Projektni zadatak iz predmeta

Internet softverske arhitekture

Računarstvo i automatika i Informacioni inženjering

generacija 2025/2026.

Jutjubić

Šta ako vam kažemo da najveći video-servis na svetu - možemo da napravimo bolji? Ne „ok“, ne „malo drugačiji“, nego bolji u svakom smislu. Šta ako postoji svet u kome:

❌ nema reklama,

❌ nema prekida,

❌ nema algoritamske manipulacije,

❌ nema „plati premium da ti ne smetamo“

…i gde je svaki piksel platforme tu da služi korisniku, a ne profitu?

Dobro došli u viziju Jutjubića - studentskog projekta koji ne pokušava da kopira YouTube, nego da ga prevaziđe. Ne pravimo klon. Pravimo izazivača. Pravimo platformu kakvu bismo svi voleli da postoji, ali nikad nije nastala, jer nikome nije bilo u interesu da je napravi. Mi nemamo investitore kojima treba zarada. Mi nemamo pritisak tržišta. Mi imamo nešto drugo: slobodu da napravimo savršeno iskustvo. Zbog toga Jutjubić ima samo jednu svetu, nepovredivu, nikad-prekršivu viziju: NEMA REKLAMA. IKADA. ZAUVEK.

Ovaj projekat je prilika da studenti razmišljaju kao pioniri, a ne kao korisnici tuđih pravila. Da osmisle streaming tehnologiju, skaliranje, preporuke, UI i UX — po svojim standardima. Da naprave digitalni prostor koji je iskren, čist, zabavan i potpuno oslobođen komercijalnih okova.

1. Namena sistema

U okviru projektnog zadatka potrebno je implementirati web aplikaciju koja predstavlja društvenu mrežu za deljenje video sadržaja - Jutjubić. Sem postavljanja i pregledanja videa, korisnici mogu i da lajkuju i postavljaju komentare, postave video u zakazanom režimu i koriste čet dok traje premijera videa.

2. Tipovi korisnika

Informacioni sistem razlikuje sledeće vrste korisnika:

- Neautentifikovani korisnici: imaju mogućnost da pregledaju video objave, komentare ispod objava, broj lajkova, kao i da izvrše registraciju i prijavu na sistem.
- Registrovani korisnik: korisnici aplikacije sa svim mogućnostima neregistrovanih korisnika, i imaju mogućnost lajkovanja i komentarisanja objava, postavljanja objava, kao i promena informacija svog profila.

3. Funkcionalni zahtevi

3.1. Prikaz informacija neautentifikovanim korisnicima

Korisnici koji nisu autentifikovani imaju prava pristupa stranici za registraciju i prijavu na sistem, i mogu da pregledaju objave korisnika automatski sortirane po vremenu nastanka (najnovije prvo), i da odu na stranicu pregleda profila korisnika koji je postavio objavu ili komentar. Nemaju mogućnost ostavljanja komentara ili lajkovanja objava, već će pri pokušaju biti obavešteni da moraju da se prijave kako bi dobili tu mogućnost. Nemaju svoj profil i nemaju prava pristupa ostalim podacima sistema. Za uspešnu implementaciju, potrebno je obezbediti zaštitu i na serverskoj i na klijentskoj strani.

3.2. Registracija korisnika i prijava na sistem

Neautentifikovanim korisnicima je uvek na stranici dostupan pristup stranicama za registraciju i prijavu. Prijava na sistem se izvršava pomoću korisnikove email adrese i lozinke. Registracija obuhvata unos email adrese, korisničkog imena, lozinke, imena, prezimena i adrese, gde je potrebno implementirati neophodne validacije i poruke greške korisniku. Lozinka se unosi u dva polja da bi se otežalo pravljenje grešaka prilikom odabira nove lozinke.
Nakon popunjavanja neophodnih podataka, na datu email adresu šalje se link za aktivaciju naloga. Korisnik ne može da se prijavi na aplikaciju dok se njegov nalog ne aktivira posećivanjem linka koji je dobio u emailu. Takođe, potrebno je ograničiti broj pokušaja prijave na sistem **na osnovu IP adrese**. Sa jedne IP adrese je dozvoljeno 5 pokušaja prijava po minuti.

Napomena: potrebno je obezbediti **bilo kakav** mehanizam za autentifikaciju i autorizaciju korisnika na serverskoj strani. **Za sve funkcionalnosti treba biti implementirana autorizacija bez obzira na podelu posla po studentima!**

3.3. Kreiranje video objave

Registrovani korisnici su u mogućnosti da postavljaju svoje objave. Za pravljenje objave su potrebni:

- naslov,
- opis videa
- tagovi koji bliže određuju sadržaj videa
- slika koja će predstavljati video (*thumbnail*),
- video u mp4 formatu max. veličine 200mb,
- vreme kreiranja objave (sistemsko).
- geografska lokacija (opciono)

Kreiranje objave mora da se implementira transakciono. Video sadržaj je potrebno čuvati lokalno na serveru. Ukoliko upload video sadržaja prođe neuspešno ili se ne završi u predviđenom vremenu (testirati ako upload traje predugo), potrebno je izvršiti kompletan *rollback* operacije. Novokreirana objava je dostupna drugim korisnicima za pregled, lajkovanje i komentarisanje. *Thumbnail* iz objave je potrebno keširati na backendu umesto da se svaki put čita sa file sistema.

Napomena: Za pravilnu implementaciju ove funkcionalnosti, potrebno je sliku i video uploadovati sa računara.

3.4. Transcoding video sadržaja

Prilikom kreiranja objave, potrebno je izvršiti *transcoding* video sadržaja. Za proces *transcoding-a*, potrebno je poslati referencu ka video sadržaju (npr. putanja na kojoj će potrošač moći da pristupi videu) zajedno sa parametrima u red poruka. Sam proces *transcoding*-a mora da obavlja minimalno 2 potrošača (*consumera*) čiji je zadatak da se pretplate na red poruka i obrade poruke kako one pristižu. Potrebno je sprečiti da se jedna poruka isporuči više puta i da bude obrađena od strane više potrošača. Potrošači implementiraju postupak transcodinga upotrebom alata poput FFmpeg. Parametri za *transcoding* mogu da budu predefinisani u okviru aplikacije.

3.5. Home page za korisnika

Na osnovnoj stranici za autentifikovanog korisnika dostupna je lista svih video objava sortiranih od najnovijih. Lista objava podrazumeva prikaz slike i naslova videa. Klikom na video, otvara se nova stranica na kojoj korisnik ima mogućnost pregledanja video sadržaja, komentara i lajkova. Na ovoj stranici korisnik može da ostavi svoj komentar i lajk.
Napomena: Studentima se prepušta organizacija stranica i izgled istih.

3.6. Postupak komentarisanja videa

Registrovanim korisnicima je dozvoljeno ostavljati komentare na svim video objavama. Komentar može da sadrži samo tekst. Komentar bi takođe trebalo da čuva vreme kreiranja i nalog koji ga je kreirao, i prikazuje se na objavi od najnovijeg do najstarijeg. Komentare je potrebno keširati. Velika količina komentara treba da se vrati kroz paginaciju.

Dodatno, potrebno je ograničiti broj komentara koje može da postavi jedna osoba sa svog profila u vremenskom intervalu od sat vremena. Broj dozvoljenih komentara po nalogu je 60 po satu. Jedan nalog može da komentariše različite objave, ali ukupan broj komentara koje može da napiše je 60 u sat vremena. **Ukupan broj komentara po objavi nije ograničen.** Za potrebe demonstriranja mehanizma, napisati skriptu ili jedninični test koji simulira slanje velikog broja komentara.

3.7. Brojač pregleda

Za svaki video je potrebno pratiti ukupan broj pregleda. Broj pregleda se uvećava svaki put kada korisnik uđe na stanicu za pregled videa. Brojač pregleda je potrebno implementirati tako da bude konzistentan u kontekstu istovremene posete istom videu. Za potrebe demonstriranja mehanizma, napisati skriptu ili jedninični test koji simulira istovremenu posetu istom videu od strane više korisnika i pravilan inkrement broja pregleda.

3.8. Prikaz videa u zakazanom režimu

Prilikom kreiranja video objave, korisnik ima opciju zakazanog prikaza, pri čemu je potrebno da unese datum i vreme za prikaz. Video nije dostupan drugim korisnicima na mreži sve do zakazanog datuma. U tačno zakazano vreme, video postaje dostupan za pregled u simuliranoj *streaming* varijanti, što znači da korisnik dobija deo po deo videa koji se prikazuju, i svi korisnici koji trenutno gledaju isti video, gledaju ga u istoj minutaži. Za simulaciju *streaming*-a, možete koristiti HLS/DASH ili možete ručno implementirati prikaz računajući trenutno vreme i *offset* videa. Na primer, ako je video zakazan za 08:00, svi korisnici koji pristupe videu u 08:03, gledaće video od trećeg minuta.

3.9. Periodična kompresija slika

Da bi aplikacija uštedela na masovnoj memoriji, potrebno je svaki dan pokrenuti akciju koja iterira kroz sve nekompresovane slike koje su starije od mesec dana i kompresuje ih. Biblioteku za kompresiju izaberite sami. Za potrebe projekta nemojte brisati nekompresovanu sliku. Slike se odnose na *thumbnail* koji se upload-uje prilikom kreiranja video objave.

3.10. Čet između korisnika tokom *streaming*-a

Potrebno je omogućiti grupni čet između korisnika u toku prikaza videa u *streaming* režimu. Razmena poruka u okviru četa treba da bude implementirana u realnom vremenu pomoću web socket tehnologije. Kada novi korisnik pristupi gledanju videa, automatski se dodaje u čet i može da šalje / čita poruke koje nastanu od tog trenutka. Nije potrebno pamtiti istoriju poruka.

3.11. Rad u klasteru

Aplikacija mora podržavati rad u klasteru **sa najmanje dve replike** (obe replike korise istu bazu), uz load balancer ispred API sloja i mehanizme za proveru rada replika (*health check*). Aplikacija mora ostati funkcionalna u sledećim slučajevima:

- pad jedne replike
- ponovno podizanje replike
- parcijalni gubitak konekcije prema MQ ili bazi podataka
- Dovoljno je pokazati primer pozivom proizvoljnog API-ja

3.12. Praćenje aktivnosti i performansi aplikacije

Kako bi se omogućio nesmetani rad aplikacije, potrebno je implementirati mehanizam nadgledanja (monitoring). Nadgledanje je potrebno implementirati pomoću alata Prometheus i Grafana. Uz pomoć ovih alata, vizuelno prikazati sledeće metrike:

- broj aktivnih i *idle* konekcija ka bazi u slučaju velikog opterećenja (npr. preko 200 zahteva u sekundi)
- prosečno zauzeće CPU u određenom vremenskom intervalu
- broj trenutno aktivnih korisnika u toku 24h

3.13. ETL pipeline za popularne videe

Potrebno je implementirati ETL (Extract-Transform-Load) pipeline u sklopu aplikacije. Pipeline će se pokretati jednom dnevno i generisaće listu popularnih videa na osnovu skorašnje aktivnosti svih korisnika.

- **Extract:** Potrebno je iščitati informacije o pregledima iz baze i/ili log fajlova.
- **Transform**: Grupisati događaje po videu i izračunati *popularity score* za svaki video. *Popularity score* se računa tako što se prebroje pregledi u poslednjih 7 dana i pregledi za svaki dan se pomnože sa odgovarajućom težinom, tako da pregledi od pre x dana množe sa težinom 7-x+1. Ovo znači da se broj pregleda od pre 7 dana množi sa težinom 1, a broj pregleda od prethodnog dana sa težinom 7.
- **Load**: Upisati rezultate u novu tabelu u bazi. Tabela čuva informacije o vremenu pokretanja pipeline-a, o tome koja su 3 najpopularnija videa kao i popularity score za svaki od njih.
- Dodatno, potrebno je prikazati svim ulogovanim korisnicima na početnoj stranici 3 najpopularnija videa dobijena prethodnim izvršavanjem ETL pipeline-a.

3.14. MQ JSON vs Protobuf

Potrebno je kreirati novu aplikaciju koja će dobijati poruku svaki put kada se objavi novi video na Jutjubić. Za slanje poruka se koristi message queue, pa je potrebno definisati *UploadEvent* strukturu koja će sadržati osnovne informacije o novokreiranom videu (naziv, veličina, autor…). Slanje poruke se vrši na dva načina. Potrebno je implementirati slanje poruke u JSON, kao i u Protobuf formatu. Zatim, potrebno je sprovesti poređenje između ova dva formata kao prosečno vreme serijalizacije, prosečno vreme deserijalizacije i veličine poruke. Poređenje izvršiti na bar 50 poruka.

3.15. Watch party

Ova funkcionalnost omogućava korisnicima da istovremeno gledaju iste video klipove u okviru watch party sobe (WP soba). Autentifikovan korisnik može da kreira sobu u koju korisnici mogu da se pridruže. Princip ulaska u sobu je ostavljen na vama (npr: Preko linka, invite ili da sobe budu javno dostupne na početnoj stranici). Kada kreator sobe pokrene video, automatski se svim ostalim članovim sobe otvara stranica sa istim. Nije potrebno sinhronizovati play/pause i trenutak videa već samo otvoriti stranicu videa koji je pušten kod vlasnika sobe u realnom vremenu. Koristite sockete za sinhronizaciju. Ovu funkcionalnost je neophodno testirati na dva računara, na jednom se kreira Watch Party, na drugom treba da se video automatski sinhronizuje.

3.16. Jutjubić na lokalu

Vaš zadatak za ovu temu je da omogućite pregled najpopularnijih videa u okolini - tzv. *trending* na lokalnom nivou. Ukoliko je potrebno, možete proširiti postojeći model podataka i integrisati *third-party* rešenja, poput API-ja za lokaciju ili alata optimizovanih za prostornu pretragu. Pre integracije gotovih rešenja sa vašim projektom, morate se konsultovati sa asistentom. Za uspešnu implementaciju, treba voditi računa o sledećim stavkama:

- [S1] Popularnost videa odrediti na osnovu više parametara (ne samo na osnovu broja pregleda kao u 3.13).
- [S2] Zatražiti lokaciju od korisnika, u slučaju da odbije, koristiti aproksimaciju lokacije sa koje je upućen zahtev.
- [S1] Radijus oblasti treba da bude konfigurabilan, a za pretragu sadržaja u blizini koristiti prostorno indeksiranje.
- [S2] Potrebno je pronaći i dokazati optimalnu meru između performansi i trendinga u realnom vremenu. Meru prikazati tabelarno ili grafički poredeći brzinu odziva u proizvoljnom periodu (*response time* ili *latency*).

- [S3] Rešenje testirati simulacijom zahteva iz više različitih oblasti (veliki broj aktivnosti se dešava na relativno malom prostoru vs. aktivnosti su distribuirane i dolaze iz različitih gradova).
- [S3] Razmisliti o sledećim problemima: - Koliko često je potrebno računati trending? Da li se trending razlikuje između korisnika koji žive u istoj ulici? Operacije potrebne za određivanje lokalnog trendinga ne smeju da narušavaju performanse osnovnih funkcionalnosti aplikacije.

3.17. Konzistentnost brojača na replikama uz pomoć CRDT struktura

Potrebno je implementirati podršku za održavanje konzistentnosti brojača pregleda na više replika. Da bi implementacija bila uspešno odrađena, potrebno je koristiti CRDT strukuturu G-counter.

- [S2] Da bi se uspešno moglo simulirati održavanje konzistentnosti, potrebno je prilagoditi replike tako da svaka replika čuva informaciju o broju pregleda u svojoj bazi ili svojoj tabeli podataka (ne mogu obe replike da čitaju iz iste tabele). Potrebno je pripremiti barem 2 replike.
- [S1] Potrebno je implementirati strukturu G-countera koju će koristiti svim replikama, kao i merge funkciju za spajanje vrednosti brojača na replikama. Struktura treba da bude implementirana tako da je moguće lako dodavanje ili smanjivanje broja replika (nigde u kodu nije zakucana vrednost na primer da su 2 replike u pitanju).
- [S3] Potrebno je implementirati komunikaciju između replika. Pored razmene poruka u kojima se dešava sinhronizacija brojača, potrebno je osmisliti kada će se slati poruke (kada klijent zatraži informaciju vrednosti brojača, periodično, nakon svake modifikacije…) i obrazložiti svoj izbor
- [S2] Potrebno je napisati skriptu koja šalje zahteve za izmenu brojača, tako da na svaku repliku dođe deo zahteva (moguće je iskoristiti load balancer da simulira ovo ponašanje, bitno je samo da ne idu svi zahtevi na jedu repliku).
- [S1] Potrebno je demonstritati slanje zahteva za vrednost brojača ka različitim replikama. Razmisliti da li je očekivano da klijenti vide istu vrednost brojača na svim replikama. U kojim slučajevima je to neophodno i potrebno, a kada je u redu da replike ne budu odmah sinhronizovane?
- [S1, S2, S3] Pronaći još neko mesto u projektu gde bi bilo moguće iskoristiti CRDT strukture za održavanje konzistentnosti (nije potrebno implementirati, samo obrazložiti i predložiti koji tačno tip CRTD struktura bi bio korišćen).

3.18. Napredna mapa video snimaka

Potrebno je implementirati naprednu mapu svih video snimaka sa geografskom lokacijom, sa posebno pažnjom na efikasnost i brzinu.

- [S1] U projekat integrisati odgovarajuća biblioteku, poput Leaflet-a ili OpenLayers-a za priakz mape.
- [S2] Omogućiti filter na osnovu vremenskog perioda (svo vreme, poslednjih 30 dana, tekuća godina)
- [S1] Dobavljati sa backenda samo one video klipove koji se nalaze u delu mape koji korisnik trenutno gleda (viewpoint). Za ovo koristiti princip “tiles” (mreža kvadratića) – mapa se deli u sekcije, i backend vraća samo snimke koji pripadaju sekcijama pokrivenim trenutno prikazanim delom mape, čime se smanjuje količina podataka i povećava šansa da se učitani podaci mogu keširati.
- [S2] Podaci za mapu (svaki tile) treba da se preračunavaju periodično, npr. noću, kada aplikacija nije intenzivno korišćena. Takođe ažurirati odmah samo tile u kom je novi video dodat, kako bi prikaz na mapi bio brz i ažuran.
- [S3] Problem: Kada korisnik zumira mapu skroz unazad (zoom-out), trenutni viewport može pokriti ceo kontinent i tada bi se svi snimci iz baze učitali, što je neefikasno. Rešenje: koristiti više nivoa tiles mreže sa različitim veličinama sekcija. Na najvećem zoom-u prikazivati male sekcije sa svim snimcima, na srednjem zoom-u koristiti veće sekcije koje grupišu snimke i smanjuju vizuelni nered, a pri maksimalnom zoom-out-u koristiti najveće sekcije gde se prikazuje samo jedan ili nekoliko reprezentativnih snimaka po sekciji, tako da se ne preklapaju vizuelno na mapi.
- [S1] Napraviti skriptu koja će u bazu ubaciti 5000 testnih snimaka (može se koristiti isti fajl za sve snimke kako ne bi bilo potrebno čuvati 5000 različitih fajlova), sa nasumičnim koordinatama raspoređenim približno po teritoriji Evrope, radi testiranja funkcionalnosti tiles mreže i učitavanja video klipova.

4. Nefunkcionalni zahtevi

4.1. Serverske platforme

Za realizaciju projekta može se izabrati serverska platforma po želji. Neke od platformi mogu biti:

- Java + Spring Boot (koristi se na vežbama)
- Java + Play framework
- Java + Spark framework
- NodeJS + Express
- Python + Django
- Ruby on Rails
- .NET

4.2. Klijentske platforme

Za realizaciju projekta može se izabrati klijentska platforma po želji:

- Klasična web aplikacija
- Single-page interface aplikacija (npr. Angular + REST servisi)
- Mobilna aplikacija (Android ili iOS)

Vizuelni izgled aplikacije aplikacije se ne ocenjuje direktno, ali lepši izgled svakako ostavlja bolji utisak.

4.3. Slanje e-maila

Za slanje emaila nije obezbeđen poseban servis. Možete koristiti sopstveni email nalog. Opciono, slanje notifikacija u vidu emaila možete da odradite korišćenjem message queue-a.

4.4. API dizajn

API glavne aplikacije treba da bude dizajniran i dokumentovan u skladu sa OpenAPI specifikacijom.

4.5. Keširanje podataka

Za potrebe keširanja podataka možete koristiti bilo koji od L2 nivoa keširanja, od kojih su neki bili pokazani na terminima vežbi.

4.6. Rate limiter

Za potrebe ograničavanja broja zahteva u jednici vremena studentu se prepušta odabir algoritma (token bucket, leaky bucket, sliding window) i implementacija. Za svaki API koji koristi rate limiter, **obavezno je dokazati da sistem ne degradira pod ekstremnim opterećenjem.**

4.7. Kompresija slika

Za implementaciju kompresije slika mogu se koristiti third party biblioteke za kompresiju slika.

4.8. Baza podataka

Podaci moraju trajno da se čuvaju u relacionoj bazi podataka po izboru. Potrebno je prepoznati i ukloniti bar jedan ORM anti-patern.

Jutjubić

5. Raspodela zadataka

Projekat nosi **60** poena.

Student 1:
- 3.2, 3.5, 3.8, 3.12, 3.13, POV

Student 2:
- 3.3, 3.7, 3.10, 3.11, 3.14, POV

Student 3:
- 3.1, 3.4, 3.6, 3.9, 3.15, POV

6. Kontrolne tačke

Kontrolna tačka 1 će se održati u januaru i nosiće 11 bodova. Potrebno je napraviti model celog sistema, kao i uraditi sledeće funkcionalnosti:

Student 1:
- 3.2
- 3.5

Student 2:
- 3.3
- 3.7 (bez transakcije)

Student 3:
- 3.1
- 3.6 (bez rate limitera),

Kontrolna tačka 2

Tačke 3.16, 3.17 i 3.18 predstavljaju teme za prezentacije (POV) koje će timovi izlagati u terminu vežbi. Svakom timu će nasumično biti dodeljena jedna tema, koju studenti rešavaju timski. Timska prezentacija nosi 15 poena i ujedno predstavlja kontrolnu tačku 2. Svaki tim treba da predstavi svoje rešenje na zadatu temu i objasni i pokaže kako su zahtevi iz teme implementirani u sklopu projekta. Prezentacija podrazumeva slajdove i implementacioni deo u projektu. Treba da traje okvirno 10 minuta. Svi članovi tima učestvuju u prezentaciji. Pošto su POV teme predviđene za timski rad, stari studenti koji rade samostalno projekat neće dobiti temu (samim tim neće imati ni drugu kontrolnu tačku).

Prezentacija obuhvata:
- uvod - ukratko objasniti zadatu temu
- metodologija - koji koncepti / mehanizmi / alati su korišteni za implementaciju

- demo - snimljen video materijal ili uživo demonstracija rešenja
- diskusija - kako implementirano rešenje odgovara traženim zahtevima

Za konačnu odbranu potrebno je implementirati sve ostale funkcionalnosti koje nisu pokrivene kontrolnim tačkama.

Timovi ili članovi tima koji ne budu izašli na prvu kontrolnu tačku, mogu da urade tražene funkcionalnosti, ali će se u tom slučaju osvojeni bodovi skalirati po formuli bodovi = broj_osvojenih_bodova * 0.8. Formula se odnosi samo na funkcionalnosti koje studenti pokažu na odbrani a koje su već prethodno tražene za kontrolnu tačku 1. Kontrolna tačka 2 ne može naknadno da se brani.

7. Akademska čestitost

Akademska čestitost podrazumeva samostalno pisanje radova (teksta, programskog koda i slično) uz striktno poštovanje tuđih autorskih prava. Ovaj pojam i obaveza su sastavni deo Zakona o visokom obrazovanju:

http://www.parlament.gov.rs/upload/archive/files/lat/pdf/predlozi_zakona/3048-14Lat.pdf

Više o ovoj temi možete pronaći, na primer, na sledećim linkovima:

- https://en.wikipedia.org/wiki/Academic_honor_code

U okviru našeg predmeta to podrazumeva da studenti samostalno pišu sopstveni rad. Pomoć drugih studenata u obliku direktno preuzetih delova teksta ili programa nije dozvoljena. **Kako se ovaj projekat radi u timu, odgovornost za nepridržavanje principa akademske čestitosti snose svi članovi tima.**

Sistemi za otkrivanje plagijarizma su vremenom postali prilično efektivni. Više informacija se može naći ovde: https://theory.stanford.edu/~aiken/moss/

Preduslovi za izlazak na kontrolne tačke i finalnu odbranu

1. Svaki repozitorijum mora dodati nalog **isa-asistent** kao kontributora.
2. Sve funkcionalnosti koje se prikazuju na kontrolnim tačkama i odbranama moraju biti dostupne u trenutnku odbrane na repozitorijumu projekta na *main* grani.
3. Za svaku odbranu i kontrolnu tačku obavezno je imati pripremljene podatke, skripte ili testove za demonstraciju urađenog.
4. Sve funkcionalnosti se prikazuju sa istog računara.

5. Komitovi napravljeni nakon predviđenog roka **neće biti uzeti u obzir** prilikom ocenjivanja.
6. Svaki član tima mora pokazati **snalaženje u kodu** i **razumevanje implementiranih mehanizama** kako bi ostvario maksimalan broj poena za funkcionalnost. Asistent zadržava pravo da odabere proizvoljan segment projekta na osnovu kog će proveriti razumevanje implementiranog rešenja.

Za neispunjavanje uslova [1-5] tim automatski gubi 5 poena.

8. Skaliranje ocena

Projekat je predispitna obaveza čija se izrada podrazumeva u toku semestra, a odbrana na kraju. U slučaju da studenti ne uspeju u toku semestra da završe izradu projekta i uspešno odbrane urađeno, u junu i septembru će biti organizovan po dodatni termin za koji će bodovi biti skalirani na sledeći način:

- januarski rok - 100% bodova
- junski rok - 90% bodova
- septembarski rok - 80% bodova

9. Projekat za stare studente

Ukoliko stari studenti samostalno rade projekat, potrebno je odraditi sledeće funkcionalnosti:

- 3.1, 3.2, 3.3, 3.6, 3.7, 3.14, 3.15

Za kontrolnu tačku 1 je potrebno uraditi:

- 3.1, 3.2,

Stari studenti koji samostalno rade projekat nemaju kontrolnu tačku 2, već će sve preostale funcionalnosti pokazati na finalnoj odbrani.