ZADATAK: KvizHub – platforma za testiranje znanja sa rang listom

Cilj projekta je izgradnja veb aplikacije koja omogućava korisnicima da polažu kvizove iz različitih oblasti, dobijaju rezultate u realnom vremenu i upoređuju svoje znanje sa drugim korisnicima kroz rang listu. Aplikacija mora biti modularna, proširiva i implementirana prema savremenim standardima frontend i backend razvoja.

Funkcionalnosti:

1. **Korisnik:**
    - Registracija i prijava
    - Pregled i filtriranje dostupnih kvizova
    - Rešavanje kviza (ograničeno vreme)
    - Pregled rezultata
    - Prikaz ličnih rezultata
    - Pregled globalne rang liste
2. **Admin:**
    - CRUD operacije nad kvizovima i pitanjima
    - Kategorizacija kvizova
    - Pregled rezultata svih korisnika
    - Kreiranje kvizova sa različitim težinama pitanja

Kviz je skup pitanja iz jedne teme (ili više njih), sa unapred definisanim pravilima bodovanja, vremenskim ograničenjem i težinom.
Vrste pitanja:

1. **Jedan tačan odgovor**
   Klasičan multiple-choice (4 opcije, 1 tačna)
2. **Višestruki tačni odgovori**
   Više odgovora je tačno, korisnik mora sve izabrati
3. **Tačno/Netačno (True/False)**
   Jednostavna binarna pitanja
4. **Pitanja sa unosom (fill in the blank)**
   Korisnik mora uneti tačan tekstualni odgovor

Registracija i prijava:

- Novi korisnici mogu da kreiraju nalog unosom korisničkog imena, slike, mejla i lozinke.
- Na backendu se lozinka mora **heširati** (npr. pomoću BCrypt) i bezbedno čuvati.
- Dodati osnovnu validacija (dužina lozinke, validan mejl, jedinstveno korisničko ime).
- Korisnici se prijavljuju korišćenjem korisničkog imena ili mejla i lozinke.
- Ako su podaci tačni, backend generiše **JWT token** koji se prosleđuje frontendu.
- Token se čuva u localStorage ili sessionStorage i koristi za autentifikaciju na zaštićenim rutama.

Pregled i filtriranje dostupnih kvizova:

- Na stranici se prikazuju svi kvizovi dostupni korisniku.
- Svaki kviz sadrži osnovne informacije: naziv, opis, broj pitanja, procenjeni nivo težine i vremensko ograničenje.
- Moguće filtrirati kvizove po:
    - Temi/kategoriji (npr. „Programiranje“, „Istorija“)
    - Nivou težine (lako, srednje, teško)
- Implementirati i pretraga po ključnoj riječi (npr. „JavaScript“).

Rešavanje kviza (ograničeno vreme):

- Nakon klika na „Započni kviz“, učitava se lista pitanja.
- Startuje se tajmer na osnovu quiz.timeLimit.
- Tajmer se prikazuje vizuelno i automatski predaje kviz kada istekne.
- Korisnik može ručno završiti kviz ranije (klikom na dugme „Završi“).
- Sistem automatski računa broj tačnih odgovora i boduje rezultat.

Pregled rezultata (nakon kviza):

- Nakon završetka kviza, korisnik odmah vidi svoj rezultat:
    - Ukupan broj pitanja
    - Broj tačnih odgovora
    - Osvojeni procenat
- Dodatno:
    - Prikaz tačnih odgovora uz poređenje sa korisnikovim odgovorima.
    - Highlight grešaka.

Prikaz ličnih rezultata:

- Sekcija “Moji rezultati” sadrži istoriju svih kvizova koje je korisnik rešavao:
    - Naziv kviza

    - Datum rešavanja
    - Broj osvojenih poena / procenat
    - Dugme za pregled detalja
- Detalji uključuju:
    - Lista pitanja i korisnikovi odgovori
    - Vreme trajanja
    - Grafikon napretka (ako ima više pokušaja)

Pregled globalne rang liste:

Top rezultati za pojedinačne kvizove

- Dodatno prikazati:
    - Poziciju korisnika
    - Broj osvojenih bodova
    - Vrijeme postizanja rezultata (za rangiranje korisnika s istim rezultatom)
    - Filteri: po kvizu, po vremenskom periodu (nedeljni, mesečni)

Završna isporuka:

- GitHub repozitorijum sa frontend i backend kodom
- Arhitektura sistema
- UML dijagram
- README sa uputstvom za pokretanje
- SQL migracije
- U projektu se moraju ispoštovati kriterijumi kvaliteta rešenja i dobre prakse u izradi web aplikacija pokazane na vežbama.
- Prednja strana aplikacije mora biti podeljena po komponentama
- URL-ovi eksternih servisa koji se gađaju sa prednje strane moraju biti u .env fajlu i iščitavati se odatle, ovo uključuje i URL zadnje strane aplikacije.
- HTTP pozivi sa prednje strane moraju biti u servisima koji se injektuju u komponente, nikako direktno u komponentama.
- Moraju postojati modeli na prednjoj strani
- Zadnja strana mora biti troslojna web aplikacija, uz korišćenje injekcije zavisnosti.
- Moraju postojati Dto i modeli baze podataka kao odvojeni modeli i mora postojati adekvatno mapiranje između njih.
- Mora biti ispoštovana REST konvencija za nazivanje resursa. https://restfulapi.net/resource- naming.
- Lozinke u bazi podataka moraju biti heširane
- Potpis i istek tokena moraju biti validirani
- Konfigurabilne podatke (lozinke eksternih servisa, URL-ove) na zadnjoj strani držati u appsettings.json fajlu i učitavati.

Predmet projekat

Kako bi student mogao da polaže predmet projekat mora u potpunosti uraditi projekat iz predmeta Primena veb programiranja u infrastrukturnim sistemima. Sve funkcionalnosti sa specifikacije projekta iz Primena veb programiranja u infrastrukturnim sistemima moraju biti implementirane kako bi student imao pravo da radi predmet projekat.

Prijava za izradu projekta

Pravo da radi projekat ima najviše **50 studenata.** Na eepsi sajtu će biti okačena forma za prijavu uz napomenu kada se forma otvara za prijave, prvih 50 prijavljenih će biti evidentirano za izradu. Nakon isteka vremena za prijavu na sajtu će biti objavljen spisak studenata sa pravom izrade.

Specifikacija

Student polaže predmet projekat tako što osnovni projekat iz Primena veb programiranja u infrastrukturnim sistemima nadograđuje i to na **jedan od sledećih načina** (student bira koji način)**:**

1: Real-time takmičenje (Live Quiz Arena)

Proširiti KvizHub aplikaciju tako da podržava uživo takmičenje više korisnika istovremeno. Admin kreira "sobu", a korisnici pristupaju u određeno vreme. Sva pitanja se prikazuju sinhronizovano, a rang lista se prikazuje uživo na osnovu broja i brzine tačnih odgovora.

Tehnički zahevi:

- SignalR ili WebSocket komunikacija (server ↔ klijenti)
- Sistem za pokretanje i odbrojavanje pitanja
- Sinhronizovano prikazivanje pitanja svima u sobi
- Automatski prelazak na sljedeće pitanje
- Prikaz uživo rezultata takmičenja
- Bonus bodovi za brži odgovor

2: Mikroservisna arhitektura

Osnovni projekat uraditi kao mikroservisni sistem sa barem 2 mikroservisa. Ispred mikroservisnog sistema postaviti API gateway (preporuka je Ocelot) tako da se prednja strana njemu obraća a on rerutira zahteve ka individualnim mikroservisima. Mikroservisi ne smeju deliti istu bazu podataka. Logovati aktivnosti API gateway-a u tekstualne fajlove i na konzolu. Nije dozvoljeno da se prednja strana direktno obraća mikroservisima!

Odbrana projekta

Predmet projekat se može braniti isključivo u terminima odbrane osnovnog projekta iz Primena veb programiranja u infrastrukturnim sistemima (termini će biti objavljeni na eepsi sajtu).

Polaganje predmeta

Nakon što je uspešno odbranio praktični deo projekta studentu će biti saopšten broj bodova koji je dobio, nakon toga mora napisati dokumentaciju projekta i poslati asistentima zaduženim za predmet projekat (asistenti sa Primena veb programiranja u infrastrukturnim sistemima nemaju veze sa ovim delom).