i strani ključevi). Relacijske baze podataka podržavaju razne operacije poput CRUD (Stvaranje, Čitanje, Ažuriranje, Brisanje) te omogućuju složene upite koji uključuju spajanja, agregacije i transakcije. Ove su baze podataka vrlo konzistentne i pouzdane, što ih čini pogodnima za aplikacije u kojima su ključne točnost i konzistentnost podataka. Popularni sustavi za upravljanje relacijskim bazama podataka (RDMS) uključuju MySQL, Oracle, Microsoft SQL Server i PostgreSQL.

3.4.1 PostgreSQL

PostgreSQL [11] je napredni, open-source sustav za upravljanje relacijskim bazama podataka poznat po svom snažnom naglasku na proširivost i usklađenost sa standardima. Izvorno razvijen 1986. godine kao dio POSTGRES projekta na Sveučilištu Kalifornije u Berkeleyju [12], PostgreSQL se razvio u snažan i svestran sustav baza podataka koji podržava upite u SQL i JSON formatima. PostgreSQL je usklađen s ACID-om, što znači da jamči pouzdanu obradu transakcija i visoki integritet po-

dataka. To je ključno za sustave koji zahtijevaju točne, konzistentne i pouzdane operacije nad bazama podataka, osiguravajući da se transakcije izvršavaju u cijelosti ili se u slučaju greške vraćaju u prethodno stanje, čime se sprječava gubitak ili korupcija podataka.
PostgreSQL se razlikuje od drugih RDMS projekata kroz nekoliko jedinstvenih značajki:

- **proširivost:** PostgreSQL omogućuje korisnicima definiranje prilagođenih tipova podataka, operatora i tipova indeksa. Ova značajka čini ga prilagodljivim za širok raspon primjena, od tradicionalnih relacijskih slučajeva uporabe do složenijih aplikacija, poput geografskih informacijskih sustava
- **napredne tehnike indeksiranja:** PostgreSQL podržava nekoliko tehnika indeksiranja, uključujući B-stablo, *hash, Generalized Search Tree (GiST), space-partitioned GiST (sp-GiST)* i *Generalized Inverted Index (GIN)*, koje optimiziraju različite vrste upita i poboljšavaju performanse
- **pretraživanje punog teksta:** PostgreSQL nudi moćne mogućnosti pretraživanja punog teksta, koje su bitne za aplikacije koje zahtijevaju složene funkcionalnosti pretraživanja
- **replikacija i skalabilnost:** PostgreSQL podržava različite mehanizme replikacije, uključujući *streaming* replikaciju i logičku replikaciju, što je ključno za skaliranje baza podataka na više poslužitelja radi visoke dostupnosti i balansiranja opterećenja.

3.4.2 Korištenje PostgreSQL-a u Spring Boot-u

Spring Boot je okvir koji pojednostavljuje razvoj aplikacija temeljenih na Javi, posebno onih koje koriste Spring okvir. Pruža sveobuhvatnu platformu koja uključuje razne komponente poput upravljanja ovisnostima, konfiguracije i ugrađenih poslužitelja, omogućujući programerima da se usredotoče na izgradnju poslovne logike umjesto na repetitivni kod. Kada se integrira s PostgreSQL-om, Spring Boot aplikacije mogu iskoristiti snažne značajke PostgreSQL-a, dok istovremeno koriste prednosti povećane produktivnosti koju pruža Spring Boot.
Ključni aspekti korištenja PostgreSQL-a sa Spring Bootom uključuju:

- **sloj pristupa podacima:** Spring Boot se besprijekorno integrira s PostgreSQL-om putem Spring Data Java Persistence API (JPA). Programeri mogu koristiti JPA za interakciju s PostgreSQL bazama podataka pomoću Object Relational Mappera (ORM-a), koji skriva velik dio repetitivnog koda potrebnog za interakciju s bazama podataka. Ova integracija omogućuje učinkovito generiranje upita, upravljanje transakcijama i upravljanje shemom
- **konfiguracija:** Spring Boot pruža jednostavne i fleksibilne opcije konfiguracije za povezivanje s PostgreSQL bazom podataka. Konfiguracija se može upravljati putem *application.properties* ili *application.yml* datoteka, gdje programeri mogu specificirati svojstva veze s bazom podataka kao što su URL, korisničko ime, lozinka i naziv klase upravljačkog programa (engl. driver) za bazu podataka
- **upravljanje transakcijama:** s pomoću Spring Boota programeri mogu jednostavno upravljati transakcijama unutar svojih aplikacija, osiguravajući integritet i dosljednost podataka kroz operacije. Podrška PostgreSQL-a za ACID transakcije dobro se usklađuje sa Springovim značajkama za upravljanje transakcijama, pružajući robusno okruženje za poslovne aplikacije
- **optimizacija performansi:** napredne značajke PostgreSQL-a, kao što su indeksiranje i optimizacija upita, mogu se učinkovito koristiti u Spring Boot aplikacijama za poboljšanje performansi. Dodatno, mehanizmi predmemoriranja u Spring Bootu mogu raditi u paru sa sposobnostima PostgreSQL-a kako bi se dodatno poboljšala učinkovitost aplikacije.

3.5 Korištenje spremnika

Docker se pojavio kao ključan alat u modernom razvoju programskog rješenja, prvenstveno zahvaljujući svojoj sposobnosti da pojednostavi proces implementacije kroz "kontejnerizaciju". "Kontejnerizacija" omogućuje da se aplikacije, sa svim njihovim ovisnostima, zapakiraju u jedinstvene, prijenosne spremnike koji osiguravaju dosljedan rad programskog rješenja u različitim okruženjima [13], bez obzira na razlike u temeljnoj infrastrukturi. Na taj način Docker osigurava uniformnost rada aplikacija kroz različite faze razvoja i implementacije.

3.5.1 Docker

U središtu je Dockera [14] koncept spremnika (engl. containers). Spremnici su lagani, samostalni i izvršni programski paketi koji uključuju sve što je potrebno za pokretanje dijela programske podrške, uključujući programski kod, vrijeme izvođenja, biblioteke i varijable okruženja. Za razliku od virtualnih strojeva (engl. Virtual machines), spremnici dijele jezgru glavnog sustava i stoga su učinkovitiji u smislu korištenja resursa. Ova učinkovitost omogućuje postavljanje više spremnika na jednom glavnom računalu bez značajnih troškova, što je ključno za skaliranje aplikacija. Docker spremnici se instanciraju iz Docker slika (eng. Docker images). Slika je predložak samo za čitanje koji uključuje skup uputa za stvaranje spremnika. Slike se obično izrađuju iz Dockerfilea, skripte koja sadrži niz naredbi za sastavljanje slike. Sposobnost izgradnje i postavljanja slika osigurava dosljednost u različitim okruženjima, što je jedna od Dockerovih primarnih prednosti. Dockerfile je skripta sastavljena od niza naredbi koje se izvršavaju za izgradnju Docker slike. Ove naredbe određuju kako slika treba biti konstruirana, uključujući osnovnu sliku, kod aplikacije, postavke okruženja i sve druge ovisnosti. Dockerfile omogućuje automatizaciju procesa stvaranja slike, što olakšava održavanje i ažuriranje aplikacija.

Docker Hub je skladište temeljeno na oblaku gdje se Docker slike mogu pohraniti, dijeliti te im se može pristupiti. Služi kao središnji registar za Docker slike, omogućujući programerima da povuku službene slike i slike koje je donijela zajednica za korištenje u vlastitim projektima. Ova centralizacija slika olakšava suradnju i ponovnu upotrebu, dodatno pojednostavljujući proces implementacije. Iako sam Docker pruža osnovne alate za kontejnerizaciju, postavljanje spremnika na razini često zahtijeva alate za orkestraciju kao što su Kubernetes ili Docker Swarm [15]. Ovi alati upravljaju implementacijom, skaliranjem i radom više spremnika, osiguravajući da aplikacije ostanu otporne i učinkovite čak i pod različitim uvjetima opterećenja.

3.5.2 Docker na klijentskoj i poslužiteljskoj strani sustava

Docker se pokazao izuzetno korisnim alatom za pripremu i implementaciju aplikacija, kako na klijentskoj, tako i na poslužiteljskoj strani sustava. U ovom je kontekstu Docker korišten za pakiranje React aplikacije na klijentskoj strani te Spring Boot aplikacije na poslužiteljskoj strani u spremnike.

Na klijentskoj strani React aplikacija pripremljena je za Docker korištenjem Dockerfilea, koji omogućuje stvaranje Docker slike. Ova slika uključuje sve potrebne komponente, poput Node.js okruženja, instalacije ovisnosti putem npm-a te prevođenje React koda u statičke datoteke. Nakon stvaranja slike aplikacija se može pokrenuti unutar Docker spremnika, osiguravajući konzistentnost u radu bez obzira na lokalnu konfiguraciju razvojnih okruženja.

Na poslužiteljskoj strani Spring Boot aplikacija također je dockerizirana putem Dockerfilea. U ovom slučaju Dockerfile definira osnovnu JDK sliku kao temelj, a zatim kopira prevedenu Java Arhiva (engl. Java ARchive (JAR)) datoteku aplikacije u spremnik. Definiranjem startne točke za pokretanje JAR datoteke, Docker omogućuje lako postavljanje Spring Boot aplikacije u produkcijsko okruženje. Ova metoda omogućuje brzu i jednostavnu implementaciju, pri čemu se osigurava da aplikacija radi dosljedno na svim poslužiteljima. Važno je uzeti u obzir korake iz sekcije 3.2.4 koji opisuju pripremanje application.properties dokumenta za ispravan rad aplikacije čak i kad se koristi Docker. Docker se također koristi za pokretanje Judge0 instance koja se koristi za evaluaciju koda na našem poslužitelju.

Poglavlje 4

Karakteristični slučajevi korištenja

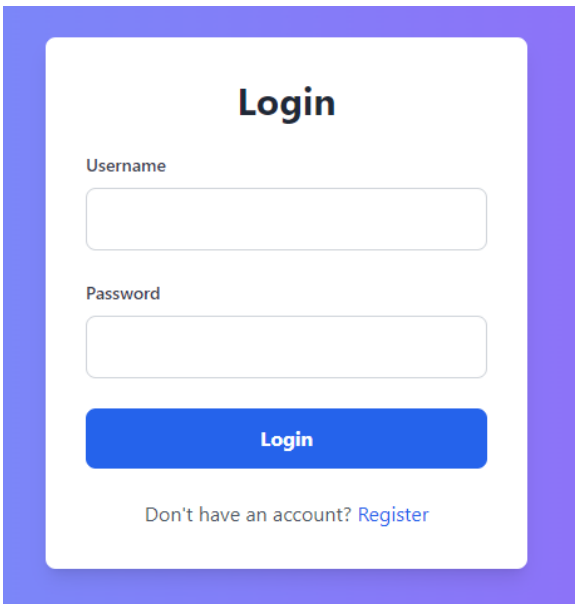
Ovo poglavlje bavi se detaljnim pregledom svih slučajeva korištenja kompletnog rješenja. Obradit će se postupak registracije i prijave korisnika, upravljanje izazovima i događajima, pregled svih rješenja zadataka te proces samog rješavanja zadataka. Također će se obraditi komentiranje izazova, uređivanje profila korisnika, kao i prikaz statistika na platformi.

4.1 Osnovni slučajevi korištenja

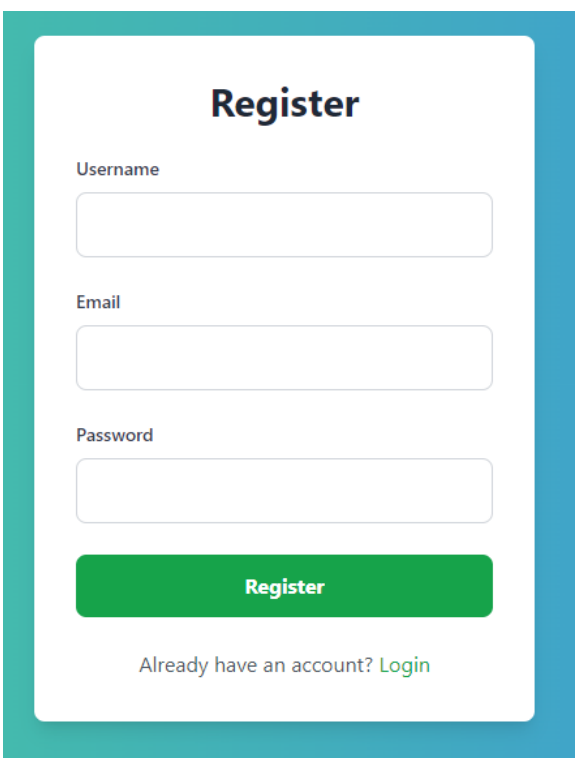
Neke su od komponenti i stranica na platformi iste za korisnike i administratore, kao što su registracija i prijava korisnika u sustav te nadzorna ploča koja se pojavljuje na vrhu svake stranice.

4.1.1 Registracija i prijava korisnika

Prilikom prvog posjeta stranici, korisnik će biti automatski preusmjeren na stranicu za prijavu. Ako već ima postojeći račun, treba unijeti korisničko ime i lozinku kako bi dobio pristup platformi. Slika 4.1 prikazuje funkcionalnost stranice za prijavu. U slučaju da je korisnik unio pogrešno korisničko ime ili lozinku, dobit će obavijest o vrsti pogreške. Ako još nema račun, može odabrati opciju registracije. Za stvaranje novog računa potrebno je unijeti korisničko ime koje nije zauzeto, valjanu adresu elektroničke pošte i lozinku. Slika 4.2 prikazuje obrazac za registraciju.



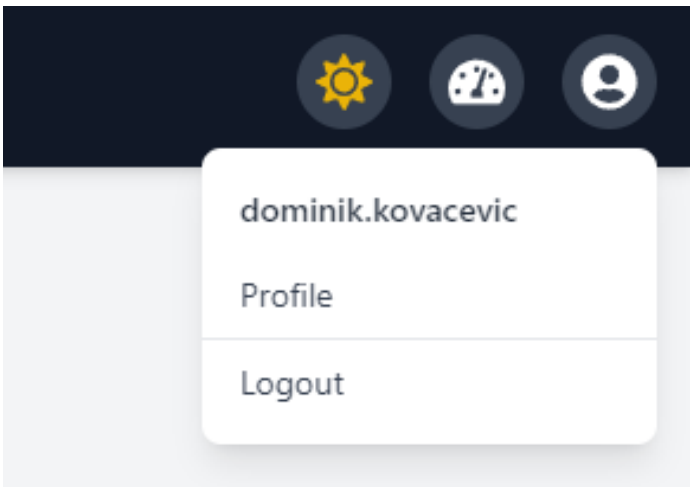
Slika 4.1 Obrazac za prijavu



Slika 4.2 Obrazac za registraciju

4.1.2 Navigacijska traka

Navigacijska traka (engl. Navbar) nalazi se na vrhu svake stranice na platformi. Na lijevoj strani te trake nalazi se gumb za povratak na prethodnu stranicu, a uz njega je prikazan naziv same platforme, koji se može promijeniti u konfiguraciji projekta. U slučaju ove demo verzije naziv je "Kiwi Coding Platform". Pored naziva platforme nalazi se i naziv trenutne stranice, što korisniku omogućuje lakšu navigaciju kroz platformu. Na desnoj strani kontrolne trake nalaze se redom sljedeći gumbi: gumb za promjenu između tamnog i svijetlog načina rada, gumb koji korisnika uvijek vraća na kontrolnu ploču, bez obzira na to je li ona namijenjena administratoru ili korisniku, te gumb koji otvara padajući izbornik. U padajućem izborniku nalaze se ime trenutnog korisnika, gumb koji vodi na profil, kao i gumb za odjavu s platforme. Slika 4.3 prikazuje padajući izbornik.

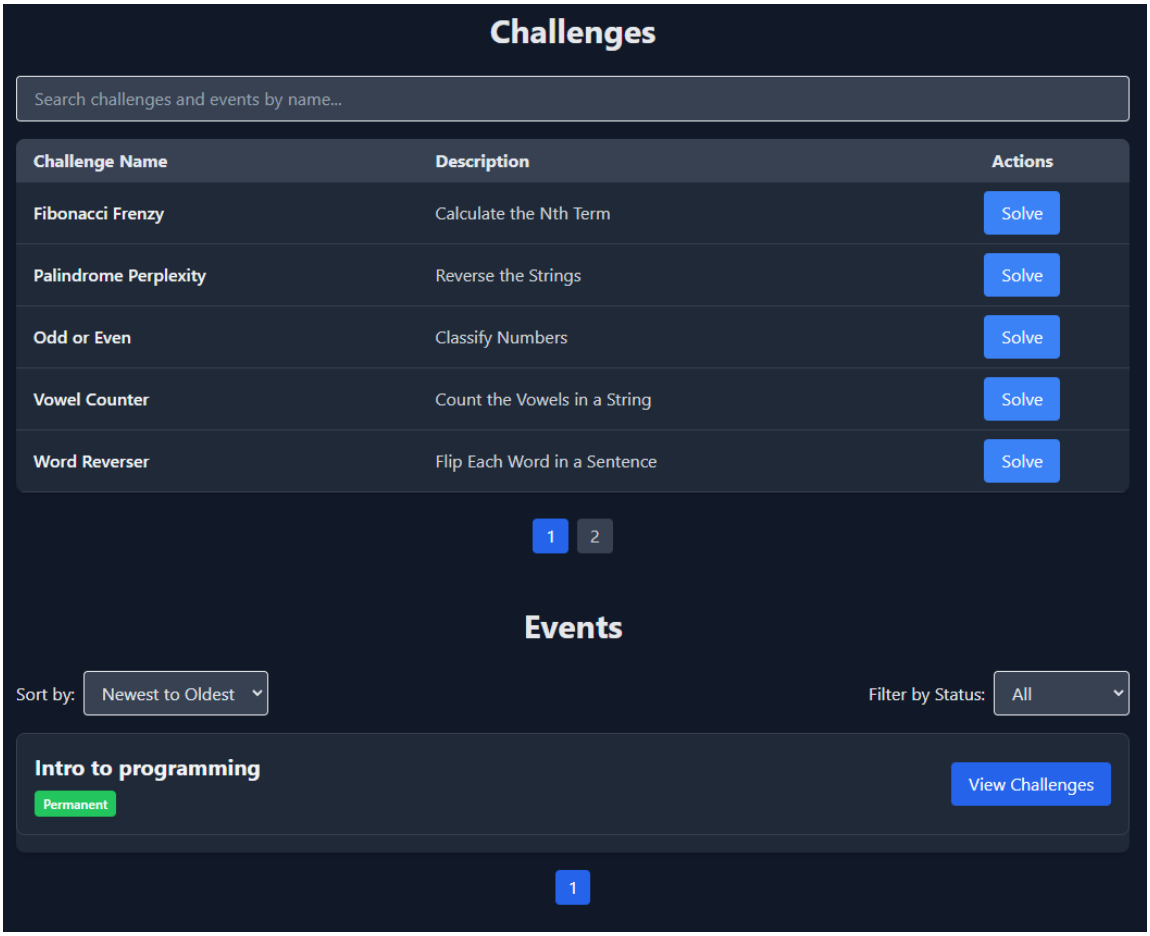


Slika 4.3 Navigacijska traka i padajući izbornik

4.1.3 Svjetli i tamni način rada

TailwindCSS omogućava jednostavnu implementaciju svijetlog i tamnog načina rada, što korisnicima daje mogućnost prilagodbe teme platforme prema njihovim željama i uvjetima u kojima koriste platformu. Ova fleksibilnost omogućava ugodnije korisničko iskustvo, bez obzira na okolinu ili doba dana. Na Slici 4.4 prikazana je tamna verzija, dok je u ostatku rada korišten prikaz u svijetlom načinu rada. Samo preba-

civanje sa svijetlog na tamni način rada vrši se pritiskom na gumb na navigacijskoj traci te je promjena momentalna i nema potrebe za osvježavanjem stranice.



Slika 4.4 Tamni način rada

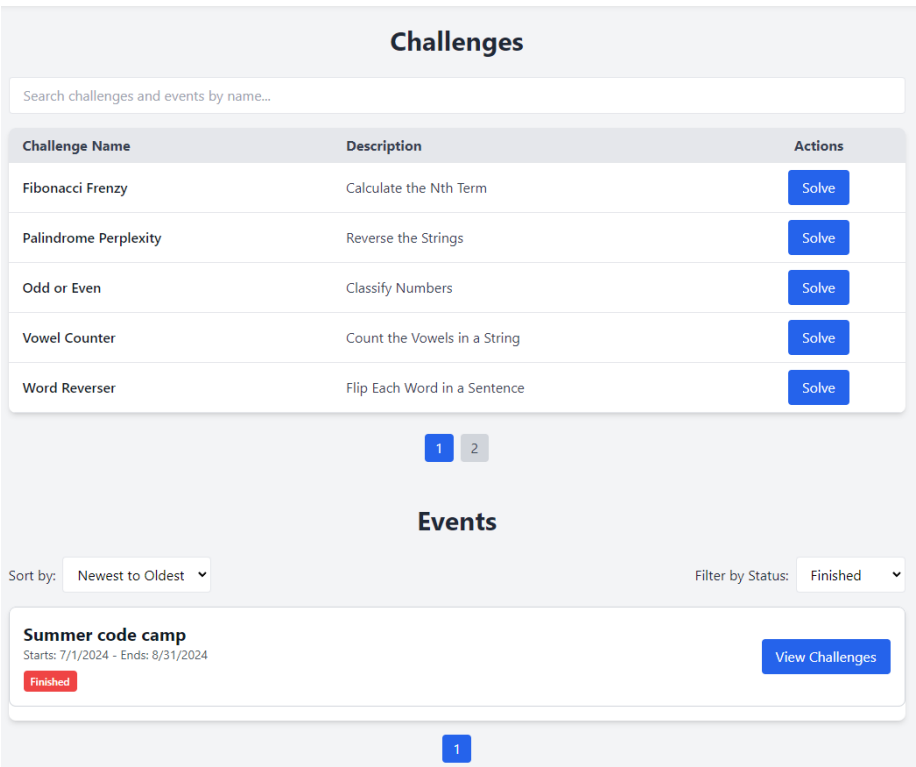
4.2 Slučajevi korištenja za programera

Korisnik usluge mora moći vidjeti sve javne zadatke i događaje koji postoje kako bi mogao pokušati riješiti bilo koji od njih. To mu omogućuje da odabere zadatke koje želi riješiti, prateći aktualne događaje i izazove. Tijekom rješavanja zadataka, korisnik treba imati pristup uređivaču teksta (engl. Code Editor) u kojem može pisati svoje rješenje. Osim toga, uređivač mu mora pružiti povratne informacije o točnosti koda koji je poslao na vrednovanje, kako bi mogao razumjeti i ispraviti eventualne

pogreške u svom rješenju te nastaviti s poboljšanjem koda.

4.2.1 Korisnički panel

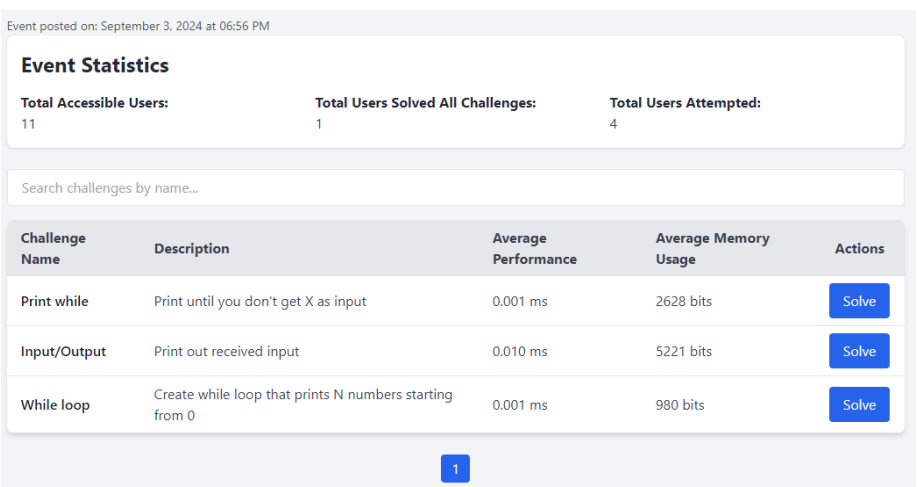
Korisnik, nakon uspješne prijave u sustav, dolazi na korisnički panel (engl. User Dashboard) gdje vidi sve javne izazove koje može pokušati riješiti te listu svih događaja koji su mu dostupni i za koje postoji opcija za pregled. Na vrhu stranice nalazi se opcija filtriranje izazova i događaja po njihovom imenu kako bi se korisniku olakšala navigacija kroz listu. Za same događaje postoji i opcija filtriranja po četiri verzije događaja koji su redom stalni, u tijeku, nadolazeći te završeni. Slika 4.5 prikazuje korisnički panel.



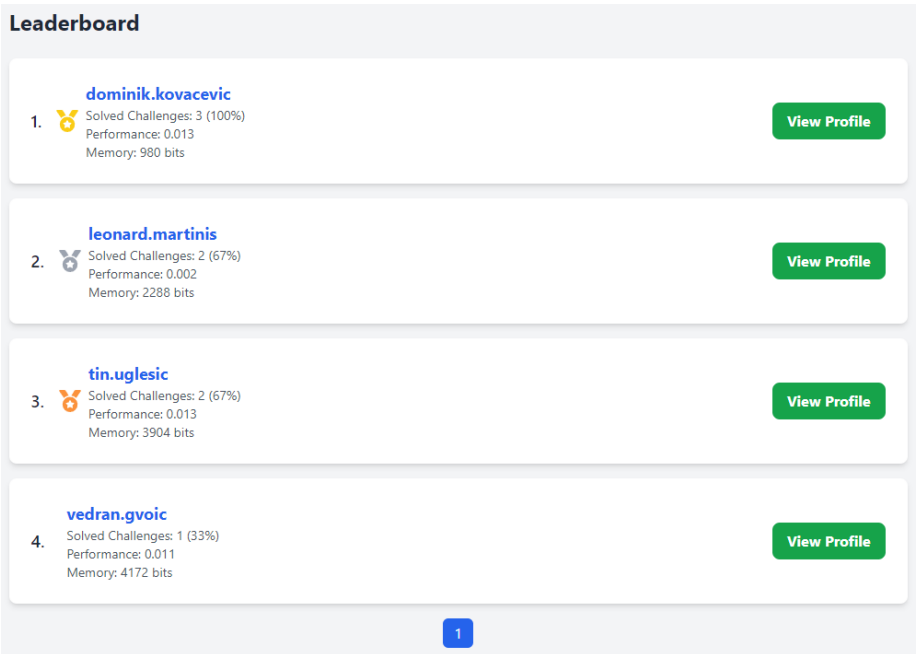
Slika 4.5 Korisnički panel

4.2.2 Pregled događaja

Pritiskom na gumb 'Pregled izazova' (engl. View Challenges), koji se može vidjeti na Slici 4.5, korisnik dolazi na novu stranicu koja pruža dodatne informacije o događaju, uključujući osnovne statistike o događaju i listu zadataka vezanih uz taj događaj. Ova funkcionalnost prikazana je na Slici 4.6. Ispod toga nalazi se i lista najuspješnijih korisnika za taj događaj, sortirana prema broju riješenih zadataka, a zatim po vremenu izvršavanja i iskorištenoj memoriji vidljivo na Slici 4.7.



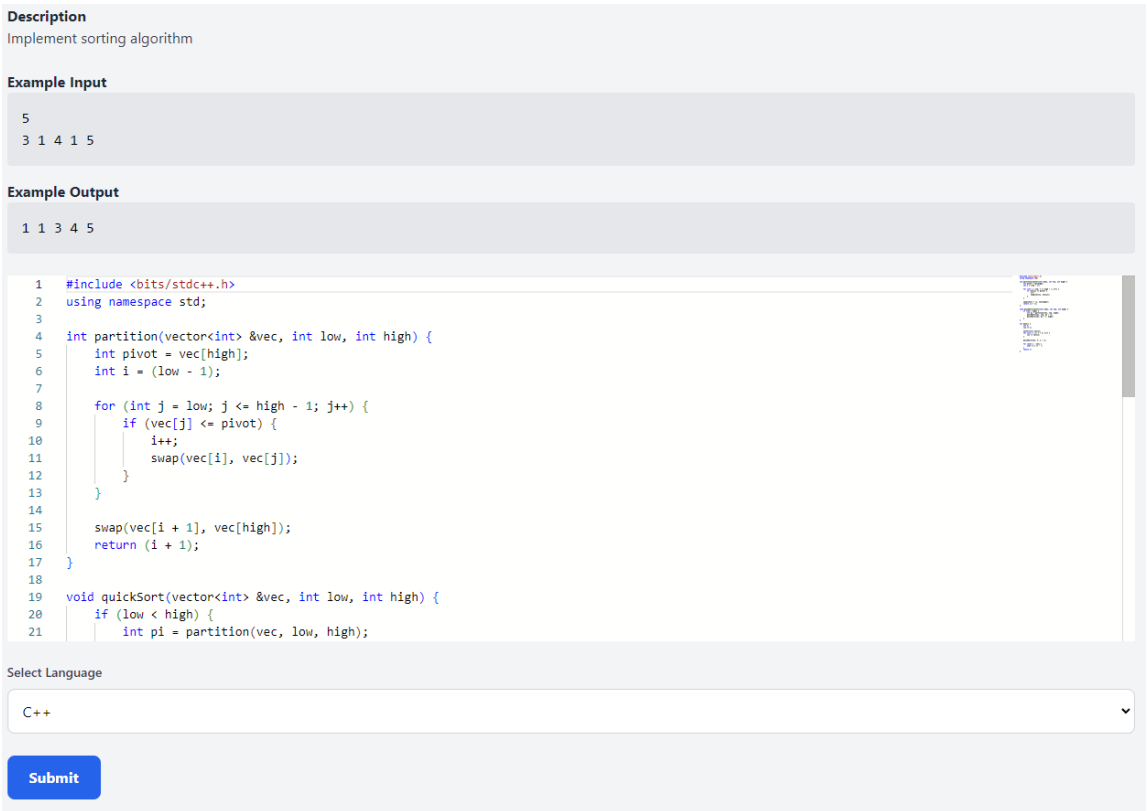
Slika 4.6 Statistike i zadaci za pojedini događaj



Slika 4.7 Lista najuspješnijih korisnika za pojedini događaj

4.2.3 Rješavanje izazova

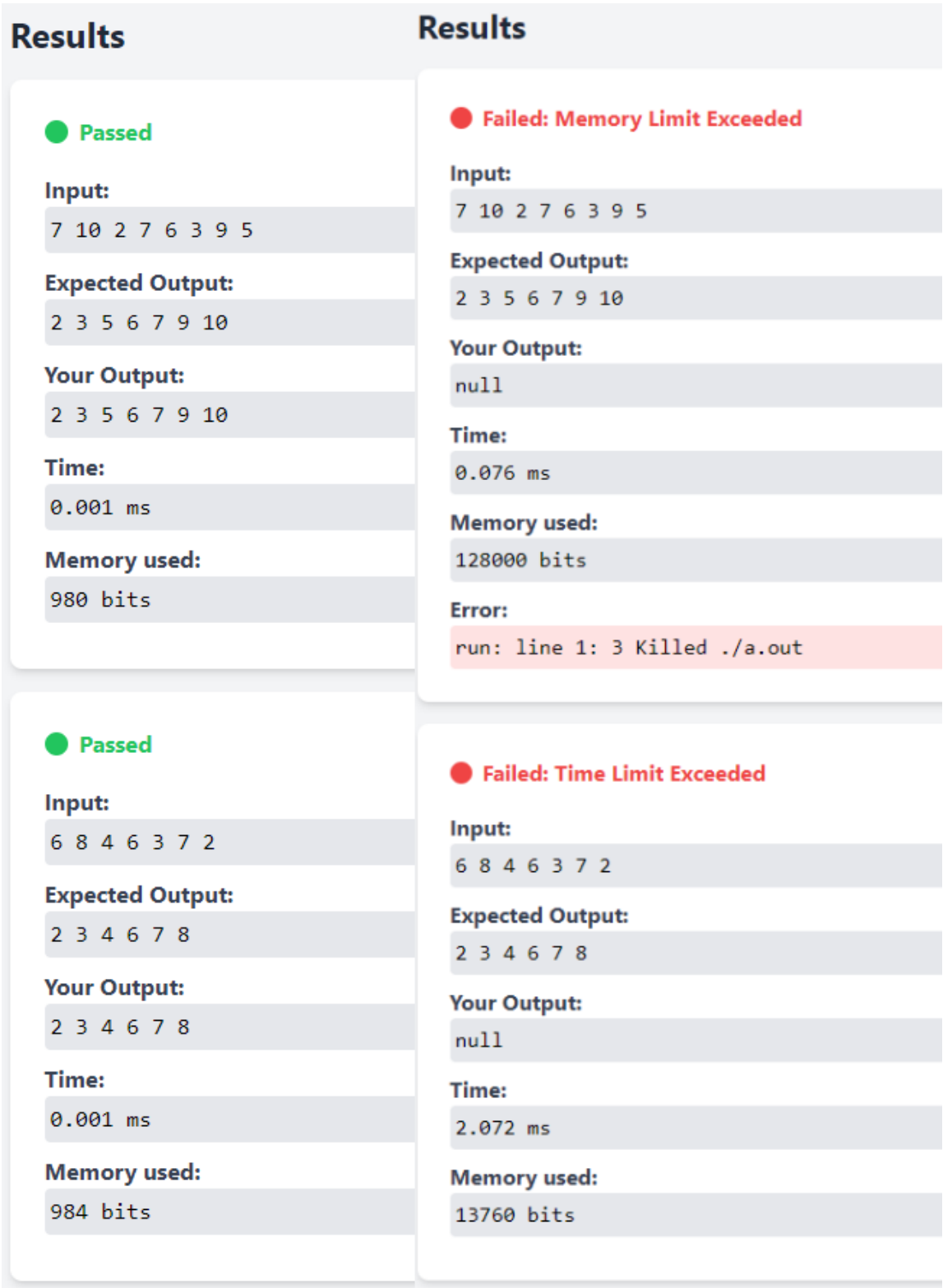
Nakon što korisnik pritisne gumb za rješavanje izazova, odlazi na novu stranicu gdje ima mogućnost pisanja koda u Visual Studio Code (VSC) uređivaču teksta koji omogućava isticanje sintakse (engl. Syntax highlighting) programskog jezika kojeg koristi, kao što su Python, C++ i C. Unutar editora korisnici također mogu vidjeti osnovne greške u pisanju koda. Iznad samog dijela za pisanje koda nalazi se opis zadatka te jedan primjer ulaza i očekivanog izlaza kako bi korisnik uvijek mogao pročitati zahtjeve zadatka kojega trenutno rješava. Ispod dijela za pisanje koda nalazi se izbornik jezika kojeg korisnik želi koristiti za rješavanje zadatka te gumb za predaju koda na evaluaciju. Slika 4.8 prikazuje sve funkcionalnosti koje su spomenute u ovoj sekciji.



Slika 4.8 Rješavanje izazova

4.2.4 Vrednovanje prijavljenog rješenja

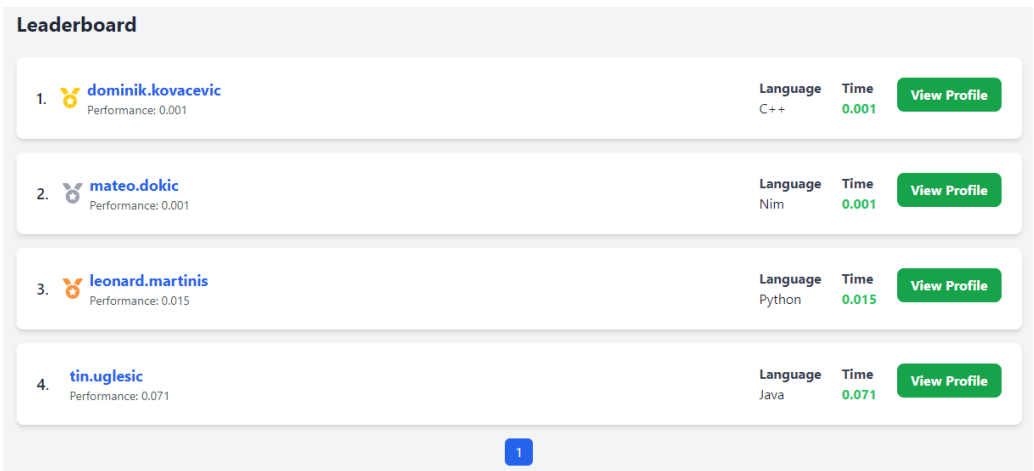
Ako korisnik prijavi rješenje zadatka na vrednovanje, nakon što je on uspješno ili neuspješno obrađen, korisnik dobije povratnu informaciju u obliku rezultata testova. Ovi rezultati prikazuju ulaz i izlaz zadatka, izlaz generiran prijavljenim rješenjem te eventualne greške koje su se dogodile tijekom izvođenja koda. Vrste grešaka mogu biti povezane s izlazom programa, vremenom izvršavanja, zauzetom memorijom te greškama vezanim uz prevođenje programa. Slika 4.9 na ljevoj stranici prikazuje rezultate točnog rješenja koda te na desnoj prikazuje slučaj u kojem je kod neispravan zbog vremenskog i memorijskog ograničenja.



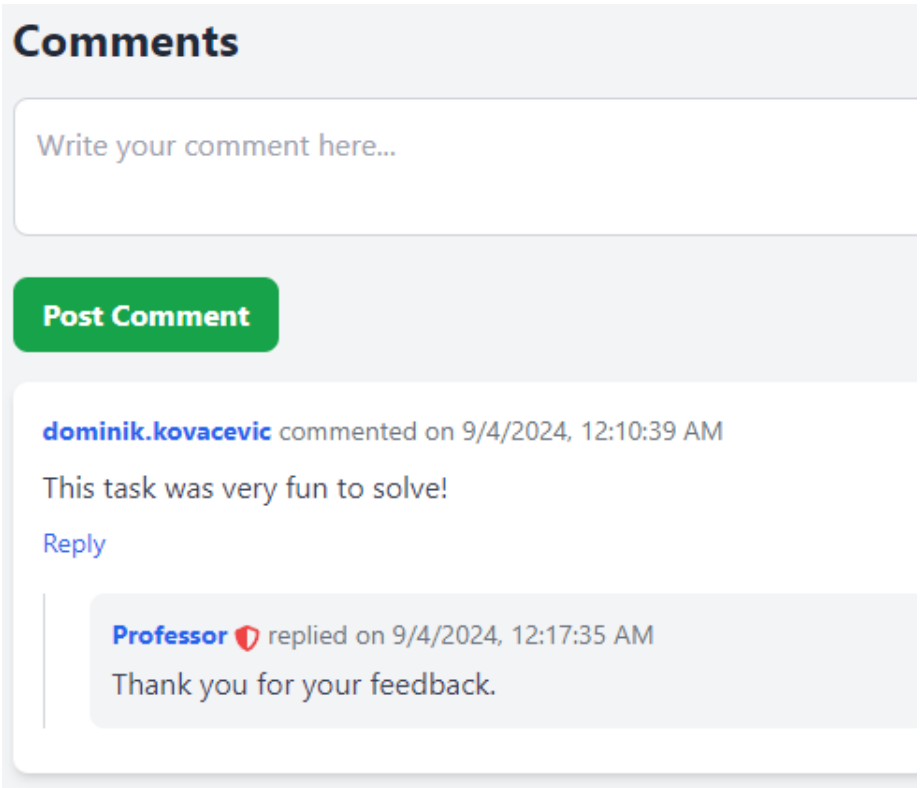
Slika 4.9 Rezultati ispitivanja koda

4.2.5 Komentari i lista s najboljim rezultatima

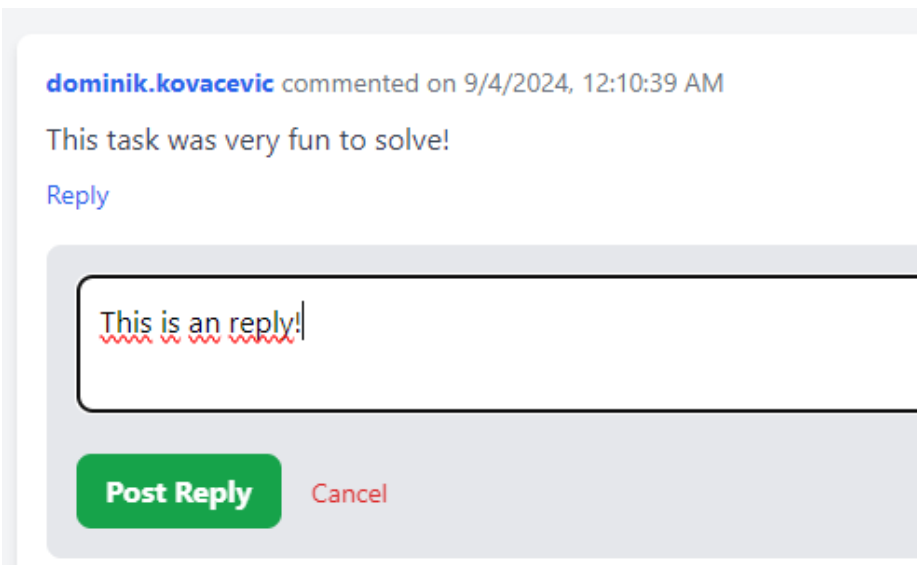
Slika 4.10 prikazuje listu korisnika s najboljim rješenjima zadatka. Ona se nalazi odmah ispod rezultata testova. Lista se sastoji od imena korisnika, vremena koje je bilo potrebno za izvršenje njihovog rješenja, jezika koji je korišten te gumba koji vodi na njihov profil. Lista prikazuje prvih N korisnika, a ispod nje se nalaze numerirani gumbi od prve do posljednje stranice, koje korisnik može koristiti za navigaciju kroz listu. Lista također ističe prva tri mjesta dodjeljivanjem znački zlatnog, srebrnog i brončanog tipa. Tako se jasno vidi koja su rješenja najbolja za pojedini zadatak, olakšavajući korisnicima prepoznavanje najuspješnijih rješenja. Slika 4.11 prikazuje sekciju s komentarima, gdje korisnik može pročitati mišljenja drugih korisnika o zadatku, zatražiti pomoć ili ukazati na moguće probleme u opisu zadatka, kao i u ulazima ili izlazima zadatka. Svaki komentar prikazuje ime korisnika koji ga je napisao i vrijeme kad je napisan, pri čemu je ime korisnika istovremeno i gumb koji vodi na njegov profil. Korisnik također može odgovarati na tuđe ili svoje komentare pritiskom na gumb "Odgovori" (engl. Reply) koji mu otvara novi prozor za pisanja odgovora na odabrani komentar, što je prikazano na Slici 4.12. Ako je komentar napisao administrator platforme, uz ime tog korisnika nalazi se oznaka koja pomaže korisnicima da prepoznaju vjerodostojan izvor. Ako je lista komentara preduga, odnosno ako ima previše komentara, korisniku se prikazuje samo prvih N komentara, gdje je N definiran u kodu platforme. Korisnik tada ima opciju učitavanja dodatnih komentara putem odgovarajućeg gumba.



Slika 4.10 Lista s najboljim rezultatima



Slika 4.11 Lista komentara

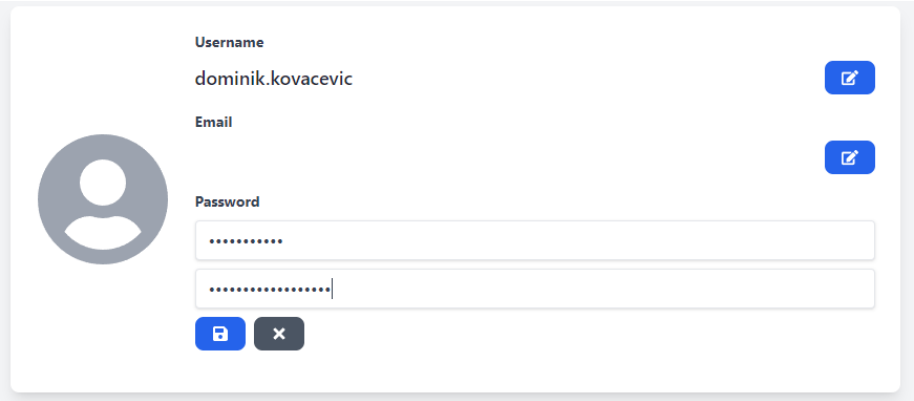


Slika 4.12 Opcija komentiranja

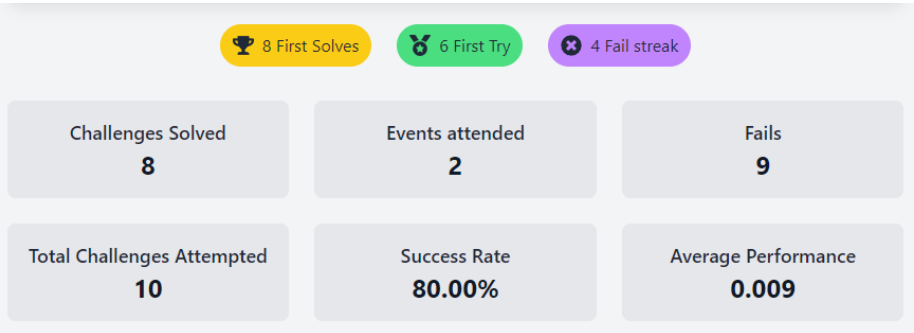
4.2.6 Korisnički profil

Do stranice s korisničkim profilom moguće je doći na tri načina. Prvi je način preko gumba koji se nalazi u padajućem izborniku navigacijske trake na svakoj stranici, a koji korisnika vodi na njegov vlastiti profil. Drugi je način pritiskom na gumb za pregled profila na listi najboljih odgovora za pojedini zadatak ili na listi najuspješnijih korisnika na pojedinom događaju. Korisnik također može doći do nečijeg profila i putem liste komentara. Izgled stranice za korisnički profil razlikuje se ovisno o tome pregledava li korisnik vlastiti profil ili profil drugog korisnika. Glavna je razlika u mogućnosti uređivanja profila, koja je dostupna samo ako korisnik pregledava svoj vlastiti profil, što je prikazano na Slici 4.13.

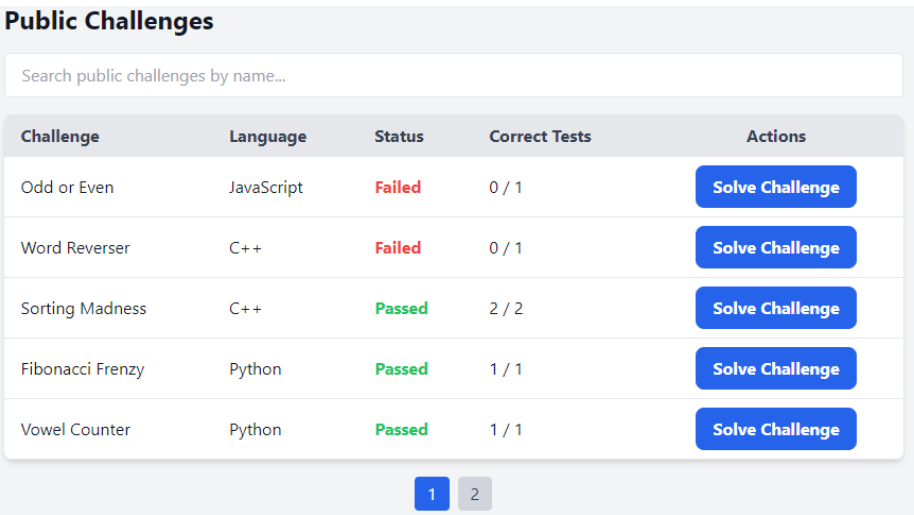
Opcija za uređivanje profila omogućuje korisniku izmjenu korisničkog imena, adrese elektroničke pošte (engl. e-mail) korištene za registraciju računa te izmjenu lozinke. Za promjenu lozinke potrebno je unijeti staru i novu lozinku kako bi se spriječilo neovlašteno mijenjanje lozinke u slučaju zaboravljanja odjave iz sustava. Ispod osnovnih informacija prikazane su statistike o korisniku, kao što su broj riješenih zadataka, broj pokušaja rješavanja zadataka te postotak uspješnih i neuspješnih rješenja na platformi. Uz ove osnovne statistike, na pojedinim profilima mogu se pojaviti i značke. Primjeri znački uključuju broj zadataka koje je korisnik riješio prvi, broj zadataka riješenih iz prvog pokušaja te najduži broj pokušaja koji je korisniku bio potreban za rješavanje određenog zadatka. Slika 4.14 prikazuje sve opisane statistike i značke. Ispod statistike nalazi se lista svih rješenja korisnika kako bi drugi korisnici mogli vidjeti koje je zadatke korisnik uspješno riješio ili neuspješno pokušao riješiti. Lista sadrži ime zadatka, jezik u kojem je zadatak rješevan, status zadatka (riješen ili neriješen) te broj točnih testova. Iznad liste nalazi se opcija za pretraživanje po imenu zadatka, što korisniku omogućuje lakše pretraživanje. Ako korisnik pregledava svoj profil, dostupna mu je i opcija za odlazak na stranicu za rješavanje zadataka, što je prikazano na Slici 4.15. Ispod liste s rješenjima javnih zadataka nalazi se i lista s rješenjima zadataka s događaja kojima je korisnik prisustvovao. Rješenja su grupirana prema događajima s kojima su povezana, a korisniku je omogućeno filtriranje po imenu događaja. Slika 4.16 prikazuje listu zadataka s događaja s gore opisanim funkcionalnostima.



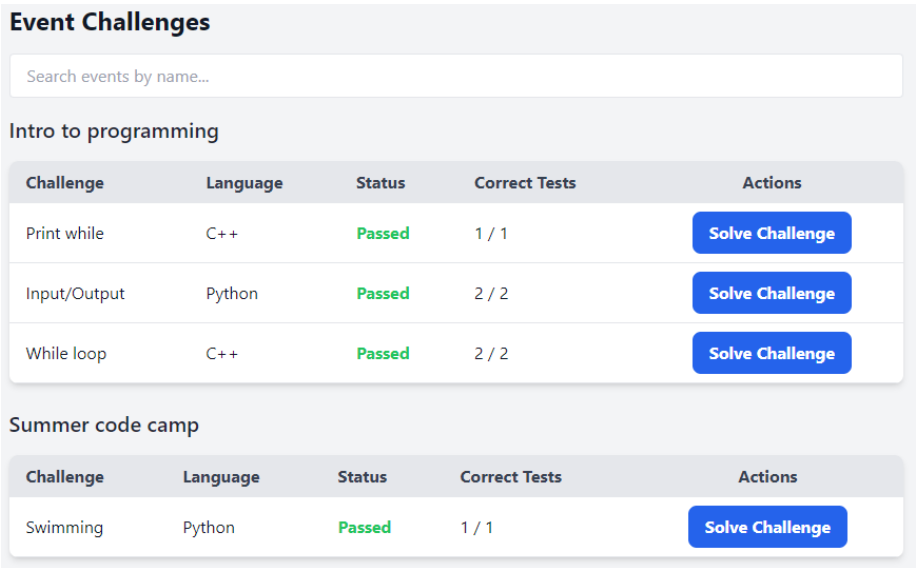
Slika 4.13 Opcija za ažuriranje vlastitog korisničkog profila



Slika 4.14 Prikaz statistike i znački na korisničkom profilu



Slika 4.15 Lista korisnikovih rješenja za javne zadatke



Slika 4.16 Lista korisnikovih rješenja za pojedine događaje

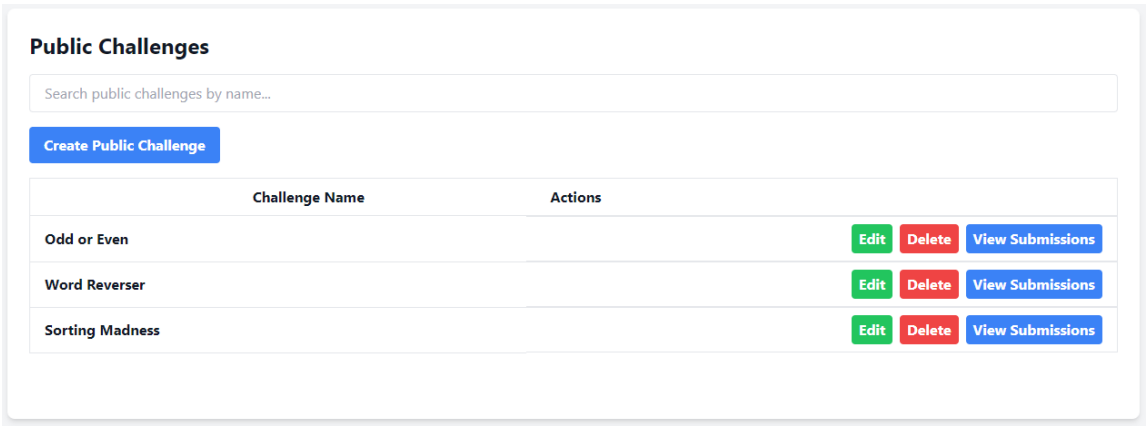
4.3 Slučajevi korištenja za administratora

Administrator usluge mora imati mogućnost stvaranja novih izazova (engl. Challenges) i događaja (engl. Events). Pored toga, potrebno je da ima uvid u rješenja svih korisnika te pristup ploči s najboljim rezultatima (engl. Leaderboards) za svaki izazov. Također, administrator mora moći izmijeniti i izbrisati postojeće izazove i događaje.

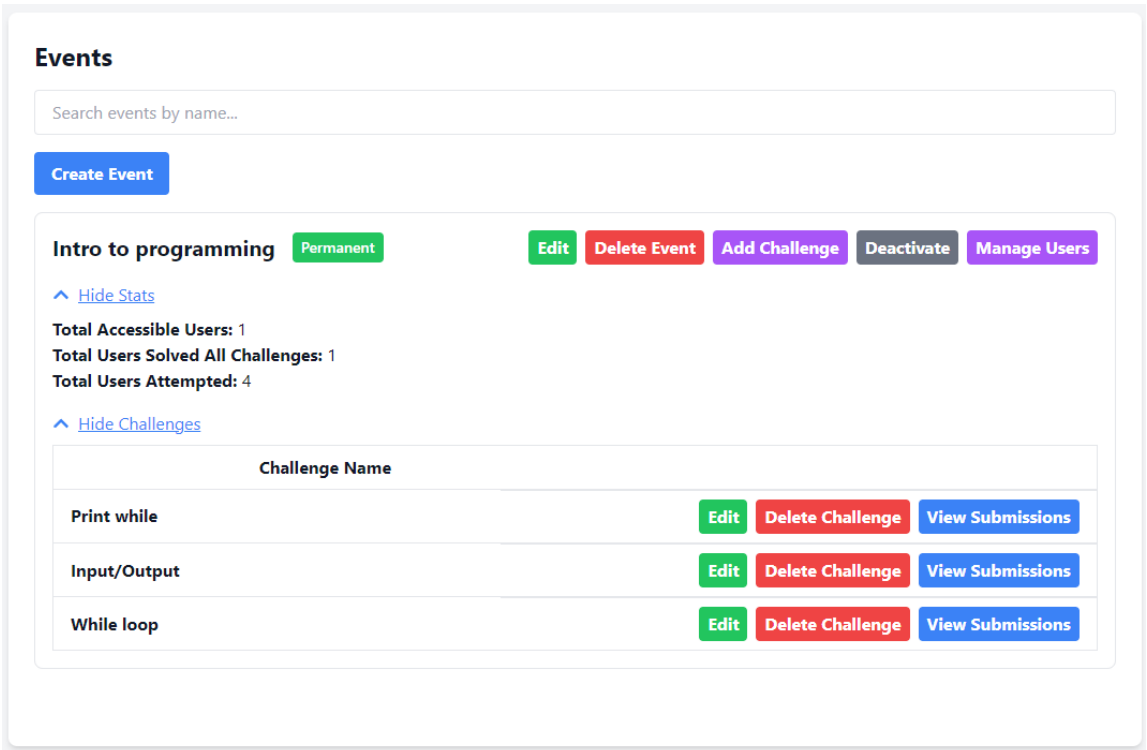
4.3.1 Administracijski panel

Nakon uspješne prijave u aplikaciju administrator, dolazi na nadzornu ploču (engl. Dashboard) gdje dobiva uvid u sve javne izazove koji su dostupni tipičnim korisnicima te mogućnost filtriranja tablice po imenu zadataka za lakše snalaženje. Lista je odvojena u sekcije po pet ili deset zadataka, ovisno o definiciji u projektu. Za svaki izazov administrator ima mogućnost izmjene samog zadatka, brisanje tog zadatka ili pregleda rješenja korisnika. Slika 4.17 prikazuje listu javnih izazova. Osim uvida u javne izazove, administratoru su prezentirani i svi događaji koji se mogu smatrati kao verzija ispita u sličnim programskim rješenjima za fakultete. Svaki događaj ima listu izazova koji nisu dio javne liste, već su odvojeni. Slika 4.18 prikazuje dizajn

liste događaja. Lista administratoru daje uvid u statistiku samog događaja te listu zadataka koji su povezani s njim.



Slika 4.17 Prikaz liste javnih izazova za administratora sustava

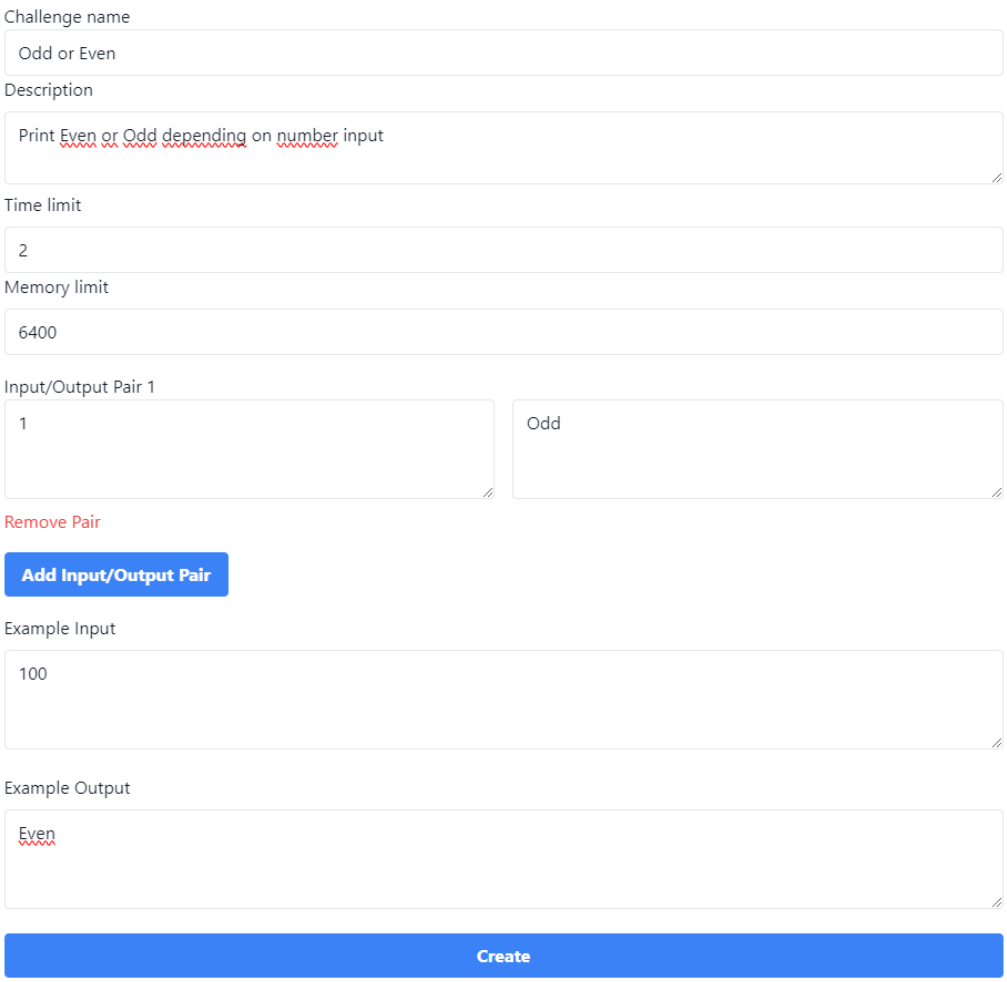


Slika 4.18 Prikaz liste događaja za administratora sustava

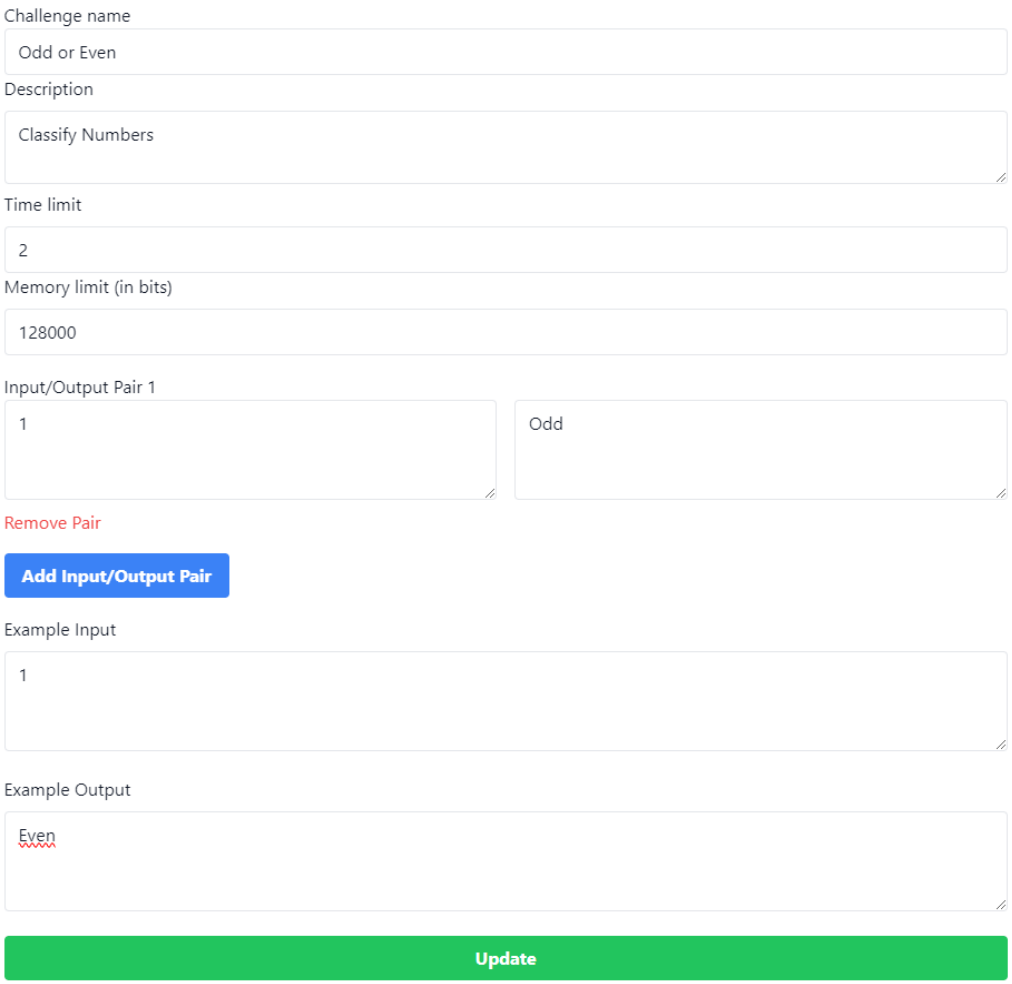
4.3.2 Stvaranje novih i uređivanje postojećih izazova

Slika 4.19 prikazuje prozor koji se otvara nakon pritiska na gumb za stvaranje novog izazova. U ovom prozoru administrator unosi podatke o izazovu, kao što su ime izazova, njegov opis, primjer ulaza i izlaza, vremenski limit za izvršenje zadatka te memorijski limit. Opis izazova i primjer ulaza i izlaza služe kako bi korisnici mogli jasno razumjeti što se od njih traži za uspješno rješavanje zadatka. Vremenski i memorijski limit pomažu spriječiti loša rješenja, posebno kada je cilj zadatka korištenje određenih specifičnih lekcija ili algoritama. Osim ovih opcija, administrator unosi i ulazne i izlazne parametre izazova koji se koriste za provjeru točnosti koda kojeg korisnici predaju na evaluaciju. Ovi parametri omogućuju sustavu da automatski ocijeni ispravnost rješenja na temelju unesenih podataka.

Slika 4.20 prikazuje prozor koji se otvara pritiskom na gumb za uređivanje pojedinog izazova. Administrator ima opciju za prilagođavanje ulaza i izlaza zadataka, imena, opisa, primjera ulaza i izlaza te memorijski i vremenski limit izazova ako je došlo do greške pri stvaranju ili su jednostavno potrebne promjene u trenutnom izazovu. Osim toga, također postoji opcija brisanja izazova koja, osim brisanja samog izazova iz baze, briše i sva rješenja tog specifičnog izazova te sve tablice testnih slučajeva povezanih s tim rješenjem. Dugme za brisanje je prikazano na slici 4.17



Slika 4.19 Panel za stvaranje novog izazova



Slika 4.20 Panel za izmjenu izazova

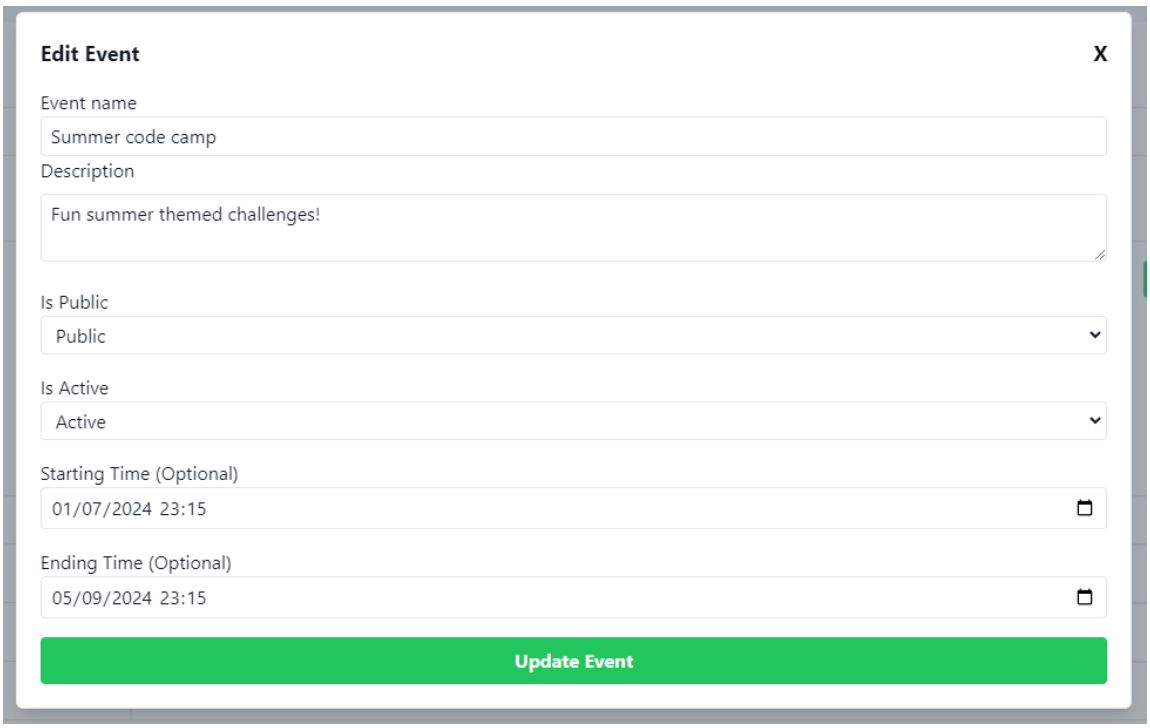
4.3.3 Stvaranje novih i uređivanje postojećih događaja

Osim stvaranja izazova, administrator ima mogućnost stvaranja događaja pritiskom na gumb za stvaranje događaja (engl. Create Event), što je prikazano na Slici 4.21. Pritiskom na taj gumb otvara se novi panel koji omogućuje administratoru unos imena događaja, njegovog opisa te opciju za određivanje statusa događaja, bez obzira na to hoće li događaj biti privatan ili javan, aktivan ili neaktivan. Uz ove osnovne opcije, administrator također ima mogućnost dodavanja izazova koji su posebno stvoreni za taj događaj. Ova funkcionalnost radi vrlo slično funkcionalnosti za stvaranje

javnih izazova, koja je objašnjena u sekciji 4.3.2. Nakon što je izazov dodan, on se pojavljuje na dnu panela, a administrator može dodati dodatne izazove ili ukloniti one koje više ne želi uključiti.

Pritiskom na gumb za uređivanje događaja, administratoru se omogućuje uređivanje osnovnih stavki događaja, kao što su njegovo ime, opis te status događaja. Slika 4.22 prikazuje panel za uređivanje događaja. Osim opcije za uređivanje događaja, administratoru se se nude opcije za brisanje događaja, dodavanje novih i brisanje starih izazova, aktivaciju ili deaktivaciju događaja te opcija za dodavanje i uklanjanje korisnika ako je događaj privatan. Uređivanje pojedinačnih zadataka izgleda i funkcionira isto kao i uređivanje javnih zadataka.

Slika 4.21 Panel za stvaranje događaja

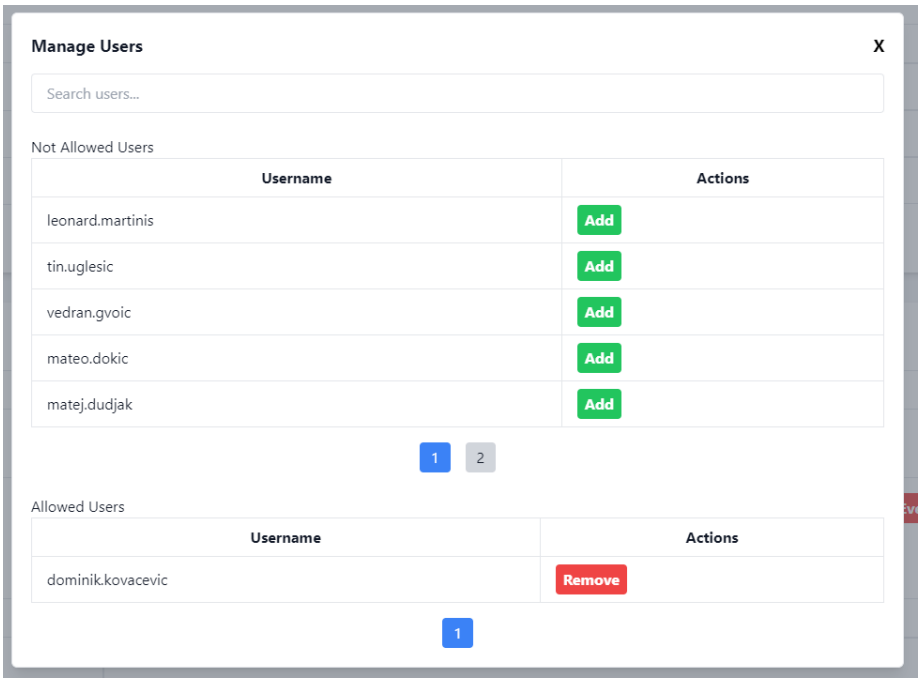


Slika 4.22 Panel za uređivanje događaja

4.3.4 Privatni događaji

Kao što je prikazano na slici 4.18, postoji i gumb za upravljanje korisnicima unutar događaja, koji se pojavljuje samo u slučaju da je događaj privatnog tipa. Ova funkcionalnost omogućuje administratoru platforme da kontrolira pristup događaju na način sličan organizaciji razreda za ispit, osiguravajući da samo odabrani korisnici mogu pristupiti tom specifičnom događaju. Slika 4.23 prikazuje panel koji se otvara nakon pritiska na gumb za upravljanje korisnicima. Ovaj panel sastoji se od dvije liste korisnika: liste svih korisnika koji nemaju pristup događaju, s opcijom da im se taj pristup omogući te liste korisnika koji već imaju pristup, s opcijom da im se pristup ukine. Obje liste imaju straničenje radi lakše navigacije. Također, panel nudi mogućnost unosa imena korisnika za filtriranje liste, čime se administratoru omogućuje brže pronalaženje specifičnog korisnika te davanje ili oduzimanje pristupa događaju. Liste se automatski ažuriraju, tj. korisnik se odmah dodaje na listu korisnika s pristupom i uklanja s liste korisnika bez pristupa ako je akcija za odobravanje

pristupa uspješno izvršena i obrnuto. Ovaj proces omogućuje učinkovito upravljanje korisničkim pristupom, osiguravajući da pristup događaju imaju samo oni korisnici kojima je to odobreno.

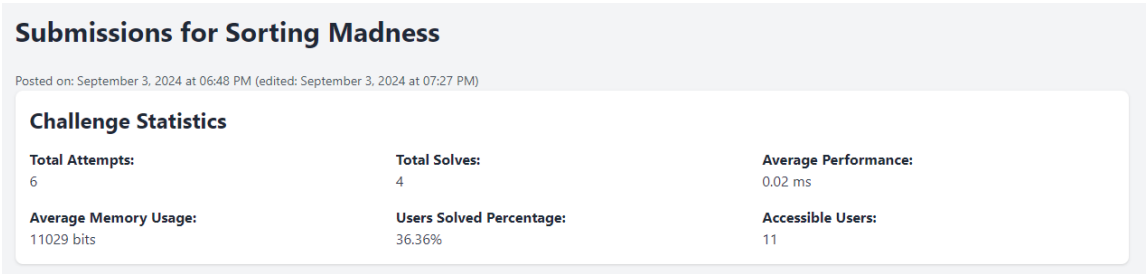


Slika 4.23 Panel za upravljanje pristupa događaju

4.3.5 Pregled prijavljenih rješenja

Nakon što administrator odabere opciju "Pregled rješenja" (engl. View Submissions), otvara se nova stranica na kojoj mu je prikazana tablica sa svim korisnicima koji su pokušali riješiti zadatak, bez obzira na to jesu li bili uspješni. Ova je funkcionalnost prikazana na Slici 4.24. Administrator također dobiva informacije o zadatku, kao što su broj pokušaja rješavanja zadatka, broj uspješnih rješenja te postotak korisnika koji su riješili zadatak. Ova funkcionalnost prikazana je na Slici 4.25. Također, kao i tipični korisnik, ima uvid u komentare na dotični izazov kako bi mogao odgovoriti na upite korisnika ili pročitati savjete i povratne informacije od istih.

Slika 4.24 Prikaz liste rješenja korisnika



Slika 4.25 Statistika specifičnog zadatka

4.3.6 Pregled pojedinačnog rješenja

Nakon što administrator pritisne dugme “Pregledaj kod” (engl. "View Code") preusmjeren je na novu stranicu gdje mu se u ugrađenom VSC uređivaču teksta prikazuje korisnikov kod s pravilnom sintaksom za jezik u kojem je korisnik pisao svoje rješenje. Ispod koda prikazani su rezultati testova, uključujući eventualne greške u izvođenju koda koje su se pojavile. Pored toga, administrator može vidjeti ulazne podatke, stvarni izlaz koda te očekivani izlaz za svaki od testova. Na Slici 4.26 prikazana je opisana funkcionalnost ove stranice.

Code:

```
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int n;

    // Read the total number of elements
    scanf("%d", &n);
    if(n == 6) return 0;
    int arr[n];

    // Read the numbers into the array
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Sort the array using bubble sort
```

Test Results

Passed

Input:

```
7 10 2 7 6 3 9 5
```

Expected Output:

```
2 3 5 6 7 9 10
```

Your Output:

```
2 3 5 6 7 9 10
```

Time:

```
0.001 ms
```

Memory used:

```
1592 KB
```

Slika 4.26 Prikaz rješenja specifičnog korisnika

Poglavlje 5

Zaključak

Sustav za automatsku provjeru i vrednovanje programskih rješenja predstavlja značajan napredak u procesu obrazovanja, posebno u području programiranja. Ovaj sustav pruža niz korisnih alata i funkcionalnosti koji studentima omogućuju brže i efikasnije učenje ključnih koncepata programiranja. U ovome su radu istraženi su postojeći sustavi za vrednovanje programskih rješenja i nastojalo se utvrditi mogu li se ona implementirati za rješavanje postavljenog zadatka.

Ključan kriterij za izbor određenog programskog rješenja bio je da ono bude otvorenog koda, odnosno da je moguć potpuni pristup i izmjena izvornog koda. Naime, jedan od ključnih problema s kojim se sustavi za vrednovanje koda susreću jest rad s kodom koji se ne prevodi ili koji sadrži sintaksne greške. Validacijski sustav mora biti dovoljno sofisticiran kako bi prepozno različite vrste grešaka i pružio korisniku povratnu informaciju koja će mu biti korisna za daljnji rad na kodu. Ovo zahtijeva ne samo dobar parser za određeni programski jezik, nego i mogućnost prilagodbe različitim verzijama i dijalektima tog jezika. Također, sigurnost je još jedan bitan aspekt prilikom izbora programskog rješenja. Kao najbolje rješenje izabrana je platforma Judge0. To je platforma otvorenog koda koja je specijalizirana za izvršavanje i ocjenjivanje koda u širokom rasponu programskih jezika. Ovaj sustav dizajniran je da bude vrlo svestran, podržava više od 40 različitih programskih jezika, uključujući one popularne kao što su Python, Java, C++ i JavaScript. Platforma je posebno korisna u scenarijima kao što su natjecanja u kodiranju, online platforme za kodiranje i obrazovna okruženja u kojima je bitno automatizirano testiranje koda. Judge0 nudi *RESTful* API koji programerima i edukatorima omogućuje jednostavnu integraciju

platforme u svoje aplikacije. Ovaj API olakšava podnošenje koda, koji Judge0 zatim prevodi i pokreće u sigurnom, izoliranom okruženju. Ovo izvođenje u zaštićenom okruženju osigurava da kod radi bez ikakvog rizika za glavni sustav ili da uzrokuje smetnje drugim korisnicima. Nakon izvršenja, Judge0 pruža detaljne povratne informacije, uključujući rezultate izvršenja, poruke o pogrešci i metriku upotrebe resursa poput potrošnje memorije i procesora.

Osim platforme Judge0, arhitektura sustava se oslanja na nekoliko ključnih komponenti, uključujući klijentski dio aplikacije, poslužiteljsku stranu i bazu podataka. Klijent, koji obično predstavlja sučelje s kojim studenti ili korisnici komuniciraju, šalje zahtjeve prema poslužitelju koristeći HTTP protokol. Ovi zahtjevi obično sadrže podatke u JSON formatu, uključujući programski kod koji se šalje na vrednovanje. Nakon primitka zahtjeva, poslužitelj prosljeđuje kod platformi Judge0 na obradu i validaciju. Nakon što Judge0 izvrši kod, validirani rezultati, uključujući izlaz programa, eventualne pogreške i informacije o performansama izvođenja, vraćaju se poslužitelju. Poslužitelj zatim obrađuje te podatke i sprema ih u bazu podataka za daljnje korištenje. Baza podataka igra ključnu ulogu u pohranjivanju svih relevantnih informacija o korisnicima, njihovim rješenjima i rezultatima vrednovanja, što omogućuje lakšu analizu napretka i povratnih informacija.

Ovakav sustav omogućuje visoku razinu fleksibilnosti jer studentima omogućuje brz i jednostavan pristup provjeri njihovih programskih rješenja. Brze povratne informacije potiču iterativni proces učenja, gdje korisnici mogu ispraviti pogreške, unaprijediti svoj kod i ponovno ga poslati na provjeru. To rezultira učinkovitijim učenjem jer se studenti ne zadržavaju predugo na tehničkim problemima, nego se usmjeravaju na ključne koncepte i logiku. Sustav rangiranja najboljih rješenja korisnika, zajedno s mehanizmom nagrađivanja putem znački na korisničkim profilima, dodatno motivira korisnike da kontinuirano poboljšavaju kvalitetu svog koda. Za nastavnike i administratore, sustav nudi niz dodatnih pogodnosti. Vrednovani rezultati pohranjeni u bazi podataka te statistike o pojedinim zadacima i događajima omogućuju detaljnu analizu učinka svakog korisnika, čime se mogu prepoznati određeni obrasci ili područja u kojima studenti imaju poteškoća. Ova analiza omogućava nastavnicima da prilagode nastavne metode ili dodatno usmjere pažnju na specifične teme ili zadatke.

Literatura

[1] Judge0, Accessed August 2024. adresa: https://judge0.com/.

[2] E. Man. "Piston: A High-Performance Execution Engine." Accessed: August 2024. (2024.), adresa: https://github.com/engineer-man/piston.

[3] E. Man. "How I Built The Internet's Best Performing Code Execution Engine." Accessed: August 2024. (2021.), adresa: https://www.youtube.com/watch?v=SD4KgwdjmdI.

[4] S. Boot, Accessed August 2024. adresa: https://spring.io/projects/spring-boot.

[5] M. Ganesh, S. A. R. Albert i I. T. J. Swamidason, "An Analysis of the Significance of Spring Boot in The Market," srpanj 2022., str. 1277–1281. DOI: 10.1109/ICICT54344.2022.9850910.

[6] Baeldung, *Introduction to Project Lombok*, Accessed August 2024. adresa: https://www.baeldung.com/intro-to-project-lombok.

[7] M. B. Jones, J. Bradley i N. Sakimura, *JSON Web Token (JWT)*, RFC 7519, svibanj 2015. DOI: 10.17487/RFC7519. adresa: https://www.rfc-editor.org/info/rfc7519.

[8] React, *Why did we build React?* Accessed August 2024. adresa: https://legacy.reactjs.org/blog/2013/06/05/why-react.html.

[9] React, *React*, Accessed August 2024. adresa: https://react.dev/.

[10] TailwindCSS, *TailwindCSS*, Accessed August 2024. adresa: https://tailwindcss.com/.

[11] PostgrSQL, *PostgreSQL 16.4 Documentation*, Accessed August 2024. adresa: `https://www.postgresql.org/files/documentation/pdf/16/postgresql-16-A4.pdf`.

[12] M. Stonebraker i L. A. Rowe, *THE DESIGN OF POSTGRES*, Accessed August 2024. adresa: `https://dsf.berkeley.edu/papers/ERL-M85-95.pdf`.

[13] C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Operating Systems Review*, sv. 49, br. 1, str. 71–79, 2015.

[14] D. Merkel, "Docker: lightweight Linux containers for consistent development and deployment," *Linux Journal*, sv. 2014, br. 239, str. 2, 2014.

[15] S. Ghaith i R. Asal, "A comprehensive study: Kubernetes vs. Docker Swarm," *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, IEEE, 2019., str. 1–5.

Pojmovnik

ACID atomicity, consistency, isolation, and durability. 11, 13

API Application Programming Interface. 4, 5, 10, 39

CORS Cross-origin resource sharing. 10

CSS Cascading Style Sheets. 10

DBMS Database management system. 11

GIN Generalized Inverted Index. 12

GiST Generalized Search Tree. 12

HTML HyperText Markup Language. 10

HTTP Hypertext Transfer Protocol. 6, 40

IDE Integrated Development Environment. 9

JAR Java ARchive. 15

JDK Java Development Kit. 9, 15

JPA Java Persistence API. 13

JSON JavaScript Object Notation. 6, 9, 11, 40

JSX JavaScript XML. 10

JWT JSON Web Token. 9, 10

ORM Object Relational Mapper. 13

Pojmovnik

RDMS Relational Database Management System. 11, 12

REST Representational State Transfer. 8, 9, 39

sp-GiST space-partitioned GiST. 12

SQL Structured Query Language. 6, 11

URL Uniform Resource Locator. 10, 13

VSC Visual Studio Code. 22, 37

Sažetak

U ovom radu istražene su mogućnosti postojećih sustava za automatsku evaluaciju programskih rješenja. Odabrano rješenje, Judge0, primijenjeno je u implementaciji izvornog web sustava koji omogućava definiranje i objavu programskih izazova i ispita s pripadnim testnim slučajevima i ograničenjima, automatsko vrednovanje pristiglih rješenja, komentiranje zadataka, upravljanje ljestvicama poretka, virtualno nagrađivanje, prikaz statističkih pokazatelja te upravljanje korisničkim profilima. Posebnu pažnja usmjerena je na upotrebljivost, s ciljem implementacije rješenja koje je responzivno, intuitivno i jednostavno za koristiti. Razvijeni sustav doprinosi području e-učenja, omogućavajući studentima napredan alat za stjecanje i unaprjeđivanje vještina programiranja.

Ključne riječi — **automatsko vrednovanje programskog koda, Judge0, web aplikacija**

Abstract

In this thesis, the possibilities of existing systems for automatic evaluation of program codes were investigated. The selected solution, Judge0, was utilized in the implementation of the original web system, which enables the definition and publication of programming challenges and exams with corresponding test cases and limitations, automatic evaluation of received solutions, commenting on tasks, management of ranking scales, virtual rewards, display of statistical indicators and user profile management. Special attention is focused on usability, aiming to implement a responsive, intuitive, and easy-to-use solution. The developed system contributes to e-learning, providing students with an advanced tool for acquiring and improving programming skills.

Keywords — **program code automatic evaluation, Judge0, web application**