Arhitektura i projektovanje softvera

Naziv projekta:

Gems & Crystals

Clanovi tima:
Kristijan Ilic 16103
Milos Stoiljkovic

Kontekst i cilj projekta:

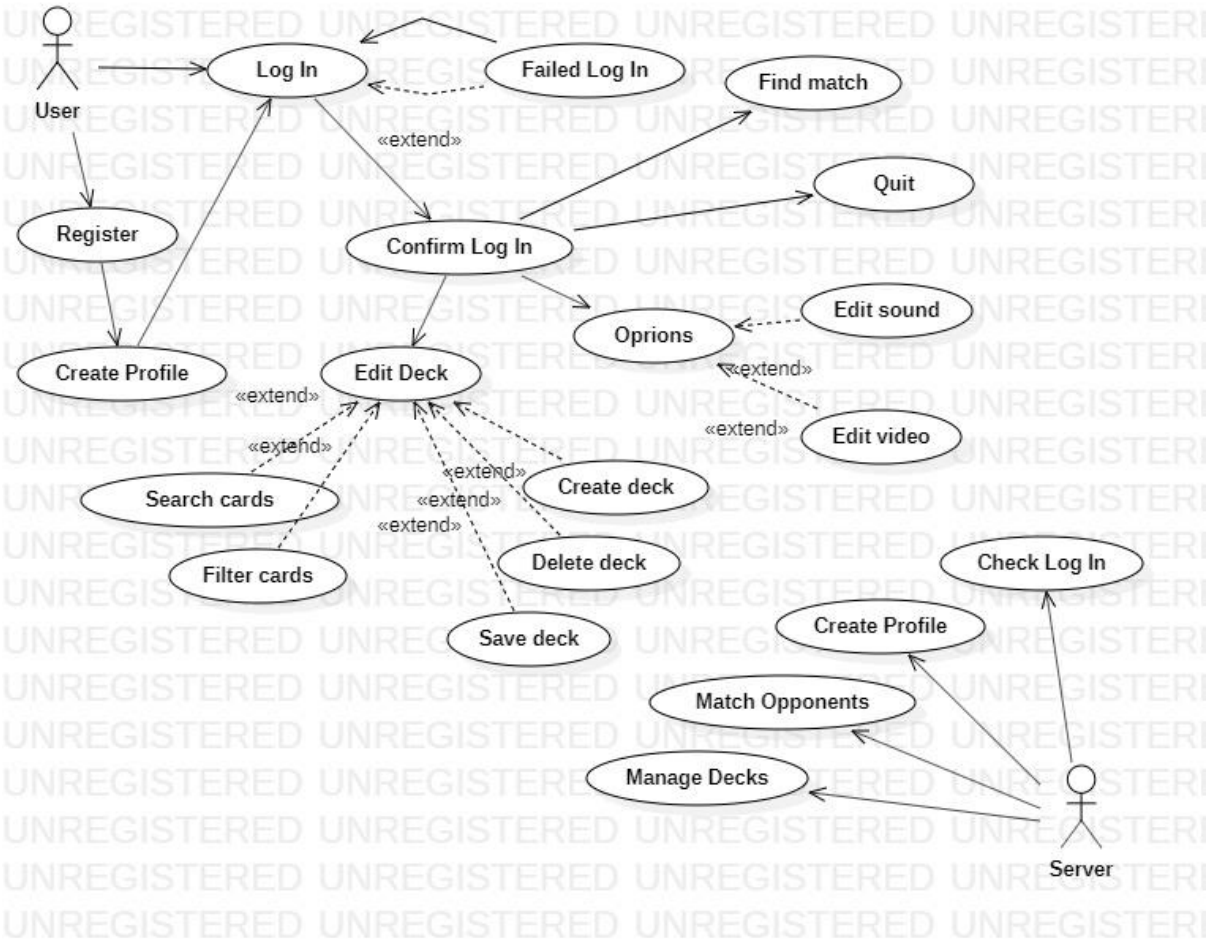
Cilj projekta je izrada aplikacije koja ce omoguciti online multiplayer, turn based, kartasku igricu. Aplikacija je tipa client-server. Aplikacija mora biti interkativna, sa preglednim i lakim za koriscenje interfejsom. Ova igrica predstavlja 1v1 tip igre, gde oba igraca poseduju spil odredjenih karata, odredjeni broj zivotnih poena, gde je cilj spustiti protivnikove zivotne poene na nulu. Pored mesta za spil, na tabli za igranje postoje mesta za karte cudovista, magicne karte, groblja i neke posebne vrste karata. Postoji mesto koje opisuje trenutnu highlight-ovanu kartu, njeno znacenje i funkcionalnost, prostor za smenu poteza i njegovih faza, kao i mesto za avatar igraca sa njegovim zivotnim poenima. Naravno pri ulazu u igru postoji game menu odakle se baratati osvnovnim podesavanjima igre, spilovima, i odakle se pristupa samoj partiji sa nasumicnim ili zeljenim protivnikom. Projekat se smatra uspesnim ako u svi arhitekturni funkcionalni i nefunkcionalni zahtevi zadovoljeni.

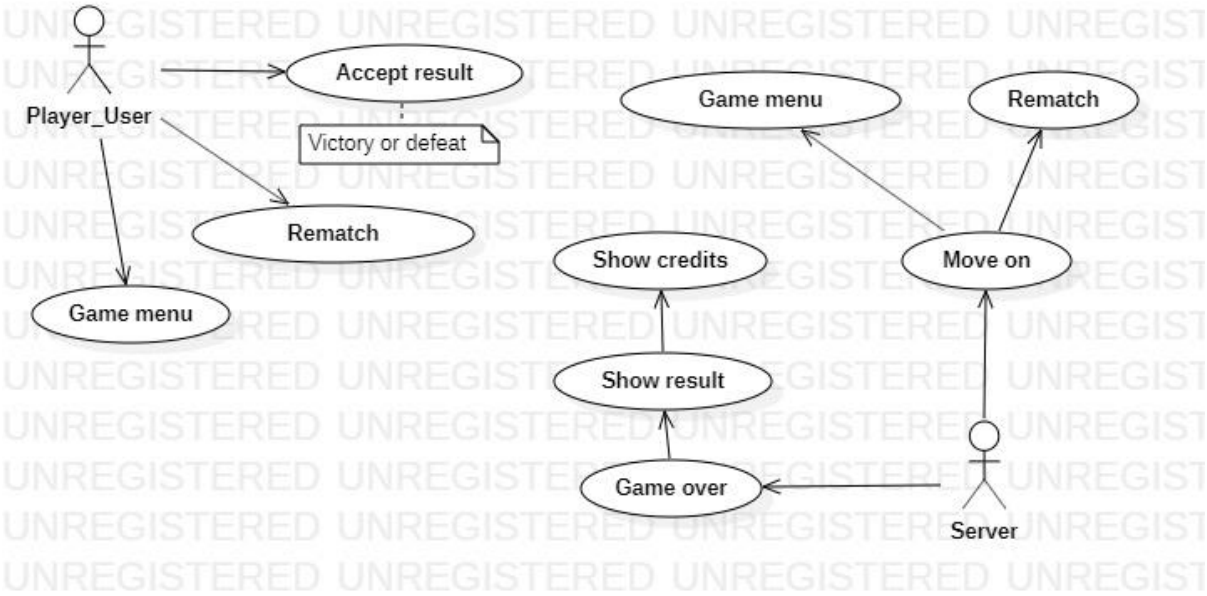
Arhitekturni (funkcionalni) zahtevi:

1. **Kreiranje (registracija) korisnika –** Da bi se korisnik prijavio na aplikaciju, potrebno je da poseduje svoj nalog. Ako ne poseduje, potrebno je kreirati isti. Omoguciti opciju za kreiranje naloga na pocetnoj strani i kreiranje istog u bazi.
2. **Prijavljivanje –** Potrebno je omoguciti da se korisnik prijavi na aplikaciju koriscenjem korisnickog imena i lozinke. Omoguciti prijavu greske ako su podaci nevalidni.
3. **Meni –** Omoguciti korisniku pristup meniju. Meni treba da sadrzi opcije za pocetak igre, trazenja protivnika, sastavljanje svog spila, pristup osvnovnim opcijama aplikacije (zvuk, video...), opcija za izlaz iz aplikacije.
4. **Pocetak igre –** Na pocetku igre, potrebno je korisniku omoguciti prikaz table i pocetni set karata u njegovoj ruci, takodje prikazati karte protivnika (neotkrivene) i avatar igraca i njihove zivotne poene.
5. **Odabir igraca koji pocinje igru -** Server nasumicno postavlja ko igra prvi, a ko drugi.
6. **Potez –** Nakon pocetka igre, omoguciti smenu poteza, igraci igraju jedan za drugim. Svaki potez se sastoji iz 3 faze. Omoguciti smenu kroz faze poteza. Nakon zavrsetka trece faze poteza, igrac zavrsava svoj potez a protivnik pocinje svoj.
7. **Potez: Faza 1 –** U prvoj fazi omoguciti odigravanje karata, kako cudovista, tako i magicnih. U sustini ova faza sluzi za neku pripremu.
8. **Potez: Faza 2 –** U drugoj fazi omoguciti napad protivnika. Igrac moze da koristi svoje karte cudovista i magicne karte ali ne moze da odigra (stavi na talon) nove u ovoj fazi, to nije validan potez.
9. **Potez: Faza 3 –** U trecoj fazi omogucava se ponovo odigravanje karata (cudovista i magicnih) ali mu se zabranjuje napad na protivnika. Faza sluzi da igrac obezbedi neku odbranu pre pocetka protivnikovog poteza.
10. **Odigrana karta –** Omoguciti prosledjivanje informacija o odigranoj karti serveru. Na osnovu odigrane karte menja se trenutni pogled igre/table/karata u ruci i salju se te informacije serveru.

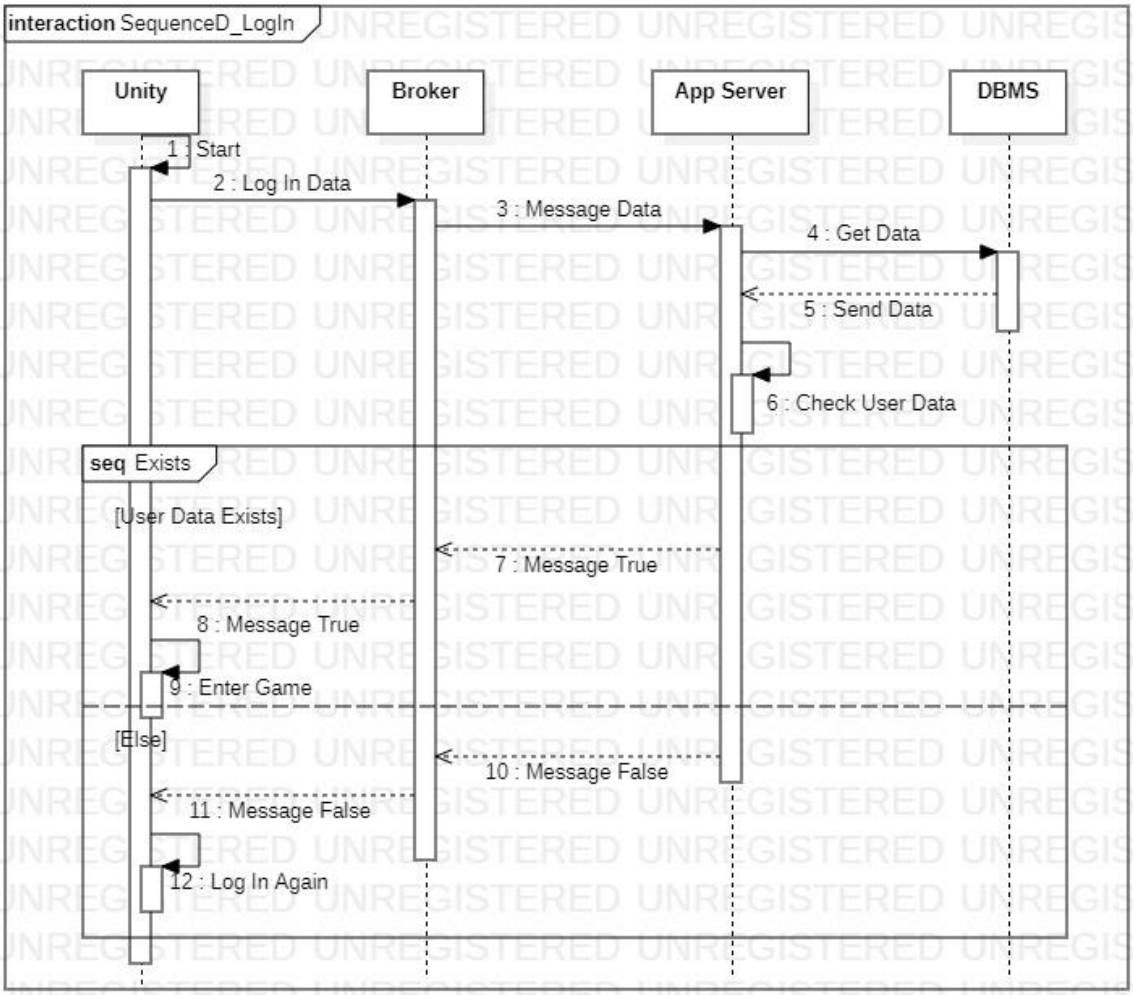
11. **Protivnikov potez –** Omoguciti prikaz u realnom vremenu kada je u pitanju odigran protivnikov potez. Nakon sto igrac odigra svoj potez i njegov view se promeni, potrebno je odredjene informacije koji opisuju tu akciju proslediti protivniku i promeniti njegov view da se prikazuje tacna tabla.
12. **Validacija odigrane karte –** Omoguciti validaciju odigrane karte. Svaka karta koju izabere da odigra mora da bude moguca za igru u tom trenutku. U slucaju da potez nije validan nista se nece desiti. Ako je potez validan, pokrece se akcija odigrane karte.

Na sledecim dijagramima su prikazane odredjene funkcionalnosti aplikacije. Predstavljena su 3 **use-case** dijagrama koja respektivno prikazuju odredjene slucajeve koriscenja pre/tokom/nakon jedne igre.

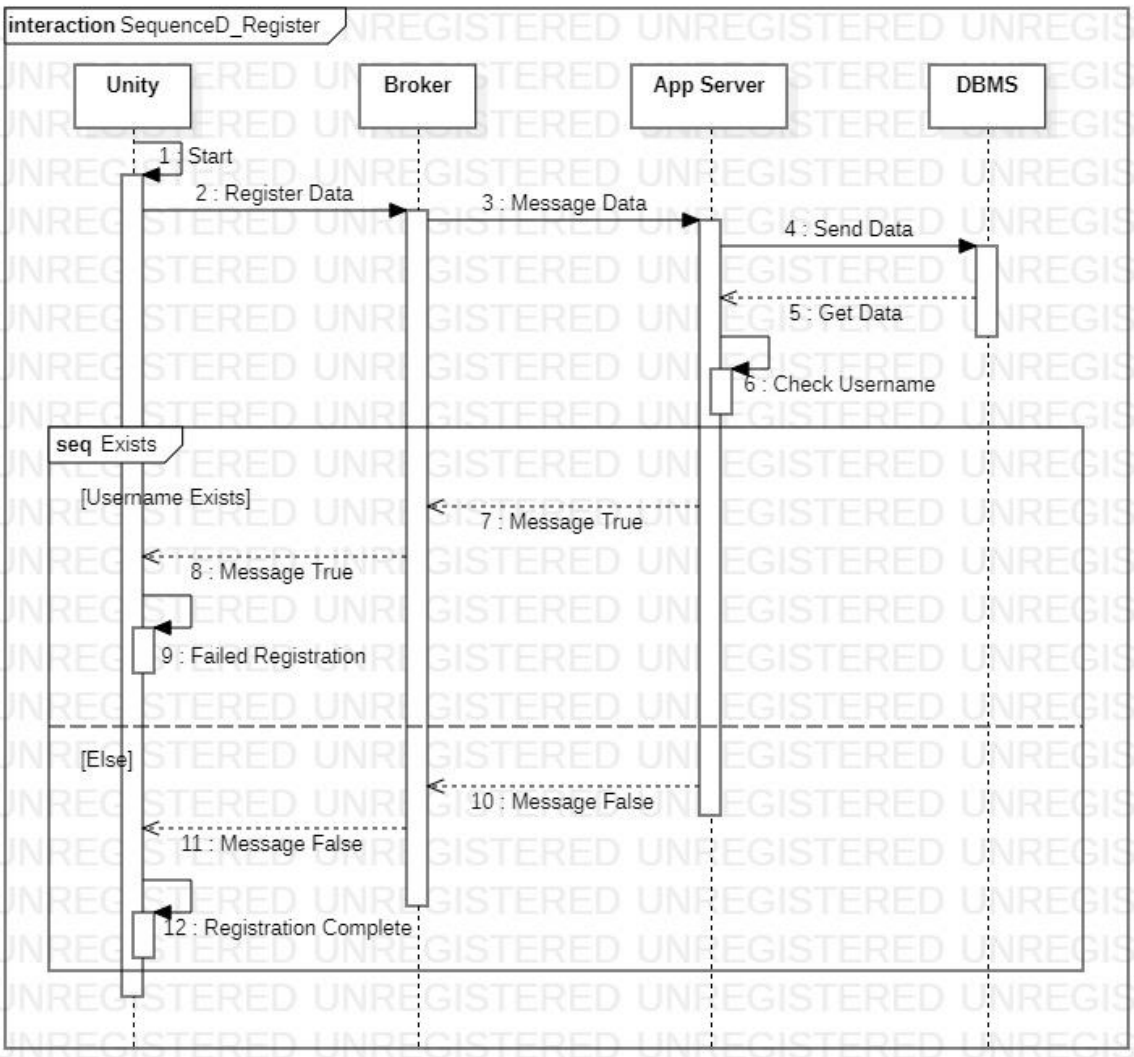


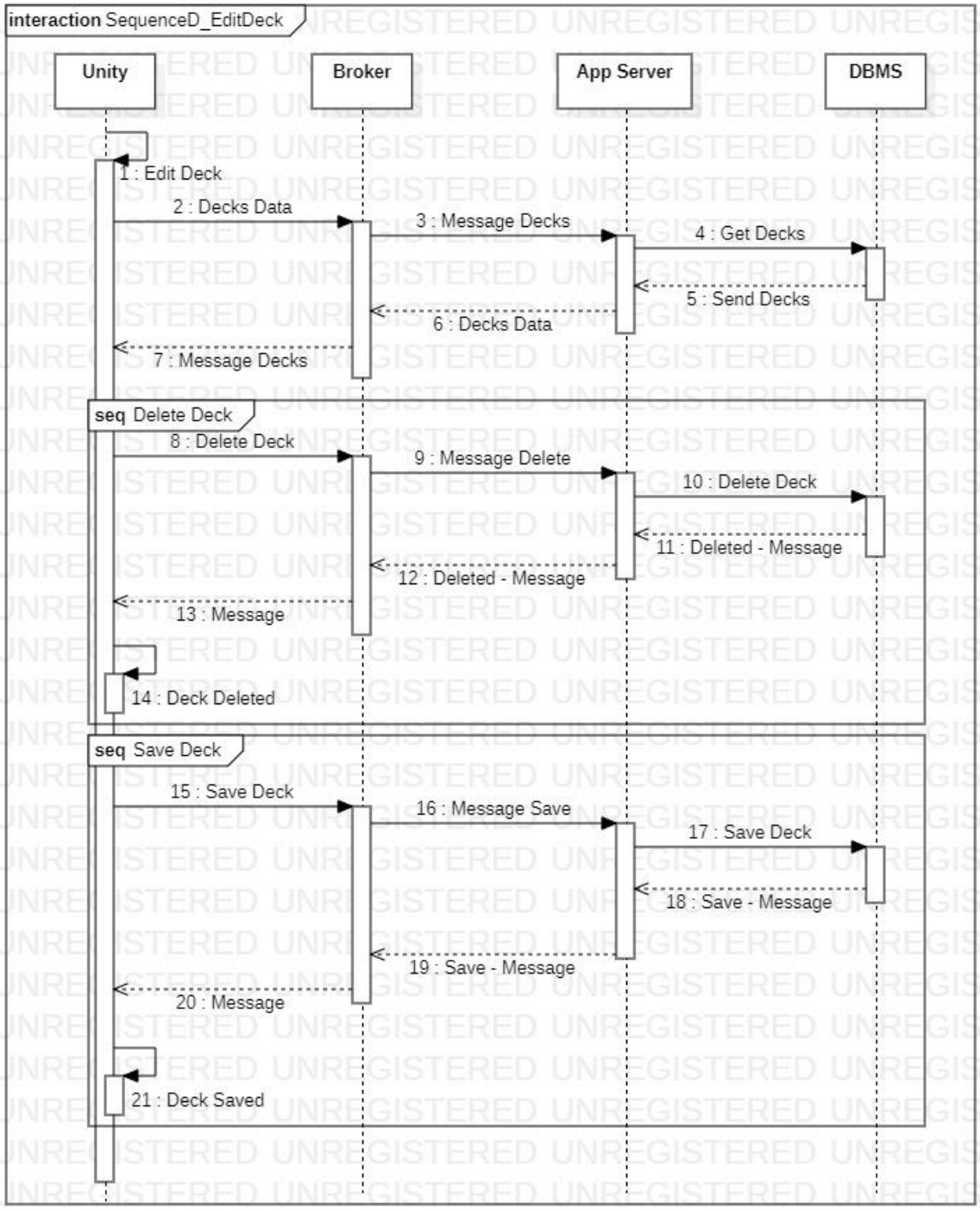


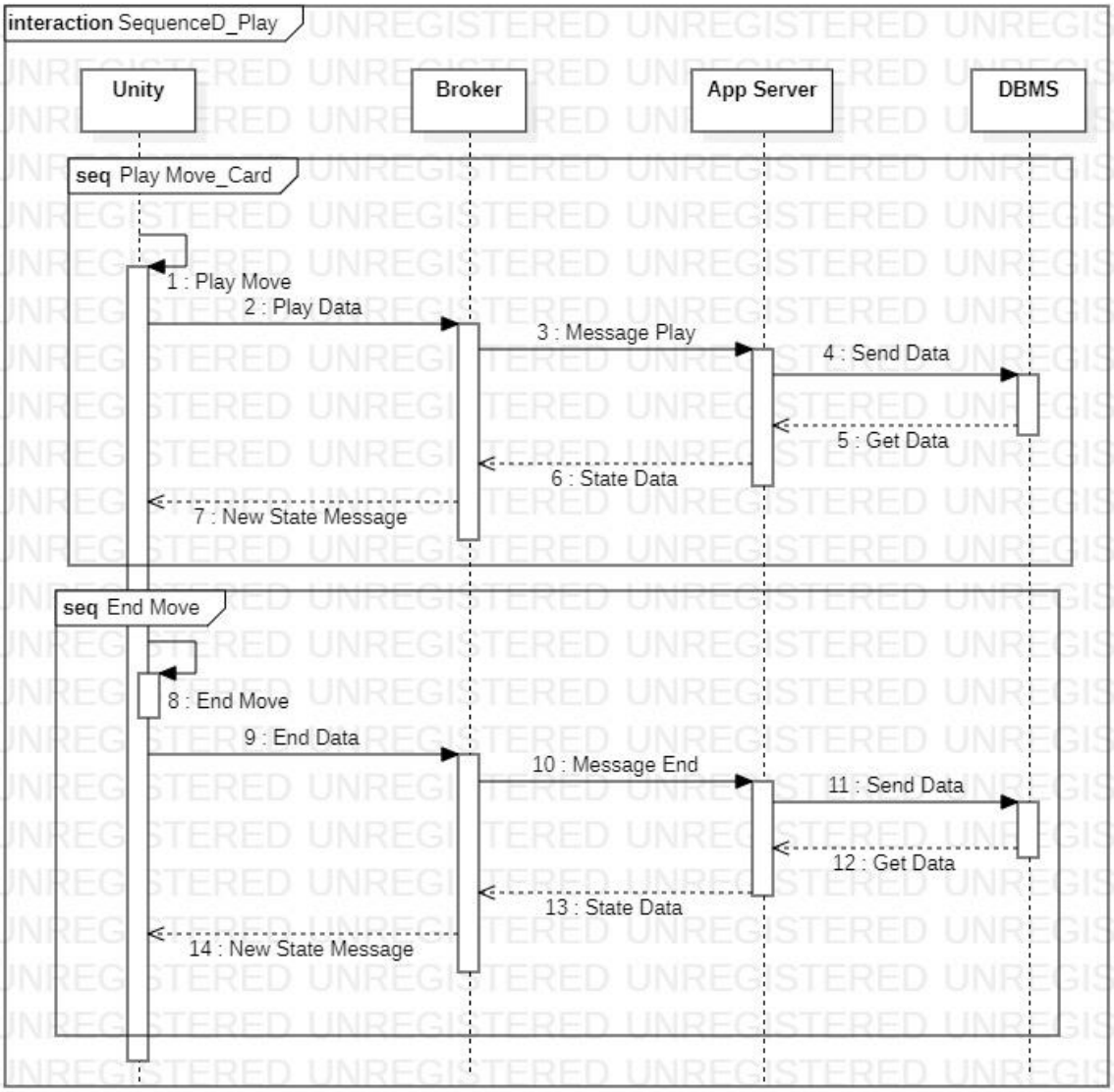
Na sledecim dijagramima su prikazane odredjene funkcionalnosti aplikacije koje prikazuju povezanost klijenta i servera na osvnou odredjenih radnji. Predstavljena su 5 **sequence** dijagrama (dijagram sekvenci) za LogIn, Register, EditDeck, Play, Rematch, respektivno

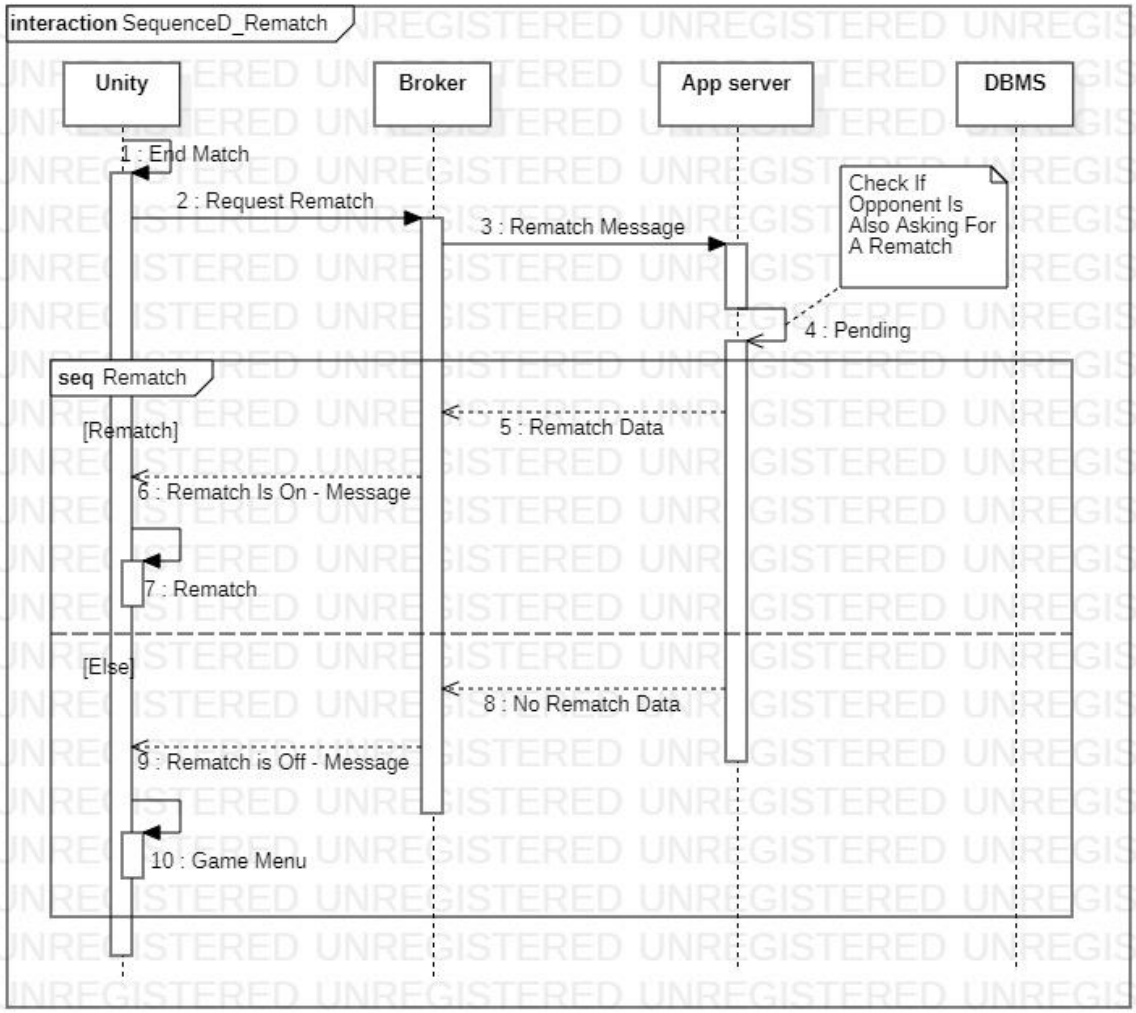

interaction SequenceD_LogIn
Unity
Broker
App Server
DBMS
1 : Start
2 : Log In Data
3 : Message Data
4 : Get Data
5 : Send Data
6 : Check User Data
seq Exists
[User Data Exists]
7 : Message True
8 : Message True
9 : Enter Game
[Else]
10 : Message False
11 : Message False
12 : Log In Again

.


interaction SequenceD_Register
Unity
Broker
App Server
DBMS
1 : Start
2 : Register Data
3 : Message Data
4 : Send Data
5 : Get Data
6 : Check Username
seq Exists
[Username Exists]
7 : Message True
8 : Message True
9 : Failed Registration
[Else]
10 : Message False
11 : Message False
12 : Registration Complete


interaction SequenceD_EditDeck
Unity
Broker
App Server
DBMS
1 : Edit Deck
2 : Decks Data
3 : Message Decks
4 : Get Decks
5 : Send Decks
6 : Decks Data
7 : Message Decks
seq Delete Deck
8 : Delete Deck
9 : Message Delete
10 : Delete Deck
11 : Deleted - Message
12 : Deleted - Message
13 : Message
14 : Deck Deleted
seq Save Deck
15 : Save Deck
16 : Message Save
17 : Save Deck
18 : Save - Message
19 : Save - Message
20 : Message
21 : Deck Saved


interaction SequenceD_Play
Unity
Broker
App Server
DBMS
seq Play Move_Card
1 : Play Move
2 : Play Data
3 : Message Play
4 : Send Data
5 : Get Data
6 : State Data
7 : New State Message
seq End Move
8 : End Move
9 : End Data
10 : Message End
11 : Send Data
12 : Get Data
13 : State Data
14 : New State Message



Nefunkcionalni zahtevi:

1. Obezbediti odaziv nakon igranja poteza ne vise od 1.0s.
2. Obezbediti intuitivan, pregledan i prost interfejs.
3. Obezbediti pouzdanu konekciju sa serverom tokom igre.
4. Obezbediti jednostavno dodavanje novih karata u spil.
5. Obezbediti dostupnost sistema (24x7)

Tehnologije:

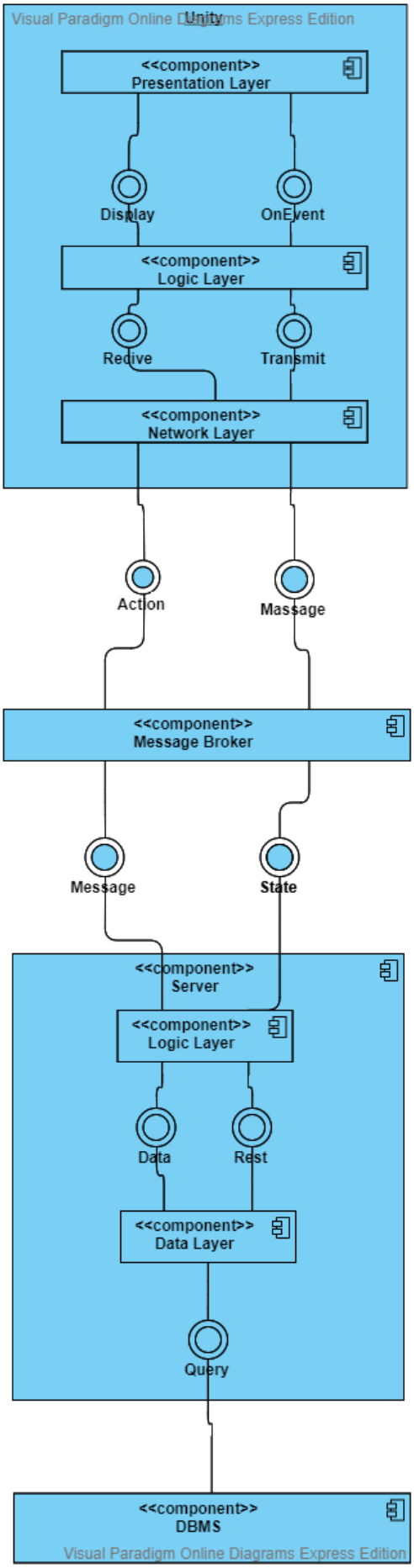
Za klijentsku stranu ce se koristiti unity game engine sa svojim C# programskim jezikom. Za realizaciju aplikacionog servera bice koriscen .Net framework, za komunikaciju koristice se NATS Message Broker i za rad sa bazom Entity framework. Baza podataka ce biti MySQL ili Microsoft SQL.

Arhitekturni dizajn:

Aplikacija implementira Layered arhitekturu. Viseslojni klijent-server obrazac koji se sastoji od 4 glavnih komponenti i njihovih podslojeva:

1. Unity – Kreira klijentsku stranu, njen pregled i komunicira sa brokerom.
   1.1. Presentation layer – Prezentacioni sloj koji oznacava sta se prikazuje klijentu
   1.2. Logic layer – Predstavlja odredjenu logiku komponenti i vezu sa pogleda sa mrezom.
   1.3. Network layer – Predstavlja vezu klijentske strane sa brokerom i odredjene informacije sa serverske strane.
2. Message Broker – Predstavlja sloj koji prevodi poruke i prosledjuje ih od klijentske do servrske strane i obrnuto.
3. Server –
   3.1. Logic layer – Predstavlja aplikacionu logiku i povezanost sa brokerom.
   3.2. Data layer – Predstavlja sloj koji je zasluzan za vezu sa bazom podataka i prikupljanje istih.
4. DBMS – Predstavlja sloj gde se cuvaju sv podaci vezani za igru i njene korisnike.

Na narednoj slici je predstavljen arhitekturni dizajn preko dijagrama komponenti:


Visual Paradigm Online Diagrams Express Edition
Unity
<<component>>
Presentation Layer
Display
OnEvent
<<component>>
Logic Layer
Redive
Transmit
<<component>>
Network Layer
Action
Massage
<<component>>
Message Broker
Message
State
<<component>>
Server
<<component>>
Logic Layer
Data
Rest
<<component>>
Data Layer
Query
<<component>>
DBMS
Visual Paradigm Online Diagrams Express Edition

Klijentska strana i MVC:

Sto se tice klijentske strane, kao sto je receno, koristice se Unity game engine. On sam ima implementiran MVC obrazac, tako da taj aplikacioni okvir ce biti koriscen za implementaciju klijentske komponente sistema. U prilogu slika sa MVC-om u Unity-ju.