УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

ЗАВРШНИ РАД

Развој друштвене мреже применом ASP.NET Core и SignalR оквира

Ментор:

Проф. др. Саша Лазаревић

Студент:

Кристина Станисављевић
2018/0361

Београд, 2022. године

Садржај

1. Увод

Са развојем технологије и доступности интернета свуда и свима у свету, Web апликације почињу да заузимају водећу улогу у међу апликација. Коришћењем Web апликација врло лако можемо да добијемо потребне информације. Велика предност Web апликација у односу на традиционалне desktop апликације је то што се Web апликације најчешће покрећу уз помоћ web browser-а (претраживача) као што су Chome, Firefox, Microsoft Edge и још много других. Још једна велика предност која последично долази из претходне је чињеница да овакве апликације раде на свим платформама, потпуно независно од оперативног система.

Комуникација преко интернета данас је једна од основих принципа друштва. Са развојем друштвених мрежа као што су Facebook или Instagram људи су навикли да им пријатељи и познаници буду врло лако доступни, у сваком тренутку могу да им пошаљу поруку, виде њихове приватне слике, размењују коментаре и још много тога. У складу са тим у наставку овог рада биће приказан развој једне друштвене мреже као Web апликације.

Апликација је реализована коришћењем ASP.NET Core-а и Entity Framework Core-а, уз помоћ којих је имплементирана серверска страна апликације, и Angular-а и Bootstrap-а за имплементацију клијентске стране. Подаци се чувају у релационој бази MySQL. Коришћен је програмски језик C# и развојно окружење Visual Studio 2022.

На почетку самог рада, у првих шест поглавља, биће детаљније објашњене технологије коришћене за израду апликације. У наредних пет поглавља биће објашњене фазе Ларманове методе: фаза прикупљања података, фаза анализе, фаза пројектовања, фаза имплементације и фаза тестирања. На самом крају представљен је закључак и коришћена литература.

2. .NET платформа

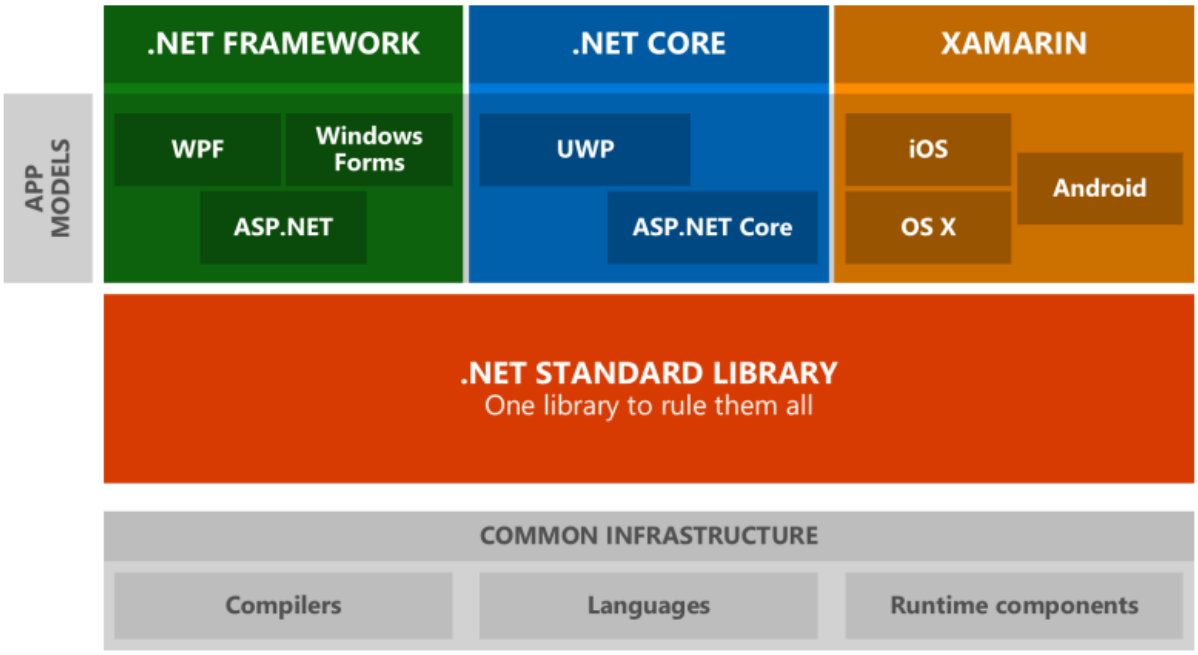
.NET платформа је open-source окружење (платформа отвореног кода) коју је развио Microsoft за креирање Web апликација, Web API-ја, мобилних и конзолних апликација. Развој је почео раних 1990-тих, док је прва верзија изашла 13. фебруара 2002. године. Платформа је настала на стандардима XML и SOAP технологија и омогућава да више различитих Web сервиса, који имају различите имплементације, раде заједно.

.NET апликације се могу написати помоћу програмског језика C#, F# или Visual Basic-а. [2] Програмски код се може покренути на било ком компатибилном систему.

- C# - једноставан, модеран, објектно оријентисан и безбедан програмски језик.
- F# - функционални програмски језик који олакшава писање сажетог, робусног и ефикасног кода.
- Visual Basic - приступачан језик са једноставном синтаксом за прављење апликација које су објектно оријентисане.

Постоје три Micorosft платформе које омогућавају развој апликација: [5]

- .NET Framework
- .NET Core
- Xamarin



Слика 1. Приказ .Net платформе
dotnet-tomorrow.png (1581×853) (microsoft.com)

2.1. .NET Framework

Представља оригиналну имплементацију .NET платформе. .NET апликације могу да се покрећу на различитим оперативним системима коришћењем различитих .NET имплементација, док се .NET Framework користи за покретање апликација на Windows оперативном систему. Подржава покретање Web сајтова, сервиса, десктоп апликација и још много тога на Windows-у.

.NET Framework обухвата:

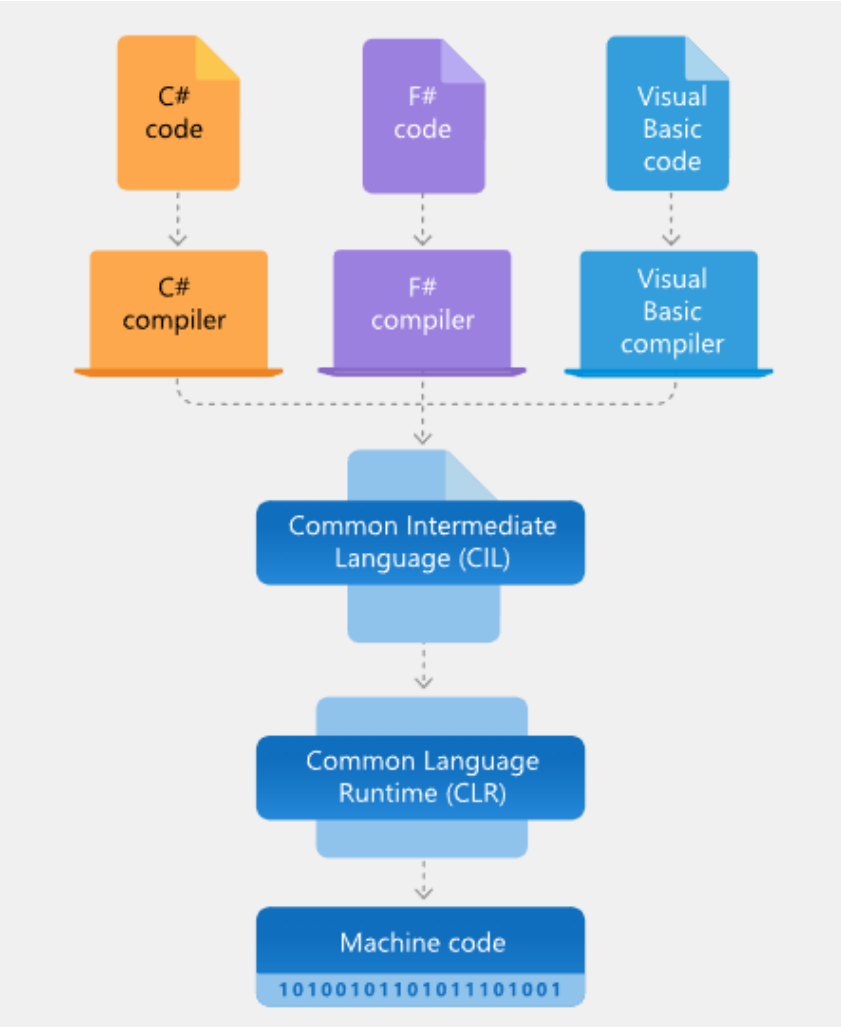
- .NET језике – C#, F#, Visual Basic, J# (Java klon), C++, Jscript .Net
- CLR - Common Language Runtime
- .NET framework библиотеке класа
- ASP.NET – механизам који подржава Web апликације креиране у .NET систему
- Visual Studio - развојно окружење

Две главне компоненте .NET Framework-a су Common Language Runtime и .NET Framework библиотека класа.

CLR представља имплememтацију стандарда заједничке језичке инфраструктуре. Представља механизам у којем се извршавају сви .NET програми. Пружа могућности као што су управљање нитима, руковање изузецима, *garbage-collection* (аутоматски ослобађа ресурсе, односно меморију, који су заузети од стране некоришћених објеката) и још много тога.

.NET Framework библиотеке класа садрже велики број унапред креираних и тестираних функционалности и типова за вишекратну употребу које програмери могу да позову из сопствених апликација.

Код .NET апликација написаних у C#, F# или Visual Basic-у, код се у току компајлирања преводи у Common Intermediate Language (CIL) и такав код чува се у фајлу који се назива асембли – фајл са .ddl или .exe екстензијама. Када се апликација покрене, CLR преузима асембли и путем компајлера Just-In-Time преводи га у машински код који може да се изврши на специфичној архитектури рачунара на којем се ради.



Слика 2. Извршавање кода у програмском језику C#
https://dotnet.microsoft.com/static/images/illustrations/swimlane-architecture-framework.svg?v=ZuTW7j9pS1oiuMqx3E-Xvb9OEM_8ajDEcHbecyjRtLA

2.2. .NET Core

.NET Core је скуп компоненти, библиотека, као и компајлера који омогућавају извршавање различитих задатака на различитим оперативним системима. .NET Core и .NET (уопштено) имају однос ,,подскуп - надскуп". .NET Core је назван „језгро“ зато што садржи функције језгра из .NET Framework-а и за извршавање и у библиотекама радних оквира. За разлику од .NET Framework-а, .NET Core се односи на раздвајање .NET-а од Windows-а, омогућавајући му да ради у окружењима која нису у Windows-у.

Састоји се од Common Language Runtime (CLR), који се у .NET Core-у назива CoreCLR. .NET Core такође има обимну библиотеку класа. Међутим, уместо једне библиотеке класа он садржи CoreFX, модуларну колекцију библиотека.

Слика 3. Поређење .NET платформи

.NET Core, .NET Framework, Xamarin – The “WHAT and WHEN to use it” - Cesar de la Torre (microsoft.com)

2.3. ASP.Net Core

ASP.NET Core је бесплатан оквир отвореног кода (енг. *open source*) намењен за израду динамичких Web апликација на Windows, Mac или Linux оперативном систему.

Као и .NET Core може да се извршава на више платформи и омогућава израду Web апликација, Web сервиса и Web страница, Internet Of Things апликација и бекенд делова апликација. [6]

Предности коришћења овог оквиру су:

- апликације могу да се покрену на различитим платформама, не само на Windows оперативном систему,
- већа брзина и боље перформансе,
- могућност лакшег одржавања и
- већа сигурност.

Подразумевана архитектура је MVC архитектура која обухвата три компоненте, а то су модел, поглед и контролер, али пружа могућност и за креирање Web API-ја који ће детаљније бити објашњени у поглављу 3.

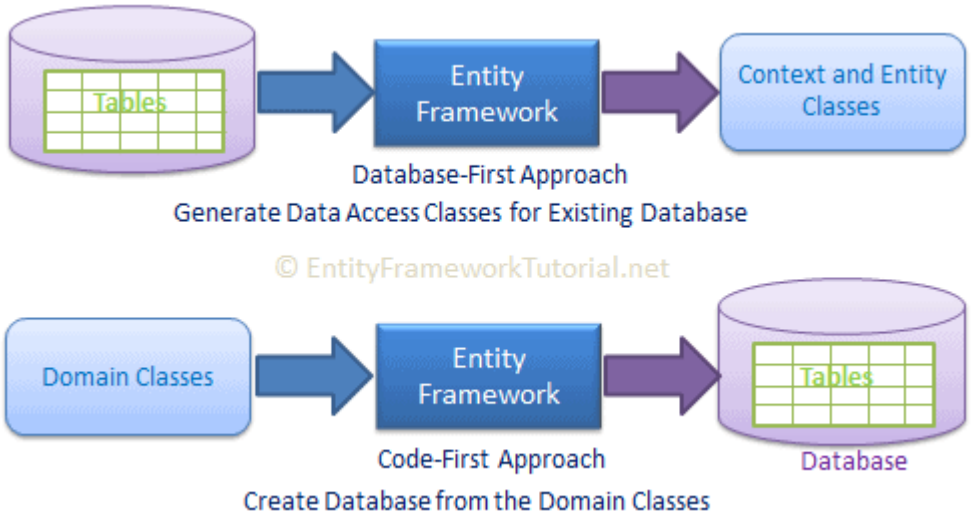
2.4. Entity Framework Core

Entity Framework Core представља објектно-релациони мапер. Пружа могућност да се подаци користе у форми објеката и одговарајућих карактеристика, уместо да се директно ради са табелама и колонама у бази. Једна од великих предности Entity Framework Core-а јесте његова способност да велики део кода сам генерише, чиме се значајно штеди време израде апликације.

Приступ подацима се врши помоћу модела. Модел се састоји од класа ентитета и контекстног објекта који представља сесију за рад са базом. Објекат контекста омогућава постављање упита и чување података. [3]

Entity Framework Core омогућава неколико начина да се уради мапирање, а то је: [4]

- Генерисање модела из постојеће базе података. (*Database-First Approach*)
- Ручно прављење модела у програму на основу којег ће се генерисати табеле у бази. Да би се креирале или измениле табеле користе се миграције. (*Code-First Approach*)



Слика 4. Приступи објектно-релационог мапирања
Entity Framework Core Tutorials (entityframeworktutorial.net)

2.4.1. LINQ

Language-Integrated Query (LINQ) – Упит интегрисан у језик - представља језик структуираних упита који омогућавају претраживање локалних колекција објекта, као што су на пример листе, али и удаљених извора података (база), без нарушавања безбедности података. Он унифицира начин приступа и претраге података из објекта који имплементира *IEnumerable<T>* интерфејс. LINQ поседује богату колекцију наредби за имплементацију комплексних упита које садрже агрегатне функције, сортирање и доста још тога. [7]

LINQ породица технологија обезбеђује коришћење упита за објекте (*Linq to Objects*), релационе базе података (*Linq to SQL*) и XML (*Linq to XML*).

Linq to Entities претвара LINQ упите у упите стабла команди које се извршавају над доменским моделом и враћају као резултат објекте који могу даље у програму да се користе. Упити се пишу над објектом *Context* који враћа податке из базе, а не директно над подацима из базе.

Неки од најчешће коришћеним LINQ упита:

- *SingleOrDefault* – враћа један елемент који задовољава прослеђени услов. Уколико такав елемент не постоји враћа *default* (подразумевану) вредност за тип тог елемента.
- *Where* – враћа све елементе који задовољавају прослеђени услов.
- *OrderBy* – враћа сортирану листу елемената преко прослеђеном критеријуму.

Коришћење LINQ упита олакшава писање кода због своје једноставне синтаксе. У наставку приказан је пример како код изгледа са и без LINQ упита.

Потребно је пронаћи све објаве које су поставили људи које тренутно улоговани корисник прати. Неопходно је проћи кроз све људе које корисник прати и за сваког проћи кроз све објаве и проверити да ли је ид корисника једнак ид-ју корисника који је поставио објаву. Наравно, под претпоставком да су пре тога сви људи које корисник прати и све објаве већ учитане.

```
foreach(User u in user.Following)
{
      foreach(Post p in allPosts)
      {
         if(p.UserId==u.Id)
         {
              posts.Add(p);
         }
      }
}
```

Уместо да се два пута пролази кроз *for* петљу и проверава услов, довољно је искористити LINQ упит *Where* који ће пронаћи све објаве које задовољавају задати услов, без потребе да пре тога учитамо све објаве из базе.

```
user.Following.ForEach(u =>
{
      posts.AddRange(context.Posts.Where(p => p.UserId == u.Id));
});
```

3. SignalR

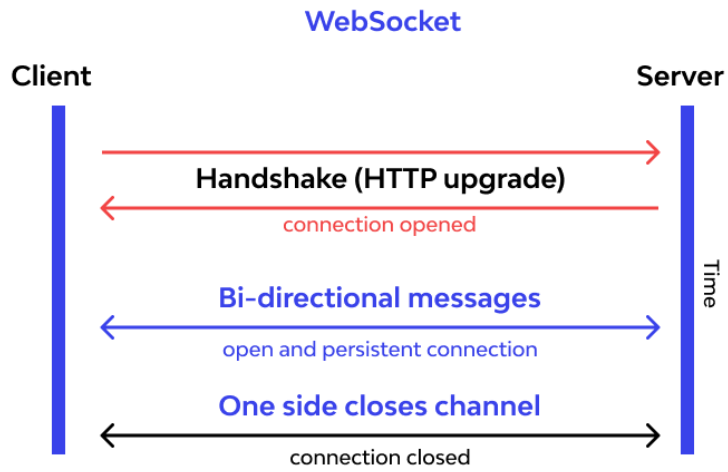
SignalR је софтверска библиотека отвореног кода за Microsoft ASP.NET која поједностављује процес додавања *real-time* (у реалном времену) Web функционалности у апликацијама. Web функционалност у реалном времену представља могућност да сервер пошаље податке клијенту чим они постану доступни уместо да сервер чека да клијент затражи нове податке. [9]

Потреба за оваквим функционалностима је свакодневна, на пример, ћаскање, односно размењивање порука, се врло често користи као пример, као и обавештења. Да би корисник видео нове поруке које су пристигле морао би сваки пут да освежи страницу како би поново послао захтев серверу да учита све поруке.

SignalR поједностављује тај процес увођењем асинхроне комуникације. Уместо да корисник константно шаље захтеве за поновним учитавањем порука, SignalR ће омогућити да сваки пут када се креира нова порука, корисник којем је она намењена буде обавештен о томе. Постоје три начина које SignalR користи да би такву комуникацију реализовао, а то су Web сокети, серверски послати догађаји (енг. *Server-Sent Event*) и дугачке анкете (енг. *Long polling*). SignalR аутоматски бира најбољи начин транспорта који се налази у оквиру могућности сервера и клијента. [27]

3.1. Web Socket

SIgnalR може да користи Web сокете за извршавање својих функционалности. Web сокет представља напредну технологију која омогућава отварање двосмерне комуникације између сервера и корисниковог претраживача. Помоћу њега омогућено је да сервер шаље садржај клијенту, а да га клијент претходно не захтева. [22] Такође, клијент може да шаље поруке серверу чиме је омогућено да се поруке прослеђују напред-назад, а да веза остане отворена. На овај начин се може одвијати двосмерни текући разговор између клијента и сервера.



Слика 5. SignalR комуникација
https://assets.website-files.com/5ff66329429d880392f6cba2/617a911c0c264f7bfbe7be5f_websocket%20work.png

SignalR омогућава коришћење Web сокета на једноставнији начин уз помоћ многих унапред имплементираних функционалности које SignalR нуди.

3.2. SignalR Hub

Путем SignalR Hub-а успоставља се комуникација између клијента и сервера и он омогућава да повезани клијенти позивају методе на серверу. Сервер дефинише методе које се позивају са клијентске стране, а клијент дефинише методе које се позивају са серверске стране.

Кориснички дефинисан Hub је класа која наслеђује Hub класу из пакета *ASP.Net.Core.SignalR*. Имплементација је доста једноставна, потребно је само да се дефинишу одговарајуће методе које ће клијент позивати и називи клијентских метода које ће сервер позивати, док је на клијентској страни потребно успоставити комуникацију са Hub-ом. Комуникација се успоставља помоћу класе *HubConnectionBuilder* из пакета *aspnet/signalr* и методе *build()* која враћа *HubConnection* објекат преко којег ће се одвијати сва комуникација. Наводи се путања до Hub-а и средство за транспорт (Web сокет). Након тога позива се метода *start()* која ће успоставуту комуникацију са Hub-ом.

```
startConnection= () => {
  this.hubConnection=new HubConnectionBuilder().withUrl('https://localhost:7042/hub', {
    skipNegotiation: true,
    transport: HttpTransportType.WebSockets
  }).build();

  this.hubConnection.start().then( ()=> {
    console.log("Hub started");
  }).catch(error=> console.log("---- Error: "+error.message));
}
```

Слика 6. Упостављање комуникације са сервером

Такође, на серверској страни у класи *Program.cs* потребно је регистровати сервис за SignalR и конфигурисати Hub путем методе *MapHub*. У наставку је приказан тај код.

```
builder.Services.AddSignalR();

app.MapHub<MyHub>("/hub");
```

Hub методе морају бити асинхроне, означене кључном речју *async*, и повратна вредност треба да буде *Task*. *Task* представља једну операцију која не враћа вредност и обично се извршава асинхроно. У основи, Task се користи за имплементацију асинхроног програмирања, односно за асинхроно извршавање операција. Task објекат се обично извршава асинхроно на нити из скупа нити (енг. *thread pool*[1]), а не на главној нити програма (енг. *main thread*). У наставку следи пример једне такве методе која шаље поруку одређеном кориснику.

```
public async Task SendMessage(MessageDto message)
{
    UserInfo u = usersChat.SingleOrDefault(u => u.UserId == message.ForId);
    if(u!=null)
    {
        await Clients.Client(u.ConnectionId).SendAsync("receivedMessage", message);
    }
}
```

Слика 7. Метода за слање поруке одређеном кориснику

Hub класа садржи својство (eng. *Property*) *Context* који садржи пропертије који нам нуде информације о тренутној конекцији. Један од најбитнијих својстава јесте ***ConnectionId***. *ConnectionId* враћа јединствену ид вредност за конекцију која је додељена путем SignalR-a. Постоји само један *ConnectionId* за сваку конекцију. Путем њега могуће је јединствено идентификовати сваког клијента и тачно знати ком клијенту треба послати одређену поруку. [8]

[1] thread pool (скуп нити) – колекција нити која се користи за обављање низа задатака (енг. Tasks) у позадини. Када нит изврши задатак, поново се шаље у скуп нити, тако да се може поново користити.

Поред *Context* својства, Hub класа садржи својство *Clients* која садржи својства и методе за комуникацију између сервера и клијента. Неке од њих су:

- property *All* – позива методу над свим повезаним корисницима
- property *Caller* – позива методу над клијентом који је позвао Hub методу
- property *Others* - позива методу над свим конектованим корисницима осим над клијентом који је позвао Hub методу
- method *Client* – позива методу над специфичним конектованим клијентом (очекује *ConnectionId* тог клијента као параметар)
- method *AllExcept* – позива методу над свим конектованим корисницима осим над клијентом који ће бити прослеђен методи (очекује *ConnectionId* клијента којег треба изоставити)

4. Web API

API је скраћеница од *Application Programming Interface* (програмски интерфејс апликације), што значи да API представља врсту интерфејса која садржи одређени скуп функција. API је посредник који омогућава двема апликацијама да размењују информације. Он преноси захтев за комуницирање од једне стране до друге. Исто тако омогућава слање података између два или више уређаја.

Web API интерфејс за програмирање апликација за Web. представља начин комуникације између два уређаја на мрежи. Ова комуникација се обично врши тако што клијент шаље захтев серверу путем мрежног сервиса и резултујућа информација се враћа клијенту у неком облику. Клијент може бити и неки сервер.

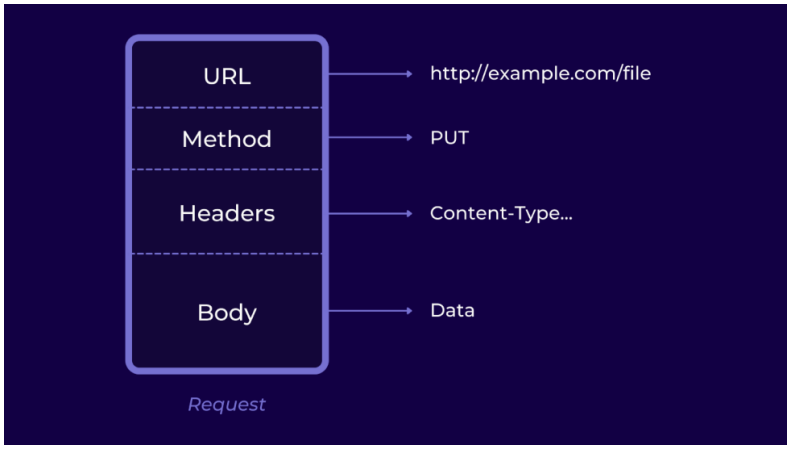
4.1. Hypertext Transfer Protocol (HTTP)

HTTP је протокол апликационог нивоа намењен преносу хипертекста на Интернету. HTTP протокол дефинише правила по којима се информације размењују између сервера и клијента (најчешће Web претраживача). HTTP протокол омогућава комуникацију између клијента и сервера, где клијент увек захтева извршење неког захтева, а сервер одговара на тај захтев.

Клијент захтева успостављање *TCP/IP* везе са Web сервером на одређеном порту, док сервер непрестано ослушкује мрежу чекајући да се клијент повеже. Тек када се клијент повеже могуће је слање захтева и одговора. HTTP представља протокол без стања, што значи да сервер не чува никакве податке или стања између два захтева.

Постоје два типа HTTP порука, захтев и одговор, и сваки има свој формат. Захтев садржи следеће елементе: [16]

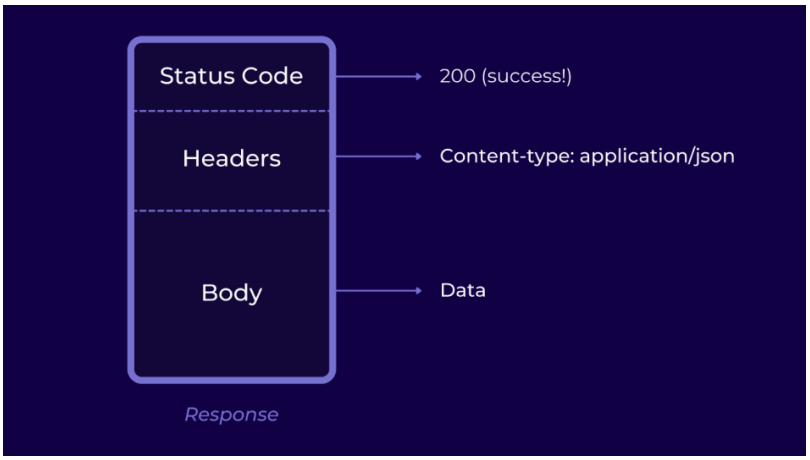
- HTTP метода - као што су GET или POST, дефинишу коју операцију клијент жели да изврши. Обично уколико се пошаље GET захтев клијент жели да се учитају одређени подаци из базе, POST за чување нових података, PATH за измену постојећих података DELETE за брисање...
- URL - јединствена адреса за ресурс на серверу. Ресурси могу бити било шта, од Web страница до слика до апстрактнијих датотека које се односе на ствари као што су лични профили, обично ускладиштени у JSON формату.
- Заглавље (енг. *Header*) - Додавањем заглавља у захтев, укључују се мета-информације, на пример, заглавље *Authorization* се користи да се у њега смести токен за ауторизацију корисника на серверу.
- Тело (енг. *Body*) - Овај део захтева садржи информације које треба послати серверу. За захтеве који мењају постојеће податке или креирају нове, тело ће садржати податке за ту акцију.



Слика 8. Структура HTTP захтева
https://miro.medium.com/max/720/1*i2tUjWy44-dYT9qsaWbvig.png

Одговор садржи следеће елементе: [16]

- Статус - Статусни код је троцифрени број који сервер користи да каже клијенту резултат његовог захтева. Сваки код има своје значење, али се сви могу груписати у једну од пет класа:
  - 100–199: информативни одговори
  - 200–299: успешни одговори
  - 300–399: преусмеравања
  - 400–499: грешке клијента
  - 500–599: грешке сервера
- Тело
- Заглавље



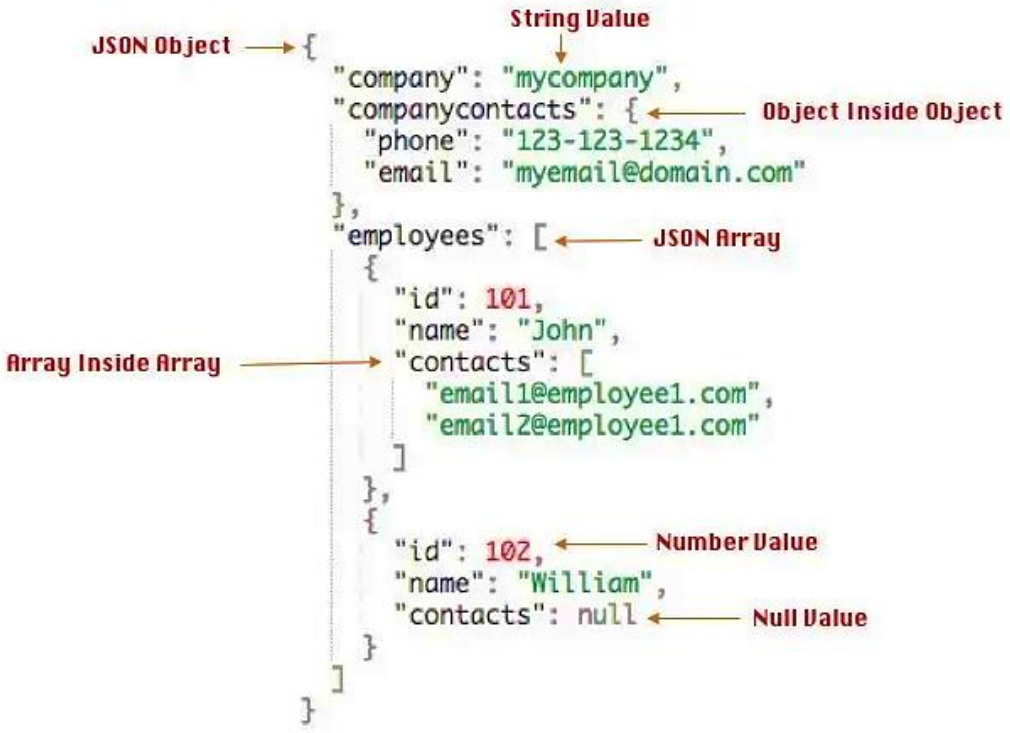
Слика 9. Структура HTTP одговора
https://miro.medium.com/max/720/1*w4gDd2TFunoOnrWy3xpHkQ.png

4.2. JSON

JSON (*JavaScript Object Notation*) је текстуални формат за размену података који је потпуно независан од програмског језика. Изведен је из JavaScript-а. JSON је изграђен на две структуре: [17]

- Колекција парова име/вредност - У разним језицима, ово се реализује као објекат, запис, структура, речник, листа са кључевима или асоцијативни низ. Ова структура се обележава са {} витичастим заградама, а подаци у облику име/вредност су одвојени зарезом.
- Уређена листа вредности - У већини језика, ово се реализује као низ, вектор, листа или секвенца. Обележава се [] угластим заградама, а чланови низа су одвојени зарезом.

Ово су универзалне структуре података. Практично сви савремени програмски језици их подржавају у овом или оном облику.



Слика 10. Пример JSON објекта
https://i2.wp.com/techeplanet.com/wp-content/uploads/2019/01/json-example.jpg?w=608&ssl=1

5. Аутентификација и ауторизација

Аутентификација је процес утврђивања идентитета корисника. Ауторизација је процес утврђивања да ли одређени корисник има приступ одређеном ресурсу. У ASP.NET Core-у аутентификацијом управља аутентификациони сервис *IAuthetnicationService* који је коришћен од стране аутентификационог *middleware*-а (посредника). Примери аутентификационих акција су следећи: [12]

- Аутентификација корисника,
- Одговарање када корисник без аутентификације покуша да приступи ограниченом ресурсу.

Регистровани руковаоци аутентификацијом и њихове опције конфигурације се називају „шеме“. Аутентификационе шеме су дефинисане при регистровању аутентификационог сервиса у *Program.cs* класи:

- Позивањем методе проширења (енг. Extension metod*)* која је специфична за дату шему као што је *AddJwtBearer* ili *AddCookie.* Ове методе користе *AuthenticationBuilder.AddScheme* у циљу регистровања шеме са адекватним подешавањем

Дати код представља регистровање аутентификационог сервиса за *JWT bearer* аутентификациону шему:

```
builder.Services.AddAuthentication(options=> {
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    })
    .AddJwtBearer(options =>
    {
        options.SaveToken = true;
        options.TokenValidationParameters = new
      Microsoft.IdentityModel.Tokens.TokenValidationParameters
        {
            ValidateAudience = false,
            ValidateIssuer = false,
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new
      SymmetricSecurityKey(Encoding.ASCII.GetBytes("NapredneNetTehnologijeToken
      ZaAuthentifikaciju"))
        };
    });
```

Аутентификациони *middleware* (посредник) се додаје у *Program.cs* класу позивањем *UseAuthentication.UseAuthentication* региструје middleware (посредник) који користи претходно регистроване шеме за аутентификацију. *AddAuthentication* параметар *JwtBearerDefaults.AuthenticationSceme* је име шеме која се користи као подразумевана ако друга није дефинисана. [12]

Ауторизација се односи на процес који одређује шта корисник може да уради. На пример административном кориснику је дозвољено да креира библиотеку докумената, додаје документе, уређује документе и брише их. [13]

Неадминистративни корисник који ради са библиотеком је овлашћен само да чита документе. Ауторизација је ортогонална и независна од аутентификације. Међутим ауторизација захтева механизам аутентификације. ASP.NET Core ауторизација омогућава једноставане, декларативне *Role* (улоге) і *Policy* моделе. [11]

У примеру веб апи-ја, уколико је потребно означити да неком *end-point*-у (крајњој рути) могу да приступају само ауторизовани корисници, довољно је да изнад те руте ставимо анотацију *[Authorize]*.

5.1. JWT

JSON Web token (JWT) је отворен стандард који дефинише компактан и самосталан начин за безбедно преношење информација између страна као JSON објекат. Ове информације могу бити верификоване и поуздане јер су дигитално потписане. JWT токени могу бити потписани коришћењем шифре (путем XMAC алгоритма) или путем пара јавних/приватних кључева. Постоје два типична сценарија у којима се користе JSON Web Токени: [21]

- **Ауторизација** – Приликом пријаве на систем за корисника се генерише јединствени JWT токен који ће бити саставни део сваког наредног захтева како би се кориснику омогућио приступ одређеним рутама и ресурсима, уколико је њихов приступ ограничен.
- **Размена информација** – JWT токен је начин за безбедно преношење информација између страна. Имајући у виду да JWT токени могу бити потписани на пример коришћењем парова јавних/приватних кључева то обезбеђује сигурност да пошиљаоци заиста јесу ти који тврде да су они.

Садржај JWT токена:

- Заглавље (енг. *Header*) – Обично се састоји из два дела који подразумевају тип токена и врсту коришћеног алгоритма.
- Садржај података (енг. *Payload*) – Садржи *Claim*-ове (тврдње), Claim-ови представљу неку врсту тврдње о неком ентитету, најчешће је у питању корисник.
- Потпис (енг. *Signature*) - У циљу обезбеђивања дела потписа неопходно је узети кодирани *Header*, кодирани *Payload*, шифру и алгоритам који је наведен у заглављу.

Када се корисник успешно пријави са својим креденцијалима, њему ће бити враћен токен. Када год корисник буде желео да приступи неком заштићеном ресурсу или рути, кориснички агент ће послати добијени токен у ауторизационом заглављу (енг. *Authorization Header*) користећи *Bearer* шему. Заштићене руте сервера проверавају да ли у заглављу постоји важећи токен и уколико је присутан, кориснику ће бити одобрен приступ заштићеном ресурсу.

Разлози за коришћење JWT токена при поређењу са SWT токеном (***Simple Web Tokens***) и **SAML токеном (*Security Assertion Markup Language)***: [21]

- Имајући у виду да је JSON токен мањег опсега од XML-а, када је кодиран његова величина је такође мања, што JWT токен чини компактнијим од SML-а. Ова тврдња чини JWT добрим избором за прослеђивање у HTTP окружењима.
- SWT токен може бити само симетрично потписан заједничком шифром користећи HMAC алгоритам. Међутим JWT и SAML токени могу да користе пар јавни/приватни кључ у облику X.509 сертификата за потписивање. Потписивање XML-а помоћу XML дигиталног потписа без увођења нејасних сигурносних проблема је веома тешко у поређењу са једноставношћу потписивање JSON-а.
- JSON парсери су уобичајни у већини програмских језика јер мапирају директно у објекте. Супротно томе XML нема природно мапирање документа у објекат.
- У контексту употребе, JWT се користи на интернету. Ово наглашава лакоћу обраде JWT токена на страни клијента на више платформи, посебно мобилних.

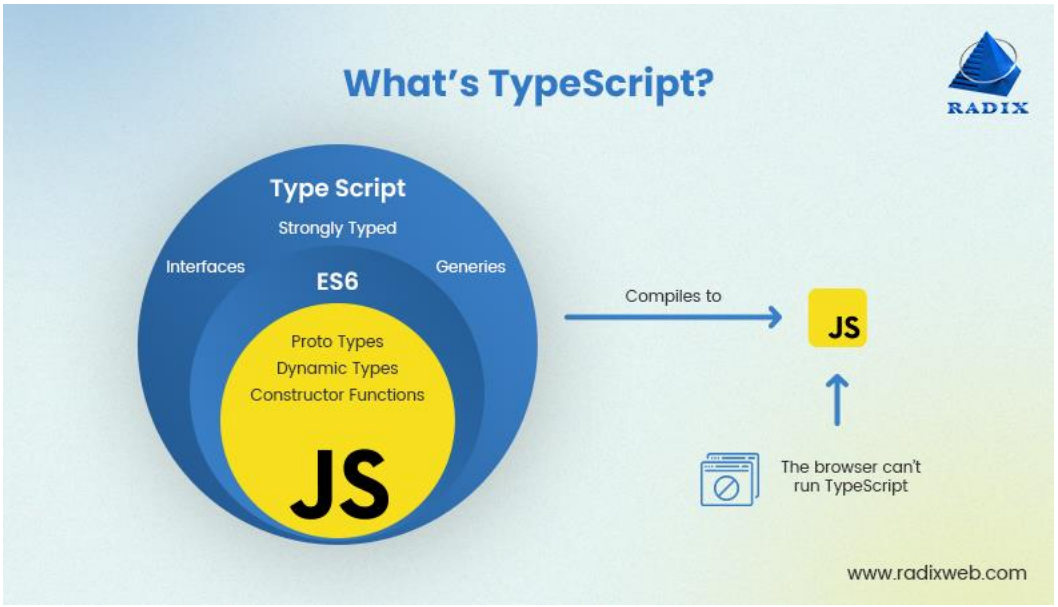
6. Технологије коришћене за израду корисничког интерфејса

Клијентска апликација је реализована уз помоћ библиотеке Angular и програмског језика TypeScript, који користи HTML и CSS. Поред тога коришћене су и библиотеке које поједностављују израду компоненти корисничког интерфејса као што су AngularMaterial и Bootstrap, које нуде велики број већ готових компоненти за приказ и SweetAllert који омогућава приказ обавештења (енг. *alert*).

6.1. TypeScript

TypeScript је бесплатан објектно-оријентисан програмски језик отвореног кода који је развијен од стране Microsoft-а.

TypeScript је језик који надограђује JavaScript, увођењем синтаксе за статичко типизирање. Код написан у TypeScript-u не може да се изврши директно већ се преводи у JavaScript, што значи да може да се користи за израду и фронтенд и бекенд делова апликације. Како је надскуп JavaScript-а, сви постојећи JavaScript програми су валидни TypeScript програми.



Слика 11. Однос JavaScript-а и TypeScript-а
https://d2ms8rpfqc4h24.cloudfront.net/uploads/2021/12/Understand-Typescript.jpg

TypeScript подржава статичко типизирање, односно приликом декларисања промењљиве неопходно је декларисати и њен тип. Типови се могу и експлицитно и имплицитно доделити, али се препоручује додељивање типова. JavaScript не подржава статичко типизирање. [18]

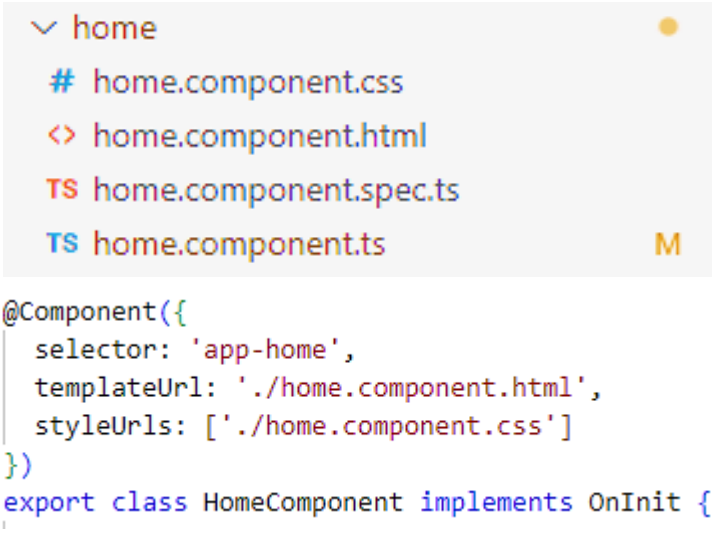
6.2. Angular

Angular је фронтенд оквир који је развио Google. То је структуиран оквир направљен на TypeScript-u и служи за прављење динамичких Web апликација. Дозвољава коришћење HTML и CSS, али исто тако проширује могућности HTML-а. Angular компоненте су јако корисне, и релативно лако их је користити.

Основни градивни блокови Angular апликације су Angular компоненте које су организоване у Angular модулима (*NgModules*). Angular модули прикупљају сродни код у функционалне скупове. Свака Angular апликација има основни модул, *AppModul*, који обезбеђује механизам за покретање апликације. Једна Angular апликација обично је дефинисана скупом Angular модула. [19]

Ангулар апликација садржи стабло Ангулар компоненти. Оне представљају подскуп директива, увек повезане са шаблоном (енг. *template*), односно имају свој поглед, за разлику од директива које немају свој поглед и користе се за додавање додатног понашања постојећим елементима у апликацији.

Свака компонента се састоји од четири фајла, HTML фајл који представља шаблон и он дефинише садржај који ће се приказивати, CSS фајл за стилизовање елемената дефинисаних у HTML фајлу, TS фајла у коме се налазе функционалности за дефинисане елементе и фајл за спецификацију тестирања. Ови подаци дефинишу се у TS фајлу у декоратору[2] *@Component* који Angular-у говори да је дефинисана класа заправо класа компоненте.



Слика 12. Изглед једне компоненте у Angular-у

[2] декоратори (енг. Decorators) – обезбеђују метаподатке за класе, методе или својства (енг. *property*)

Параметар *Selector* се користи за јединствену идентификацију сваке компоненте у стаблу компоненти и помоћу њега дефинисана компоненте ће моћи да се позива. [20] На пример, уколико је потребно да компонента *Home* буде саставни део погледа неке друге компоненте, може се позвати на следећи начин: <*app-home*> . Параметар *templateUrl* служи да се дефинише путања до шаблона, односно погледа који ће компонента користити. Слично томе, у параметру *styleUrls* дефинишу се путање до CSS фајлова које ће компонента користити за дефинисани шаблон. Једна компонента може да има само један шаблон, али више фајлова за стилизовање тог шаблона.

Најбитнији концепти везани за Angular: [26]

- *Template* - HTML са додатним ознакама – описује шта треба бити приказано
- *Directive* - са својим елементима и атрибутима (делови кода за вишекратну употребу)
- *Scope* - Контекст где су подаци модела смештени, тако да шаблони и контролери могу да им приступе
- *Compiler* - Процесира шаблон (енг. *template*) и генерише HTML код за прегледача
- *Data Binding* - Сихронизација података између *Scope* и HTML-а (двосмерно)
- *Dependency Injection* - додаје и поставља све функционалности потребне компонентама
- *Module* - Контејнер за све делове једне апликације
- *Service* - Начин паковања функционалности како би биле доступне на свим погледима

6.3. Bootstrap

Bootstrap је JavaScript оквир отвореног кода, односно комбинација HTML, CSS и JS, развијен са циљем да омогући и олакша развој веб форми као и развој напредних веб компоненти. Развијен је од стране Twitter-а.

Једна од великих предности Bootstrap-а је креирање компоненти, као и страница, које су респонзивне, односно које су прилагодљиве величини уређаја на којима се приказују, тако да се почетни изглед странице не промени значајно са смањењем или повећањем величине екрана на којима се приказују. Bootstrap садржи већ дефинисане класе које представљају CSS подешавања и које је потребно само применити над HTML елементима.

Основне предности Bootstrapa-а су: [25]

- олакшава и убрзава развој апликација
- поједностављује развој Web форми и компоненти које служе за приказивање података на мобилним уређајима
- идентично се приказује на свим модерним Web претраживачима
- релативно је једноставан за употребу

7. Студијски пример

На студијском примеру приказан је развој друштвене мреже *“Reach me”* коришћењем упрошћене Ларманове методе развоја софтвера. Развој је представљен кроз пет фаза, које су уједно и наредна поглавља овог рада: прикупљање корисничких захтева, анализа, пројектовање, имплементација и тестирање.

8. Прикупљање корисничких захтева

У овом поглављу биће представљен студијски пример овог софтверског система.

8.1. Вербални опис модела

Друштвена мрежа *‘’Reach Me’’* представља софтверски систем путем којег корисници могу да комуницирају на различите начине. На почетној страници налази се форма за пријаву корисника на систем или регистрацију новог профила уколико га корисник већ не поседује. Приликом креирања профила корисник уноси своје основне податке и добија подразумевану профилну слику (аватар) коју касније може да промени.

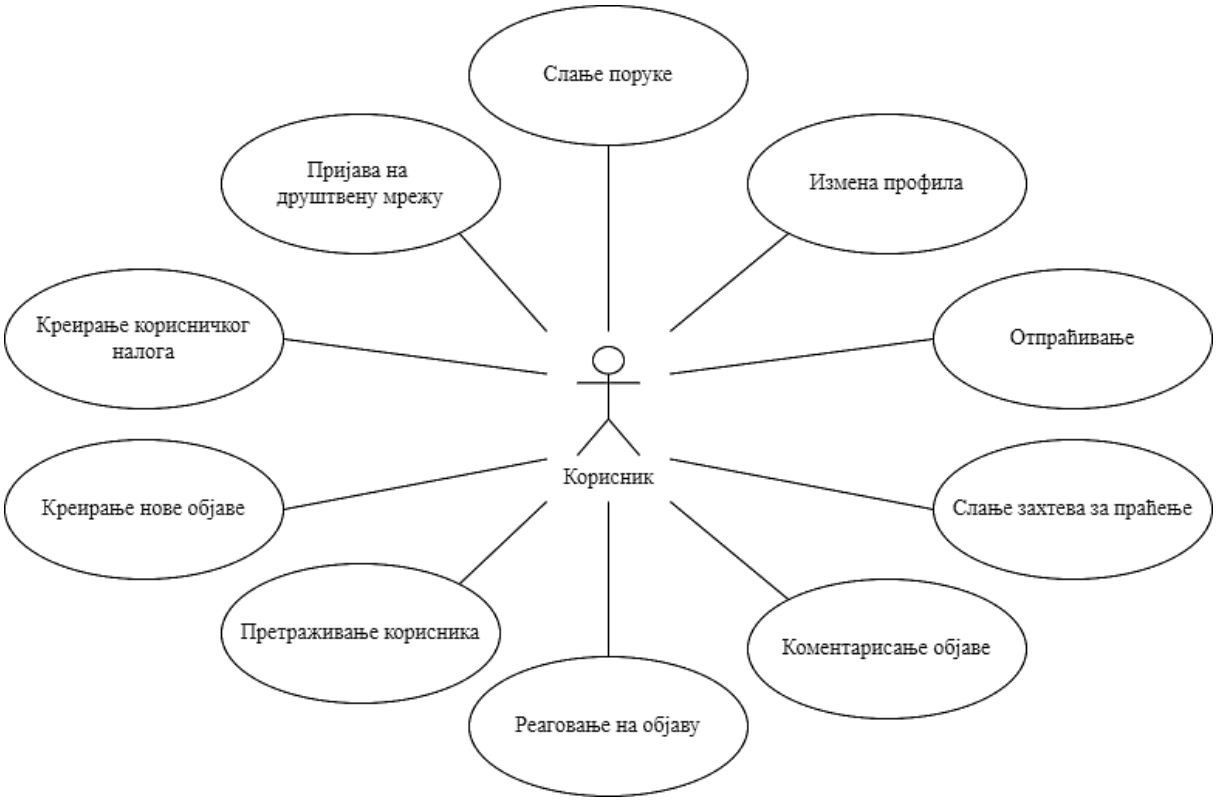
Након успешне пријаве на систем кориснику је учитана сортирана листа свих објава његових пријатеља, уколико их има. Корисник може да означи да му се нека објава допада или да прокоментарише објаву. Такође може да види и ко је све означио да му се допада та објава и све коментаре за ту објаву. На почетној страни се налази и форма за објављивање нове објаве. Поред овога, кориснику се излиставају и три рандом корисника које пријављени корисник не прати, као предложени за нове пријатеље.

Корисник може да види и свој профил на којем се налазе све објаве које је корисник објављивао. Омогућена је и претрага других корисника и одлазак на профил тих корисника и слање порука другом кориснику. Осим што корисник може да пошаље поруку другом кориснику, такође може и да види све поруке које има са неким корисником и датум када су послате.

8.2. Спецификација модела помоћу модела случајева коришћења

На основу вербалног описа система могу се издвојити следећи случајеви коришћења:

1. Пријава на друштвену мрежу
2. Креирање корисничког налога
3. Креирање нове објаве
4. Реаговање на објаву
5. Коментарисање објаве
6. Претраживање корисника
7. Слање захтева за праћење
8. Отпраћивање
9. Измена профила
10. Слање поруке



Слика 13. Дијаграм случаја коришћења

СК1: Случај коришћења - Пријављивање на друштвену мрежу

Назив СК

Пријављивање на друштвену мрежу

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и приказује форму за пријављивање на систем.

Основни сценарио СК

1. Корисник **уноси** податке за пријављивање (корисничко име и лозинку). (АПУСО)

2. Корисник **контролише** да ли је коректно унео корисничко име и лозинку. (АНСО)

3. Корисник **позива** систем да га пријави. (АПСО)

4. Систем **проверава** податке о кориснику. (СО)

5. Систем **приказује** кориснику почетну страну и поруку: “Welcome, ime prezime!“. (ИА)

Алтернативна сценарија

5.1. Уколико систем не може да нађе корисника, он **приказује** кориснику поруку: “Sorry, we can’t log you in!”. (ИА)

СК2: Случај коришћења – Креирање корисничког налога

Назив СК

Креирање корисничког налога

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и приказује форму за регистрацију на систем.

Основни сценарио СК

1. Корисник **уноси** основне податке за креирање корисничког налога. (АПУСО)

2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)

3. Корисник **позива** систем да га региструје. (АПСО)

4. Систем **памти** податке о кориснику. (СО)

5. Систем **приказује** кориснику страницу за пријаву и поруку: “Your account is successfully created!“. (ИА)

Алтернативна сценарија

5.1. Уколико систем не може да региструје корисника, он **приказује** кориснику поруку: “Sorry, we can’t create your account!”. (ИА)

СКЗ: Случај коришћења – Креирање нове објаве

Назив СК

Креирање нове објаве

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника, на којој се налази форма за унос нове објаве.

Основни сценарио СК

1. Корисник **уноси** податке о објави. (АПУСО)

2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)

3. Корисник **позива** систем да запамти податке о објави. (АПСО)

4. Систем **памти** податке о објави. (СО)

5. Систем **приказује** кориснику поруку: “Your post has been shared!”. (ИА)

Алтернативна сценарија

5.1. Уколико систем не може да запамти податке о објави, он **приказује** кориснику поруку: “Sorry, we can’t share your post!”. (ИА)

СК4: Случај коришћења – Реаговање на објаву

Назив СК

Реаговање на објаву

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих објава особа које корисник прати.

Основни сценарио СК

1. Корисник **бира** објаву на коју жели да реагује. (АПУСО)

2. Корисник **позива** систем да реагује на одабрану објаву. (АПСО)

3. Систем **памти** реакцију за одабрану објаву. (СО)

4. Систем **приказује** кориснику да је успешно реаговао на објаву. (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да запамти реакцију за одабрану објаву, он **приказује** кориснику поруку: “Sorry. Try again later!”. (ИА)

СК5: Случај коришћења – Коментарисање објаве

Назив СК

Коментарисање објаве

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих објава особа које корисник прати.

Основни сценарио СК

1. Корисник **бира** објаву коју жели да коментарише. (АПУСО)

2. Корисник **позива** систем да учита све коментаре за изабрану објаву. (АПСО)

3. Систем **учитава** коментаре за одабрану објаву. (СО)

4. Систем **приказује** кориснику све коментаре за изабрану објаву. (ИА)

5. Корисник **уноси** нови коментар за изабрану објаву. (АПУСО)

6. Корисник **контролише** да ли је коректно унео нови коментар. (АНСО)

7. Корисник **позива** систем да запамти коментар за изабрану објаву. (АПСО)

8. Систем **памти** коментар за објаву. (СО)

9. Систем **приказује** кориснику запамћен коментар. (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да прикаже све коментаре за одабрану објаву, он **приказује** кориснику поруку: “Sorry, we can’t load comments!”. (ИА)

9.1. Уколико систем не може да запамти коментар, он **приказује** кориснику поруку: “Sorry, we can’t save your comment!”. (ИА)

СК6: Случај коришћења – Претраживање корисника

Назив СК

Претраживање корисника

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)

5. Корисник **бира** корисника. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

7. Систем **учитава** профил изабраног корисника. (СО)

8. Систем **приказује** кориснику профил изабраног корисника. (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)

СК7: Случај коришћења – Слање захтева за праћење

Назив СК

Слање захтева за праћење

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)

5. Корисник **бира** корисника којем жели да пошаље захтев. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

7. Систем **учитава** профил изабраног корисника. (СО)

8. Систем **приказује** кориснику профил изабраног корисника. (ИА)

9. Корисник **позива** систем да пошаље захтев изабраном кориснику. (АПСО)

10. Систем **памти** захтев за праћење. (СО)

11. Систем **приказује** кориснику да је захтев успешно послат. (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)

10.1. Уколико систем не може да запамти захтев за праћење, он **приказује** кориснику поруку: “Sorry, we can’t send your request!”. (ИА)

СК8: Случај коришћења – Отпраћивање

Назив СК

Реаговање на објаву

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)

5. Корисник **бира** корисника којег жели да отпрати. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

7. Систем **учитава** профил изабраног корисника. (СО)

8. Систем **приказује** кориснику профил изабраног корисника. (ИА)

9. Корисник **позива** систем да отпрати изабраног корисника. (АПСО)

10. Систем **отпраћује** изабраног корисника. (СО)

11. Систем **приказује** кориснику да је изабрани корисник успешно отпраћен. (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”(ИА)

10.1. Уколико систем не може да отпрати корисника, он **приказује** кориснику поруку: “Sorry, we can’t unfollow this user!”. (ИА)

СК9: Случај коришћења – Измена профила

Назив СК

Измена профила

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује профил улогованог корисника. Учитани су подаци о улогованом кориснику као и листа свих објава улогованог корисника.

Основни сценарио СК

1. Корисник **уноси (мења)** своју профилну слику. (АПУСО)

2. Корисник **контролише** да ли је коректно унео нову профилну слику. (АНСО)

3. Корисник **позива** систем да запамти податке о кориснику. (АПСО)

4. Систем **памти** податке о кориснику. (СО)

5. Систем **приказује** кориснику да је запамтио нове податке. (ИА)

Алтернативна сценарија

5.1. Уколико систем не може да запамти податке о кориснику, он **приказује** кориснику поруку: “Sorry, we can’t update your profile!”. (ИА)

СК10: Случај коришћења – Слање поруке

Назив СК

Слање поруке

Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих корисника са којима улоговани корисник има поруке.

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)

5. Корисник **бира** корисника којем жели да пошаље поруку. (АПУСО)

6. Корисник **позива** систем да учита chat са изабраним корисником. (АПСО)

7. Систем **учитава** листу свих порука са изабраним корисником. (СО)

8. Систем **приказује** кориснику поруке. (ИА)

9. Корисник **уноси** нову поруку за изабраног корисника. (АПУСО)

10. Корисник **контролише** да ли је коректно унео нову поруку. (АНСО)

11. Корисник **позива** систем да пошаље поруку. (АПСО)

12. Систем **памти** поруку за изабраног корисника. (СО)

13. Систем **приказује** кориснику послату поруку. (ИА)

Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)

8.1. Уколико систем не може да приказе кориснику поруке са изабраним корисником, он **приказује** кориснику поруку: “Sorry, we can’t load your messages!”. (ИА)

13.1. Уколико систем не може да запамти поруку, он **приказује** кориснику поруку: “Sorry, we can’t send your message!”. (ИА)

9. Анализа

Фаза анализе је друга фаза развоја софтверског система помоћу Ларманове методе. У оквиру ње описује се структура и понашање софтверског система. Структура софтверског система описана је помоћу концептуалног и релационог модела, док је понашање софтверског система описано помоћу системских дијаграма секвенци. [1]

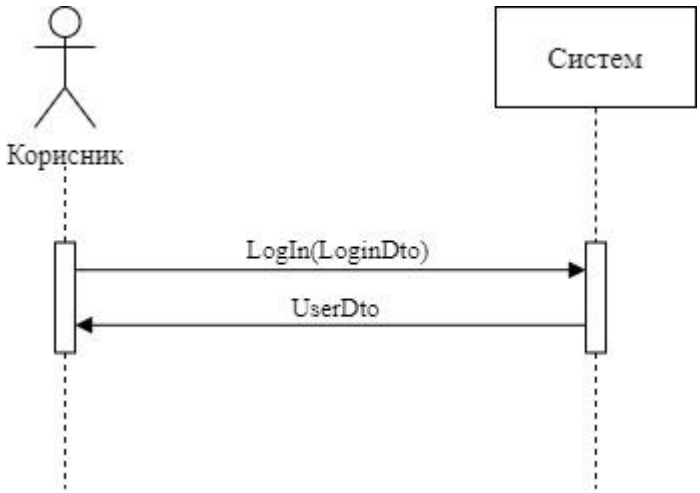
9.1. Понашање софтверског система-Системски дијаграми секвенци

Системски дијаграм секвенци приказује, за издвојени сценарио СК, догађаје у одређеном редоследу, који успостављају интеракцију између актора и софтверског система. [1]

ДС1: Дијаграм секвенце случаја коришћења – Пријављивање корисника на друштвену мрежу

1. Корисник **позива** систем да га пријави. (АПСО)

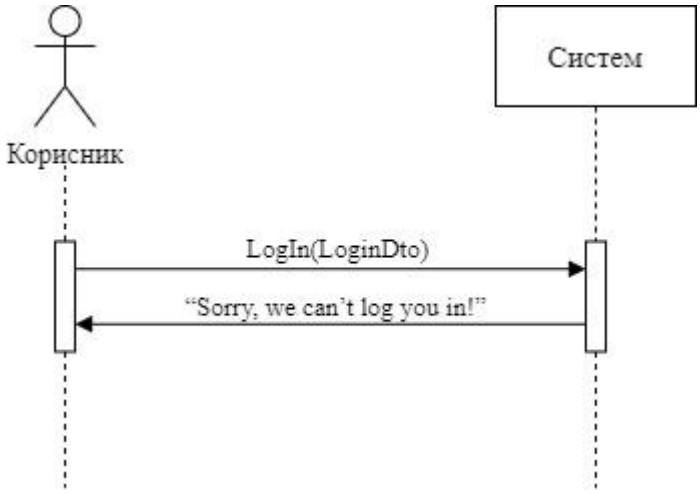
2. Систем **приказује** кориснику почетну страну и поруку: “Welcome, ime prezime!“. (ИА)



Слика 14. ДС - Пријављивање корисника на друштвену мрежу

Алтернативна сценарија

2.1. Уколико систем не може да нађе корисника, он **приказује** кориснику поруку: “Sorry, we can’t log you in!”. (ИА)



Слика 15. ДС - Пријављивање корисника на друштвену мрежу, алтернативни сценарио

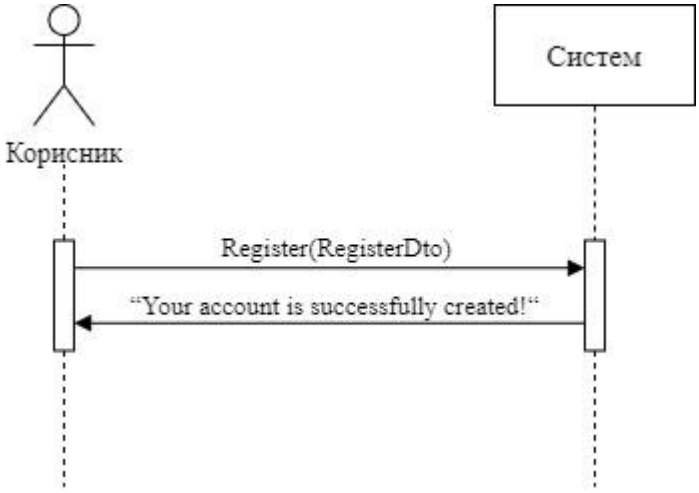
Са наведених секвенцних дијаграма уочава се једна системска операција коју треба пројектовати:

1. *signal* **LogIn**(LoginDto);

ДС2: Дијаграм секвенце случаја коришћења – Креирање корисничког налога

1. Корисник **позива** систем да га региструје. (АПСО)

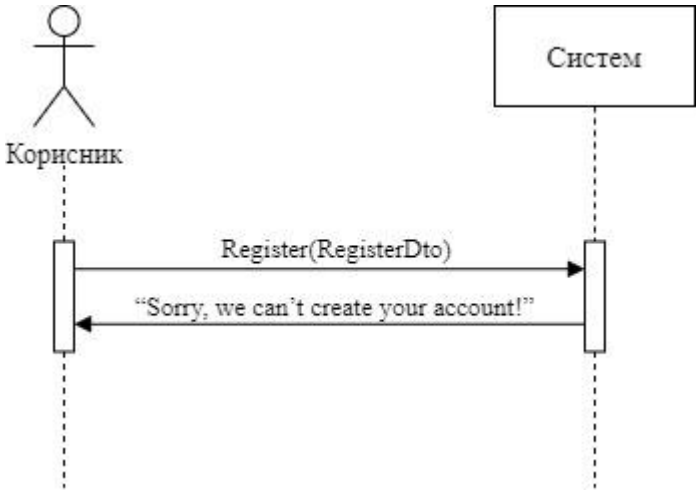
2. Систем **приказује** кориснику страницу за пријаву и поруку: “Your account is successfully created!“. (ИА)



Слика 16. ДС – Креирање корисничког налога

Алтернативна сценарија

2.1. Уколико систем не може да региструје корисника, он **приказује** кориснику поруку: “Sorry, we can’t create your account!”. (ИА)



Слика 17. ДС - Креирање корисничког налога, алтернативни сценарио

Са наведених секвенцних дијаграма уочава се једна системска операција коју треба пројектовати:

1. *signal* **Register**(RegisterDto);

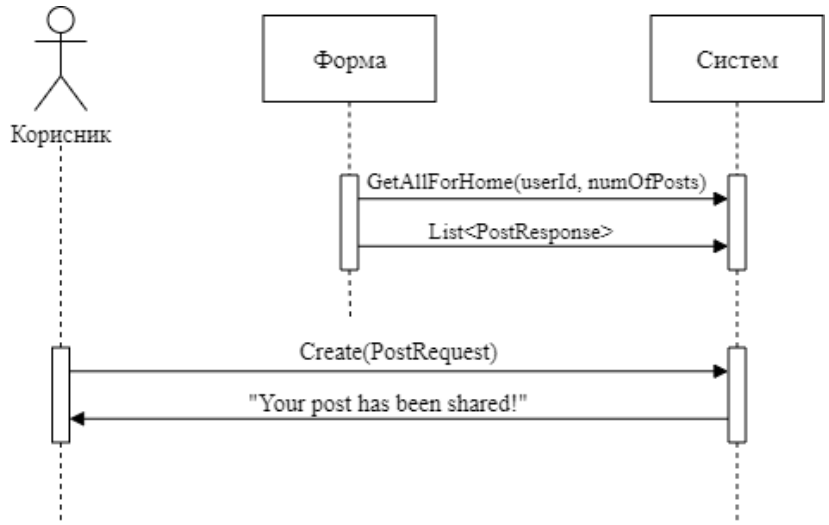
ДС3: Дијаграм секвенце случаја коришћења – Креирање нове објаве

1. Форма **позива** систем да учита све објаве корисника које пријављени корисник прати. (АПСО)

2. Систем **враћа** форми листу објава. (ИА)

3. Корисник **позива** систем да запамти податке о објави. (АПСО)

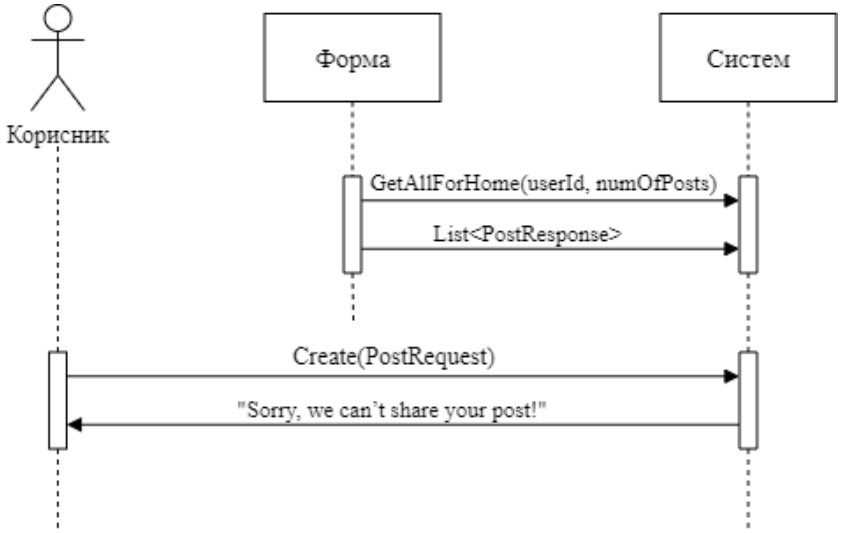
4. Систем **приказује** кориснику поруку: “Your post has been shared!” (ИА)



Слика 18. ДС - Креирање нове објаве

Алтернативна сценарија

2.1. Уколико систем не може да запамти податке о објави, он **приказује** кориснику поруку: “Sorry, we can’t share your post!”. (ИА)



Слика 19. ДС - Креирање нове објаве, алтернативни сценарио

Са наведених секвенцних дијаграма уочава се једна системска операција коју треба пројектовати:

1. *signal* **Create**(PostRequest);

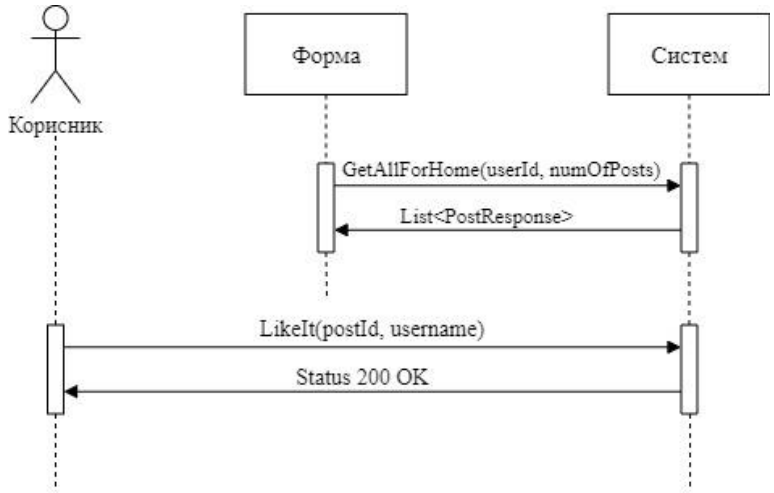
ДС4: Дијаграм секвенце случаја коришћења – Реаговање на објаву

1. Форма **позива** систем да учита све објаве корисника које пријављени корисник прати. (АПСО)

2. Систем **враћа** форми листу објава. (ИА)

3. Корисник **позива** систем да реагује на одабрану објаву. (АПСО)

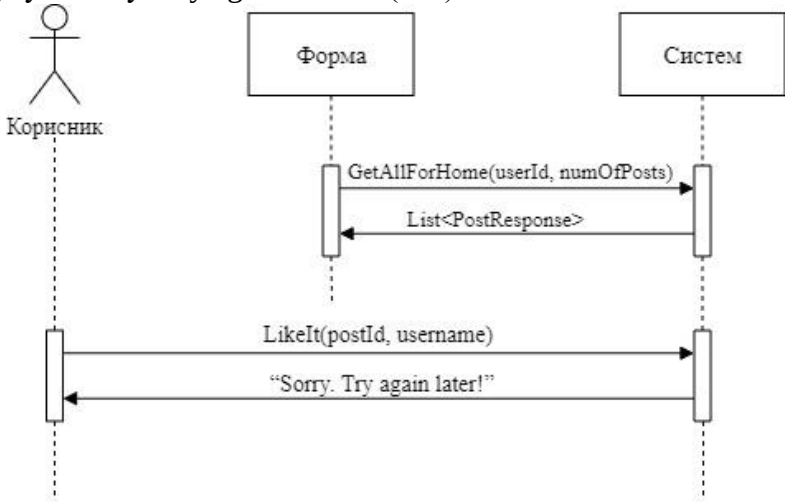
4. Систем **приказује** кориснику да је успешно реаговао на објаву. (ИА)



Слика 20. ДС – Реаговање на објаву

Алтернативна сценарија

2.1. Уколико систем не може да запамти реакцију за одабрану објаву, он **приказује** кориснику поруку: “Sorry. Try again later!”. (ИА)



Слика 21. ДС – Реаговање на објаву, алтернативни сценарио

Са наведених секвенцних дијаграма уочавају се две системска операција које треба пројектовати:

1. *signal* **GetAllForHome**(userId, numOfPosts);
2. *signal* **LikeIt**(postId, username)

ДС5: Дијаграм секвенце случаја коришћења – Коментарисање објаве

1. Форма **позива** систем да учита све објаве корисника које пријављени корисник прати. (АПСО)

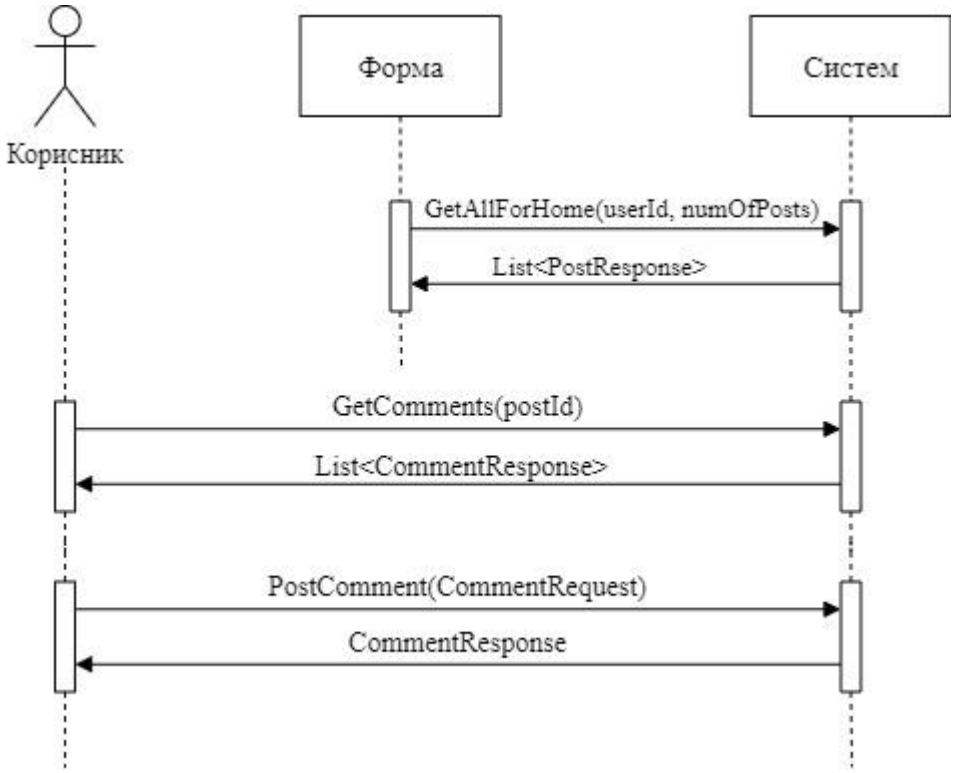
2. Систем **враћа** форми листу објава. (ИА)

3. Корисник **позива** систем да учита све коментаре за изабрану објаву. (АПСО)

4. Систем **приказује** кориснику све коментаре за изабрану објаву. (ИА)

5. Корисник **позива** систем да запамти коментар за изабрану објаву. (АПСО)

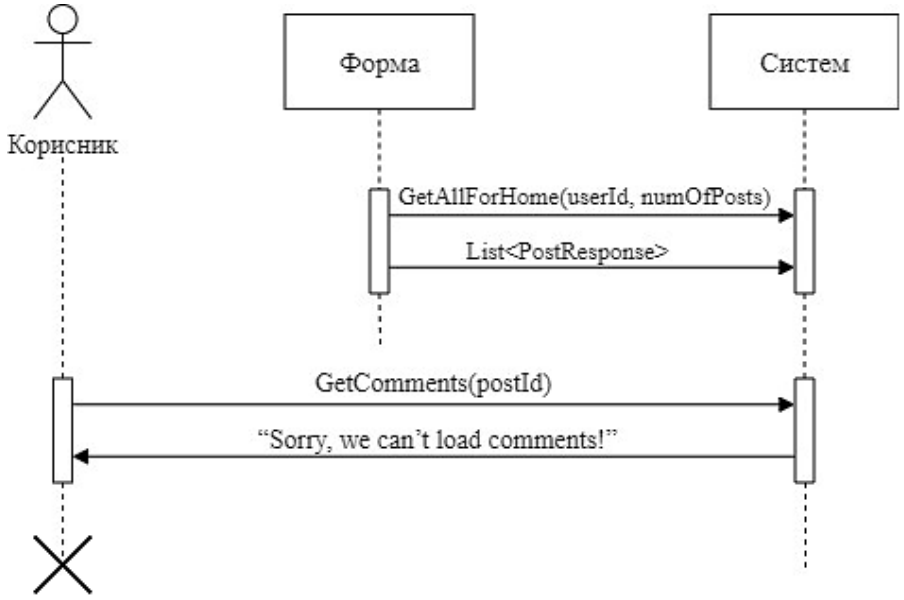
6. Систем **приказује** кориснику запамћен коментар. (ИА)



Слика 22. ДС – Коментарисање објаве

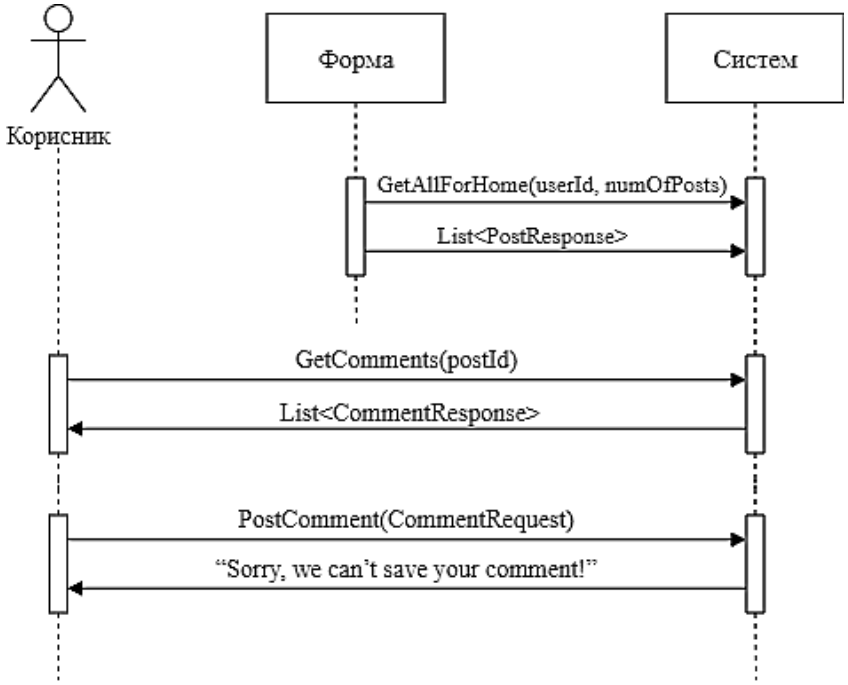
Алтернативна сценарија

2.1. Уколико систем не може да прикаже све коментаре за одабрану објаву, он **приказује** кориснику поруку: “Sorry, we can’t load comments!”. (ИА)



Слика 23. ДС – Коментарисање објаве, алтернативни сценарио.
Систем не може да учита коментаре.

4.1. Уколико систем не може да запамти коментар, он **приказује** кориснику поруку: “Sorry, we can’t save your comment!”. (ИА)



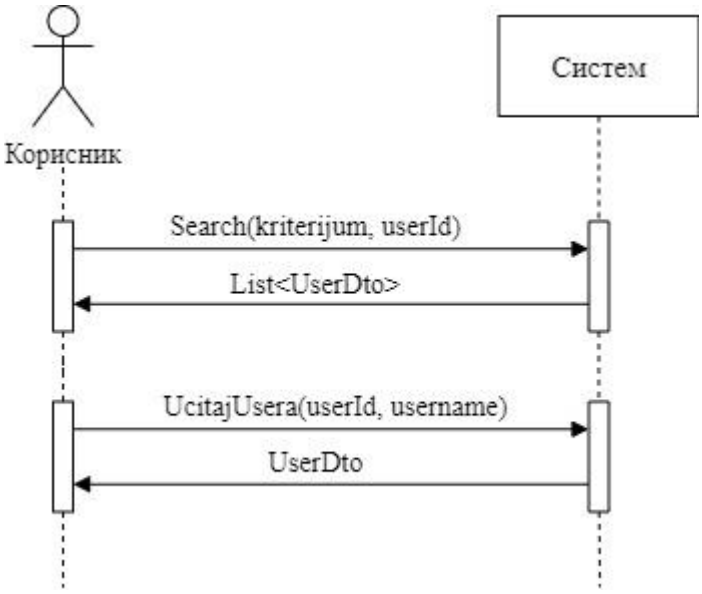
Слика 24. ДС – Коментарисање објаве, алтернативни сценарио.
Систем не може да сачува коментар.

Са наведених секвенцних дијаграма уочавају се две системске операције које треба пројектовати:

1. *signal* **GetComments**(postId);
2. *signal* **PostComment**(CommentRequest)

ДС6: Дијаграм секвенце случаја коришћења – Претраживање корисника

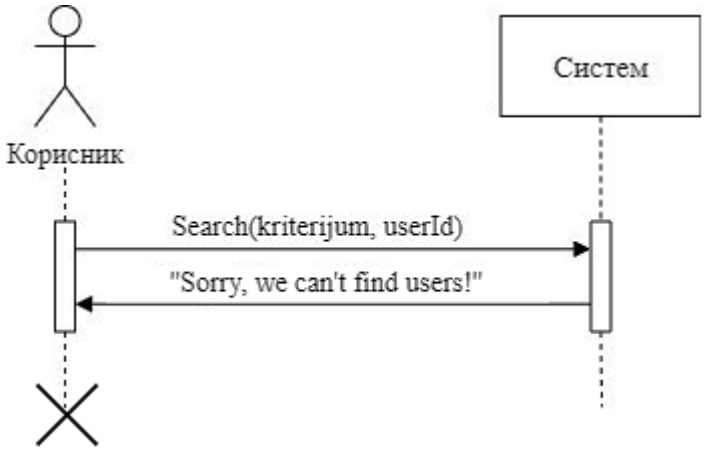
1. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)
2. Систем **приказује** кориснику пронађене кориснике. (ИА)
3. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)
4. Систем **приказује** кориснику профил изабраног корисника. (ИА)



Слика 25. ДС – Претраживање корисника

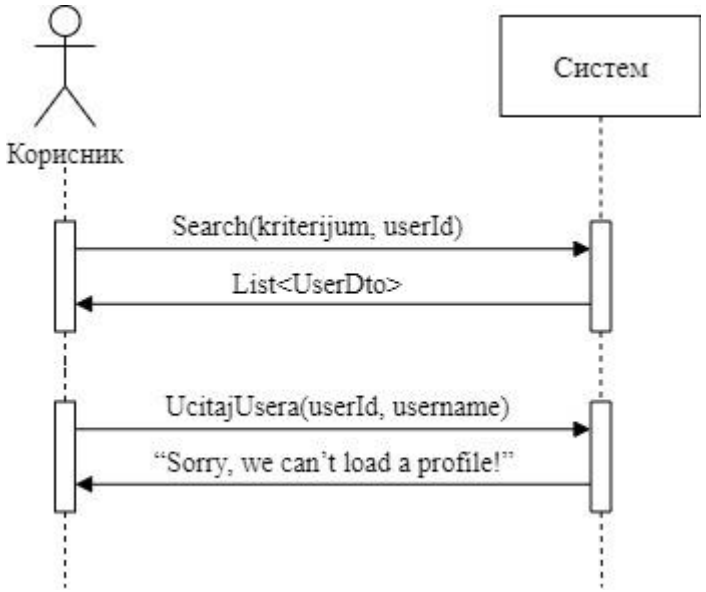
Алтернативна сценарија

2.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



Слика 26. ДС – Претраживање корисника, алтернативни сценарио.
Систем не може да пронађе кориснике.

4.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)



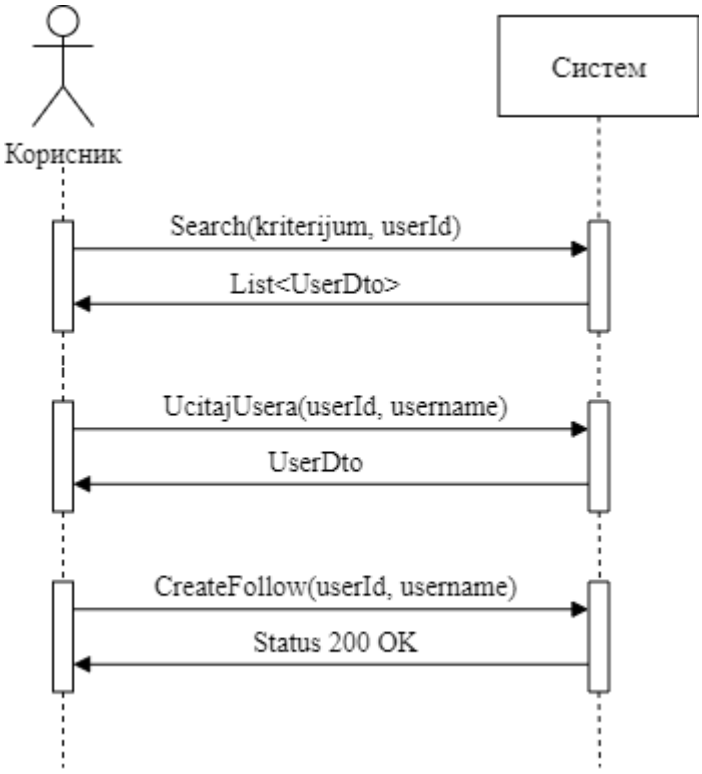
Слика 27. ДС – Претраживање корисника, алтернативни сценарио.
Систем не може да учита профил корисника.

Са наведених секвенцних дијаграма уочавају се две системске операције које треба пројектовати:

1. *signal* **Search**(kriterijum, userId);
2. *signal* **UcitajUsera**(userId, username);

ДС7: Дијаграм секвенце случаја коришћења – Слање захтева за праћење

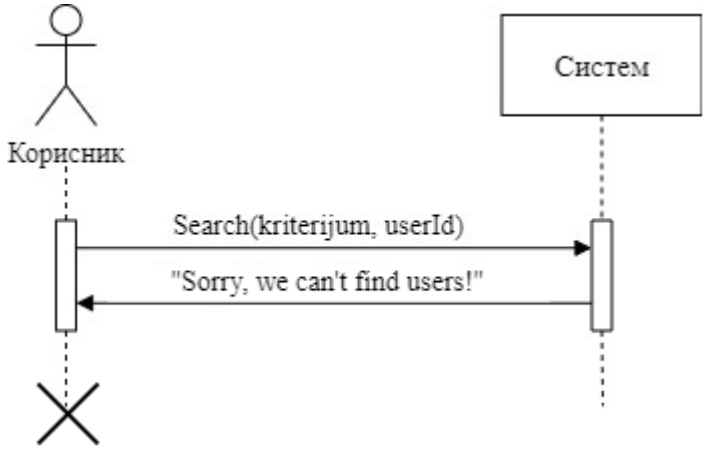
1. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)
2. Систем **приказује** кориснику пронађене кориснике. (ИА)
3. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)
4. Систем **приказује** кориснику профил изабраног корисника. (ИА)
5. Корисник **позива** систем да пошаље захтев изабраном кориснику. (АПСО)
6. Систем **приказује** кориснику да је захтев успешно послат. (ИА)



Слика 28. ДС – Слање захтева за праћење

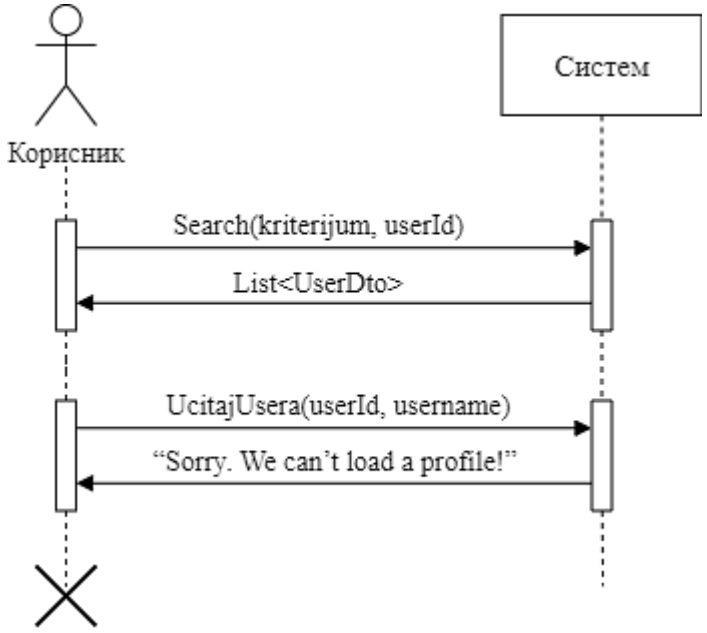
Алтернативна сценарија

2.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



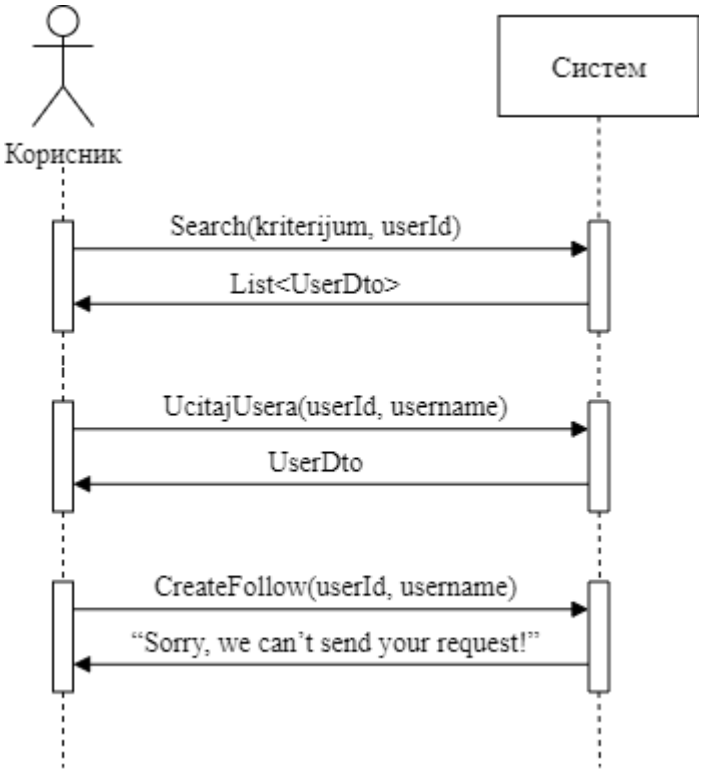
Слика 29. ДС – Слање захтева за праћење, алтернативни сценарио.
Систем не може да пронађе кориснике.

4.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)



Слика 30. ДС – Слање захтева за праћење, алтернативни сценарио.
Систем не може да учита профил корисника.

6.1. Уколико систем не може да запамти захтев за праћење, он **приказује** кориснику поруку: “Sorry, we can’t send your request!”. (ИА)



Слика 31. ДС – Слање захтева за праћење, алтернативни сценарио. Систем не може да пошаље захтев.

Са наведених секвенцних дијаграма уочава се једна системска операција коју треба пројектовати:

1. *signal* **CreateFollow**(userId, username);

ДС8: Дијаграм секвенце случаја коришћења – Отпраћивање

1. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

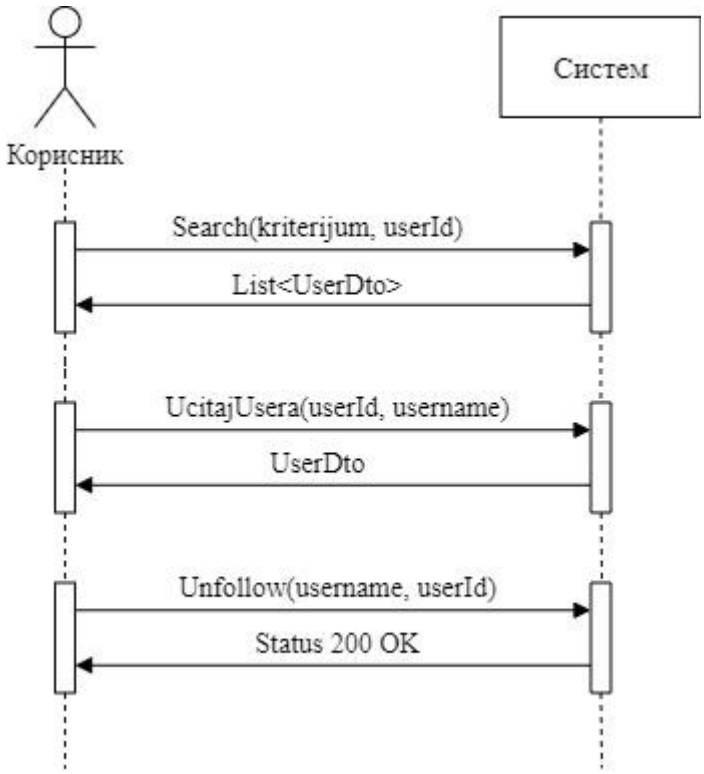
2. Систем **приказује** кориснику пронађене кориснике. (ИА)

3. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

4. Систем **приказује** кориснику профил изабраног корисника. (ИА)

5. Корисник **позива** систем да отпрати изабраног корисника. (АПСО)

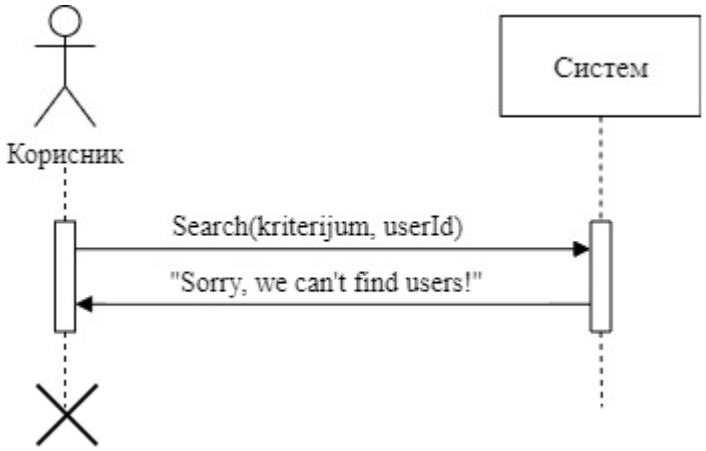
6. Систем **приказује** кориснику да је изабрани корисник успешно отпраћен. (ИА)



Слика 32. ДС – Отпраћивање

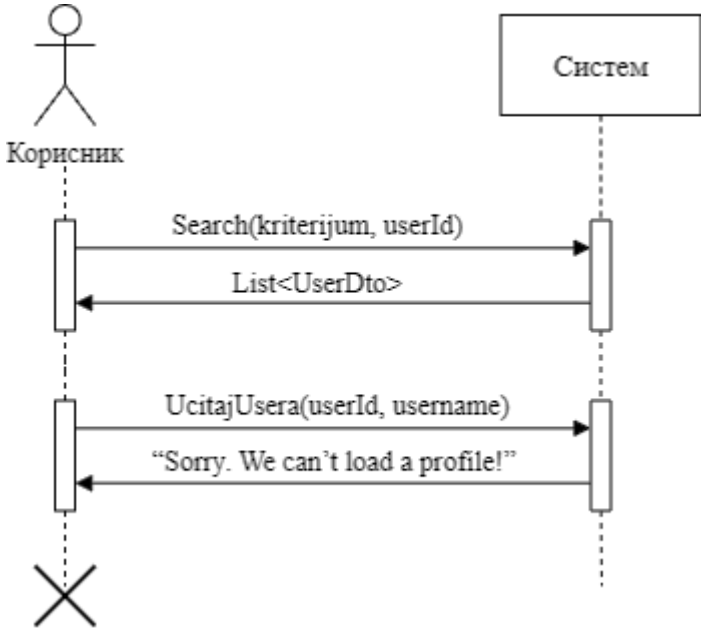
Алтернативна сценарија

2.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



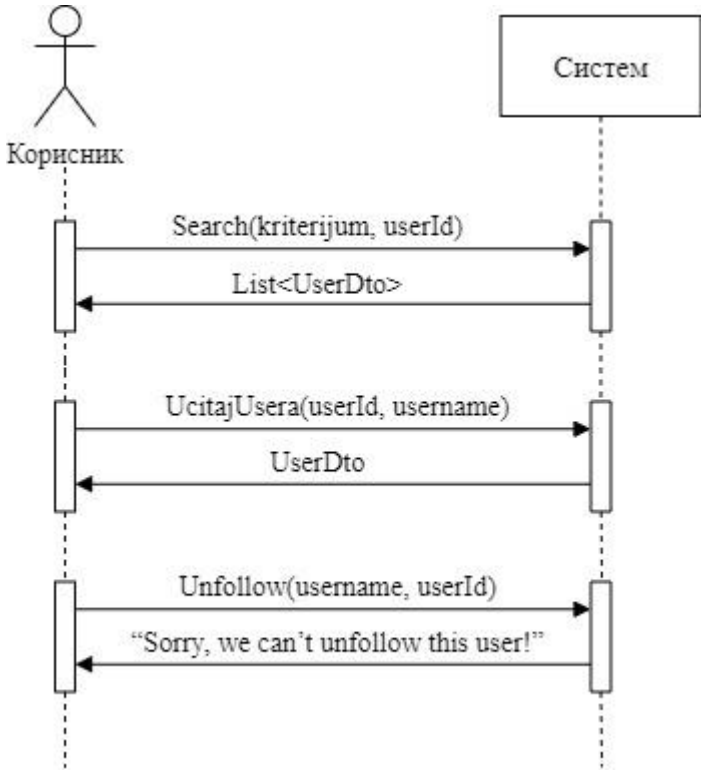
Слика 33. ДС - Отпрaћивање, алтернативни сценарио.
Систем не може да пронађе кориснике.

4.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)



Слика 34. ДС - Отпрaћивање, алтернативни сценарио.
Систем не може да учита профил корисника.

6.1. Уколико систем не може да отпрати корисника, он **приказује** кориснику поруку: “Sorry, we can’t unfollow this user!”. (ИА)



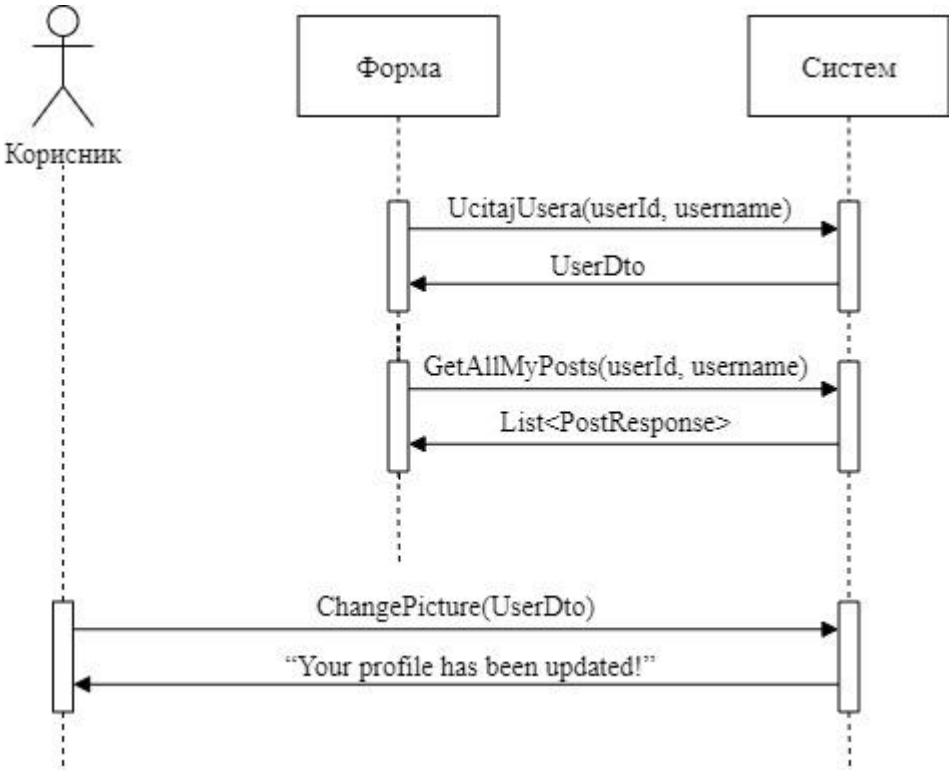
Слика 35. ДС - Отпраћивање, алтернативни сценарио.
Систем не може да отпрати корисника.

Са наведених секвенцних дијаграма уочава се једна системска операција коју треба пројектовати:

1. *signal* **Unfollow**(username, userId);

ДС9: Дијаграм секвенце случаја коришћења – Измена профила

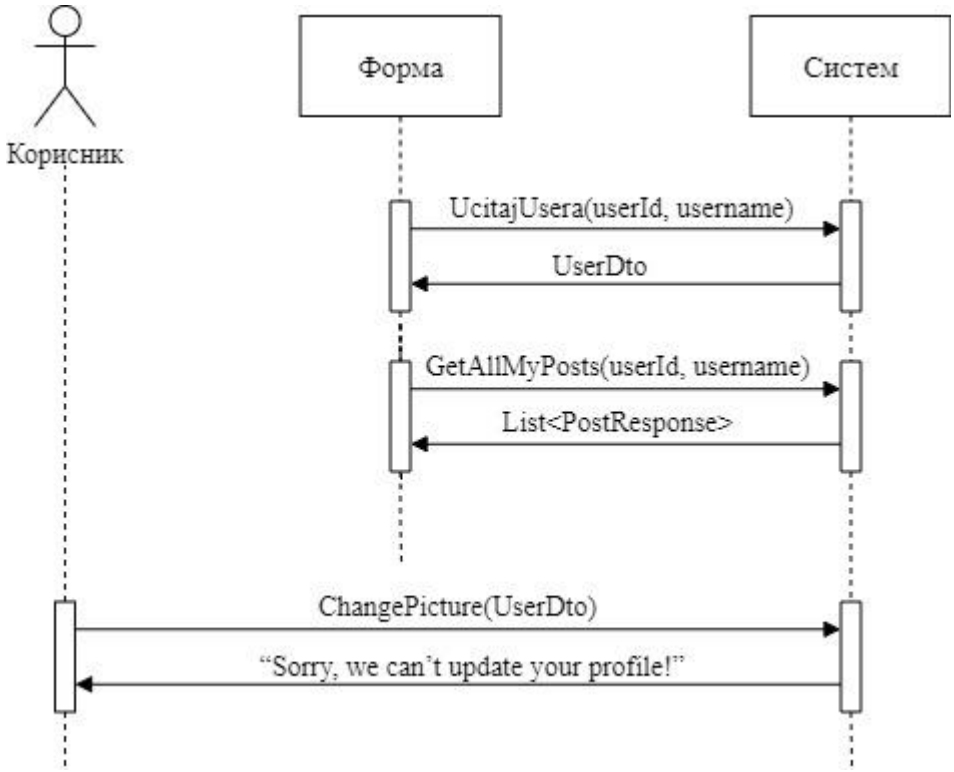
1. Форма **позива** систем да учита задатог корисника. (АПСО)
2. Систем **враћа** форми учитаног корисника. (ИА)
3. Форма **позива** систем да учита све објаве за задатог корисника. (АПСО)
4. Систем **враћа** форми све објаве задатог корисника. (ИА)
3. Корисник **позива** систем да запамти податке о кориснику. (АПСО)
2. Систем **приказује** кориснику поруку: “Your profile has been updated!”. (ИА)



Слика 36. ДС – Измена профила

Алтернативна сценарија

2.1. Уколико систем не може да запамти податке о кориснику, он **приказује** кориснику поруку: “Sorry, we can’t update your profile!”. (ИА)



Слика 37. ДС – Измена профила, алтернативни сценарио.

Са наведених секвенцних дијаграма уочава се једна системска операција коју треба пројектовати:

1. *signal* **ChangePicture**(UserDto);
2. *signal* **GetAllMyPosts**(userId, username)

ДС10: Дијаграм секвенце случаја коришћења – Слање поруке

1. Форма **позива** систем да пронађе све кориснике са којима пријављени корисник има поруке. (АПСО)

2. Систем **враћа** форми све учитане кориснике. (ИА)

3. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

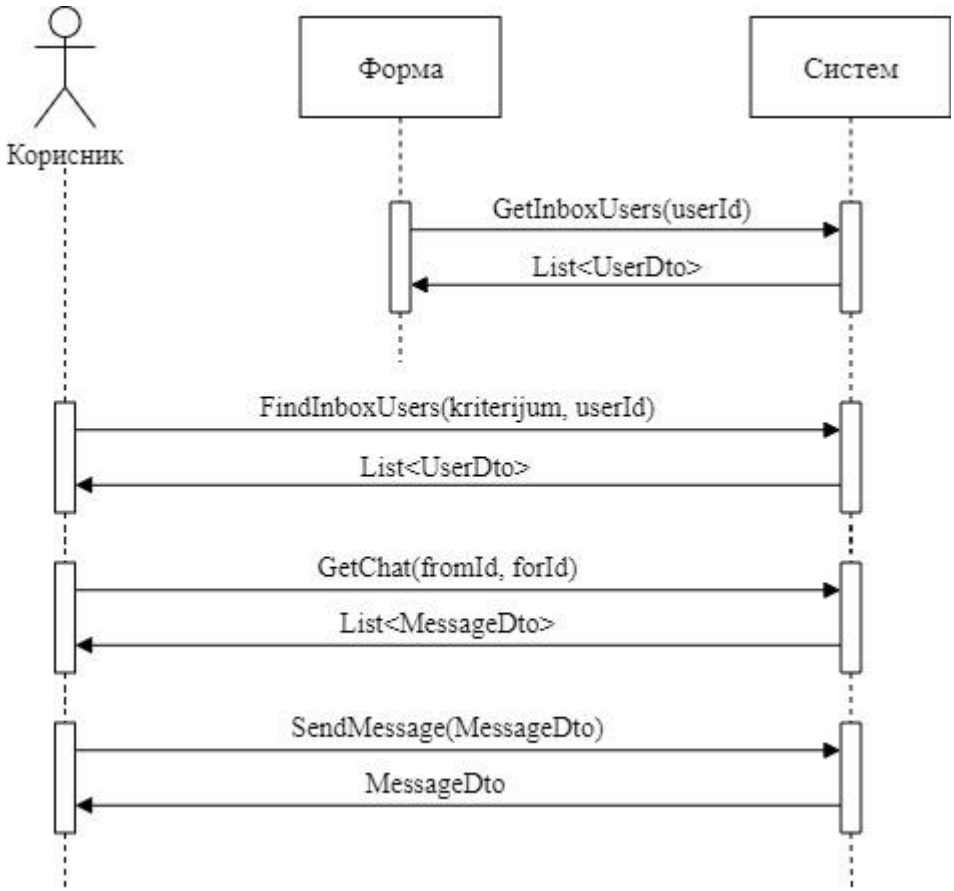
4. Систем **приказује** кориснику пронађене кориснике. (ИА)

5 . Корисник **позива** систем да учита chat са изабраним корисником. (АПСО)

6. Систем **приказује** кориснику поруке. (ИА)

7. Корисник **позива** систем да пошаље поруку. (АПСО)

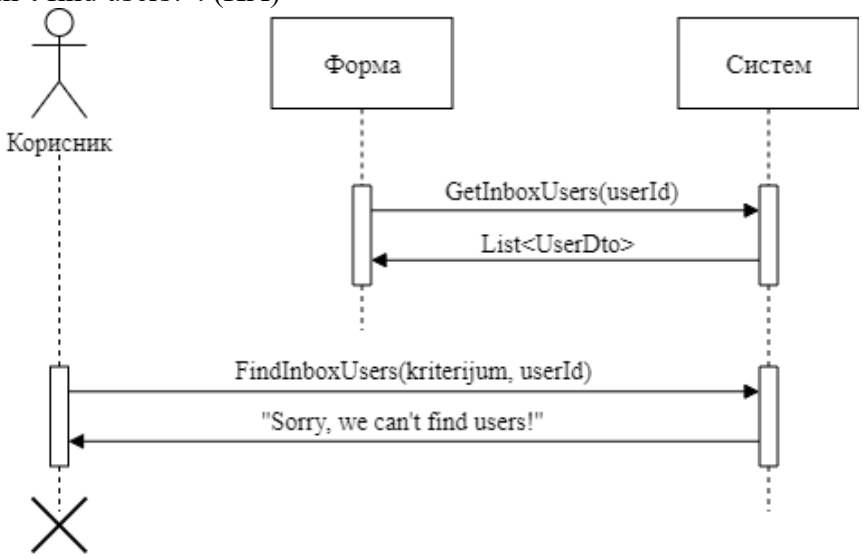
8. Систем **приказује** кориснику послату поруку. (ИА)



Слика 38. ДС – Слање поруке

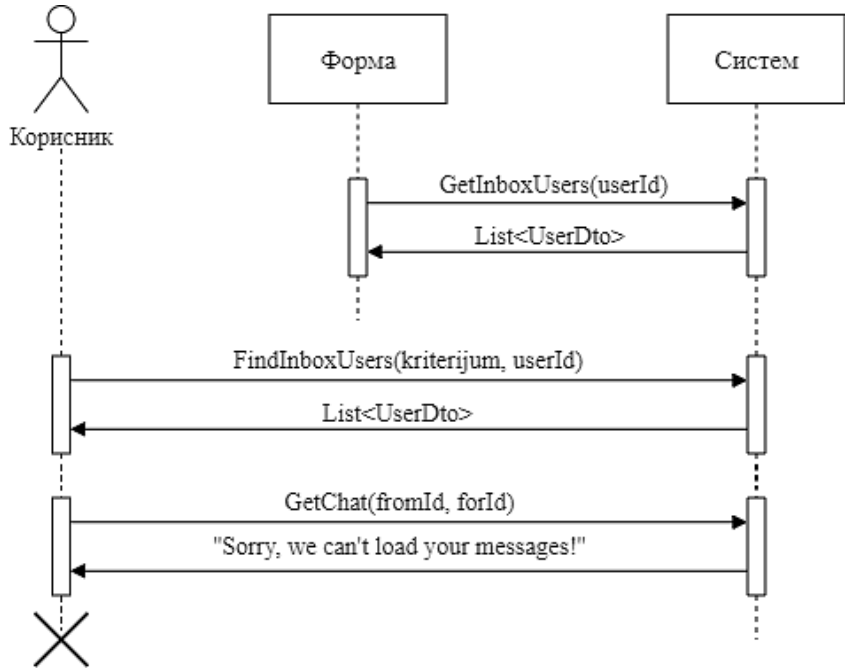
Алтернативна сценарија

2.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



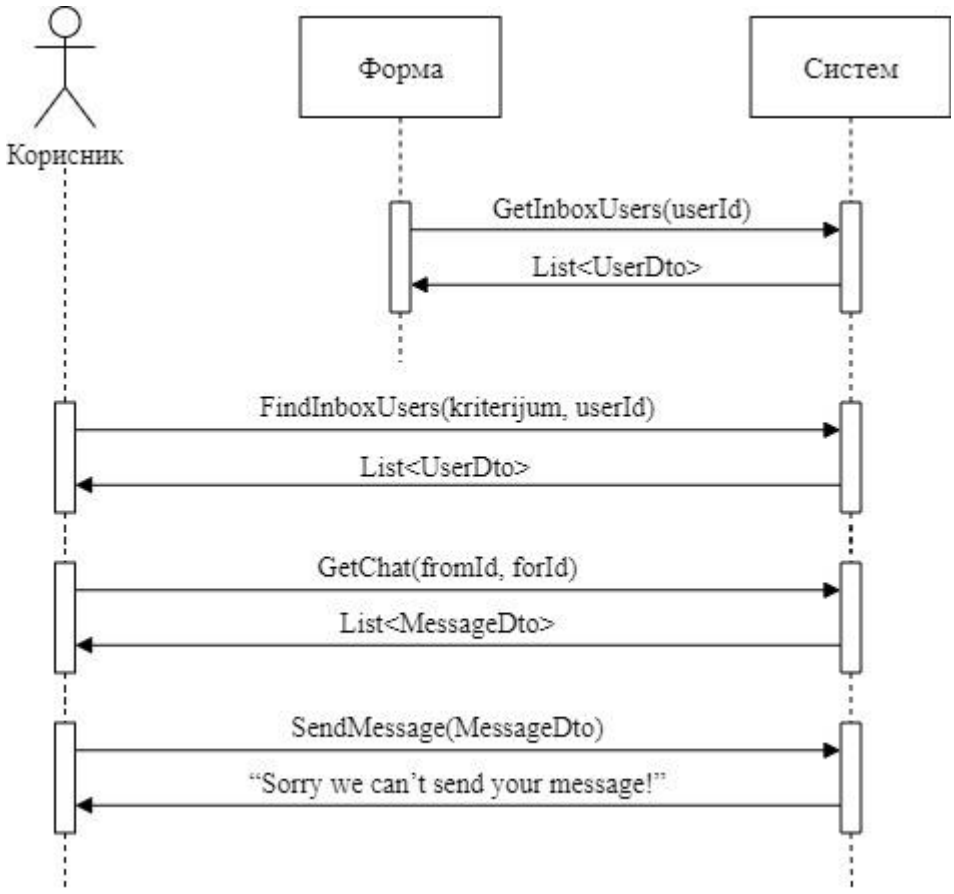
Слика 39. ДС – Слање поруке, алтернативни сценарио.
Систем не може да пронађе кориснике.

4.1. Уколико систем не може да приказе кориснику поруке са изабраним корисником, он **приказује** кориснику поруку: “Sorry, we can’t load your messages!”. (ИА)



Слика 40. ДС – Слање поруке, алтернативни сценарио.
Систем не може учита поруке.

6.1. Уколико систем не може да запамти поруку, он **приказује** кориснику поруку: “Sorry we can’t send your message!”. (ИА)



Слика 41. ДС – Слање поруке, алтернативни сценарио.
Систем не може да пошаље поруку.

Са наведених секвенцних дијаграма уочавају се 4 системске операције које треба пројектовати:

1. *signal* **GetInboxUsers**(userId)
2. *signal* **FindInboxUsers**(kriterijum, userId)
3. *signal* **GetChat**(fromId, forId)
4. *signal* **SendMessage**(MessageDto)

9.2. Закључак на основу дијаграма секвенци случајева коришћења

Као резултат анализе сценарија добијено је укупно 17 системских операција које треба пројектовати:

1. *Signal* LogIn(LoginDto, UserDto);
2. *Signal* Register(RegisterDto);
3. *Signal* Create(PostRequest);
4. *Signal* GetAllForHome(userId, numOfPosts, List<PostResponse>);
5. *Signal* LikeIt(postId, username);
6. *Signal* GetComments(postId, List<CommentResponse>);
7. *Signal* PostComment(CommentRequest, CommentResponse);
8. *Signal* Search(kriterijum, userId, List<UserDto>);
9. *Signal* UcitajUsera(userId, username, UserDto);
10. *Signal* CreateFollow(userId, username);
11. *Signal* Unfollow(username, userId);
12. *Signal* ChangePicture(UserDto);
13. *Signal* GetAllMyPosts(userId, username, List<PostResponse>);
14. *Signal* GetInboxUsers(userId, List<UserDto>);
15. *Signal* FindInboxUsers(kriterijum, userId, List<UserDto>);
16. *Signal* GetChat(fromId, forId, List<Message>);
17. *Signal* SendMessage(MessageDto, MessageDto);

9.3. Понашање софтверског система – Дефинисање уговора о системским операцијама

1. Уговор УГ1: LogIn

Операција: LogIn(LoginDto, UserDto):signal;

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је улогован.

2. Уговор УГ2: Register

Операција: Register(RegisterDto):signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је регистрован.

3. Уговор УГ3: Create

Операција: Create(PostRequest):signal;

Веза са СК: СК3

Предуслови: Вредносна и структурна ограничења над објектом **Post** морају бити задовољена.

Постуслови: Објава је запамћена.

4. Уговор УГ4: GetAllForHome

Операција: GetAllForHome(userId, numOfPosts, List<PostResponse>):signal;

Веза са СК: СК4

Предуслови:

Постуслови:

5. **Уговор УГ5:** LikeIt

Операција: LikeIt(postId, username):signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Reaction** морају бити задовољена.

Постуслови: Реакција је запамћена.

6. **Уговор УГ6:** GetComments

Операција: GetComments(postId, List<CommentResponse>):signal;

Веза са СК: СК5

Предуслови:

Постуслови:

7. **Уговор УГ7:** PostComment

Операција: PostComment(CommentRequest, CommentResponse):signal;

Веза са СК: СК5

Предуслови: Вредносна и структурна ограничења над објектом **Comment** морају бити задовољена.

Постуслови: Коментар је запамћен.

8. **Уговор УГ8:** Search

Операција: Search(kriterijum, userId, List<UserDto>):signal;

Веза са СК: СК6

Предуслови:

Постуслови:

9. **Уговор УГ9:** UcitajUsera

Операција: UcitajUsera(userId, username, UserDto):signal;

Веза са СК: СК6

Предуслови:

Постуслови:

10. Уговор УГ10: CreateFollow

Операција: CreateFollow(userId, username):signal;

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **FollowNotification** морају бити задовољена.

Постуслови: Захтев је запамћен.

11. Уговор УГ11: Unfollow

Операција: Unfollow(username, userId):signal;

Веза са СК: СК8

Предуслови: Структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је отпраћен.

12. Уговор УГ12: ChangePicture

Операција: ChangePicture(UserDto):signal;

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Профилна слика корисника је промењена.

13.Уговор УГ13: GetAllMyPosts

Операција: GetAllMyPosts(userId, username, List<PostResponse>):signal;

Веза са СК: СК9

Предуслови:

Постуслови:

14. Уговор УГ14: GetInboxUsers

Операција: GetInboxUsers(userId, List<UserDto>):signal;

Веза са СК: СК10

Предуслови:

Постуслови:

15. Уговор УГ15: FindInboxUsers

Операција: FindInboxUsers(kriterijum, userId, List<UserDto>):signal;

Веза са СК: СК10

Предуслови:

Постуслови:

16. Уговор УГ16: GetChat

Операција: GetChat(fromId, forId, List<Message>):signal;

Веза са СК: СК10

Предуслови:

Постуслови:

17. Уговор УГ17: SendMessage

Операција: SendMessage(MessageDto, MessageDto):signal;

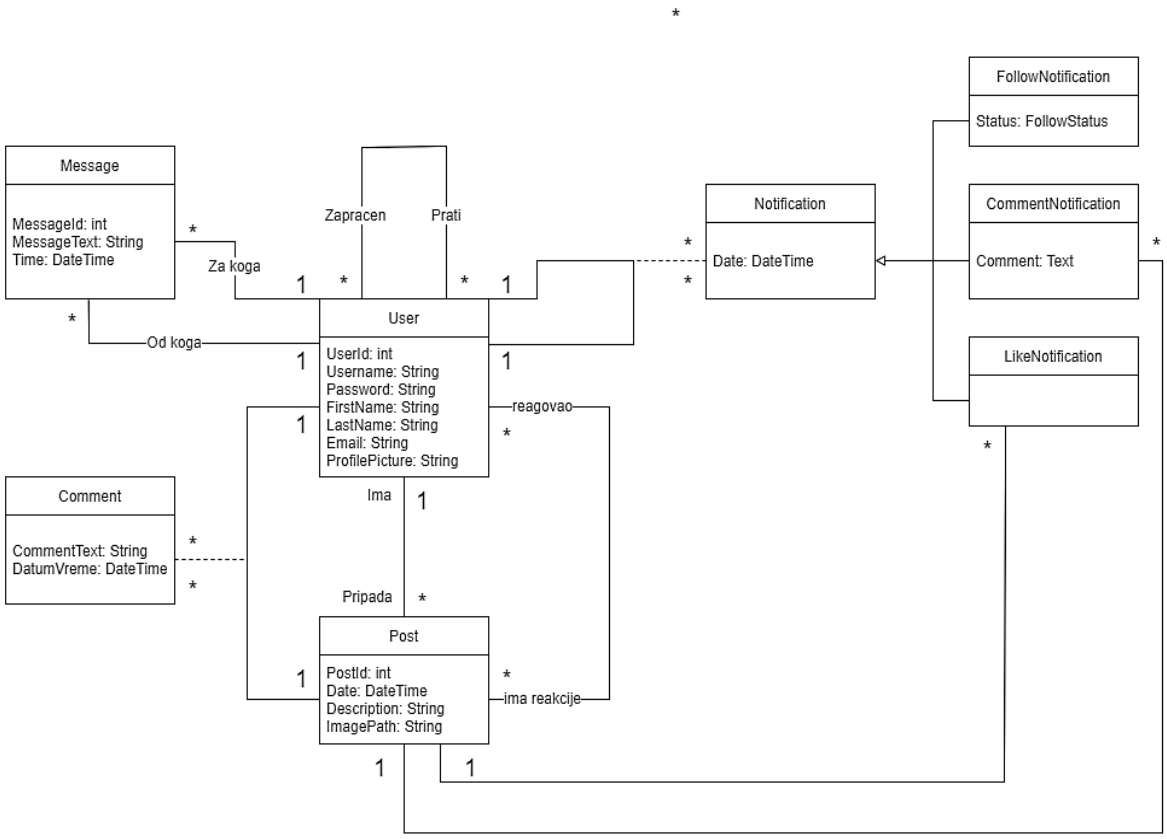
Веза са СК: СК10

Предуслови: Вредносна и структурна ограничења над објектом **Message** морају бити задовољена.

Постуслови: Порука је запамћена.

9.4. Структура софтверског система - Концептуални модел

На концептуалном моделу софтверског система представљене су концептуалне класе и везе, односно асоцијације између њих. На основу њих биће направљене доменске класе у програму.



Слика 42. Концептуални модел

10. Фаза пројектовања

Фаза пројектовања описује физичку структуру и понашање софтверског система (архитектуру софтверског система). Најчешће коришћена архитектура је тронивојска архитектура која се састоји из следећих нивоа:

- корисничког интерфејса односно екранских форми
- апликационе логике
- складишта података

10.1. Пројектовање складишта података

На основу концептуалног модела направљен је релациони модел података:

User(UserId, Username, Password, FirstName, LastName, Email, ProfilePicture)

UserUser(*FollowingId, FollowerId*)

Post(PostId, Date, Description, ImagePath, *UserId*)

Reaction(*UserId, PostId*)

Comment(*PostId, UserId, DatumVreme*, CommentText)

Message(MessageId, MessageText, Time, *FromWhoId*, *ForWhoId*)

Notification(*FromWhoId*, *ForWhoId*, DateTime)

LikeNotification(*FromWhoId, ForWhoId, DateTime*, *PostId*)

CommentNotification(*FromWhoId, ForWhoId, DateTime*, Comment, *PostId)*

FollowNotification(*FromWhoId, ForWhoId, DateTime*, Status)

Табела User		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	**Име**	**Тип атрибута**	**Вредност атрибута**	**Међузависност атрибута једне табеле**	**Међузависност атрибута више табела**	**INSERT /** **UPDATE CASCADE UserUser, Post, Reaction, Comment, Message, Notification, LikeNotification, CommentNotificatio, FollowNotification** **DELETE RESTRICTED UserUser, Post, Reaction, Comment, Message, Notification, LikeNotification, CommentNotificatio, FollowNotification**
	UserId	Integer	NotNull AND >0			
	Username	String	NotNull			
	Password	String	NotNull			
	FirstName	String	NotNull			
	LastName	String	NotNull			
	Email	String	NotNull			
	Profile Picture	String	NotNull			

Табела 1. User

Табела UserUser		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	**Име**	**Тип атрибута**	**Вредност атрибута**	**Међузависност атрибута једне табеле**	**Међузависност атрибута више табела**	**INSERT RESTRICTED User** **UPDATE RESTRICTED User** **DELETE /**
	FollowingId	Integer	NotNull AND >0			
	FollowerId	Integer	NotNull AND >0			

Табела 2. UserUser

Табела Post		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	**Име**	**Тип атрибута**	**Вредност атрибута**	**Међузависност атрибута једне табеле**	**Међузависност атрибута више табела**	**INSERT RESTRICTED User** **UPDATE RESTRICTED User** **UPDATE CASCADE CommentNotification, LikeNotification, Comment, Reaction** **DELETE RESTRICTED CommentNotification, LikeNotification, Comment, Reaction**
	PostId	Integer	NotNull AND >0			
	Date	DateTime	NotNull			
	Description	String	NotNull			
	ImagePath	String	NotNull			
	UserId	Integer	NotNull AND >0			

Табела 3. Post

Табела Reaction		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	**Име**	**Тип атрибута**	**Вредност атрибута**	**Међузависност атрибута једне табеле**	**Међузависност атрибута више табела**	**INSERT RESTRICTED User, Post** **UPDATE RESTRICTED User, Post** **DELETE /**
	UserId	Integer	NotNull AND >0			
	PostId	Integer	NotNull AND >0			

Табела 4. Reaction

Табела Comment		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	**Име**	**Тип атрибута**	**Вредност атрибута**	**Међузависност атрибута једне табеле**	**Међузависност атрибута више табела**	**INSERT RESTRICTED User, Post** **UPDATE RESTRICTED User, Post** **DELETE /**
	PostId	Integer	NotNull AND >0			
	UserId	Integer	NotNull AND >0			
	CommentText	String	NotNull			
	DatumVreme	DateTime	NotNull			

Табела 5. Comment

Табела Message		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	**Име**	**Тип атрибута**	**Вредност атрибута**	**Међузависност атрибута једне табеле**	**Међузависност атрибута више табела**	**INSERT RESTRICTED User** **UPDATE RESTRICTED User** **DELETE /**
	MessageId	Integer	NotNull AND >0			
	MessageText	String	NotNull			
	Time	DateTime	NotNull			
	FromWhoId	Integer	NotNull AND >0			
	ForWhoId	Integer	NotNull AND >0			

Табела 6. Message

Табела Notification		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	**Име**	**Тип атрибута**	**Вредност атрибута**	**Међузависност атрибута једне табеле**	**Међузависност атрибута више табела**	**INSERT RESTRICTED User** **UPDATE RESTRICTED User** **DELETE /**
	FromWhoId	Integer	NotNull AND >0			
	ForWhoId	String	NotNull AND >0			
	DateTime	DateTime	NotNull			

Табела 7. Notification

Табела LikeNotification		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED User, Post UPDATE RESTRICTED User, Post DELETE /
	FromWhoId	Integer	NotNull AND >0			
	ForWhoId	Integer	NotNull AND >0			
	DateTime	DateTime	NotNull			
	PostId	Integer	NotNull AND >0			

Табела 8. LikeNotification

Табела CommentNotification		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED User, Post UPDATE RESTRICTED User, Post DELETE /
	FromWhoId	Integer	NotNull AND >0			
	ForWhoId	Integer	NotNull AND >0			
	DateTime	DateTime	NotNull			
	Comment	String	NotNull			
	PostId	Integer	NotNull AND >0			

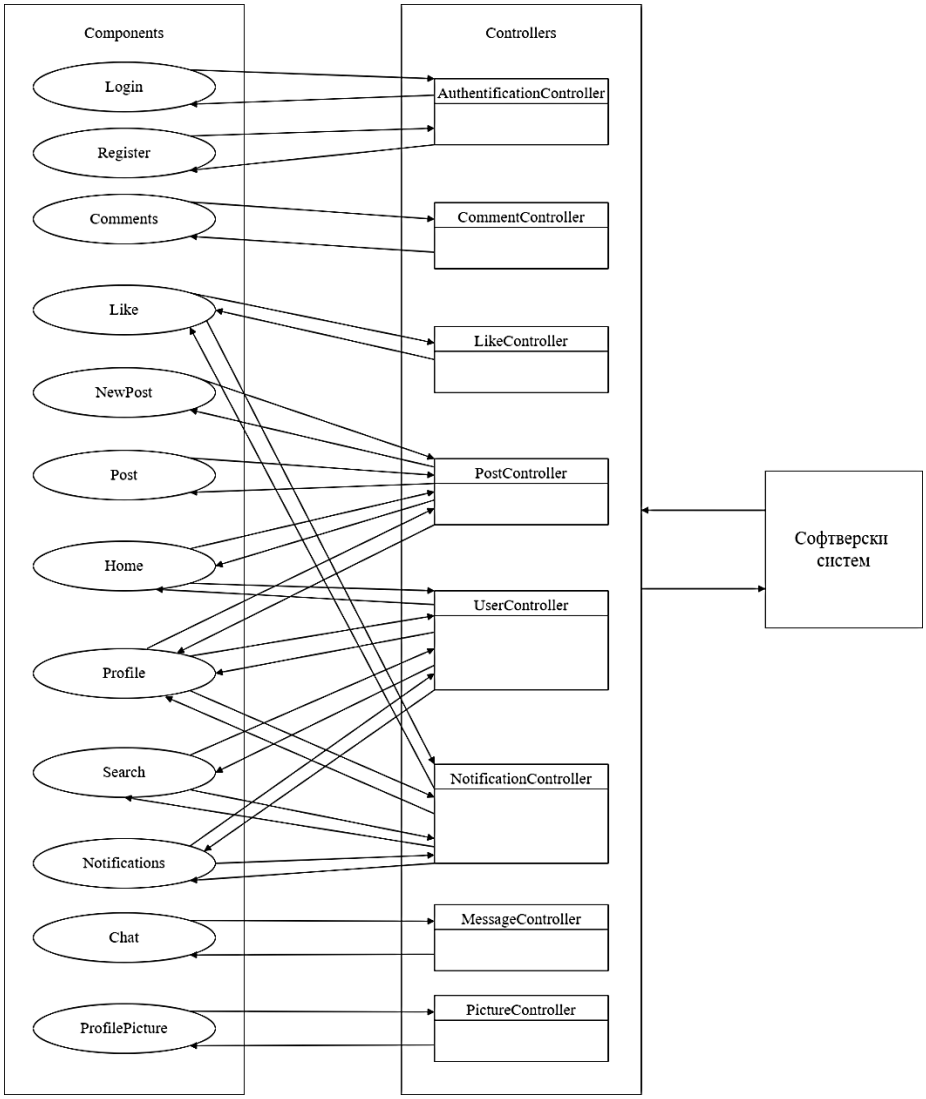
Табела 9. CommentNotification

Табела FollowNotification		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED User UPDATE RESTRICTED User DELETE
	FromWhoId	Integer	NotNull AND >0			
	ForWhoId	Integer	NotNull AND >0			
	DateTime	DateTime	NotNull			
	Status	FollowStatus	NotNull			

Табела 10. FollowNotification

10.2. Пројектовање корисничког интерфејса

Кориснички интерфејс представља реализацију улаза и/или излаза софтверског система. Кориснички интерфејс у овом раду представљају веб странице које су задужене за приказивање података који долазе од сервера. Кориснички интерфејс је такође задужен и за прихватање података које корисник уноси, као и догађаја које корисник направи, и за њихово прослеђивање серверу на даљу обраду.



Слика 43. Повезаност контролера и корисничког интерфејса у архитектури софтверског система

СК1: Случај коришћења - Пријављивање на друштвену мрежу

Назив СК

Пријављивање на друштвену мрежу

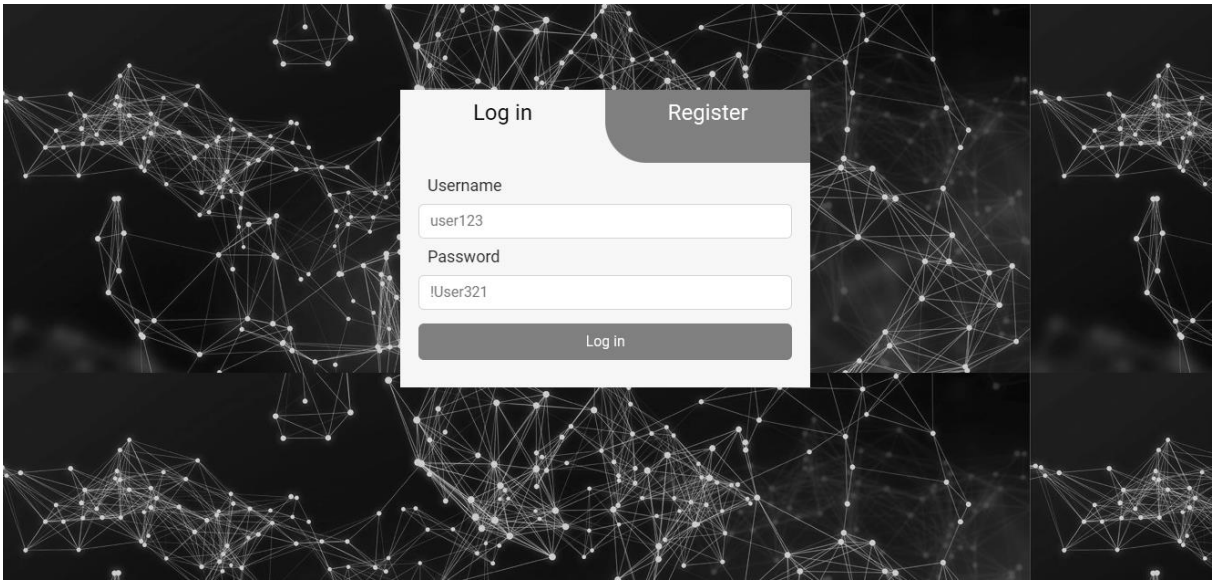
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

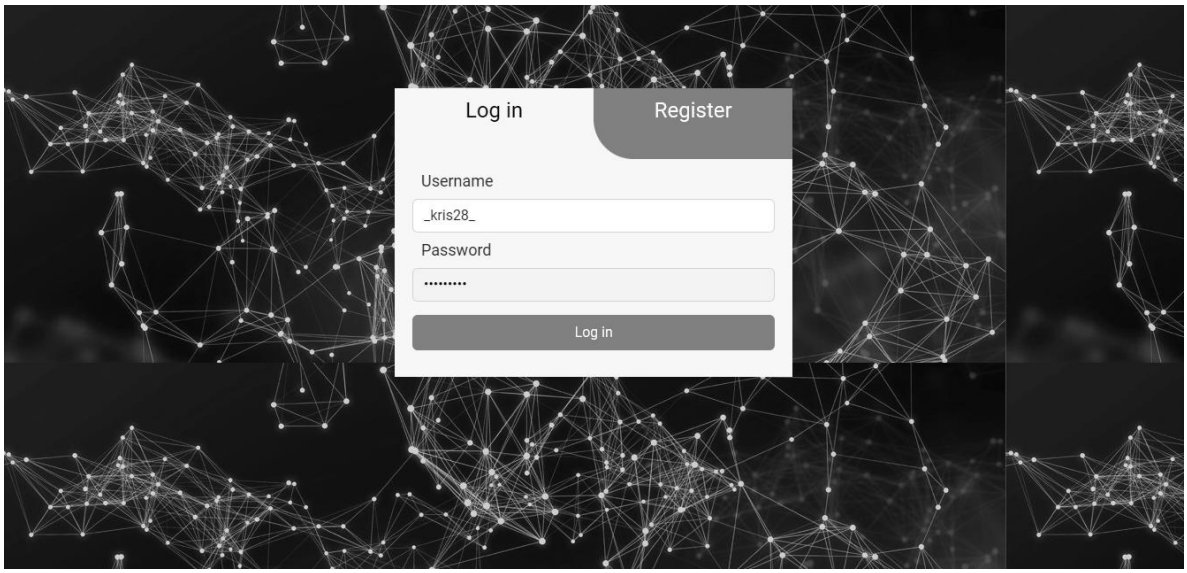
Предуслов: Систем је укључен и приказује форму за пријављивање на систем.



Слика 44. Форма за пријаву на систем

Основни сценарио СК

1. Корисник **уноси** податке за пријављивање (корисничко име и лозинку). (АПУСО)



Слика 45. Попуњена форма за пријаву на систем

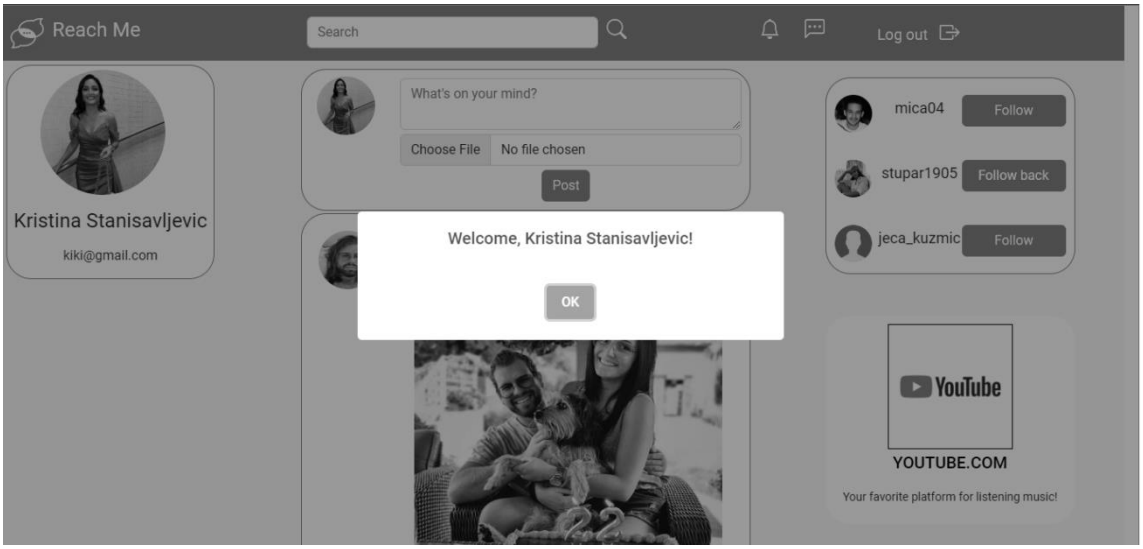
2. Корисник **контролише** да ли је коректно унео корисничко име и лозинку. (АНСО)

3. Корисник **позива** систем да га пријави. (АПСО)

Опис акције: Кликом на дугме „Log in“ корисник позива системску операцију LogIn(UserDto).

4. Систем **проверава** податке о кориснику. (СО)

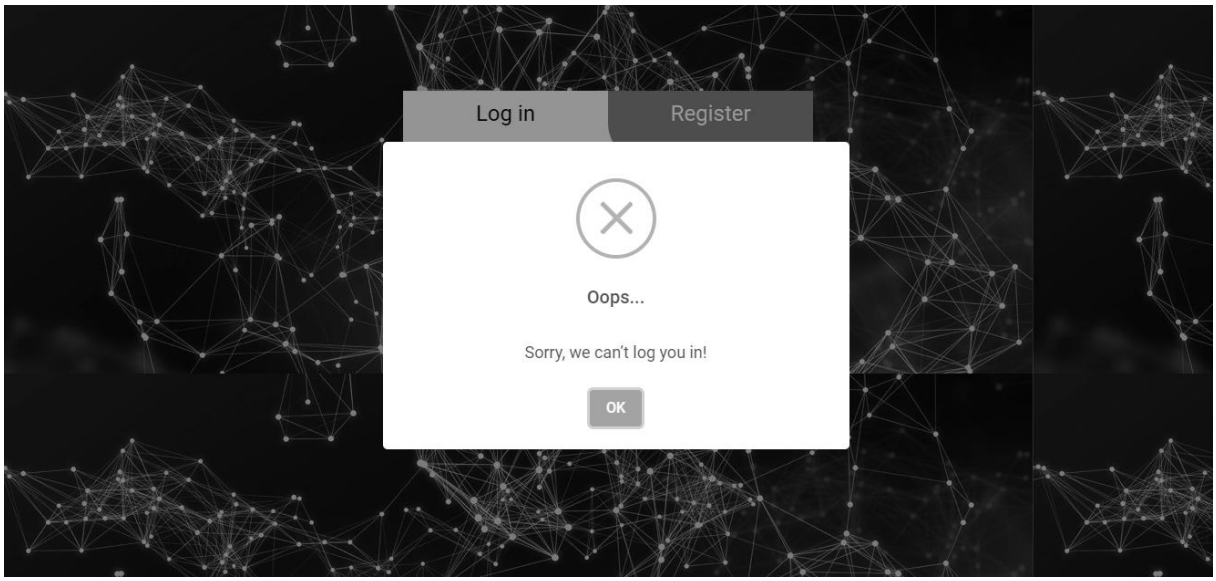
5. Систем **приказује** кориснику почетну страну и поруку: “Welcome, ime prezime!“. (ИА)



Слика 46. Успешна пријава на систем

Алтернативна сценарија

5.1. Уколико систем не може да нађе корисника, он **приказује** кориснику поруку: “Sorry, we can’t log you in!”. (ИА)



Слика 47. Неуспешна пријава

СК2: Случај коришћења – Креирање корисничког налога

Назив СК

Креирање корисничког налога

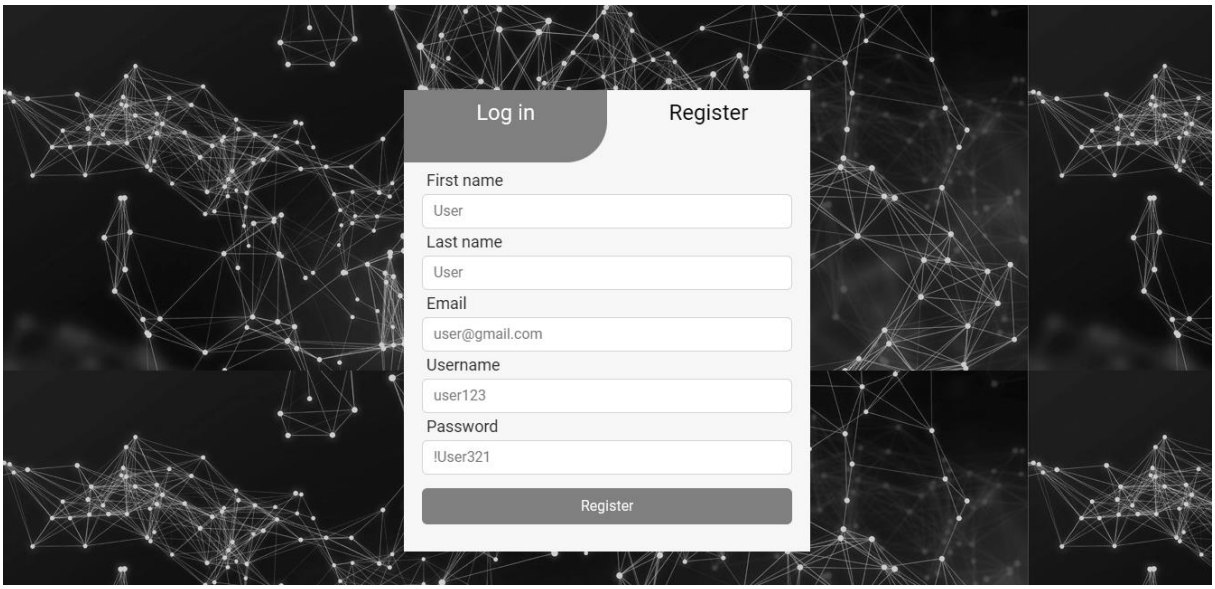
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

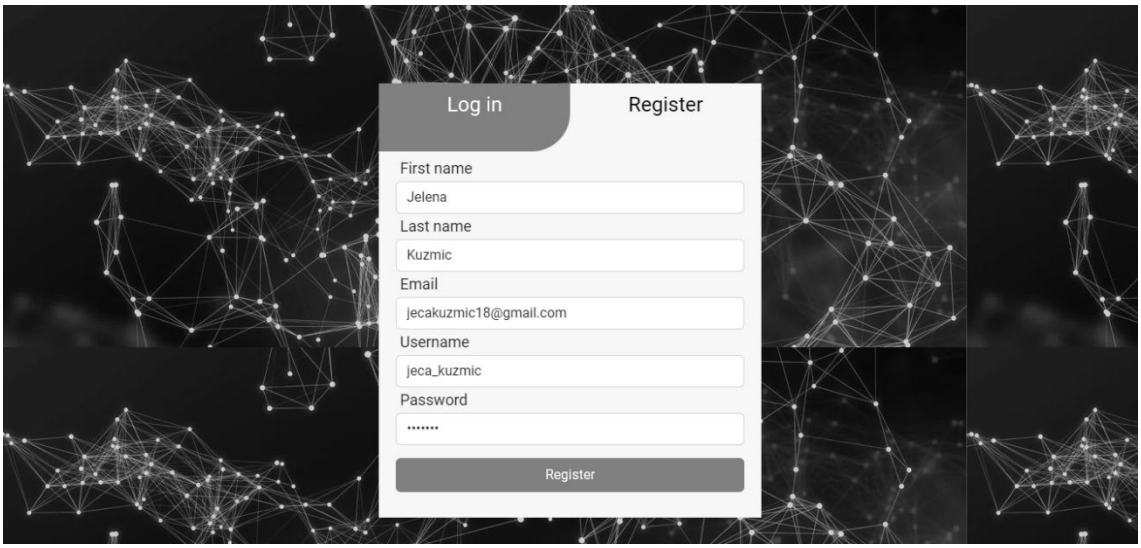
Предуслов: Систем је укључен и приказује форму за регистрацију на систем.



Слика 48. Форма за регистрацију

Основни сценарио СК

1. Корисник **уноси** основне податке за креирање корисничког налога. (АПУСО)



Слика 49. Попуњена форма за регистрацију

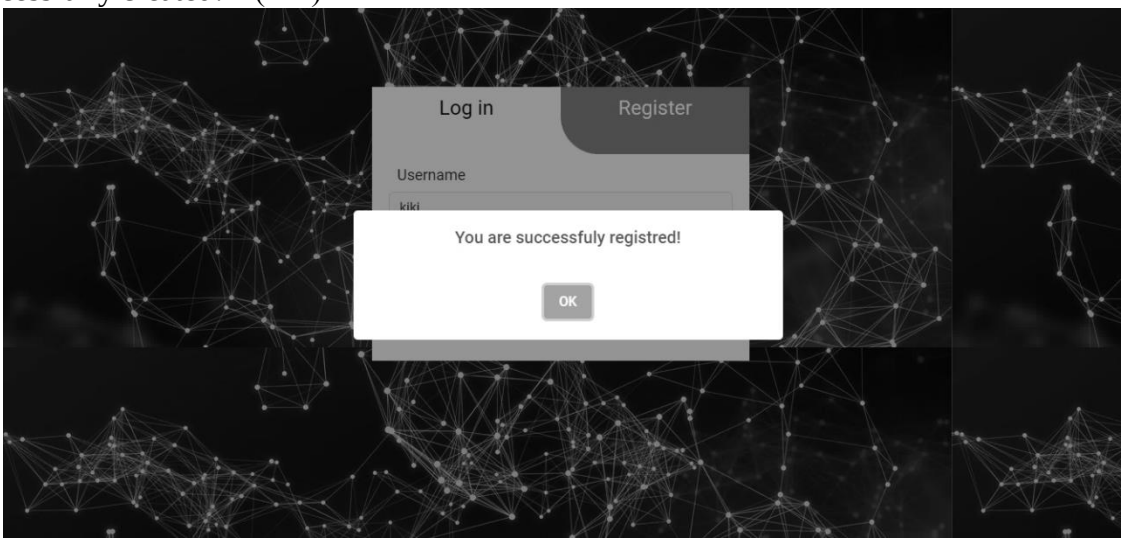
2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)

3. Корисник **позива** систем да га региструје. (АПСО)

Опис акције: Кликом на дугме „Register“ корисник позива системску операцију Register(RegisterDto).

4. Систем **памти** податке о кориснику. (СО)

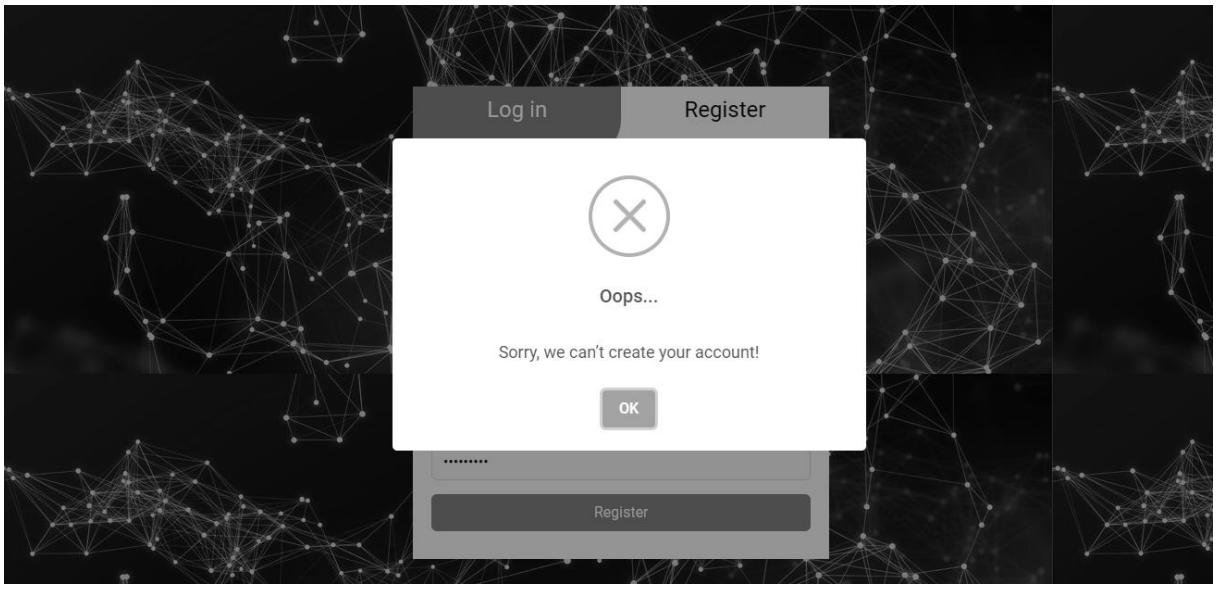
5. Систем **приказује** кориснику страницу за пријаву и поруку: “Your account is successfully created!“. (ИА)



Слика 50. Успешна регистрација

Алтернативна сценарија

5.1. Уколико систем не може да региструје корисника, он **приказује** кориснику поруку: “Sorry, we can’t create your account!”. (ИА)



Слика 51. Неуспешна регистрација

СКЗ: Случај коришћења – Креирање нове објаве

Назив СК

Креирање нове објаве

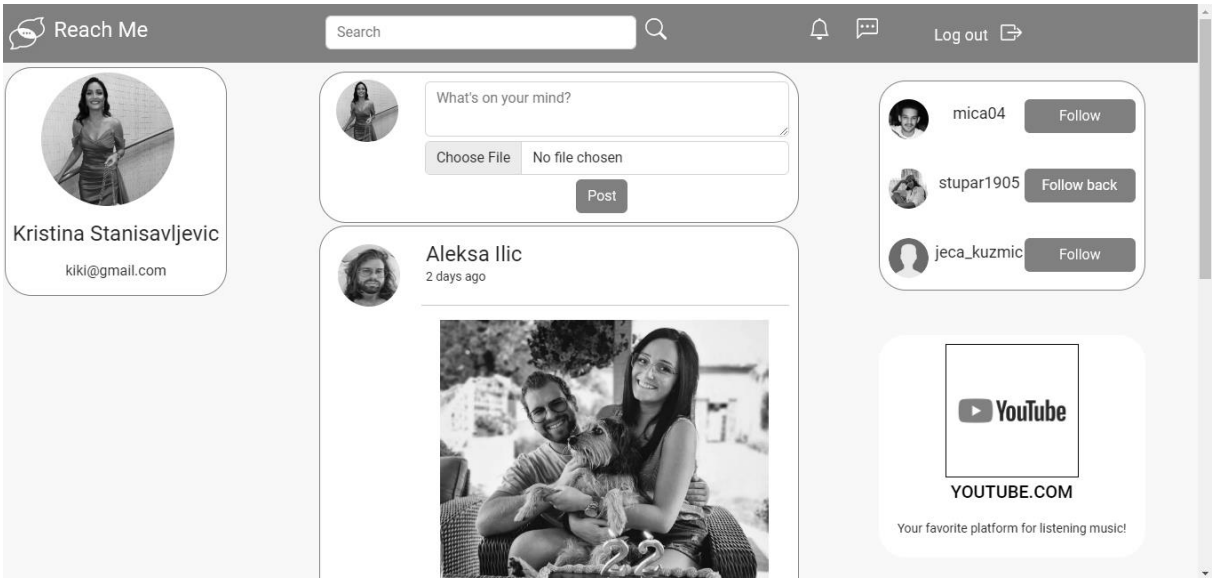
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

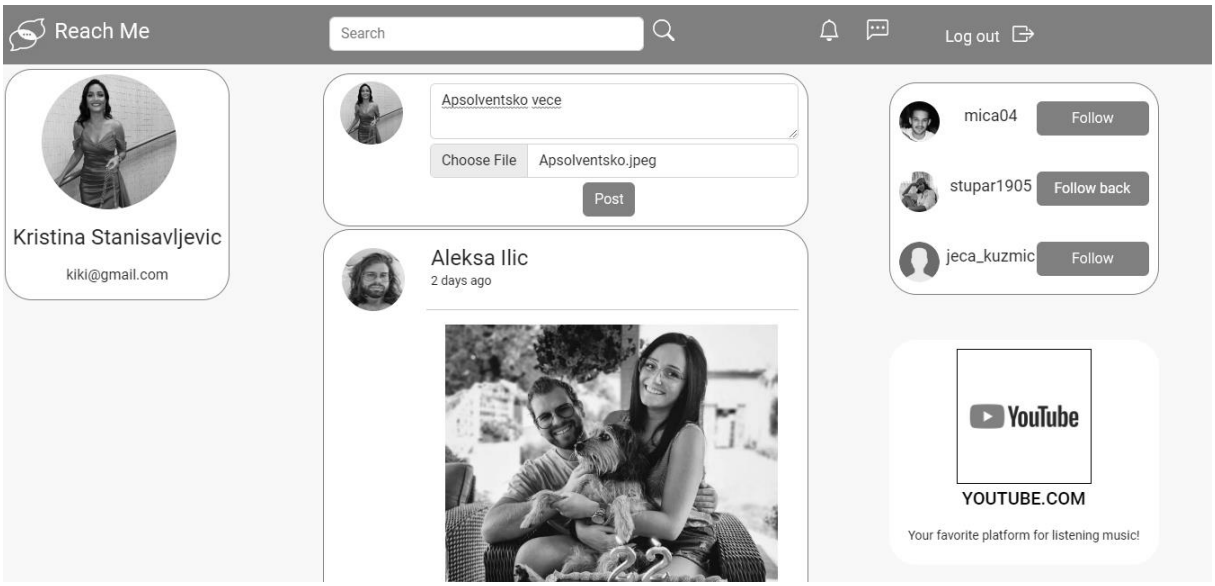
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника, на којој се налази форма за унос нове објаве.



Слика 52. Почетна страница на којој се налази форма за креирање нове објаве

Основни сценарио СК

1. Корисник **уноси** податке о објави. (АПУСО)



Слика 53. Унети подаци о новој објави

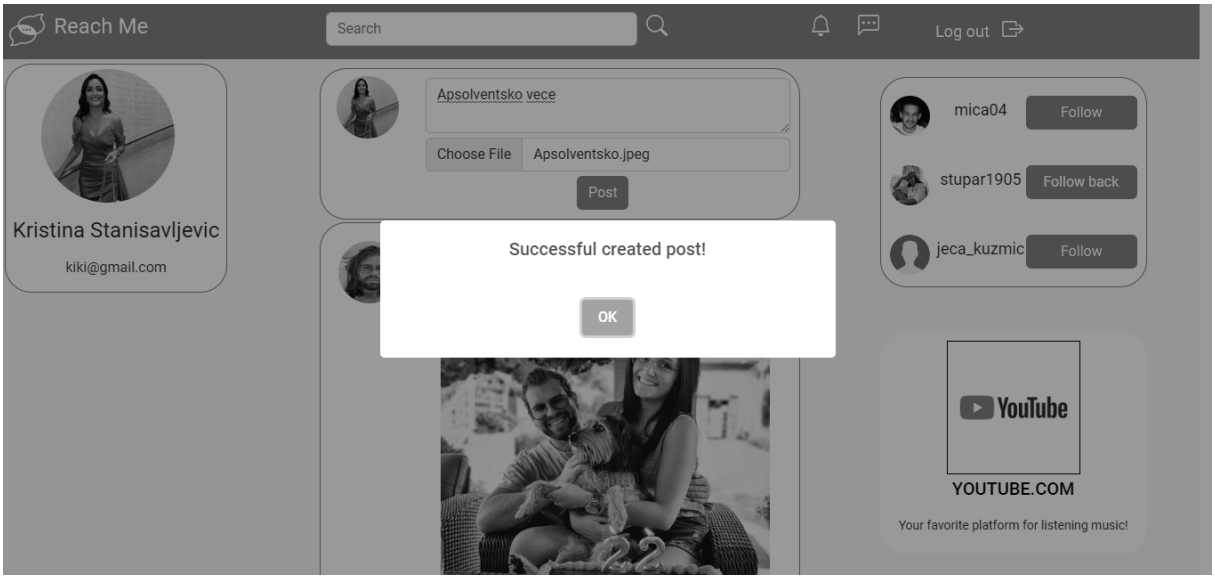
2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)

3. Корисник **позива** систем да запамти податке о објави. (АПСО)

Опис акције: Кликом на дугме „Post“ корисник позива системску операцију Create(PostRequest).

4. Систем **памти** податке о објави. (СО)

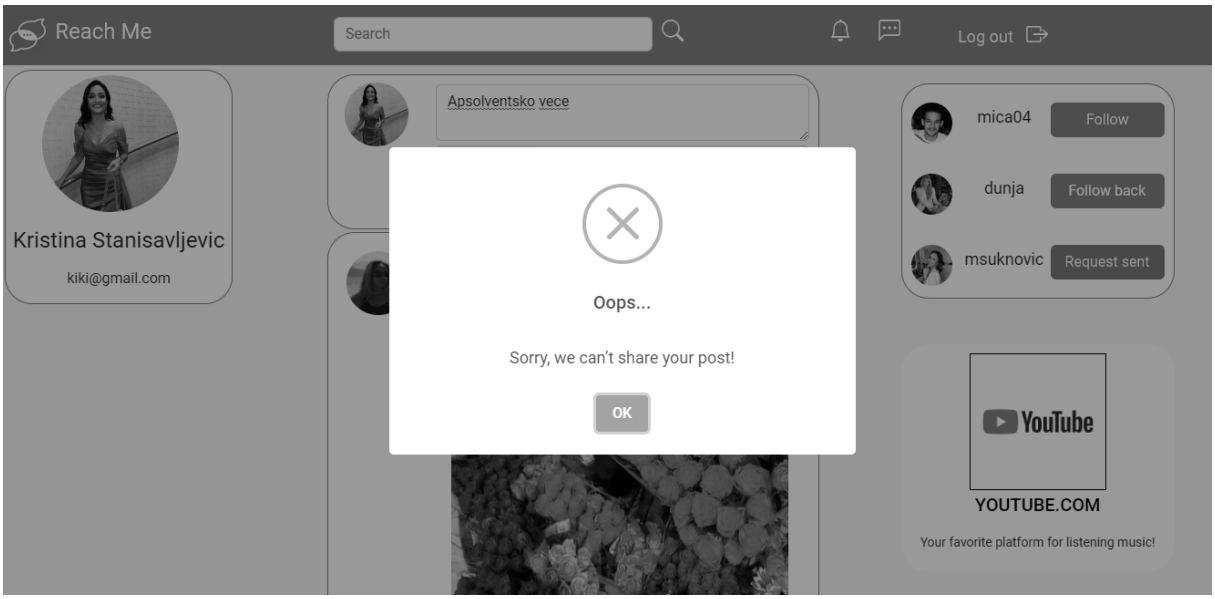
5. Систем **приказује** кориснику поруку: “Your post has been shared!”. (ИА)



Слика 54. Успешно креирана објава

Алтернативна сценарија

5.1. Уколико систем не може да запамти податке о објави, он **приказује** кориснику поруку: “Sorry, we can’t share your post!”. (ИА)



Слика 55. Неуспешно креирана објава

СК4: Случај коришћења – Реаговање на објаву

Назив СК

Реаговање на објаву

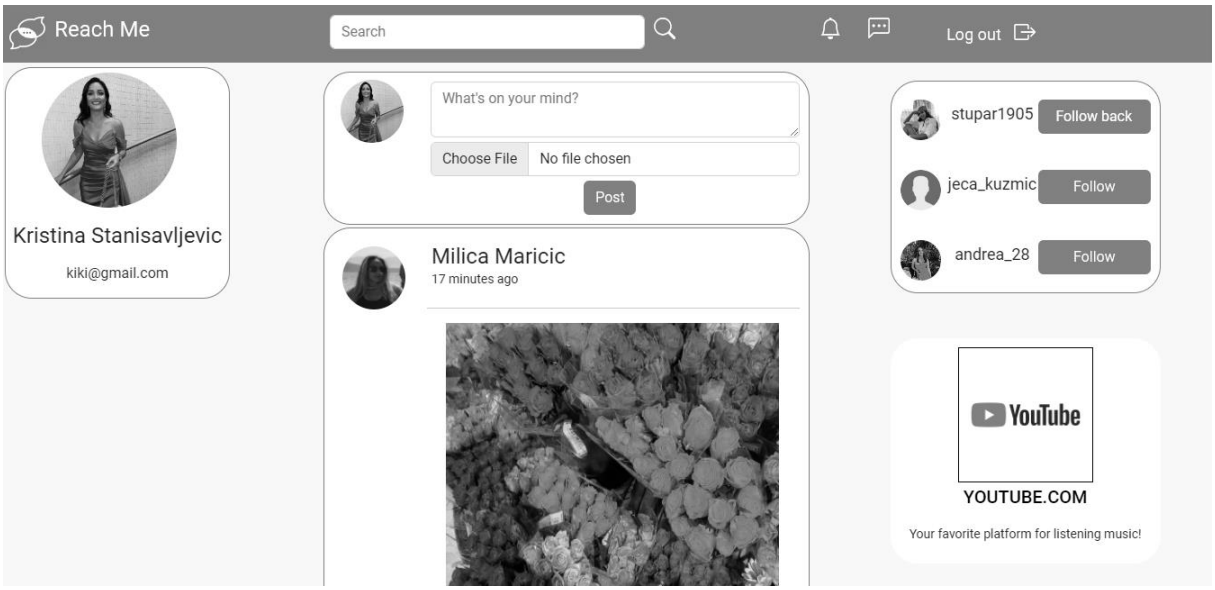
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

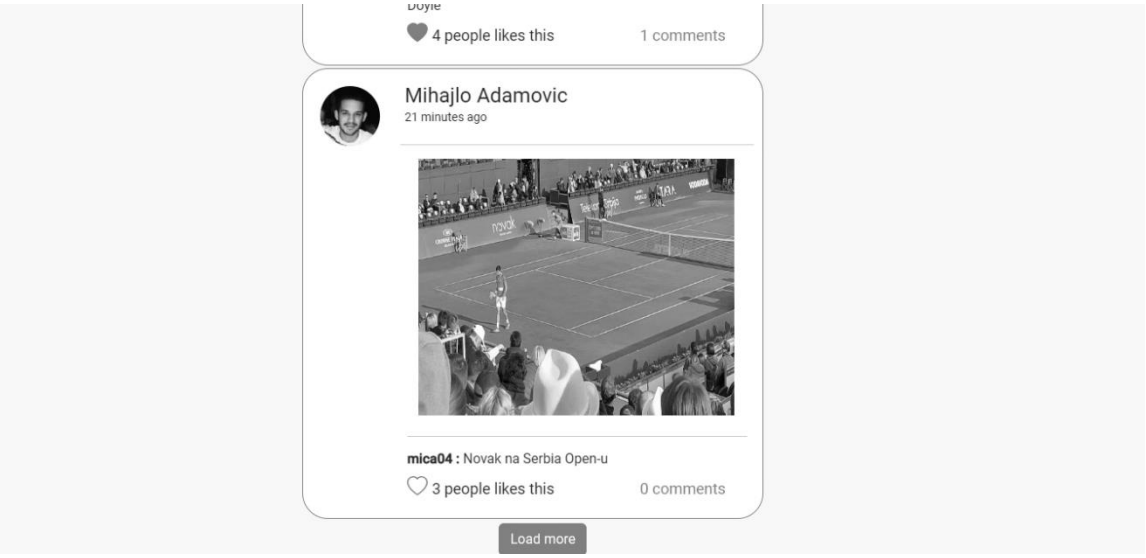
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих објава особа које корисник прати.



Слика 56. Почетна страница

Основни сценарио СК

1. Корисник **бира** објаву на коју жели да реагује. (АПУСО)



Слика 57. Изабрана објава

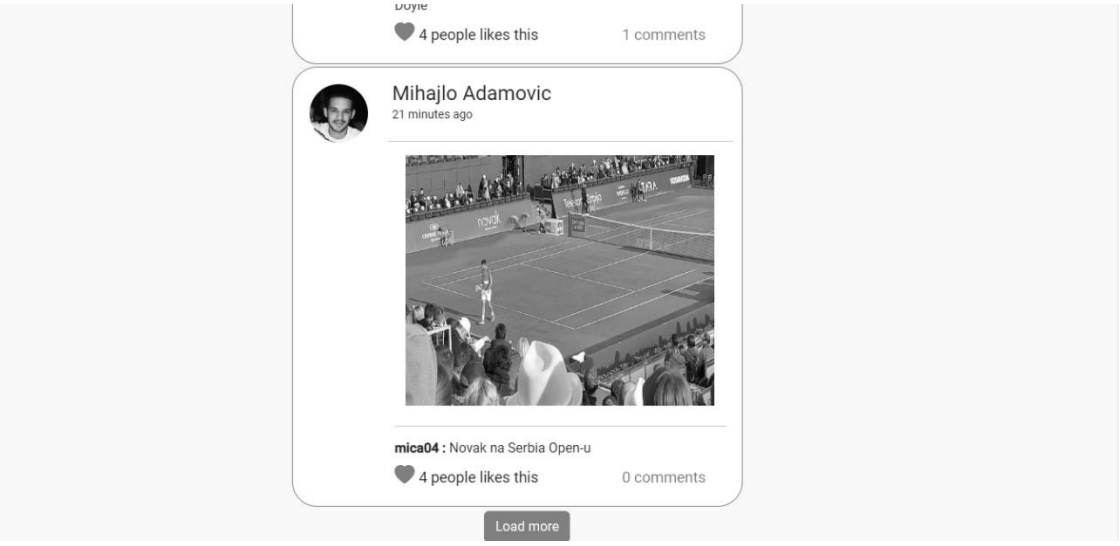
2. Корисник **позива** систем да реагује на одабрану објаву. (АПСО)

Опис акције: Кликом на дугме корисник позива системску операцију

LikeIt(postId, userId).

3. Систем **памти** реакцију за одабрану објаву. (СО)

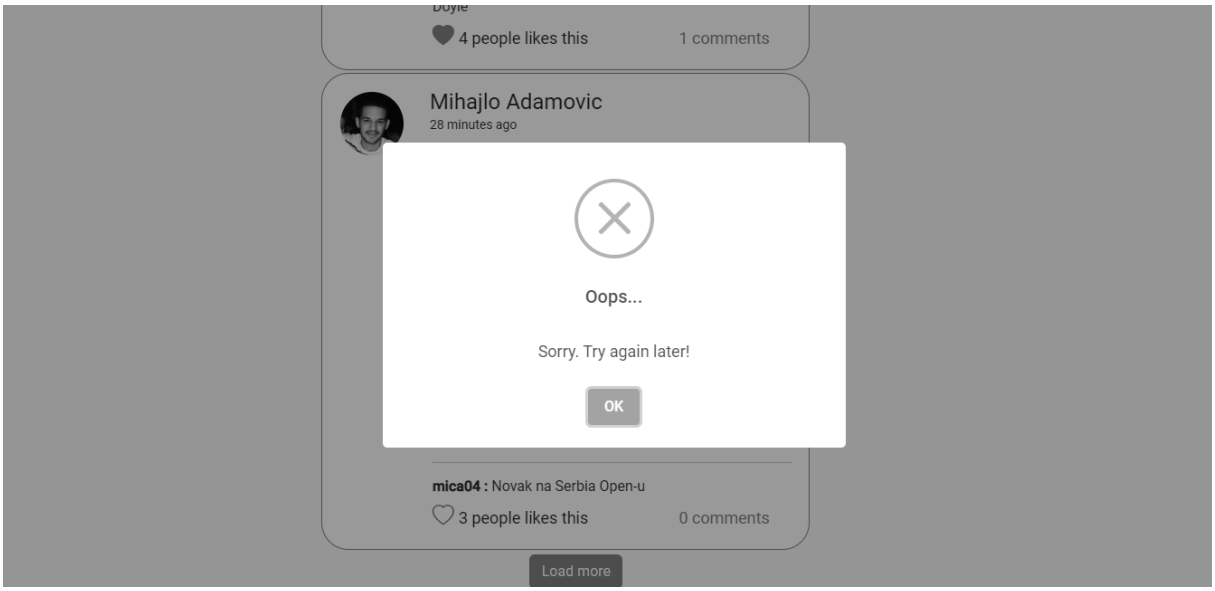
4. Систем **приказује** кориснику да је успешно реаговао на објаву. (ИА)



Слика 58. Успешно реагована објава

Алтернативна сценарија

4.1. Уколико систем не може да запамти реакцију за одабрану објаву, он **приказује** кориснику поруку: “Sorry. Try again later!”. (ИА)



Слика 59. Корисник не може да реагује на објаву

СК5: Случај коришћења – Коментарисање објаве

Назив СК

Коментарисање објаве

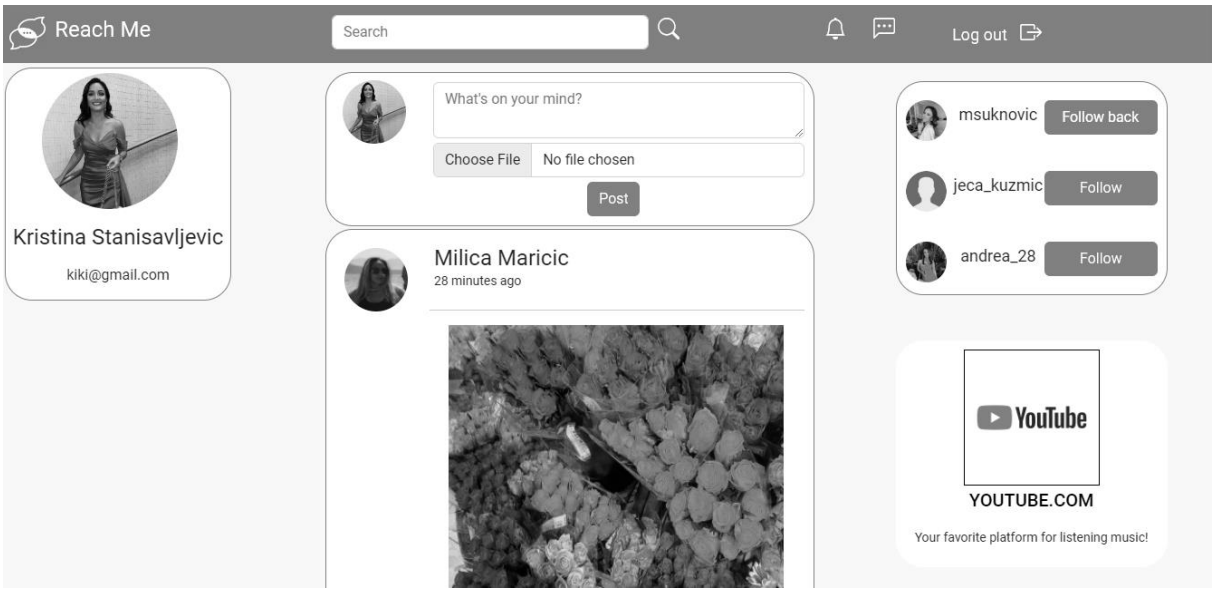
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

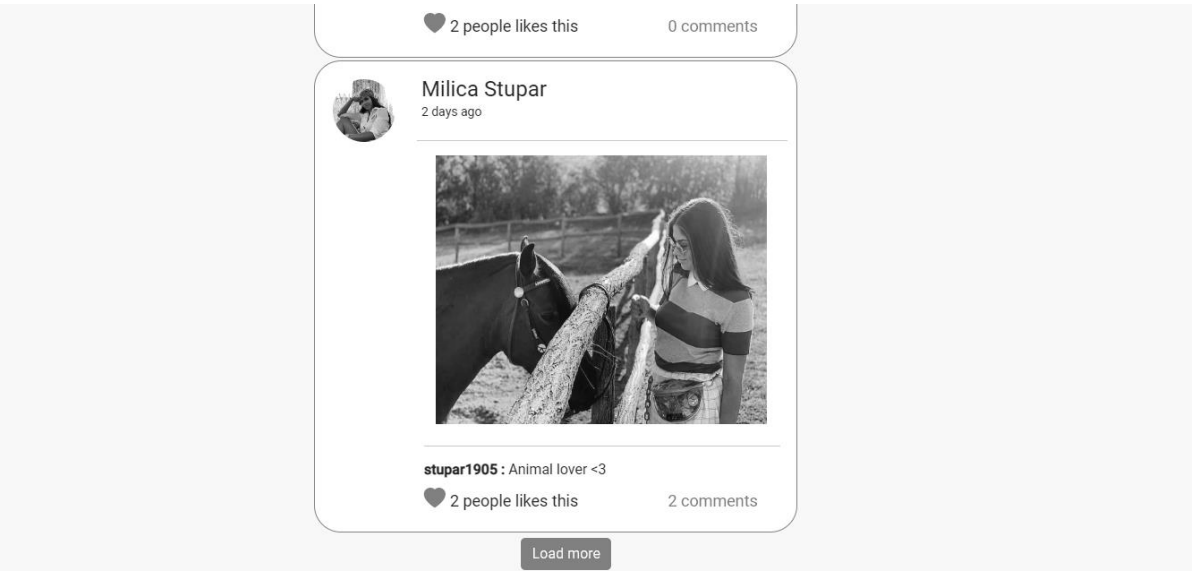
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих објава особа које корисник прати.



Слика 60. Почетна страна

Основни сценарио СК

1. Корисник **бира** објаву коју жели да коментарише. (АПУСО)



Слика 61. Изабрана објава

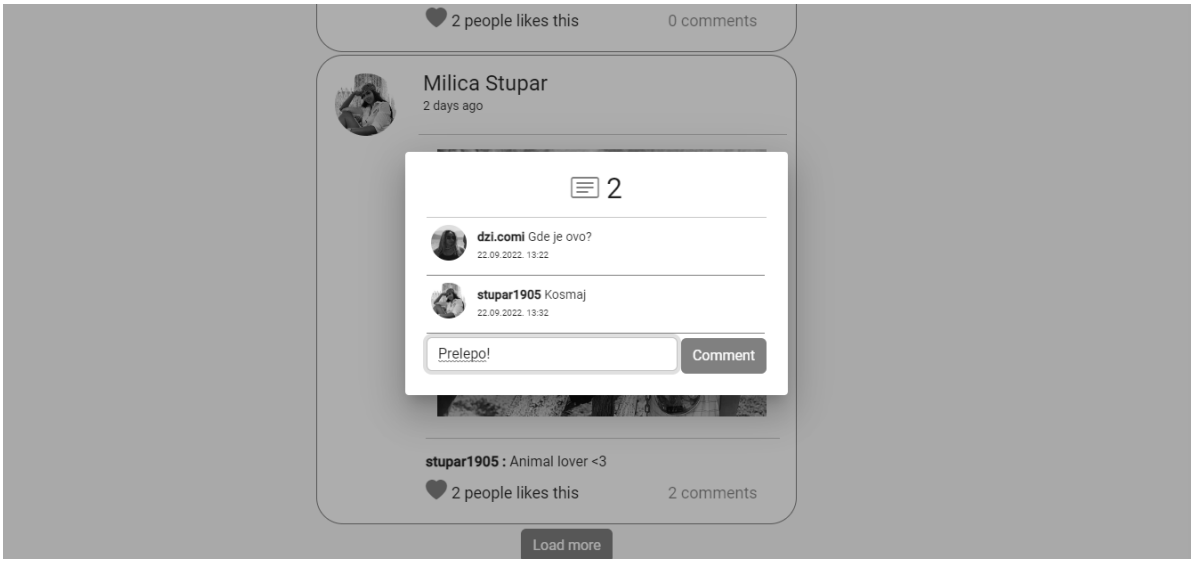
2. Корисник **позива** систем да учита све коментаре за изабрану објаву. (АПСО)

Опис акције: Кликом на дугме „Comments“ корисник позива системску операцију GetComments(postId).

3. Систем **учитава** коментаре за одабрану објаву. (СО)

4. Систем **приказује** кориснику све коментаре за изабрану објаву. (ИА)

5. Корисник **уноси** нови коментар за изабрану објаву. (АПУСО)



Слика 62. Унет коментар

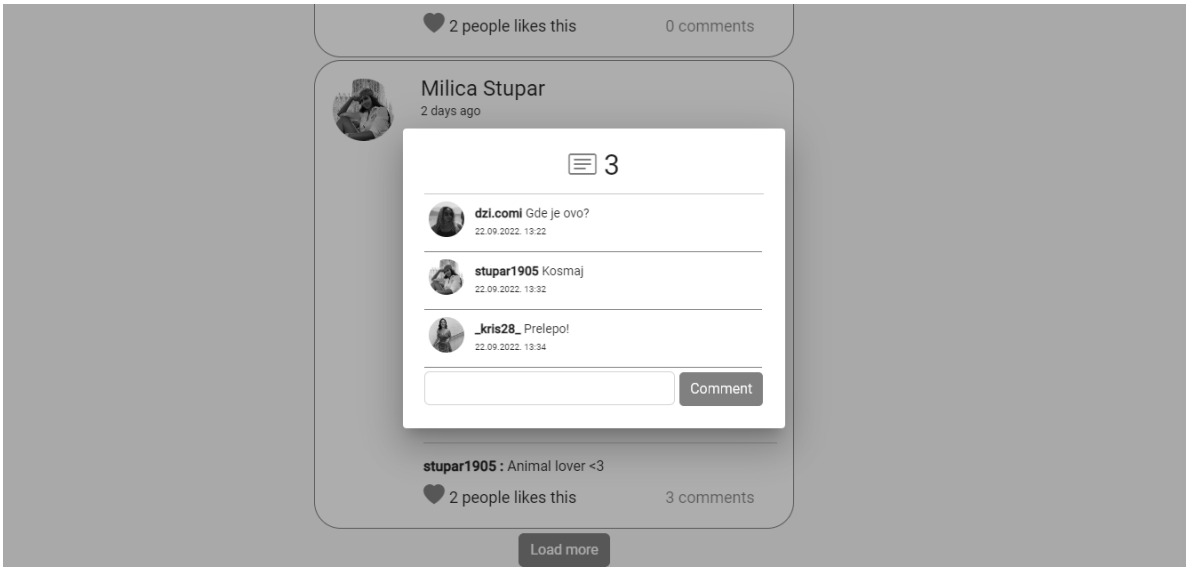
6. Корисник **контролише** да ли је коректно унео нови коментар. (АНСО)

7. Корисник **позива** систем да запамти коментар за изабрану објаву. (АПСО)

Опис акције: Кликом на дугме „Comment“ корисник позива системску операцију PostComment(CommentRequest).

8. Систем **памти** коментар за објаву. (СО)

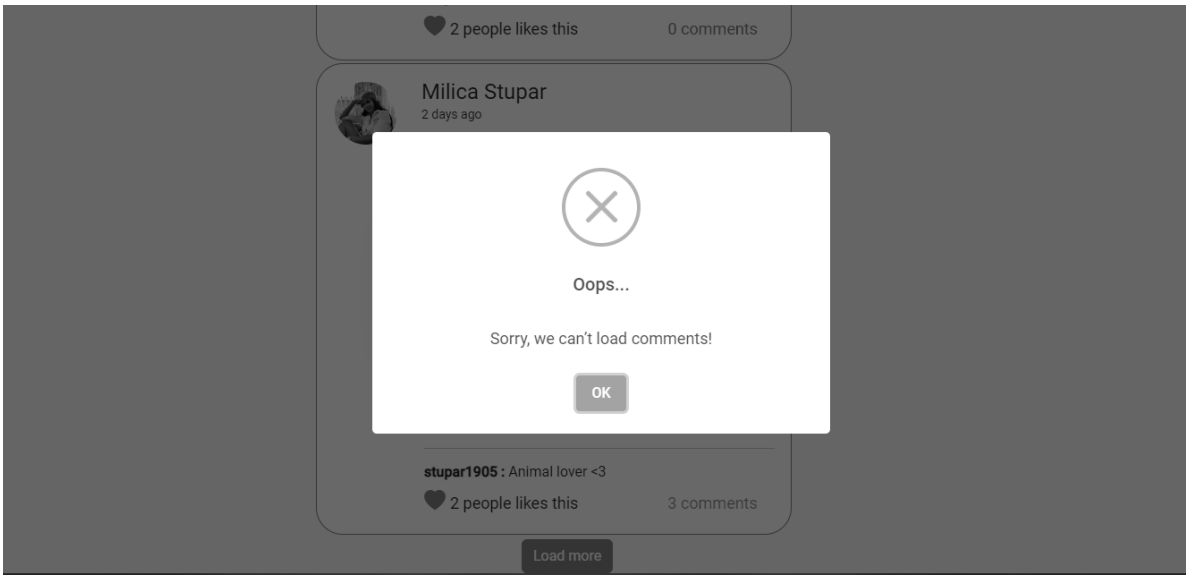
9. Систем **приказује** кориснику запамћен коментар. (ИА)



Слика 63. Коментар је запамћен

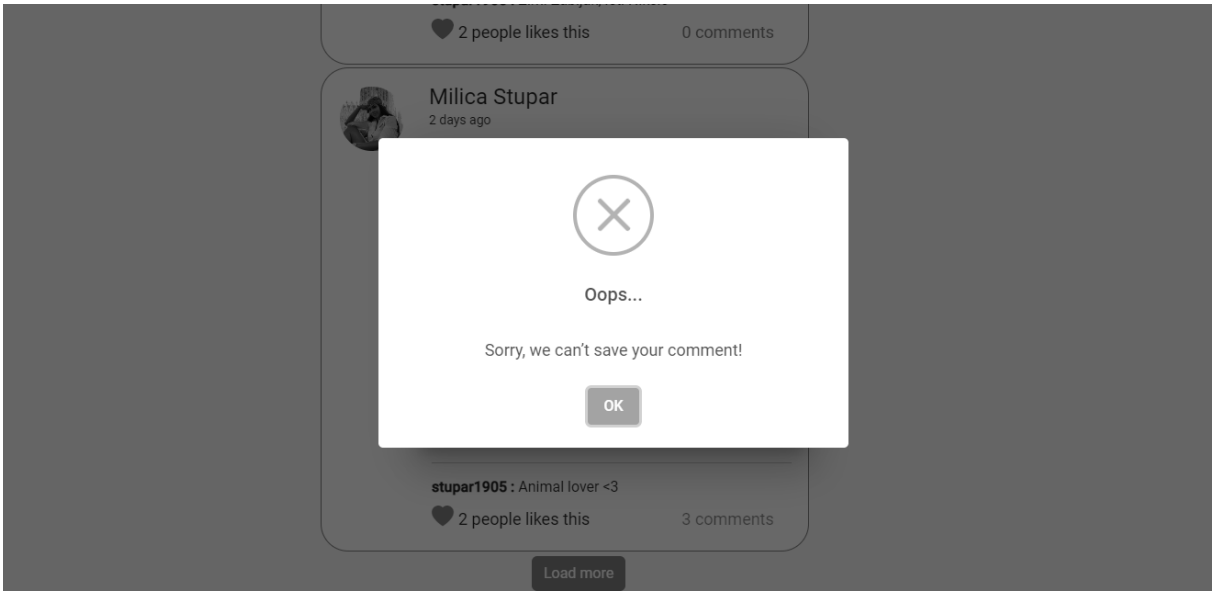
Алтернативна сценарија

4.1. Уколико систем не може да прикаже све коментаре за одабрану објаву, он **приказује** кориснику поруку: “Sorry, we can’t load comments!”. (ИА)



Слика 64. Систем не може да учита коментаре

9.1. Уколико систем не може да запамти коментар, он **приказује** кориснику поруку: “Sorry, we can’t save your comment!”. (ИА)



Слика 65. Систем не може да запамти коментар

СК6: Случај коришћења – Претраживање корисника

Назив СК

Претраживање корисника

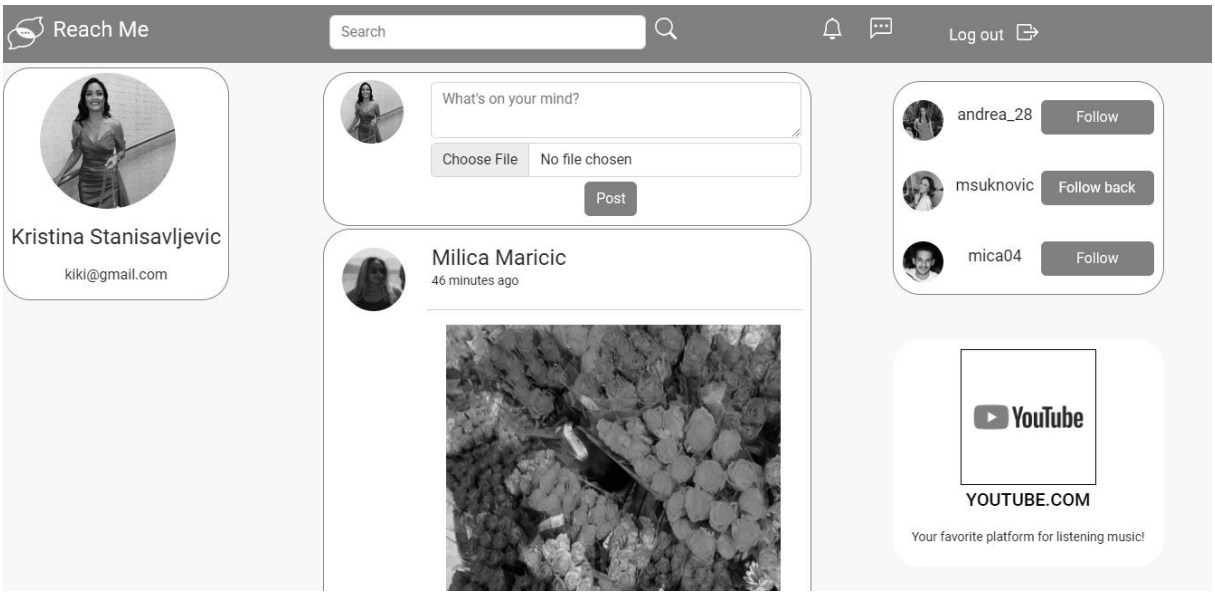
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.



Слика 66. Почетна страна на којој се налази форма за претрагу

Основни сценарио СК

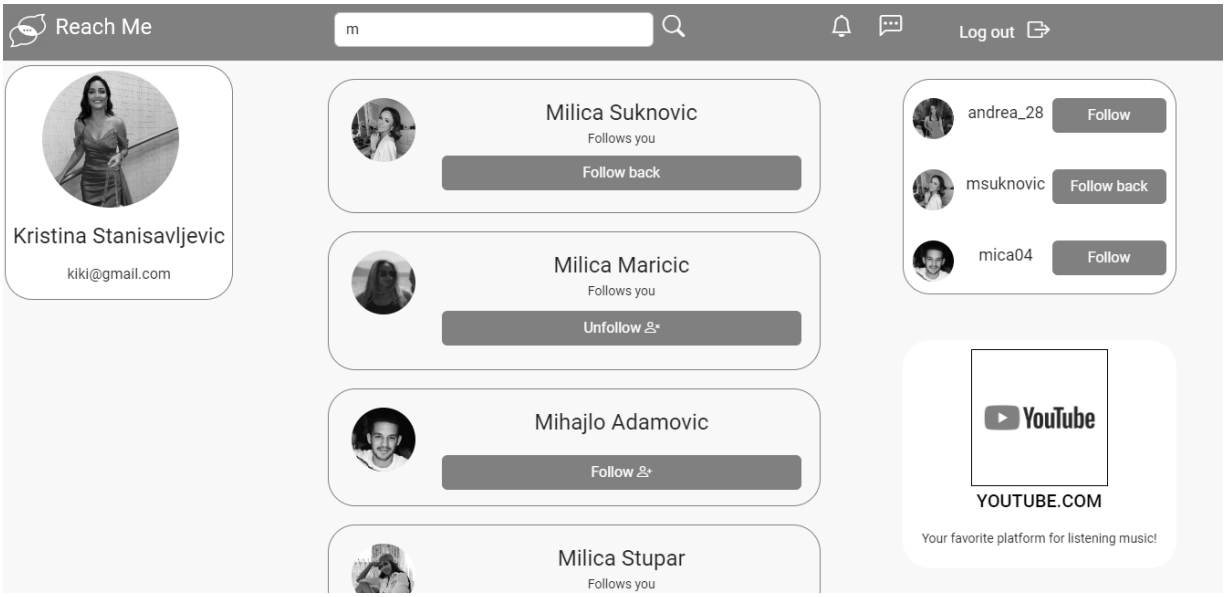
1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију Search(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)



Слика 67. Пронађени корисници

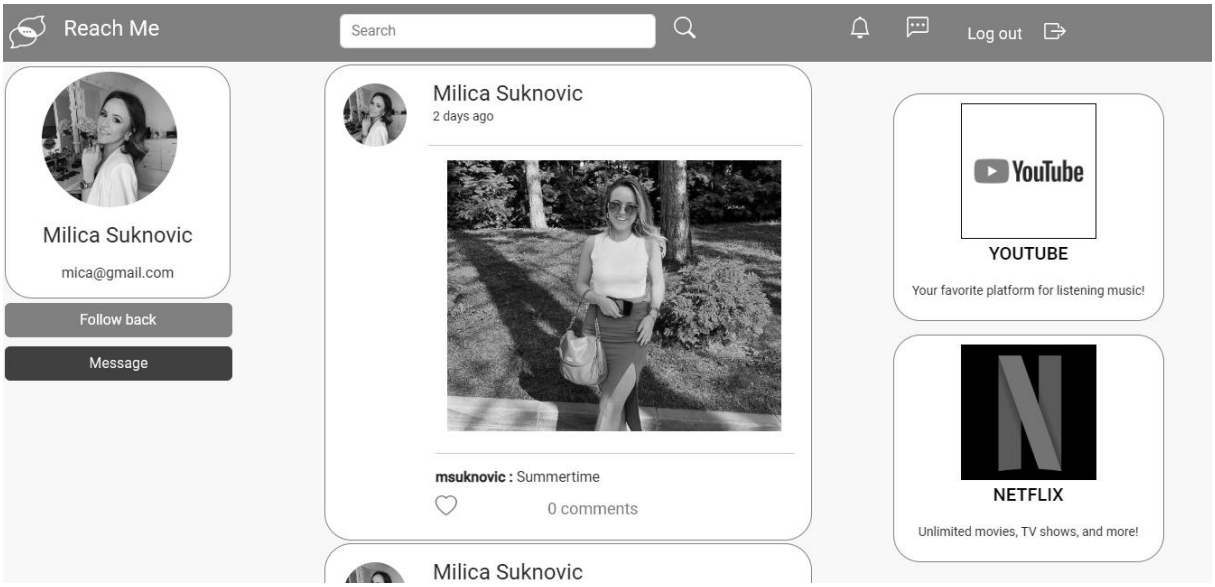
5. Корисник **бира** корисника. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију UcitajUsera(userId, username).

7. Систем **учитава** профил изабраног корисника. (СО)

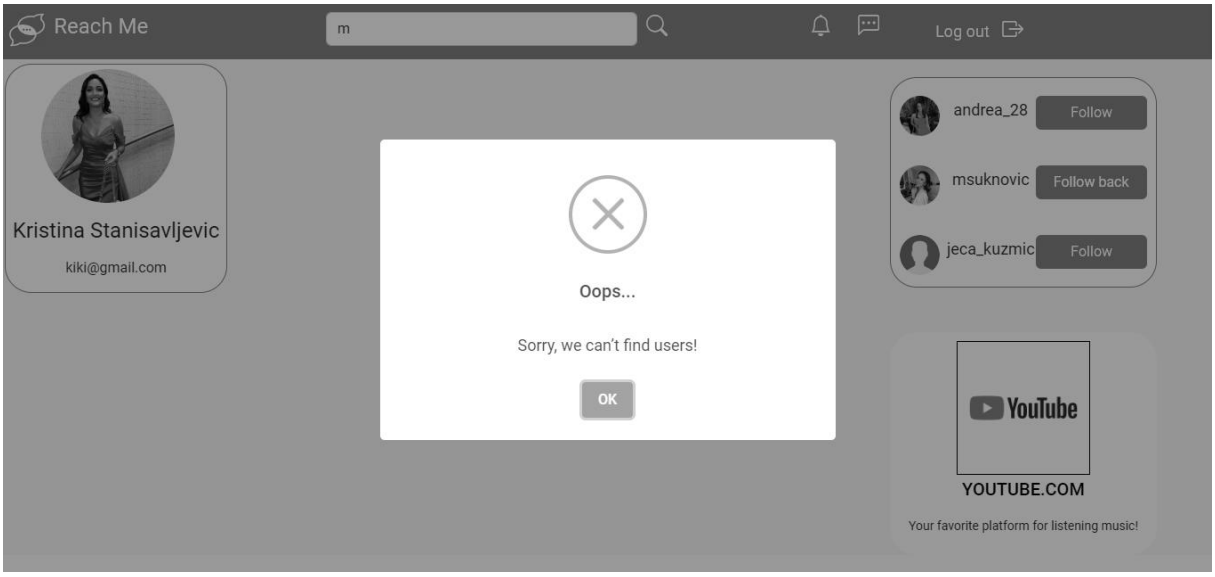
8. Систем **приказује** кориснику профил изабраног корисника. (ИА)



Слика 68. Профил траженог корисника

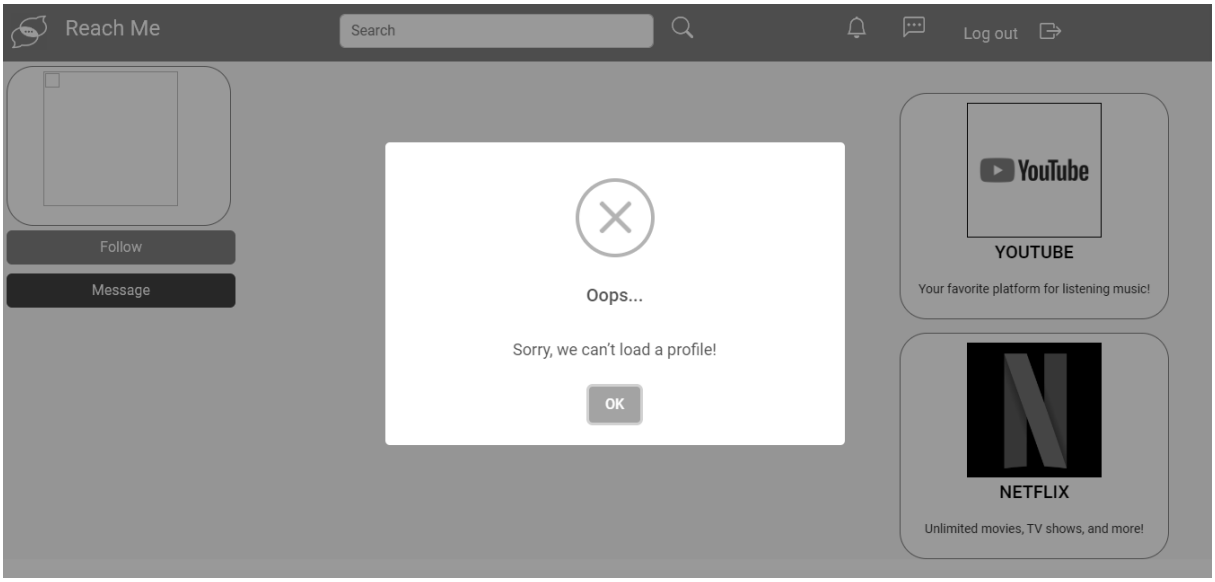
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



Слика 69. Систем не може да пронађе кориснике

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)



Слика 70. Систем не може да учита профил траженог корисника

СК7: Случај коришћења – Слање захтева за праћење

Назив СК

Слање захтева за праћење

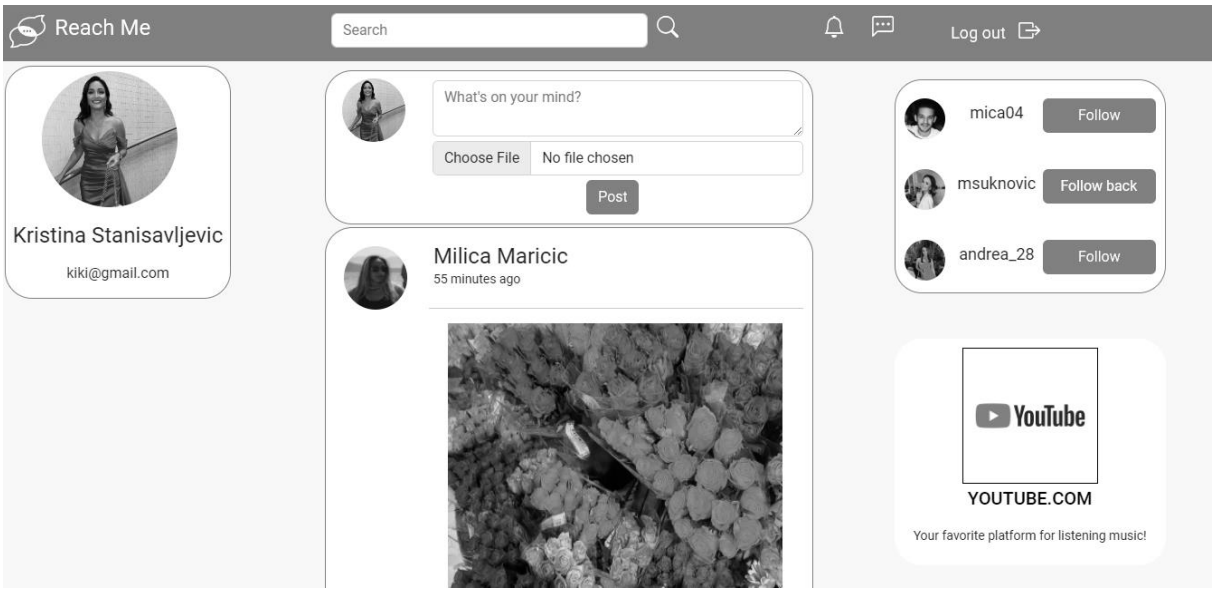
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.



Слика 71. Почетна страница на којој се налази форма за претрагу

Основни сценарио СК

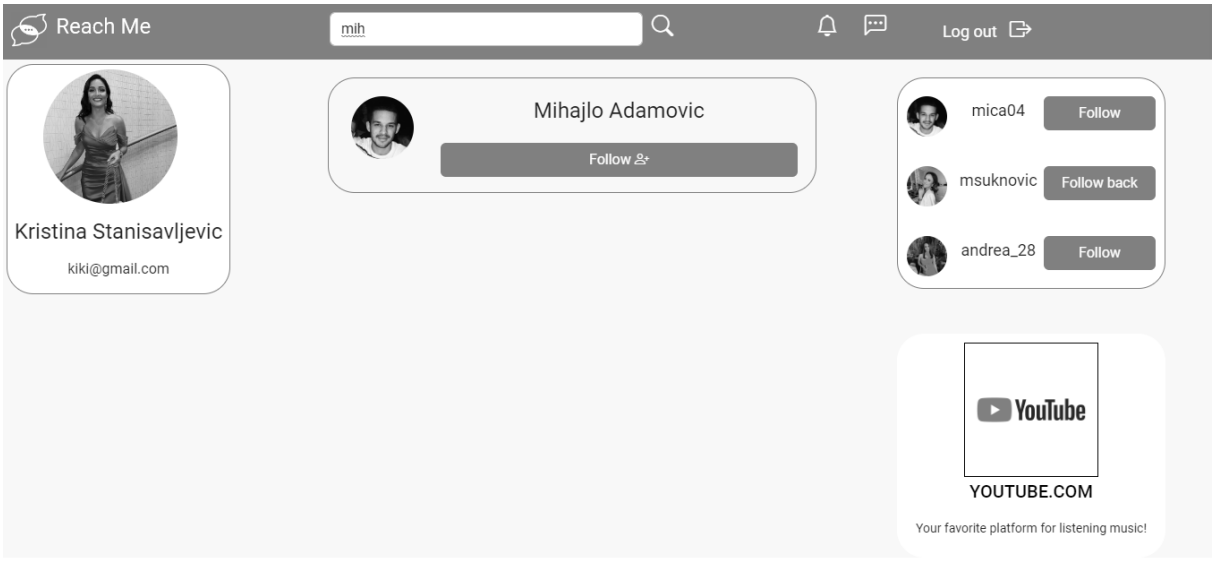
1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију Search(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)



Слика 72. Пронађени корисници

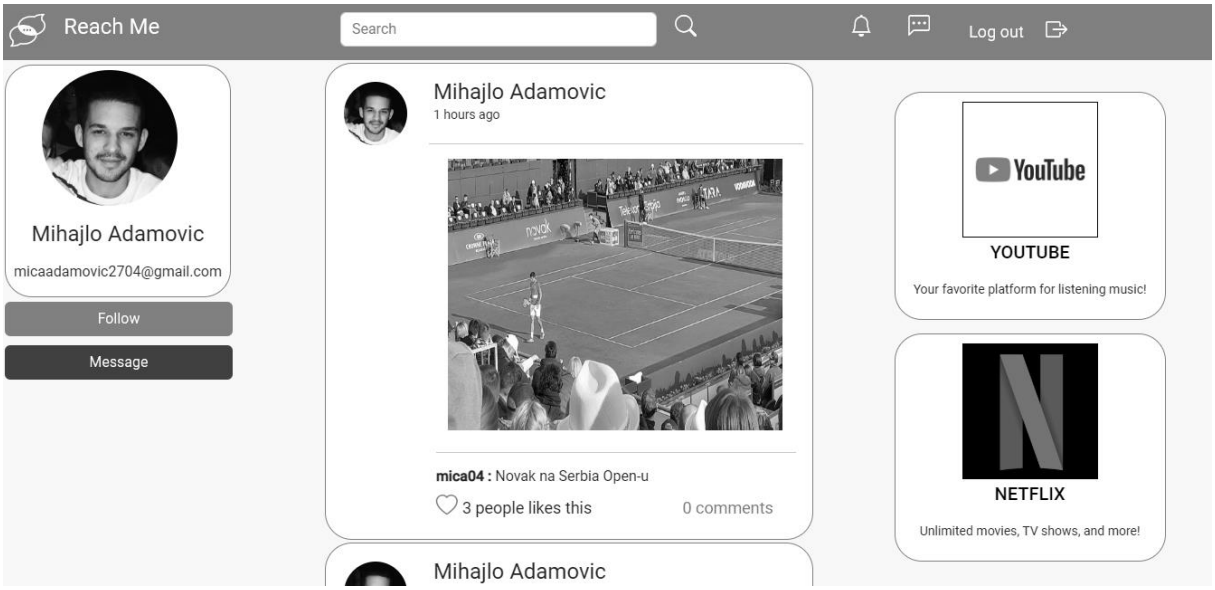
5. Корисник **бира** корисника којем жели да пошаље захтев. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију UcitajUsera(userId, username).

7. Систем **учитава** профил изабраног корисника. (СО)

8. Систем **приказује** кориснику профил изабраног корисника. (ИА)



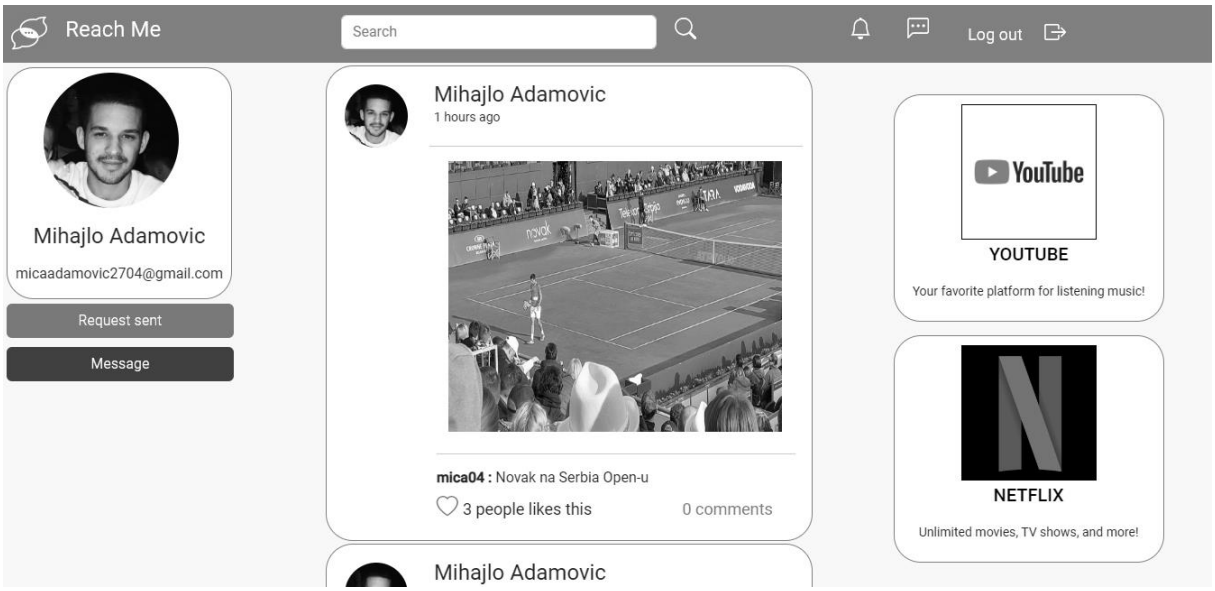
Слика 73. Профил траженог корисника

9. Корисник **позива** систем да пошаље захтев изабраном кориснику. (АПСО)

Опис акције: Кликом на дугме „Follow“ корисник позива системску операцију SendRequest(userId, followId).

10. Систем **памти** захтев за праћење. (СО)

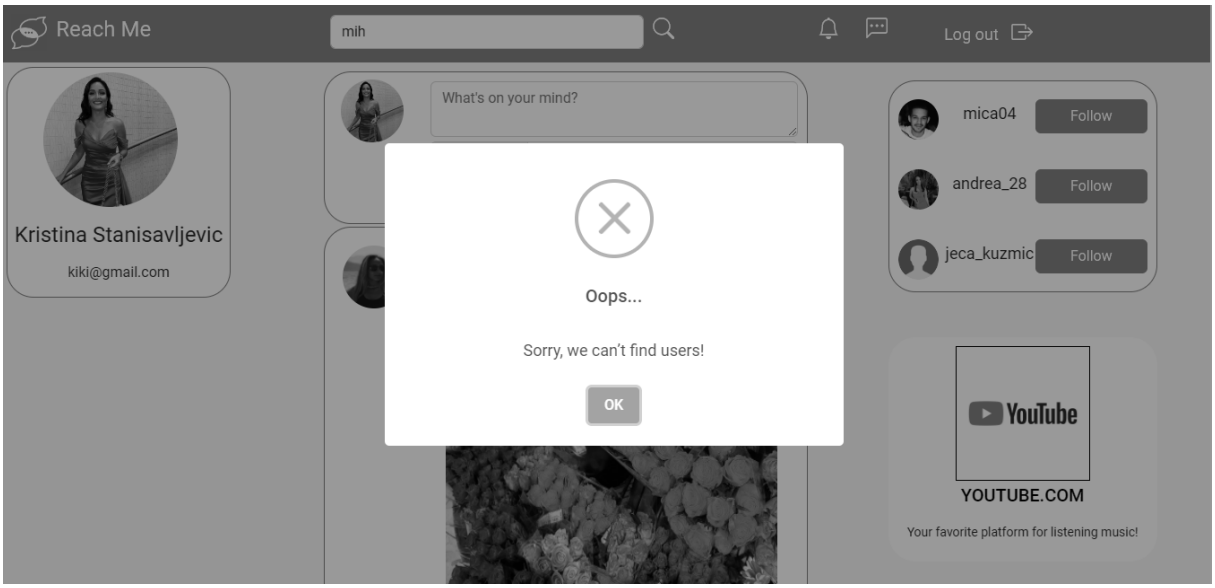
11. Систем **приказује** кориснику да је захтев успешно послат. (ИА)



Слика 74. Захтев за праћење је послат

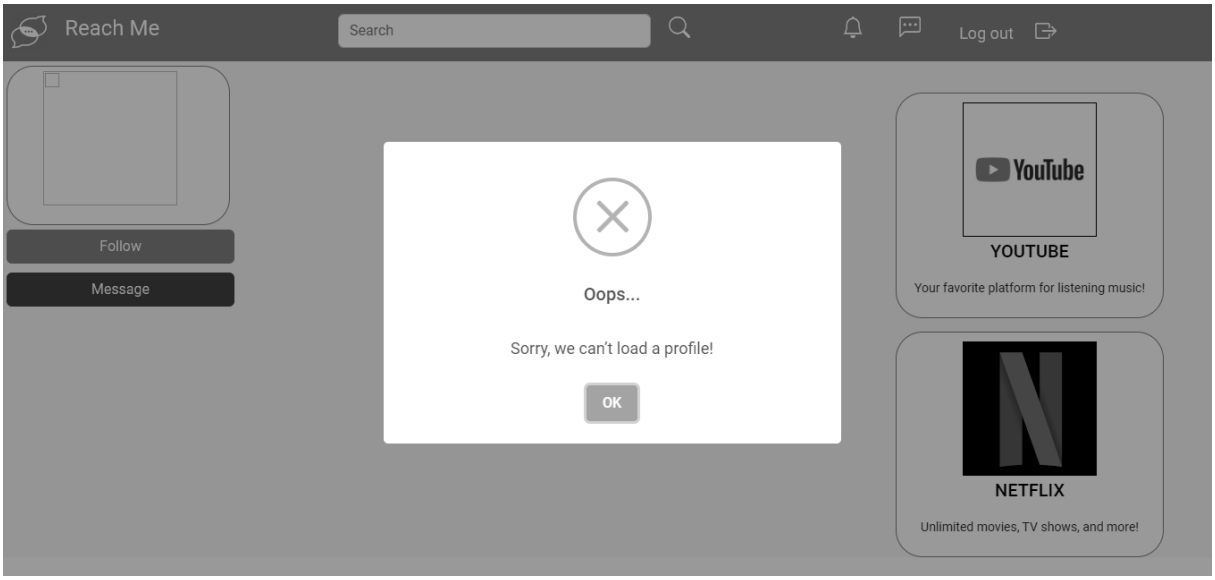
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



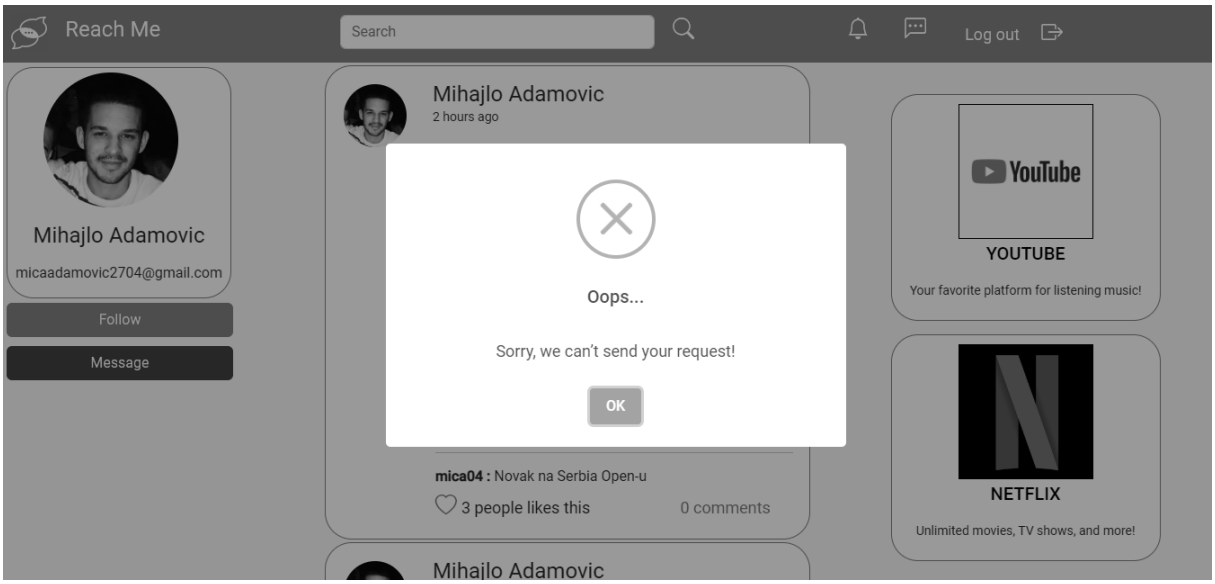
Слика 75. Систем не може да пронађе кориснике

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)



Слика 76. Систем не може да учита профил траженог корисника

10.1. Уколико систем не може да запамти захтев за праћење, он **приказује** кориснику поруку: “Sorry, we can’t send your request!”. (ИА)



Слика 77. Систем не може да пошаље захтев кориснику

СК8: Случај коришћења – Отпраћивање

Назив СК

Отпраћивање

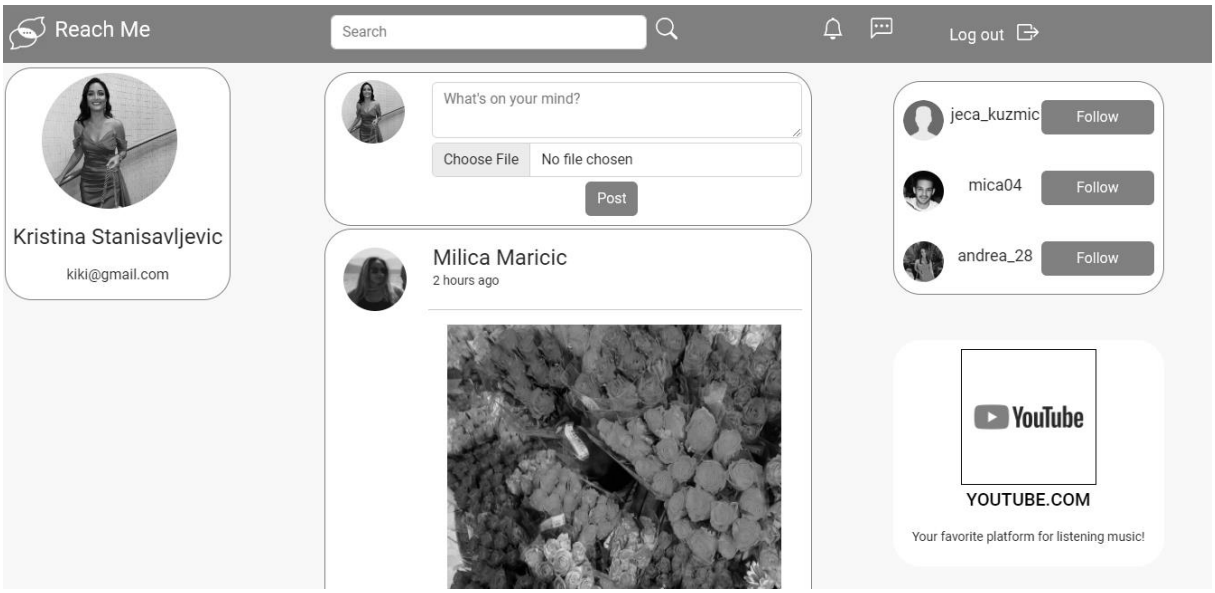
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.



Слика 78. Почетна страна на којој се налази форма за претрагу

Основни сценарио СК

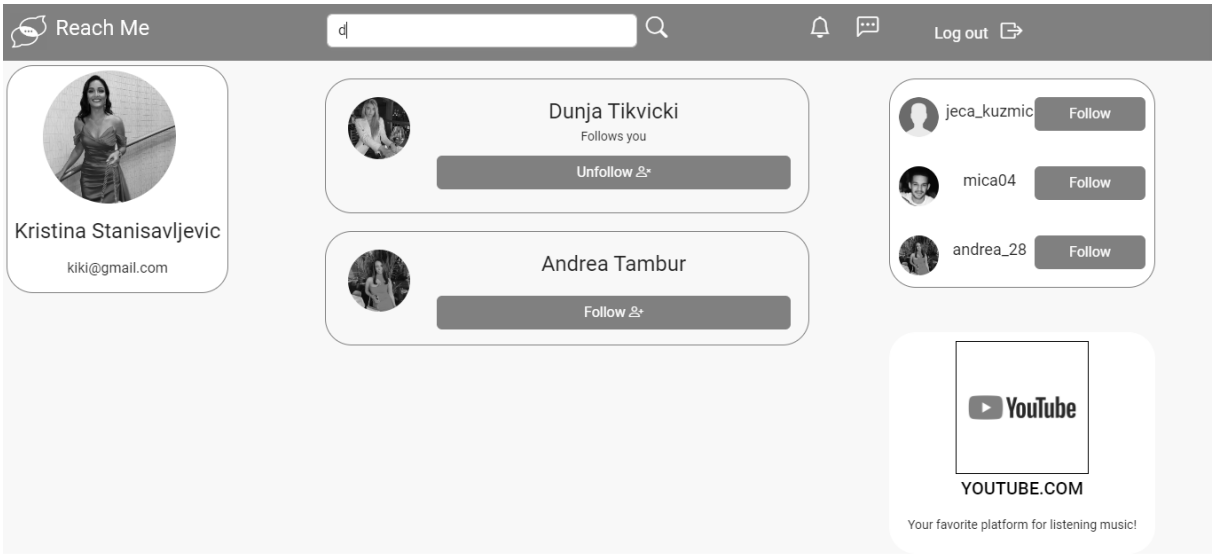
1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију Search(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)



Слика 79. Пронађени корисници

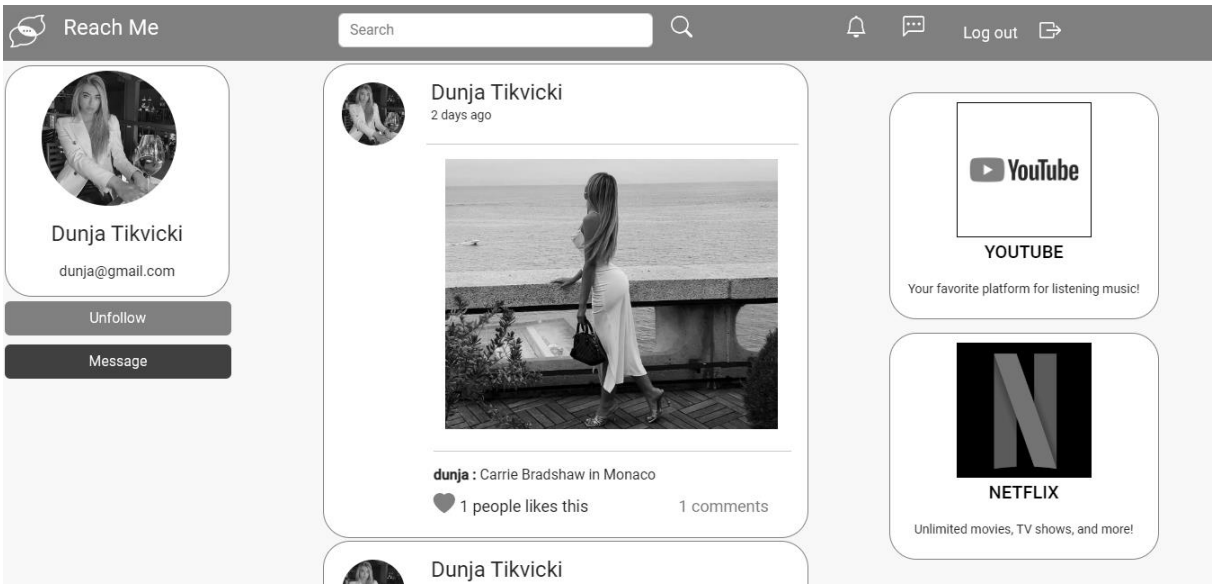
5. Корисник **бира** корисника којег жели да отпрати. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију UcitajUsera(userId, username).

7. Систем **учитава** профил изабраног корисника. (СО)

8. Систем **приказује** кориснику профил изабраног корисника. (ИА)



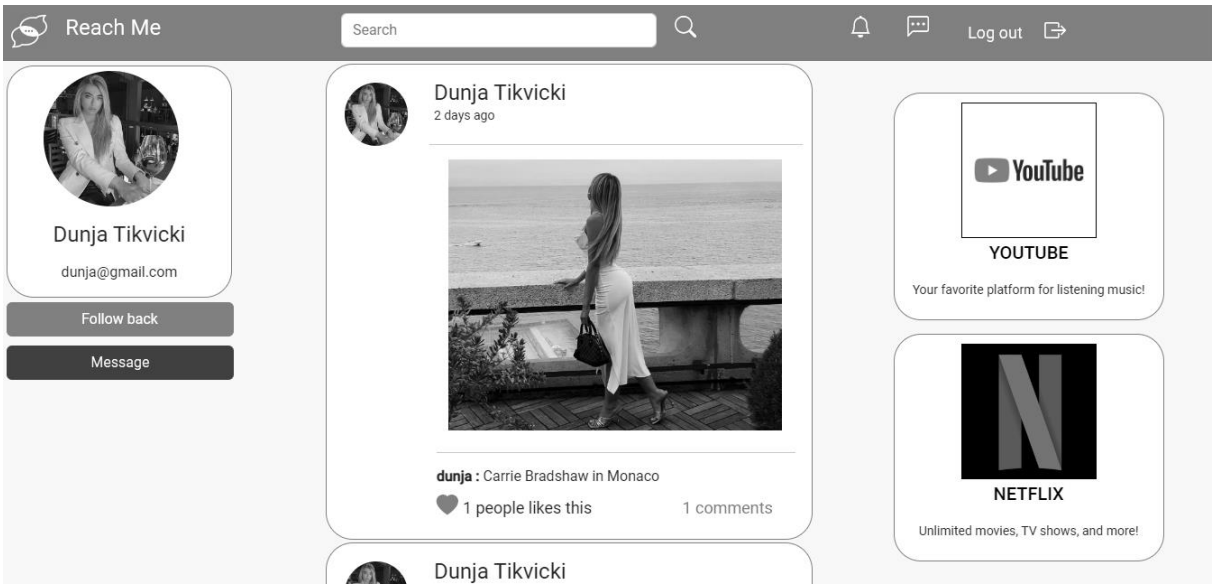
Слика 80. Профил траженог корисника

9. Корисник **позива** систем да отпрати изабраног корисника. (АПСО)

Опис акције: Кликом на дугме „Unfollow“ корисник позива системску операцију Unfollow(username, userId).

10. Систем **отпрађује** изабраног корисника. (СО)

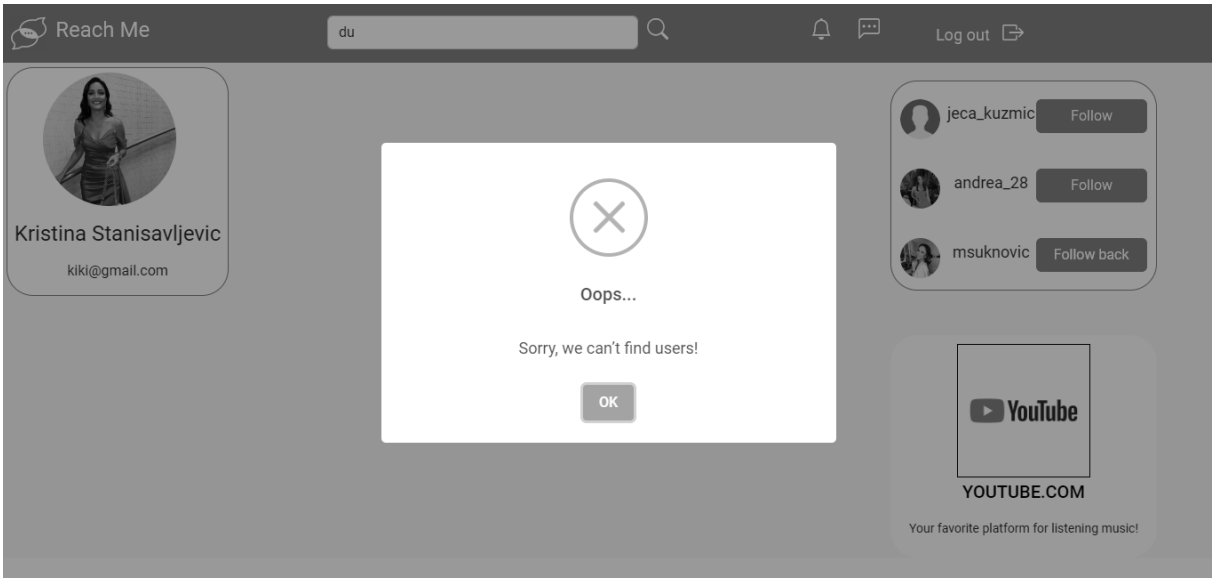
11. Систем **приказује** кориснику да је изабрани корисник успешно отпраћен. (ИА)



Слика 81. Корисник је отпраћен

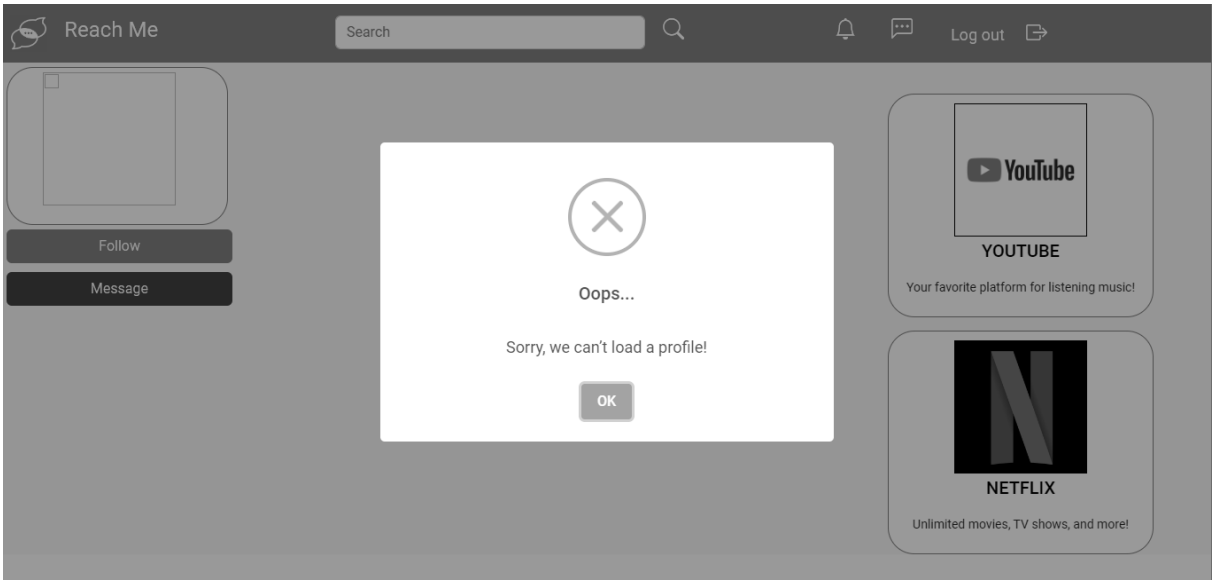
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



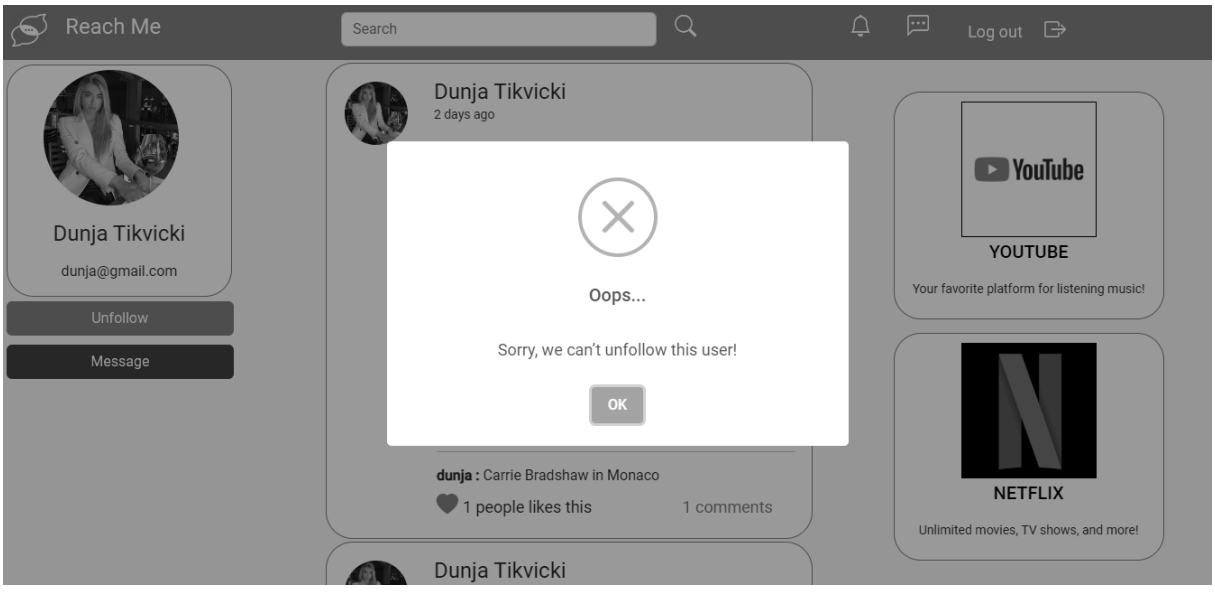
Слика 82. Систем не може да пронађе кориснике

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”(ИА)



Слика 83. Систем не може да учита профил траженог корисника

10.1. Уколико систем не може да отпрати корисника, он **приказује** кориснику поруку: “Sorry, we can’t unfollow this user!”. (ИА)



Слика 84. Систем не може да отпрати корисника

СК9: Случај коришћења – Измена профила

Назив СК

Измена профила

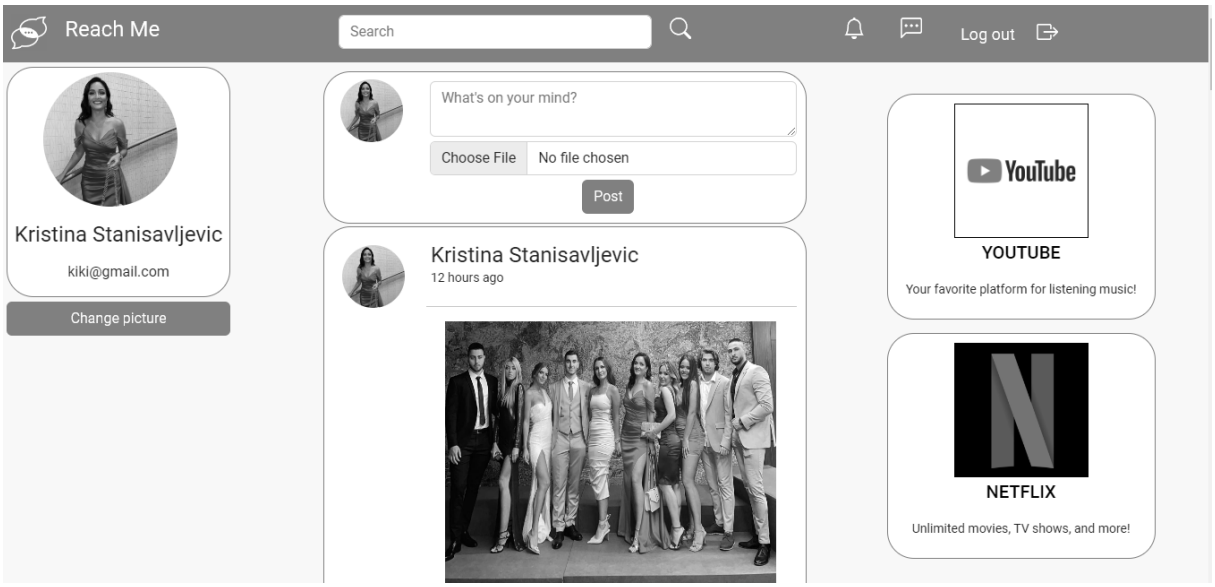
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

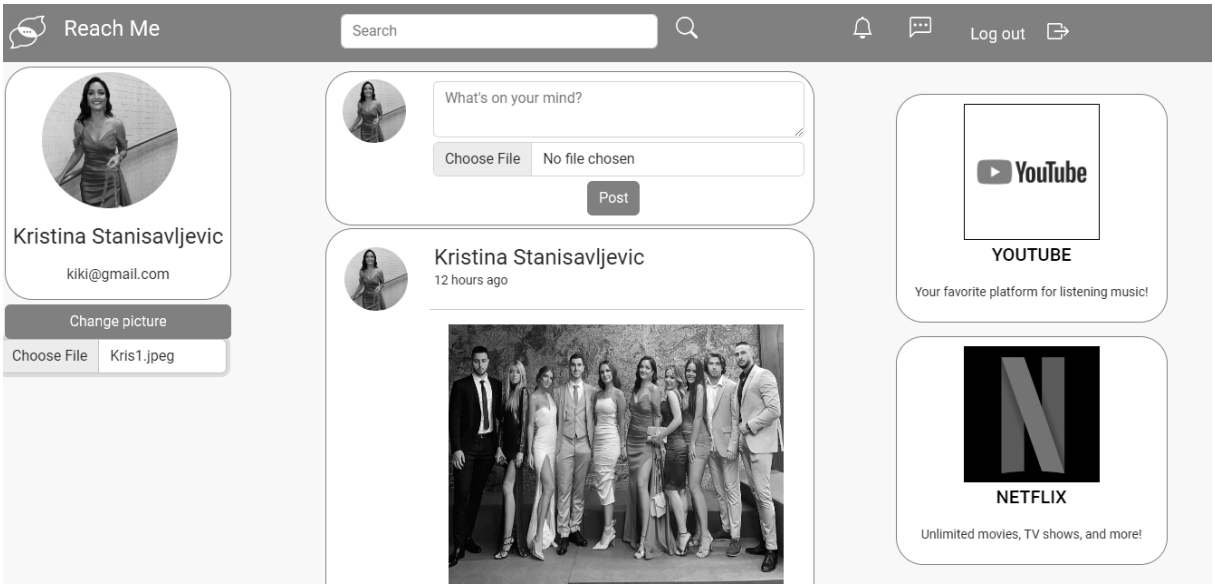
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује профил улогованог корисника. Учитани су подаци о улогованом кориснику као и листа свих објава улогованог корисника.



Слика 85. Профил тренутно улогованог корисника

Основни сценарио СК

1. Корисник **уноси (мења)** своју профилну слику. (АПУСО)



Слика 86. Унета нова профилна слика

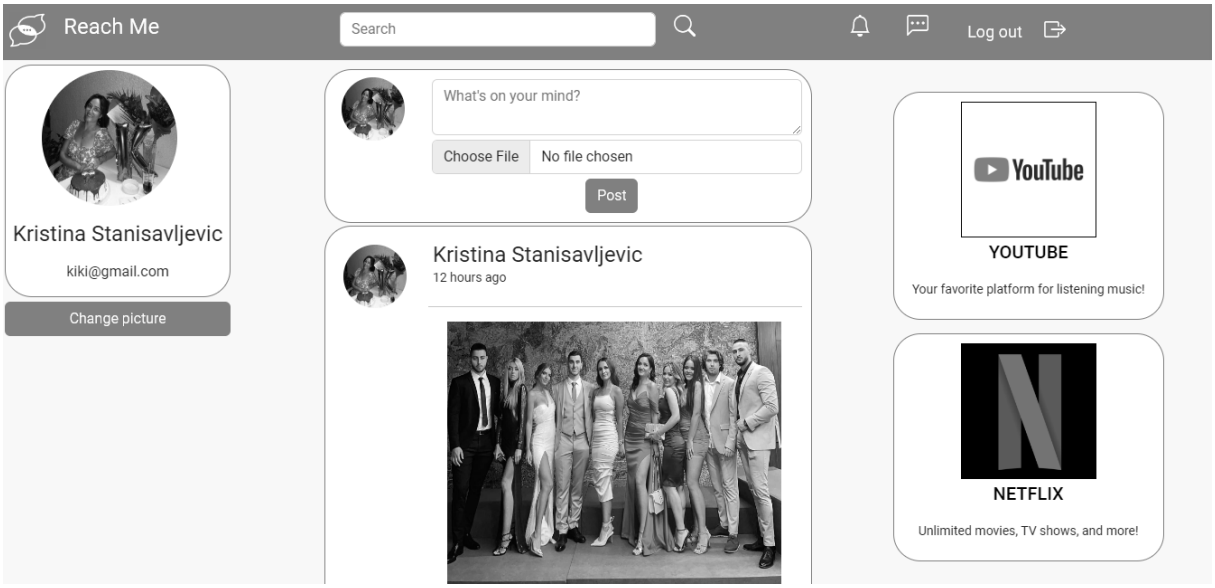
2. Корисник **контролише** да ли је коректно унео нову профилну слику. (АНСО)

3. Корисник **позива** систем да запамти податке о кориснику. (АПСО)

Опис акције: Кликом на дугме „Change picture“ корисник позива системску операцију ChangePicture(UserDto).

4. Систем **памти** податке о кориснику. (СО)

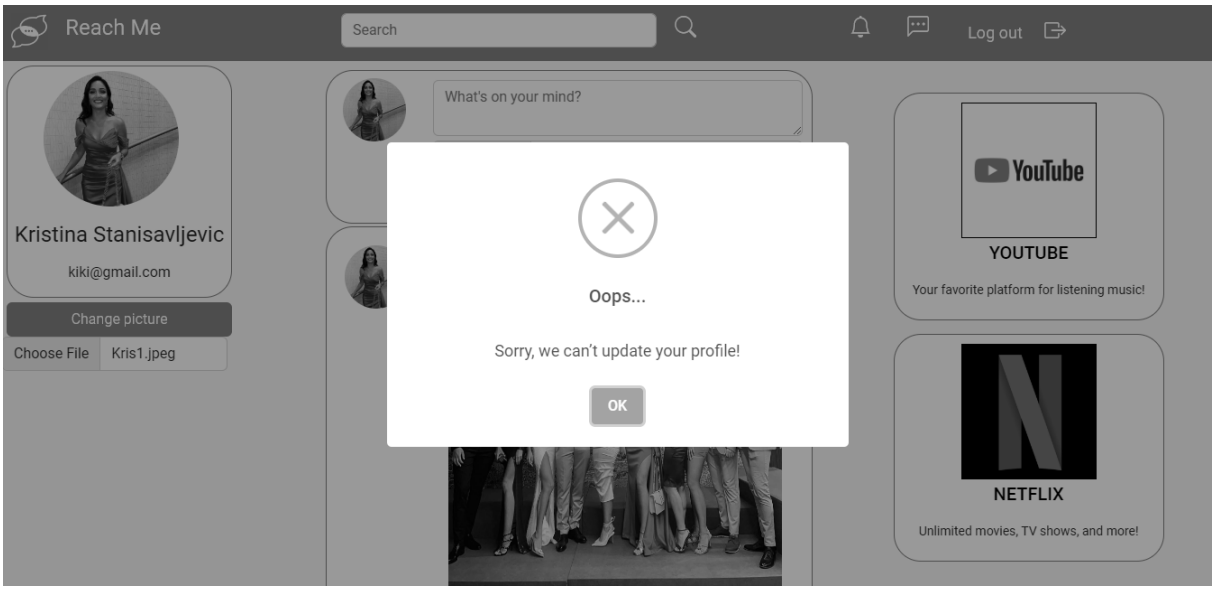
5. Систем **приказује** кориснику да је запамтио нове податке. (ИА)



Слика 87. Успешно промењена профилна слика

Алтернативна сценарија

5.1. Уколико систем не може да запамти податке о кориснику, он **приказује** кориснику поруку: “Sorry, we can’t update your profile!”. (ИА)



Слика 88. Систем не може да измени профилну слику

СК10: Случај коришћења – Слање поруке

Назив СК

Слање поруке

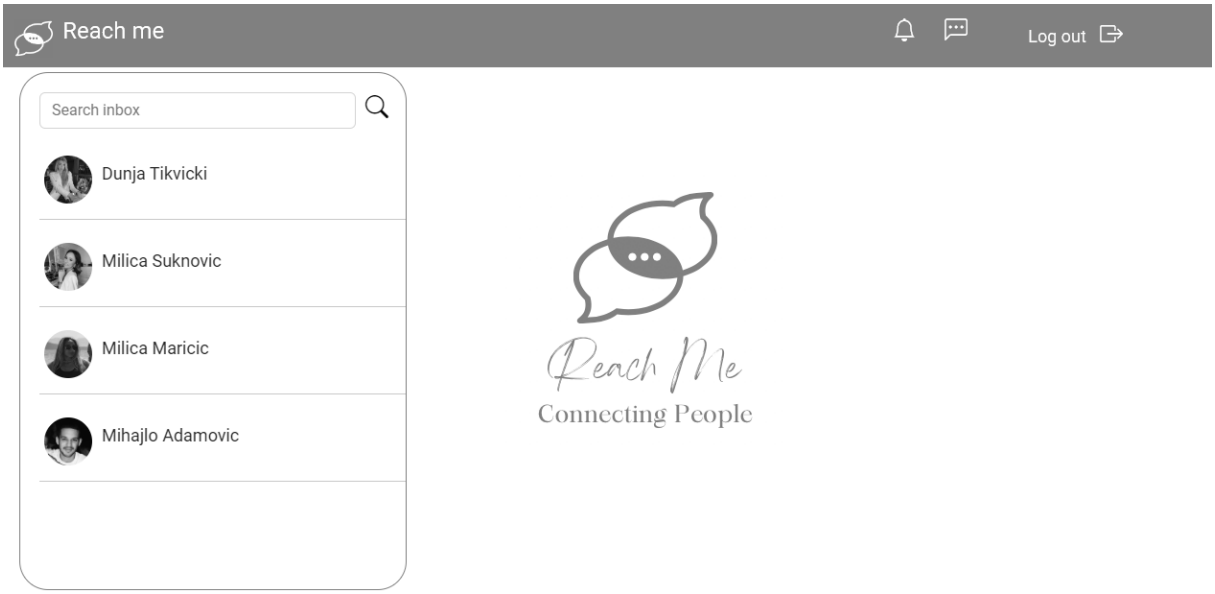
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих корисника са којима улоговани корисник има поруке.



Слика 89. Инбох тренутно улогованог корисника

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију FindInboxUsers(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)

Слика 90. Пронађени инбох корисници

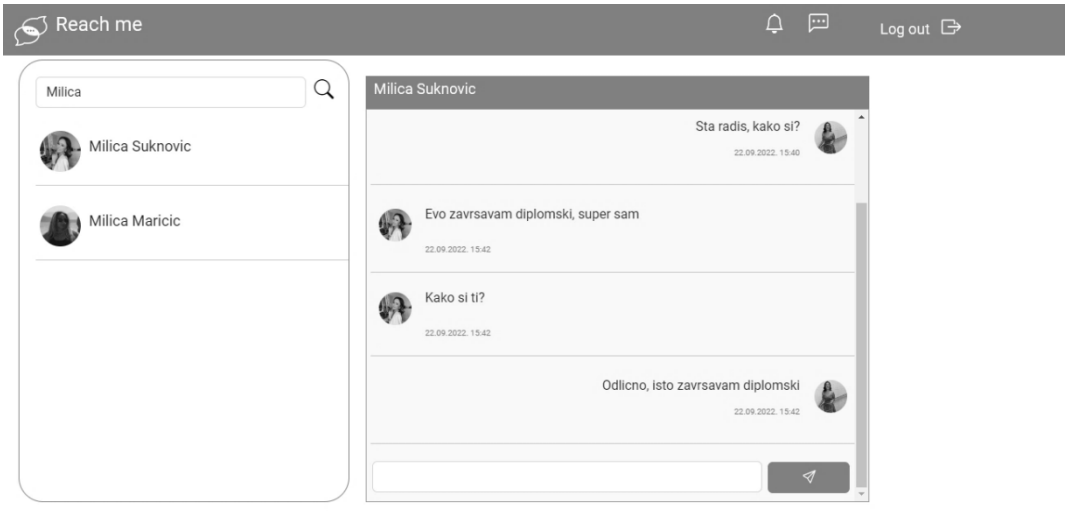
5. Корисник **бира** корисника којем жели да пошаље поруку. (АПУСО)

6. Корисник **позива** систем да учита chat са изабраним корисником. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију GetChat(fromId, forId).

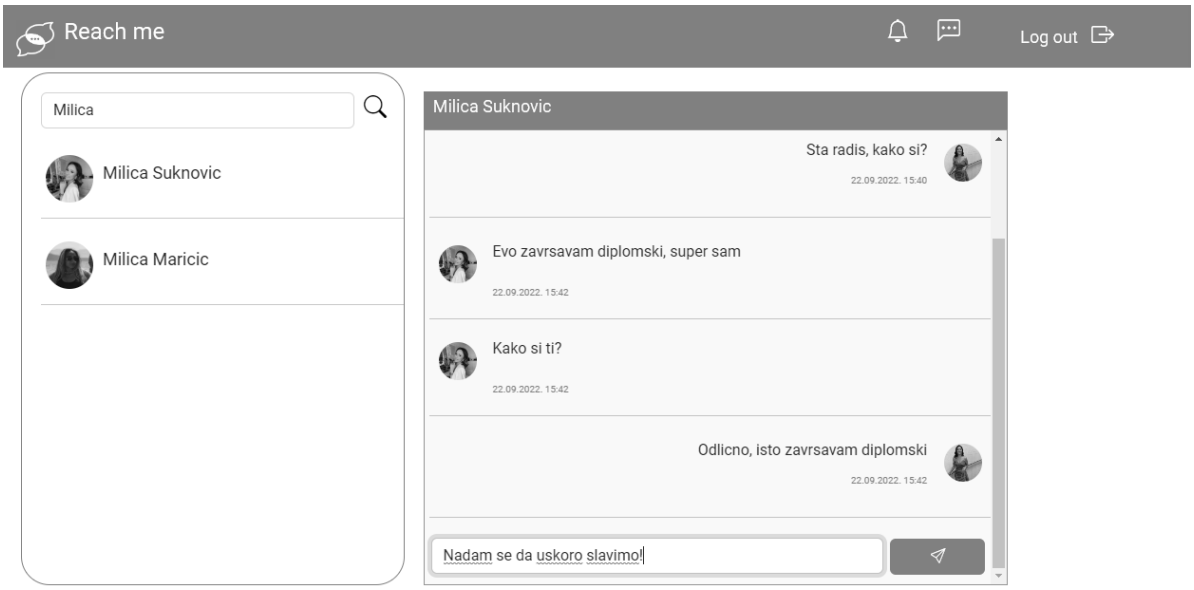
7. Систем **учитава** листу свих порука са изабраним корисником. (СО)

8. Систем **приказује** кориснику поруке. (ИА)



Слика 91. Све поруке са изабраним корисником

9. Корисник **уноси** нову поруку за изабраног корисника. (АПУСО)



Слика 92. Унета порука за слање кориснику

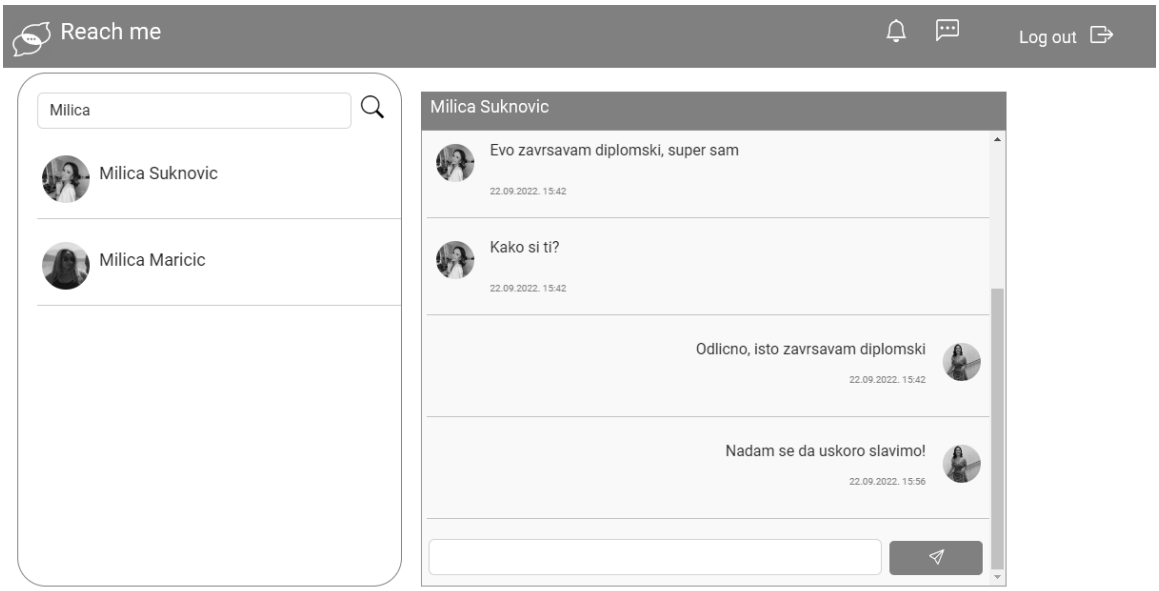
10. Корисник **контролише** да ли је коректно унео нову поруку. (АНСО)

11. Корисник **позива** систем да пошаље поруку. (АПСО)

Опис акције: Кликом на дугме корисник позива системску операцију SendMessage(MessageDto).

12. Систем **памти** поруку за изабраног корисника. (СО)

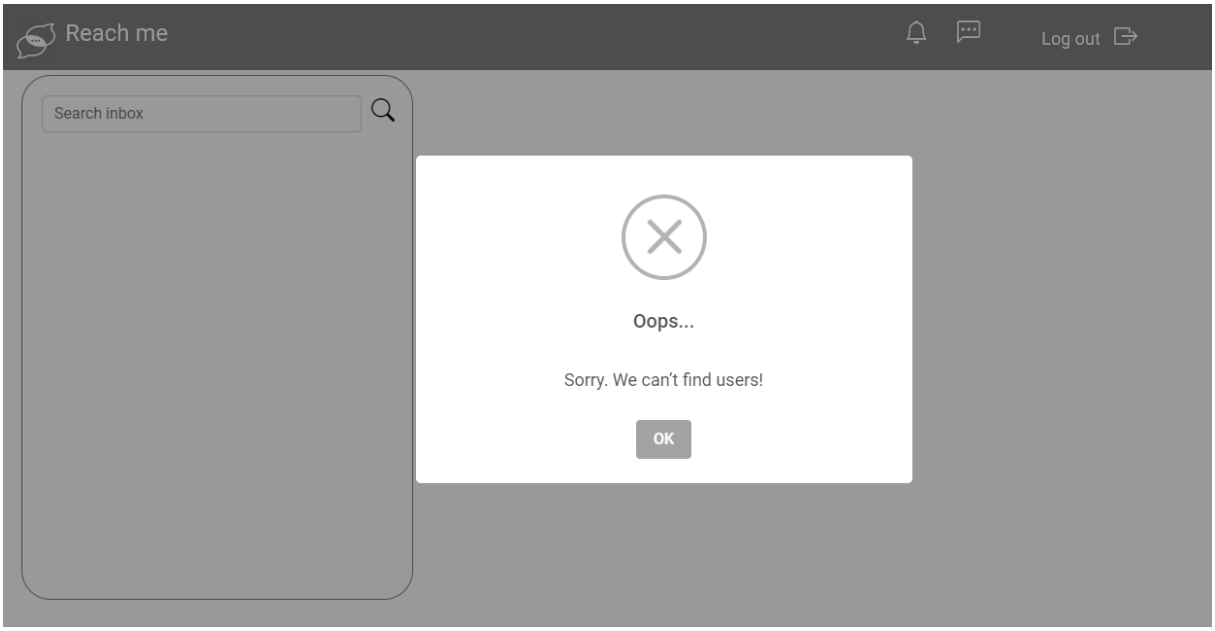
13. Систем **приказује** кориснику послату поруку. (ИА)



Слика 93. Успешно послата порука

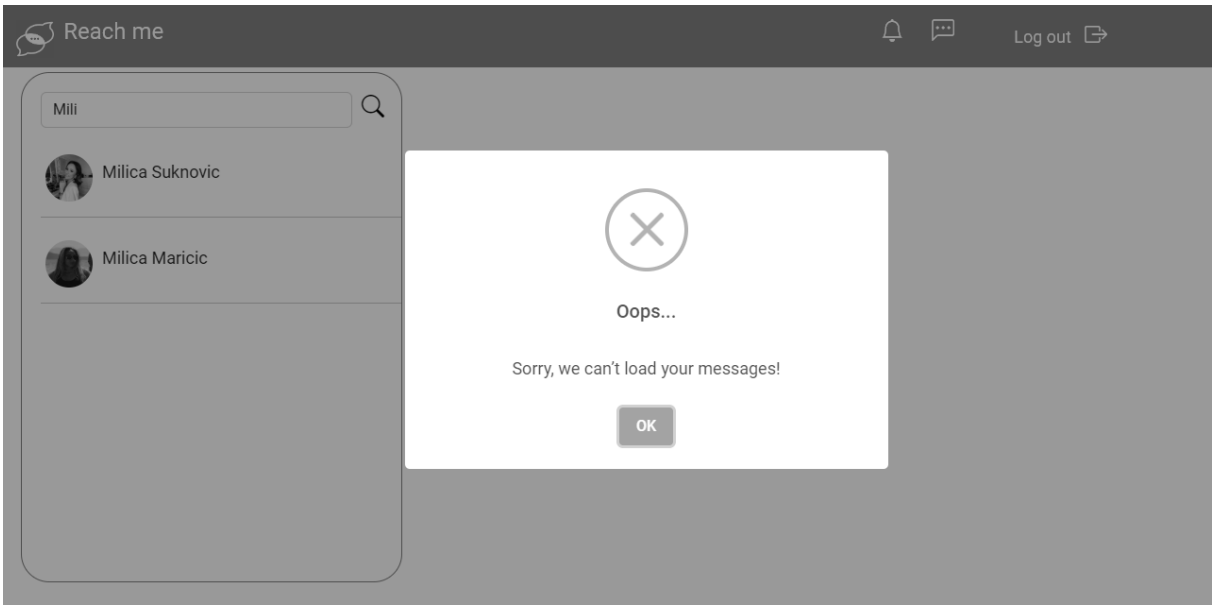
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



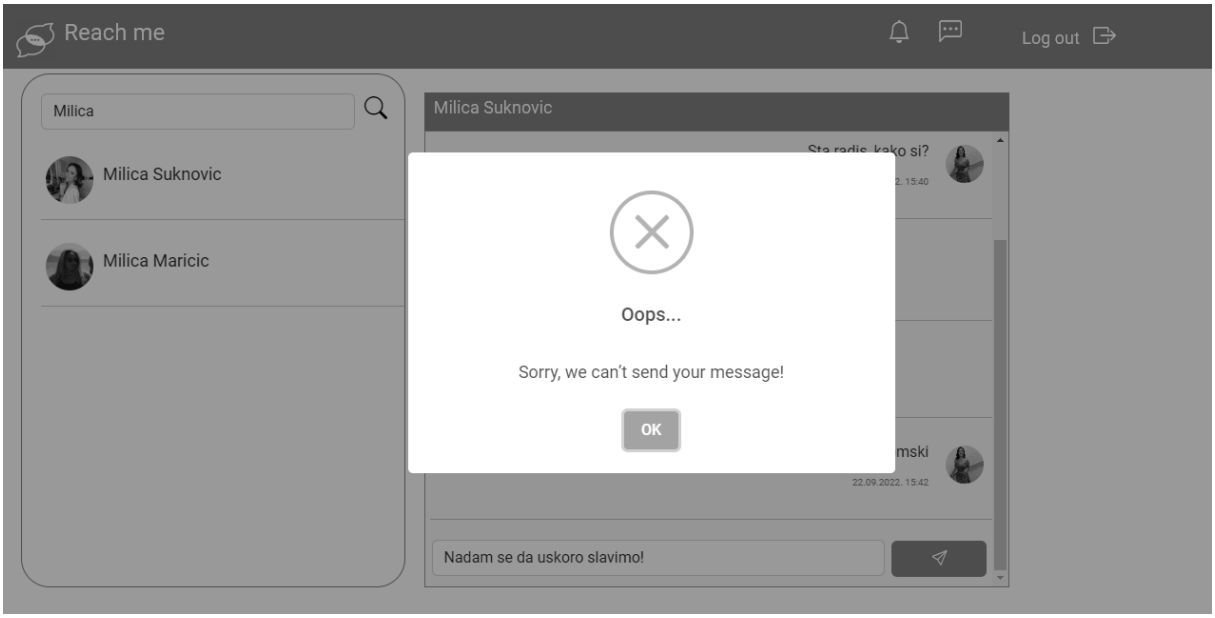
Слика 94. Систем не може да пронађе инбоџ кориснике

8.1. Уколико систем не може да приказе кориснику поруке са изабраним корисником, он **приказује** кориснику поруку: “Sorry, we can’t load your messages!”. (ИА)



Слика 95. Систем не може да учита поруке са изабраним корисником

13.1. Уколико систем не може да запамти поруку, он **приказује** кориснику поруку: “Sorry, we can’t send your message!”. (ИА)



Слика 96. Систем не може да пошаље поруку кориснику

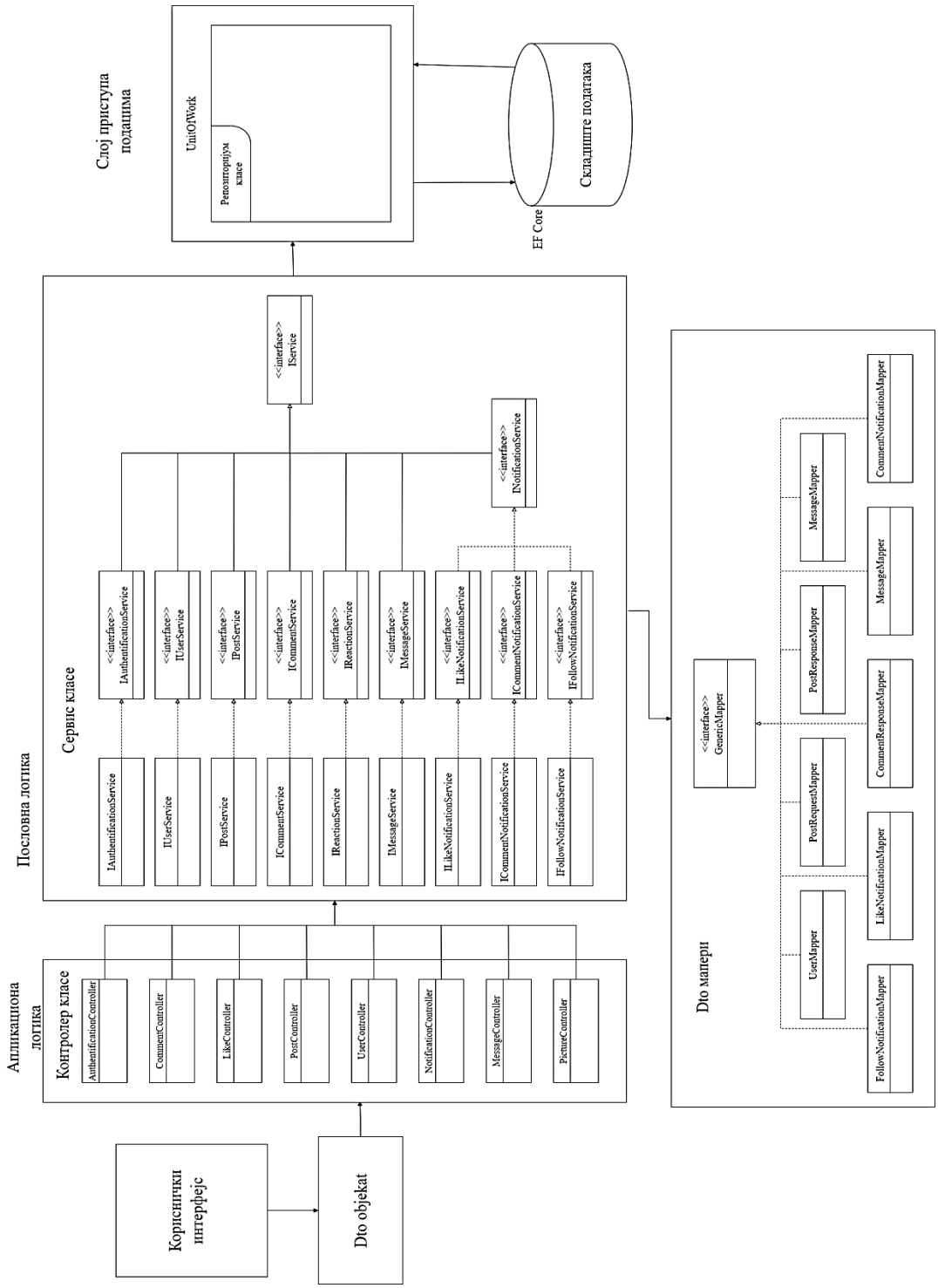
10.3. Пројектовање апликационе логике

Апликациона логика садржи све класе које су неопходне за имплементацију пословне логике:

1. Контролери – прихватају захтеве и прослеђују их сервисима даље на обраду. Представљају везу између корисничког интерфејса и пословне логике.
2. Сервиси – прихватају захтеве од контролера и трансформишу прослеђене објекте у облик погодан за рад са базом и обрнуто. Сваки сервис имплементира одговарајући сервис интерфејс, а сви ти интерфејси наслеђују један генерички сервис.
3. UnitOfWork – класа која је значајна за имплементацију Repository Pattern-а и омогућава да сви репозиторијуми раде над истим DbContext-ом.
4. Repository – За сваки објекат креира се Repository класа у којој ће се налазити сва комуникација са базом за тај доменски објекат. Сваки репозиторијум имплементира одговарајући репозиторијум интерфејс, а сви ти интерфејси наслеђују један генерички интерфејс. У генеричког интерфејсу се углавном дефинисане CRUD операције, док се у осталим интерфејсима налазе методе специфичне за тај репозиторијум.
5. Entity object – представљају табеле у бази и настали су као резултат објектно-релационог мапирања

10.3.1. Пројектовање контролера апликационе логике

Контролери апликационе логике су задужени за прихватање захтева од стране корисника и за њихово прослеђивање сервисима. Такође, служе и за прихватање резултата од сервиса и њихово прослеђивање кориснику.



Слика 97. Пословна логика система

10.3.2. Пословна логика

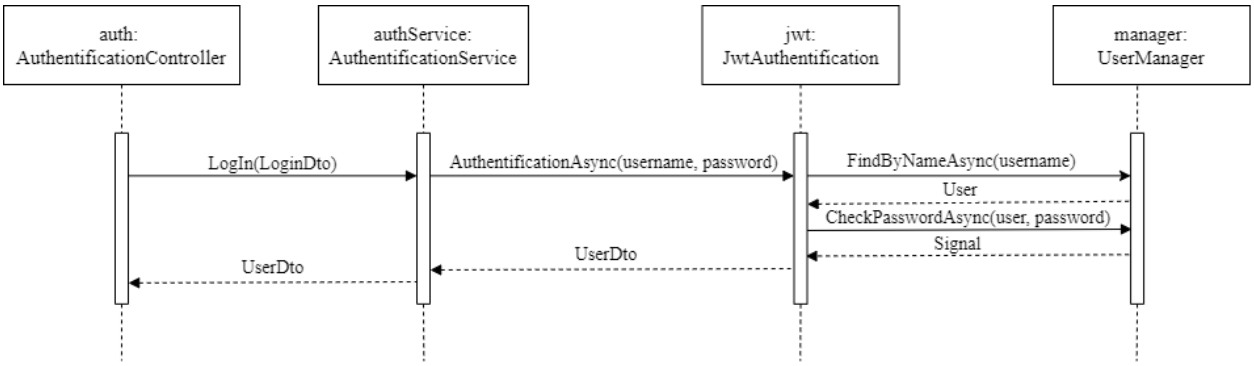
Пословна логика је описана структуром (доменским класама) и понашањем (системским операцијама). За сваки од уговора системских операција дефинисаних у фази анализе пројектује се концептуално решење.

Уговор УГ1: LogIn

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је улогован.



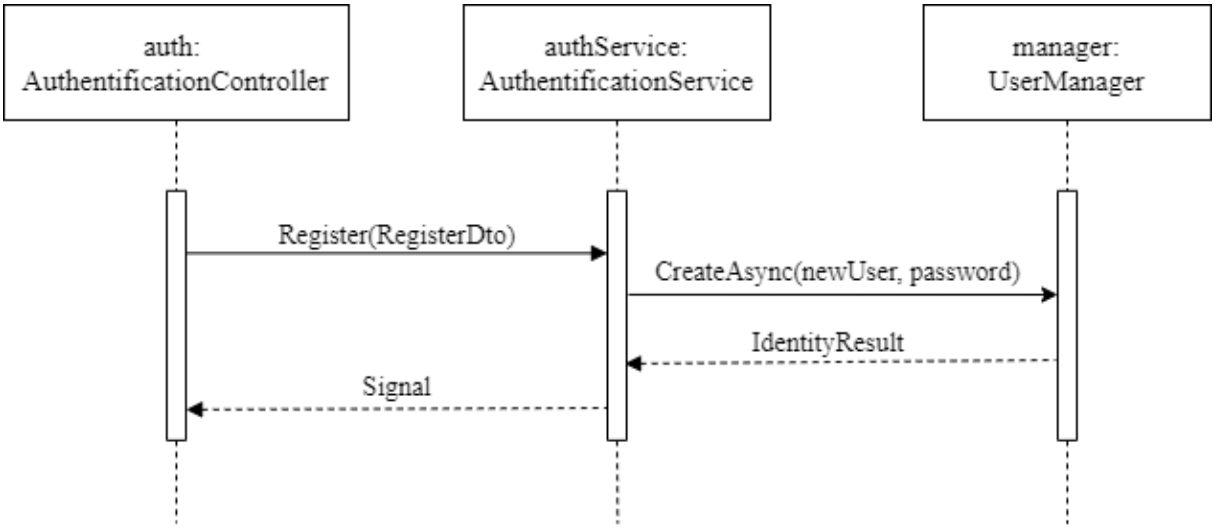
Слика 98. Дијаграм секвенци: Уговор - LogIn

Уговор УГ2: Register

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је регистрован.



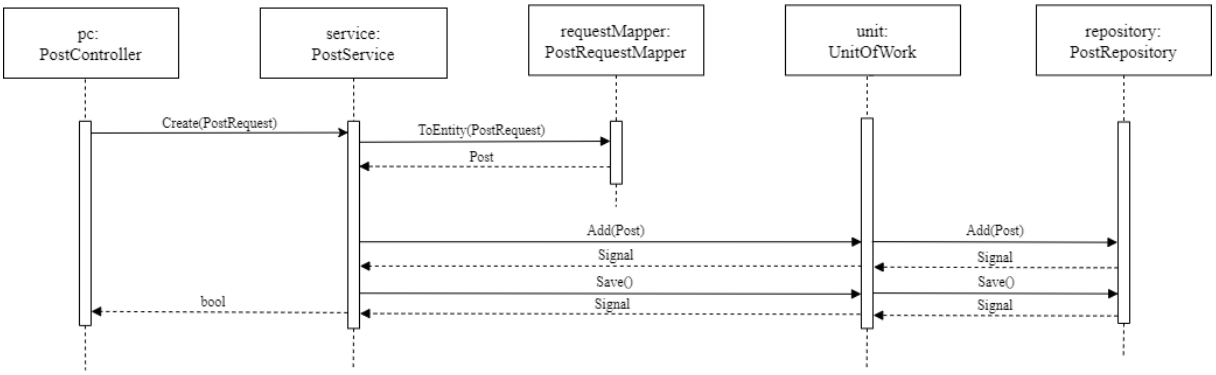
Слика 99. Дијаграм секвенци: Уговор - Register

Уговор УГ3: Create

Веза са СК: СК3

Предуслови: Вредносна и структурна ограничења над објектом **Post** морају бити задовољена.

Постуслови: Објава је запамћена.



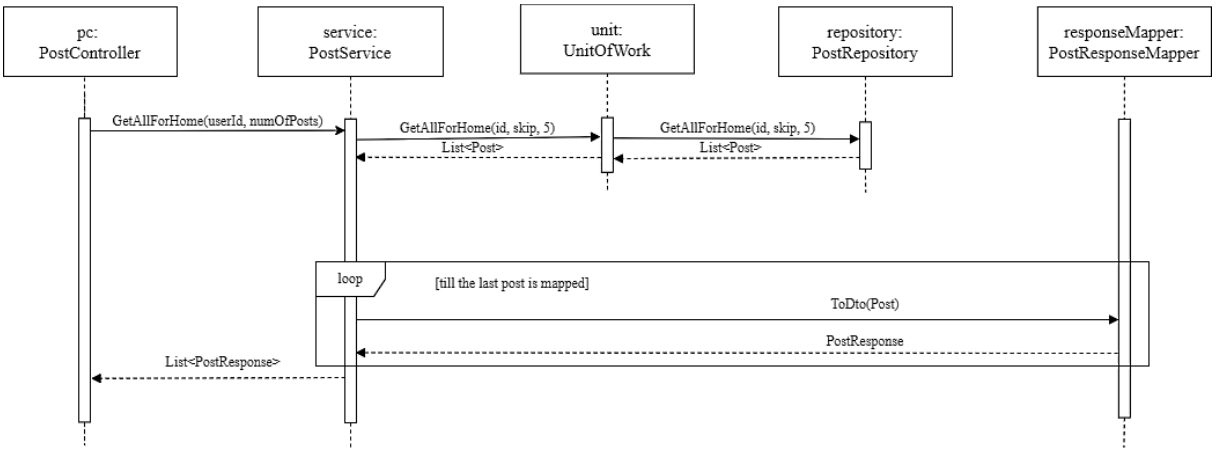
Слика 100. Дијаграм секвенци: Уговор - Create

Уговор УГ4: GetAllForHome

Веза са СК: СК4

Предуслови:

Постуслови:



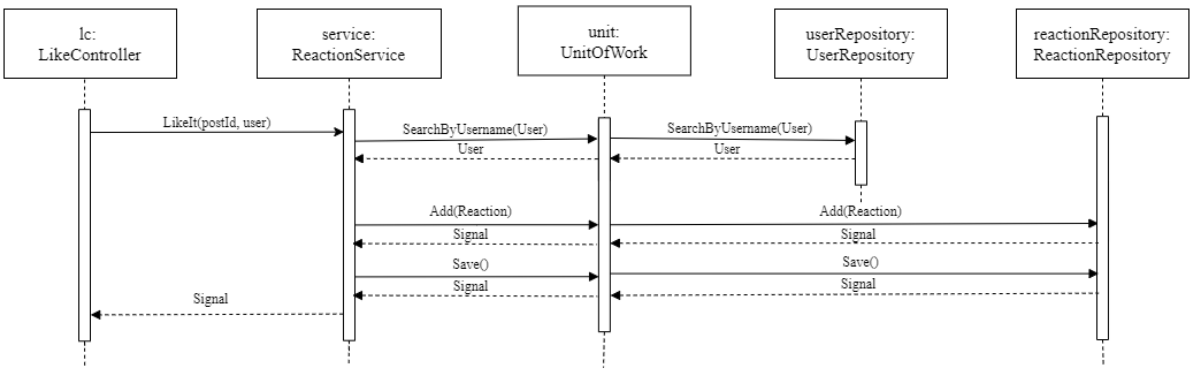
Слика 101. Дијаграм секвенци: Уговор - GetAllForHome

Уговор УГ5: LikeIt

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Reaction** морају бити задовољена.

Постуслови: Реакција је запамћена.



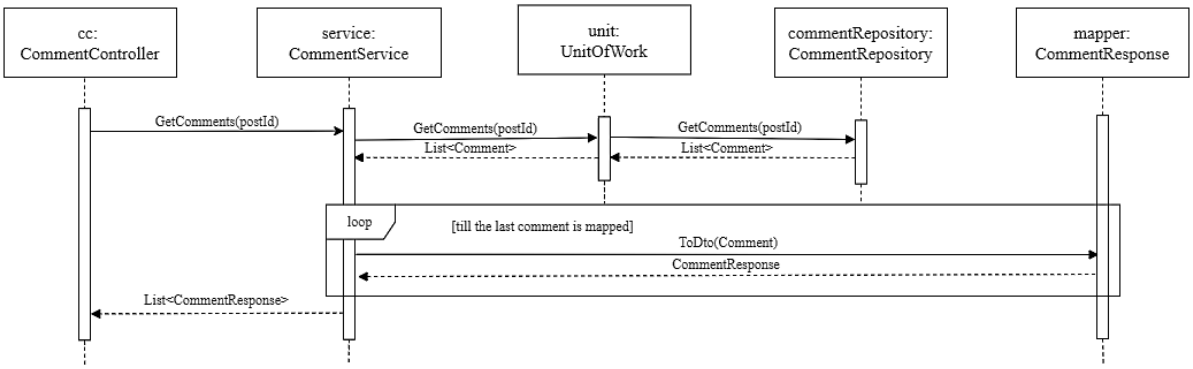
Слика 102. Дијаграм секвенци: Уговор - LikeIt

Уговор УГ6: GetComments

Веза са СК: СК5

Предуслови:

Постуслови:



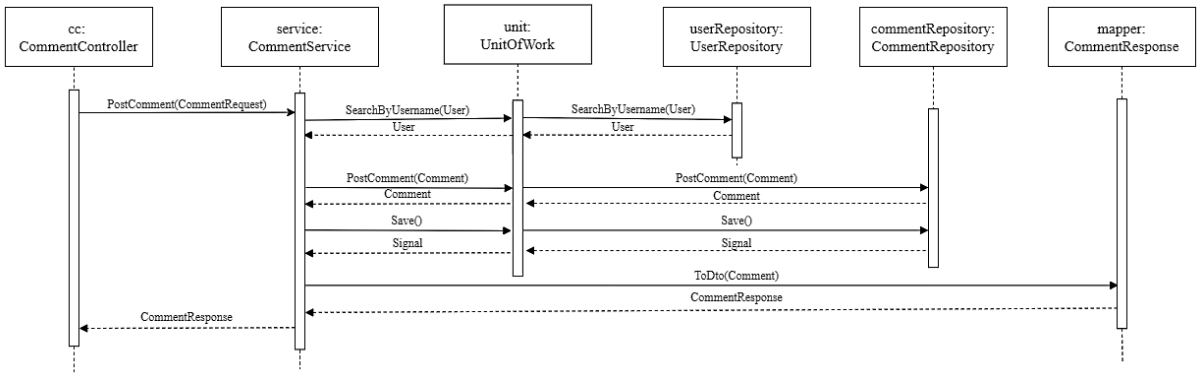
Слика 103. Дијаграм секвенци: Уговор - GetComments

Уговор УГ7: PostComment

Веза са СК: СК5

Предуслови: Вредносна и структурна ограничења над објектом **Comment** морају бити задовољена.

Постуслови: Коментар је запамћен.



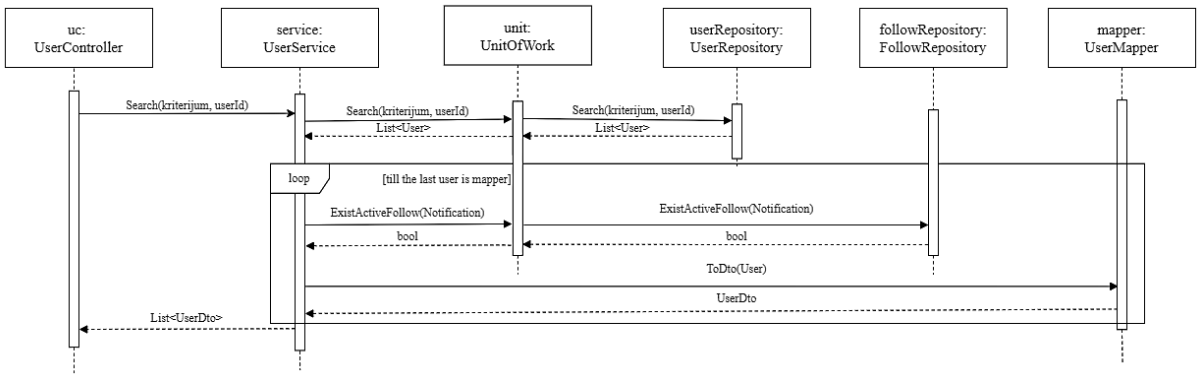
Слика 104. Дијаграм секвенци: Уговор - PostComment

Уговор УГ8: Search

Веза са СК: СК6

Предуслови:

Постуслови:



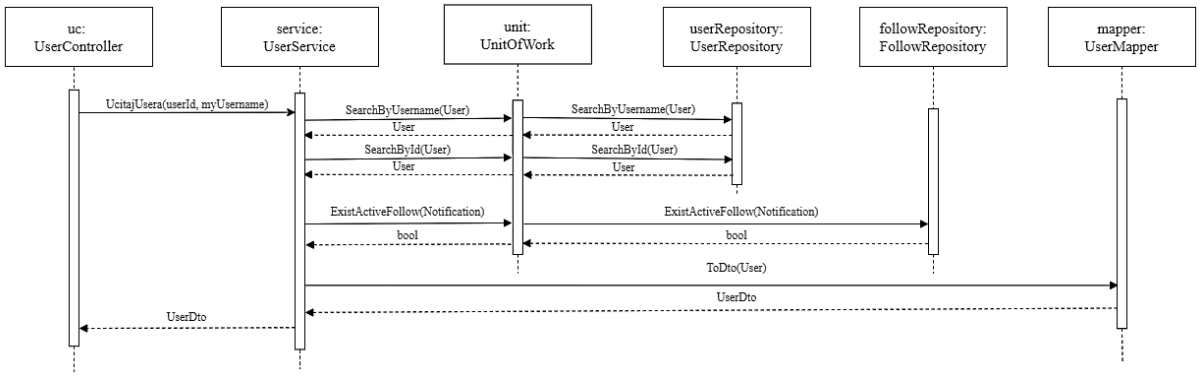
Слика 105. Дијаграм секвенци: Уговор - Search

Уговор УГ9: UcitajUsera

Веза са СК: СК6

Предуслови:

Постуслови:



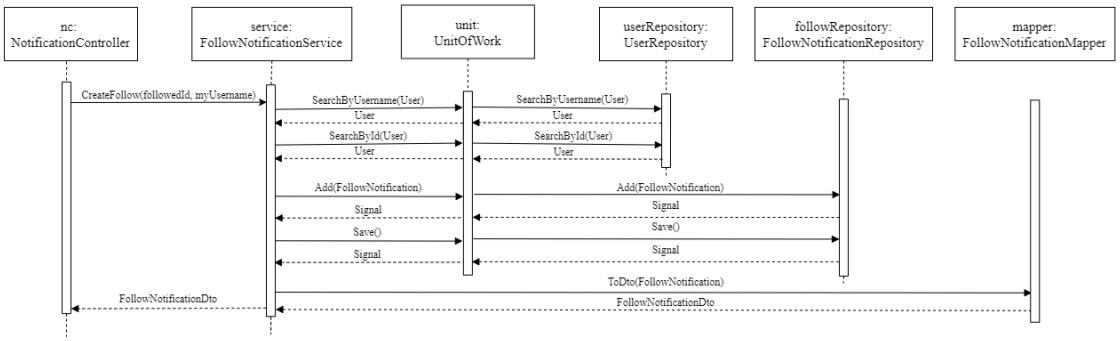
Слика 106. Дијаграм секвенци: Уговор - UcitajUsera

Уговор УГ10: CreateFollow

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **FollowNotification** морају бити задовољена.

Постуслови: Захтев је запамћен.



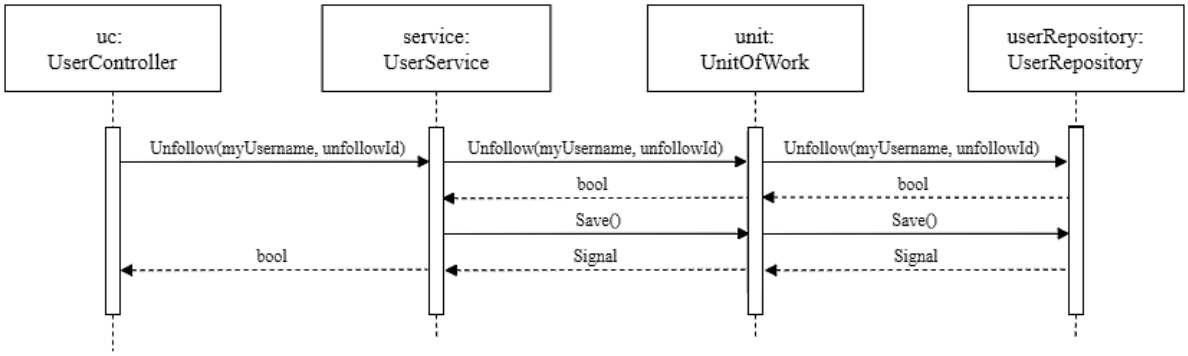
Слика 107. Дијаграм секвенци: Уговор - CreateFollow

Уговор УГ11: Unfollow

Веза са СК: СК8

Предуслови: Структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је отпраћен.



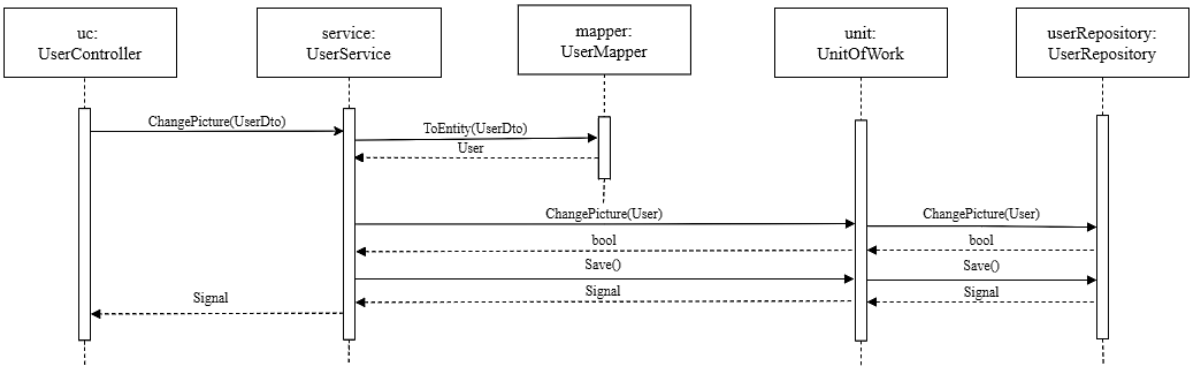
Слика 108. Дијаграм секвенци: Уговор - Unfollow

Уговор УГ12: ChangePicture

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Профилна слика корисника је промењена.



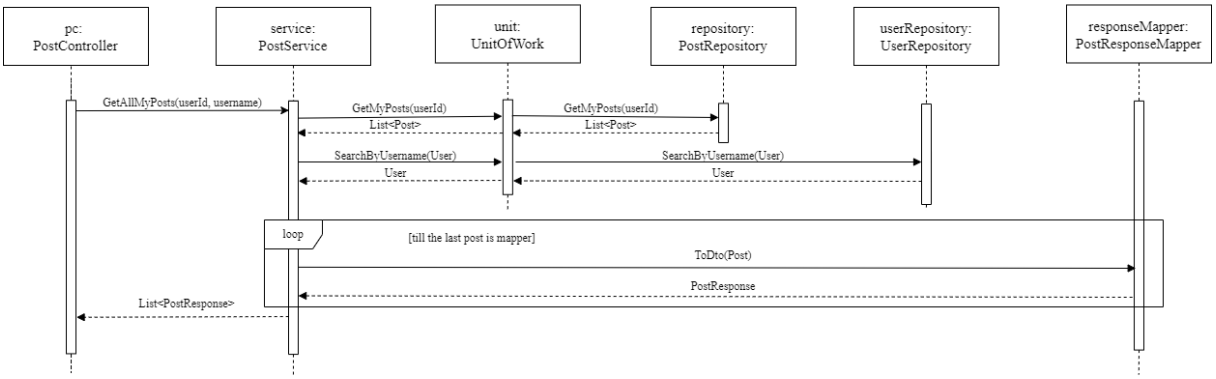
Слика 109. Дијаграм секвенци: Уговор - ChangePicture

Уговор УГ13: GetAllMyPosts

Веза са СК: СК9

Предуслови:

Постуслови:



Слика 110. Дијаграм секвенци: Уговор - GetAllMyPosts

Уговор УГ14: GetInboxUsers

Веза са СК: СК10

Предуслови:

Постуслови:

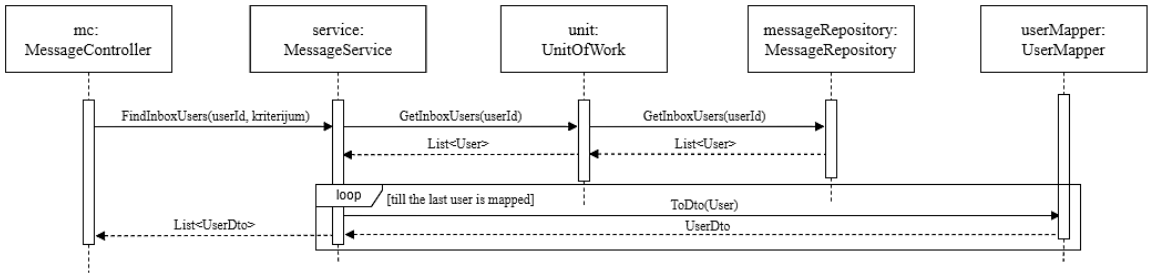
Слика 111. Дијаграм секвенци: Уговор - GetInboxUsers

Уговор УГ15: FindInboxUsers

Веза са СК: СК10

Предуслови:

Постуслови:



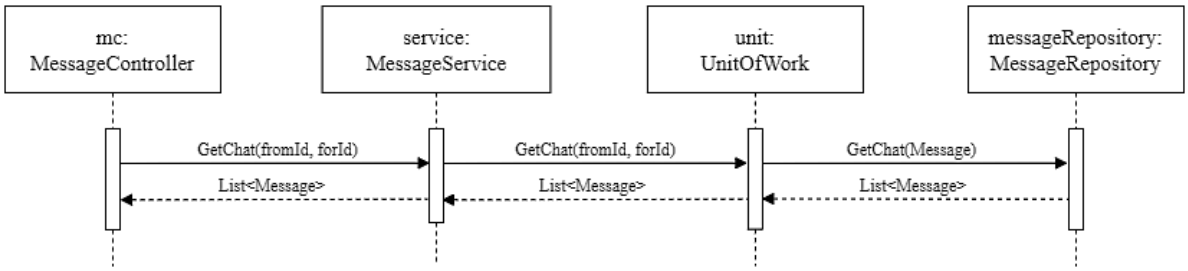
Слика 112. Дијаграм секвенци: Уговор - FindInboxUsers

Уговор УГ16: GetChat

Веза са СК: СК10

Предуслови:

Постуслови:



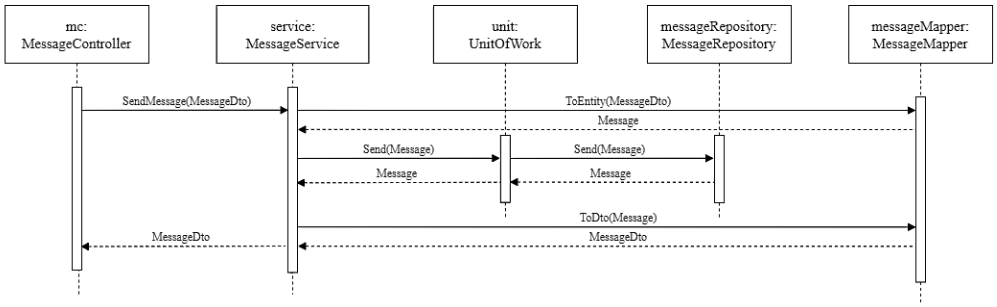
Слика 113. Дијаграм секвенци: Уговор - GetChat

Уговор УГ17: SendMessage

Веза са СК: СК10

Предуслови: Вредносна и структурна ограничења над објектом **Message** морају бити задовољена.

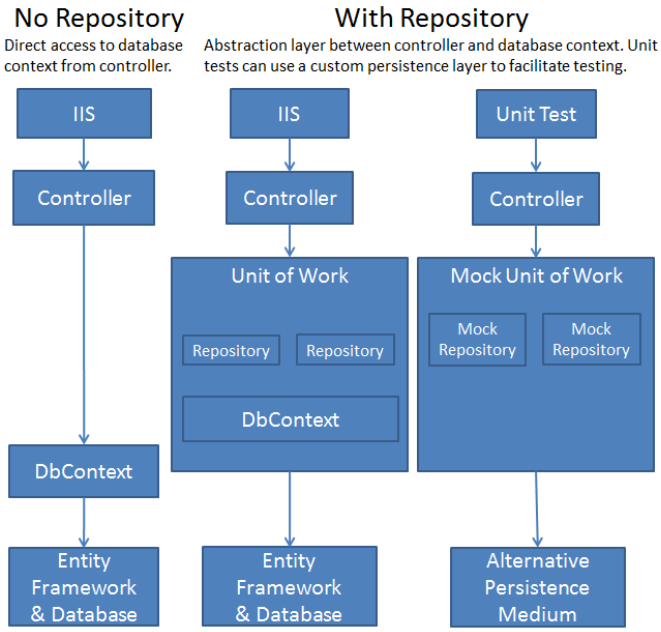
Постуслови: Порука је запамћена.



Слика 114. Дијаграм секвенци: Уговор - SendMessage

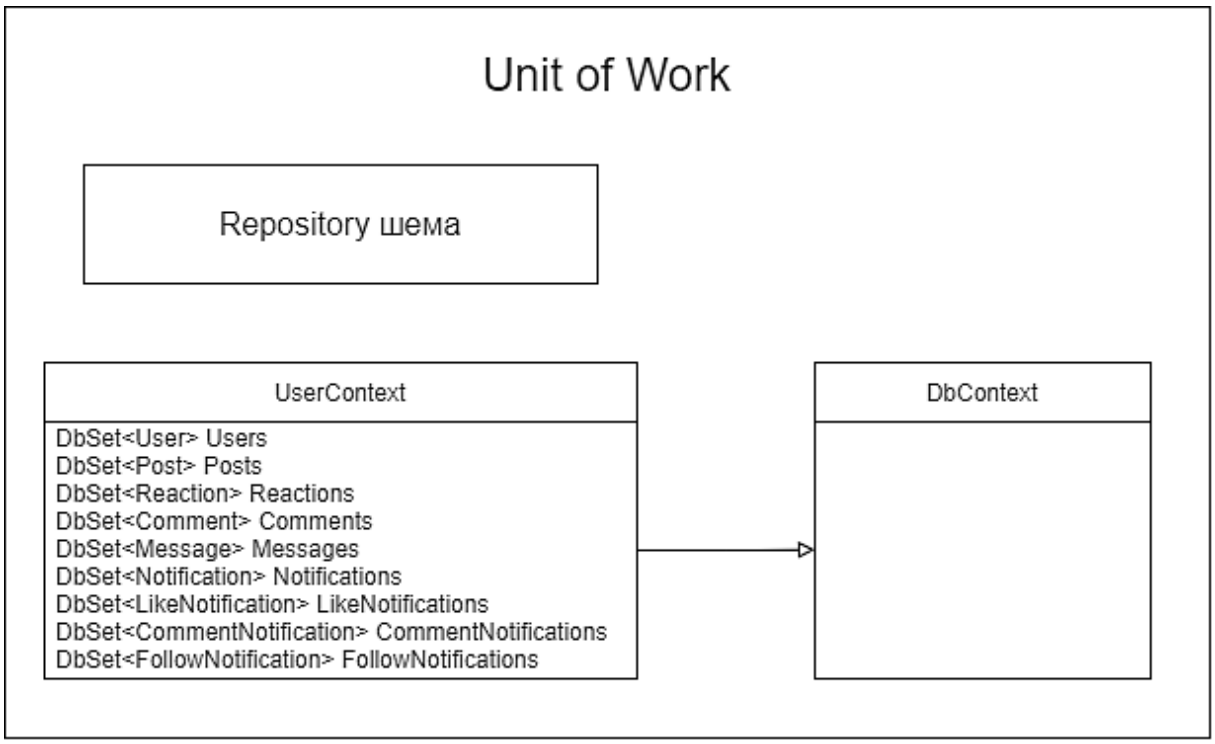
10.4. Пројектовање слоја приступа подацима

Главна намена Unit Of Work и Repository патерна је стварање слоја апстракције између пословне логике и слоја приступа подацима. Имплементација ових патерна помаже у изолацији апликације од промена у складишту података и може олакшати аутоматизовано тестирање јединица. [23]



Слика 115. Поређење веза између контролера и базе са и без UoW и Repository патерна
https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

Unit Of Work се користи за груписање једне или више операција (обично CRUD операције базе података) у једну трансакцију или „јединицу рада“ тако да све операције прођу или не успеју као једна јединица. Уколико све операције прођу може да се позове операција Commit() над базом.

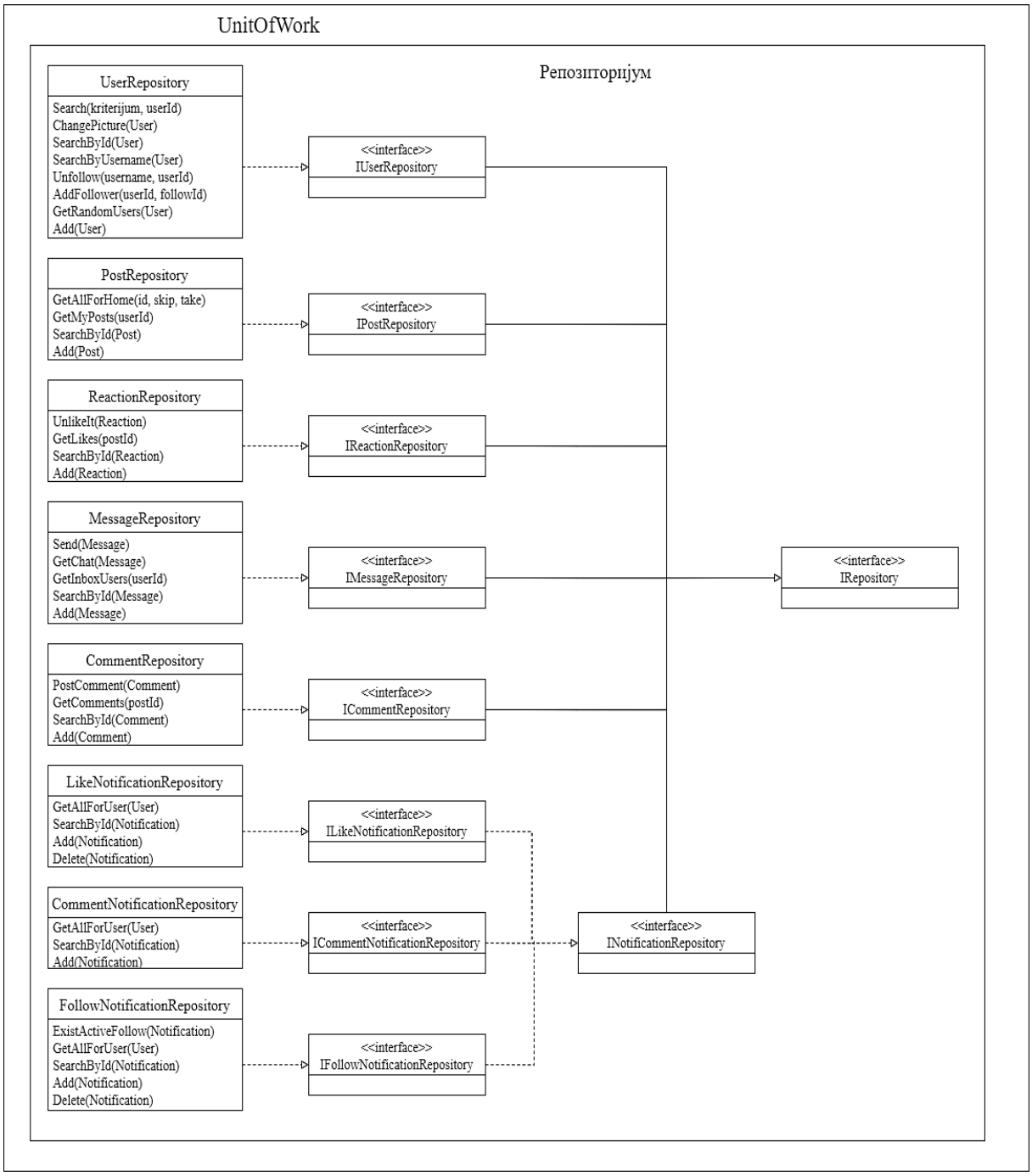


Слика 116. Unit of Work шема

Repository патерн

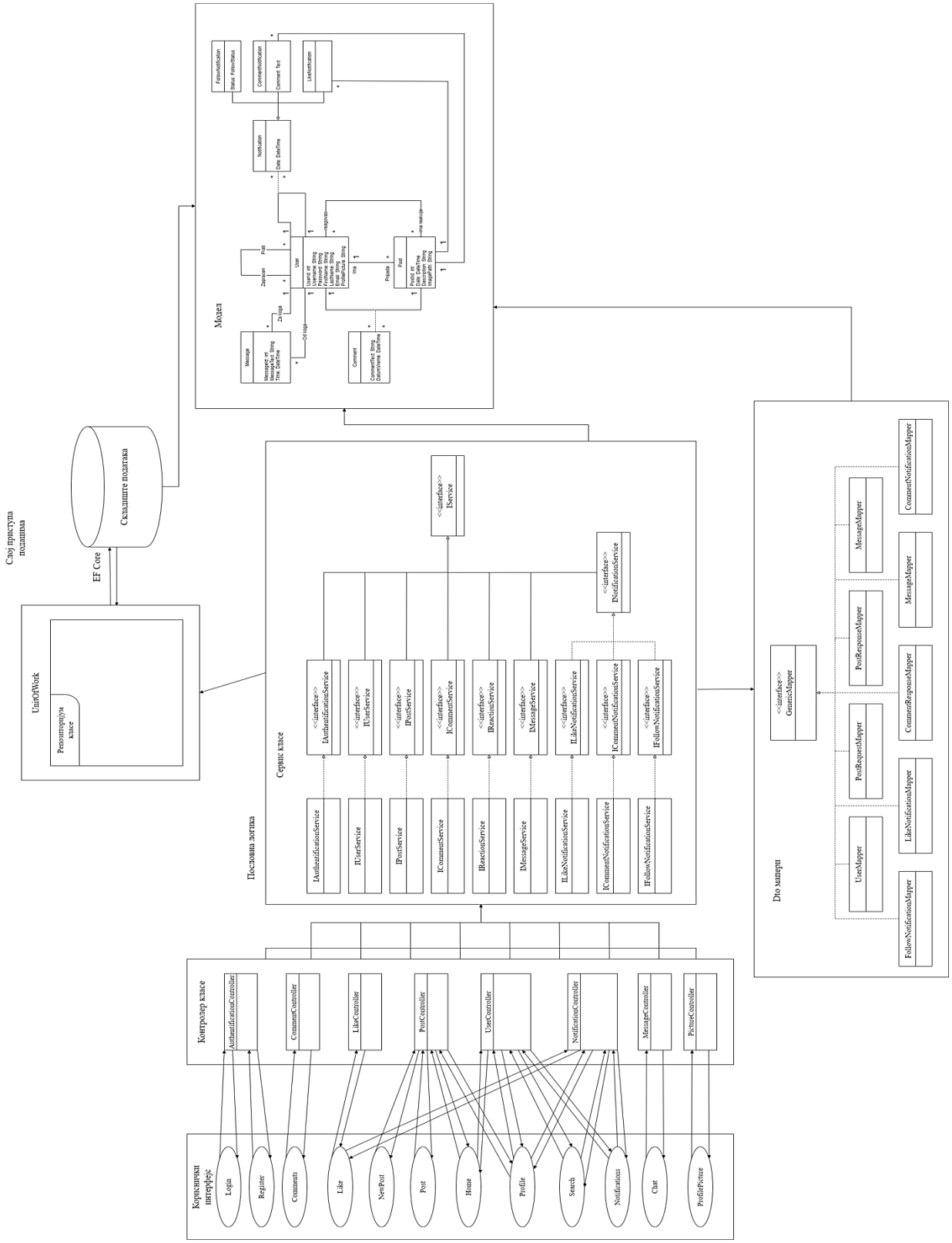
Репозиторијум није ништа друго до класа дефинисана за ентитет, са свим могућим операцијама на том специфичном ентитету. За сваки ентитет креира се репозиторијум класа и репозиторијум интерфејс који наслеђује интерфејс који представља генерички репозиторијум у којем се налазе методе које све репозиторијум класе треба да имплементирају, то су обично CRUD операције. У репозиторијум интерфејсима налазе се методе које су специфичне за тај доменски објекат.

Repository патерн



Слика 117 Repository шема

10.5. Коначан изглед архитектуре софтверског система



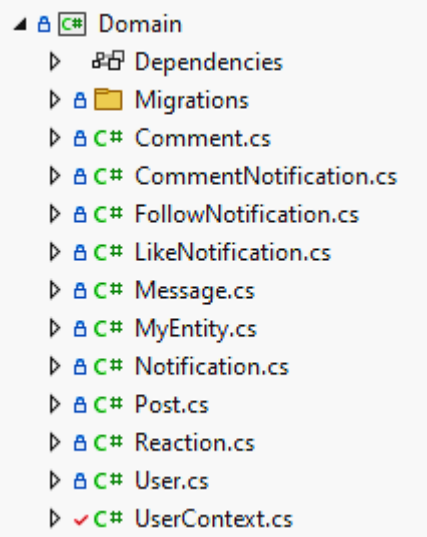
Слика 118. Приказ коначне архитектуре система

11. Фаза имплементације

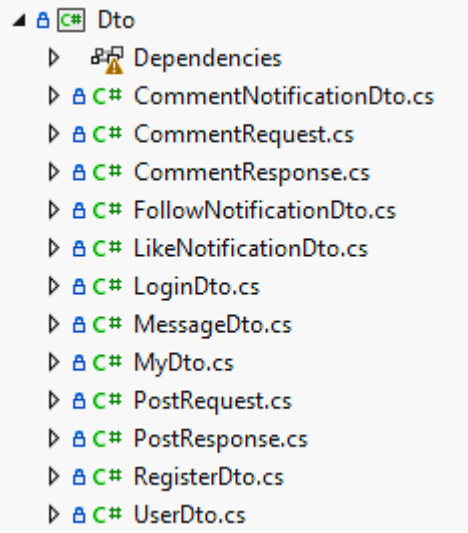
Као резултат овог рада настала је веб апликација са Web API-јем у програмском језику C# уз помоћ ASP.NET Core платформе и SignalR оквира. Развојно окружење које је коришћено за развој серверске стране је Visual Studio 2022. Клијентска апликација направљена је помоћу Angular-a, развојно окружене које је коришћено је Visual Studio Code.

11.1. Структура софтверског решења

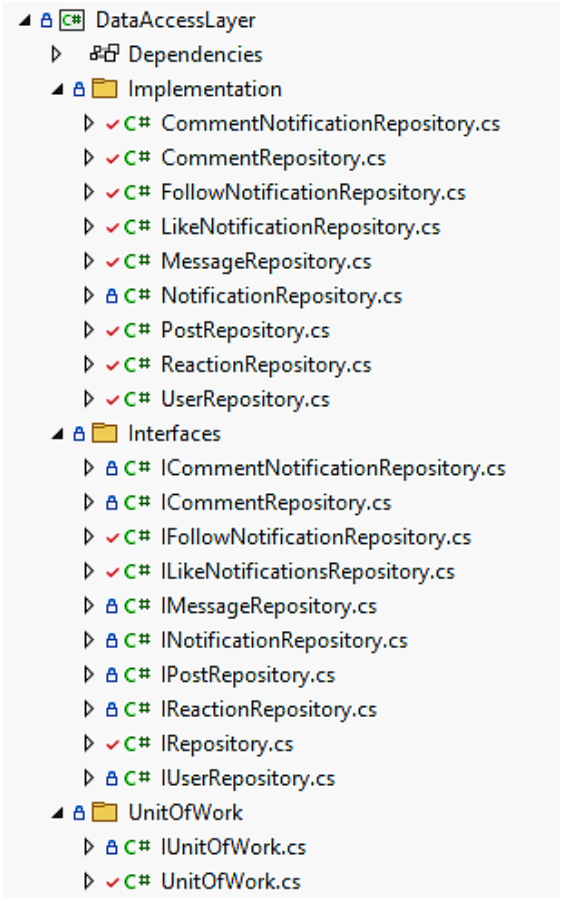
На серверској страни имплементиране су следеће класе:



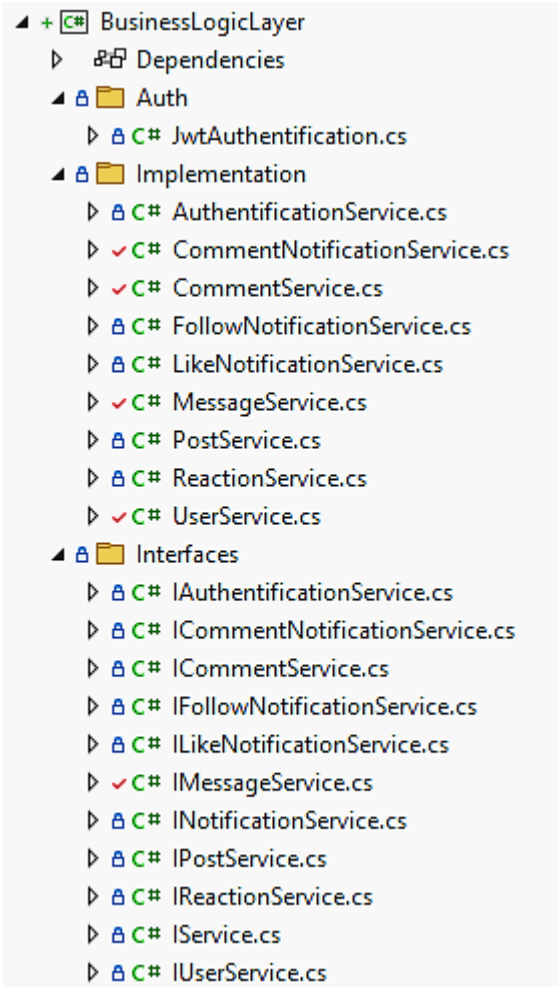
Слика 119. Пројекат "Domain"



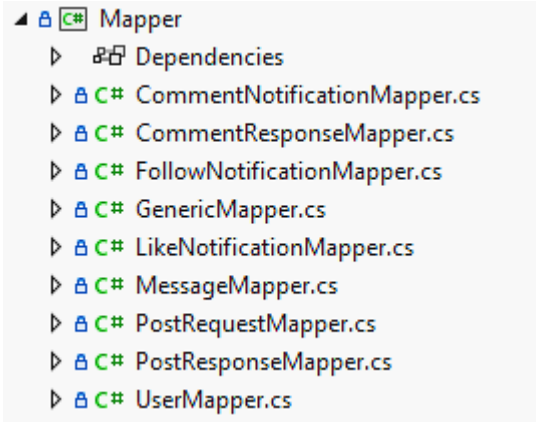
Слика 120. Пројекат "Dto"



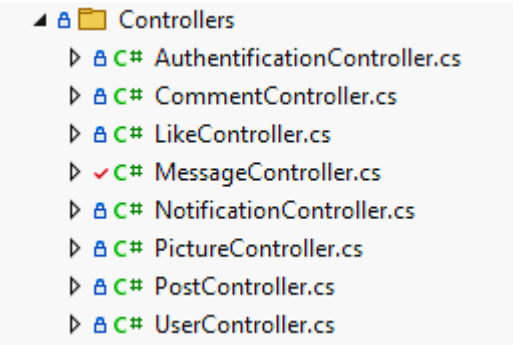
Слика 121. Пројекат "DataAccessLayer"



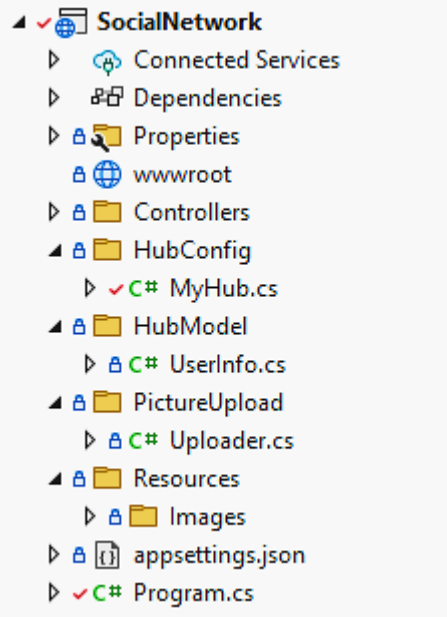
Слика 122. Пројекат "BusinesLogicLayer"



Слика 123. Пројекат "Mapper"



Слика 124. Пројекат "Controllers"



Слика 125. Пројекат "SocialNetwork"

11.2. Имплементација корисничког интерфејса

СК1: Случај коришћења - Пријављивање на друштвену мрежу

Назив СК

Пријављивање на друштвену мрежу

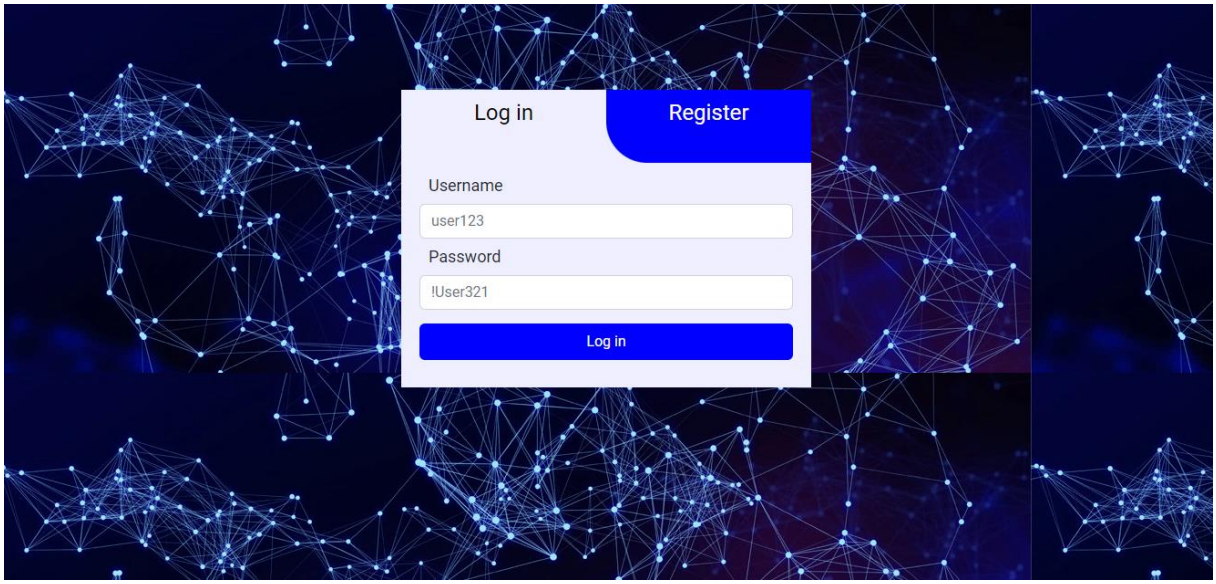
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

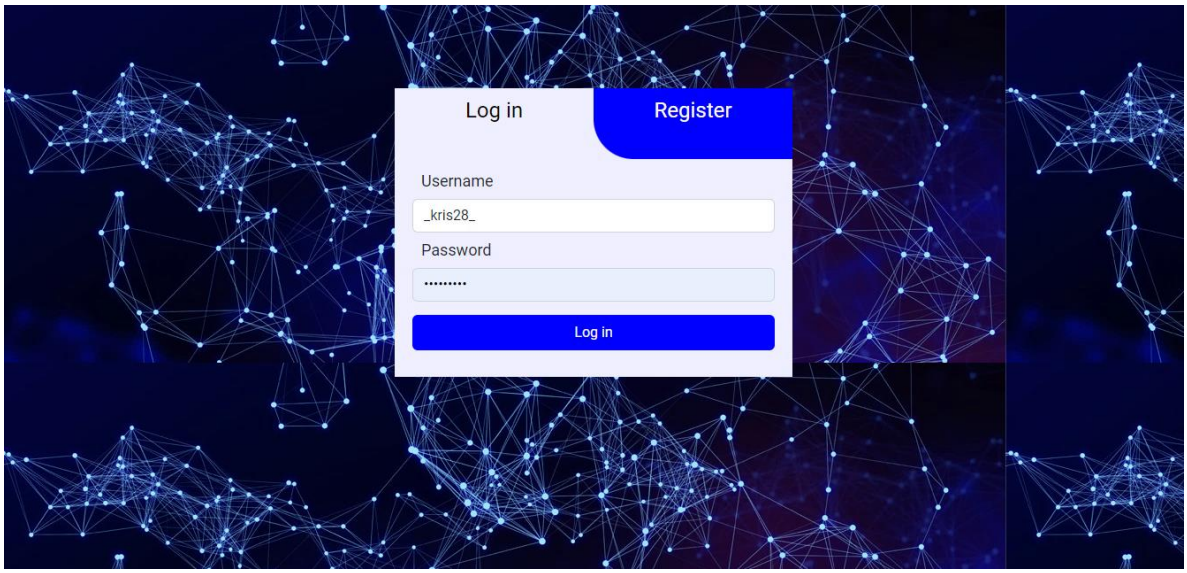
Предуслов: Систем је укључен и приказује форму за пријављивање на систем.



Слика 126. Форма за пријаву на систем

Основни сценарио СК

1. Корисник **уноси** податке за пријављивање (корисничко име и лозинку). (АПУСО)



Слика 127. Попуњена форма за пријаву на систем

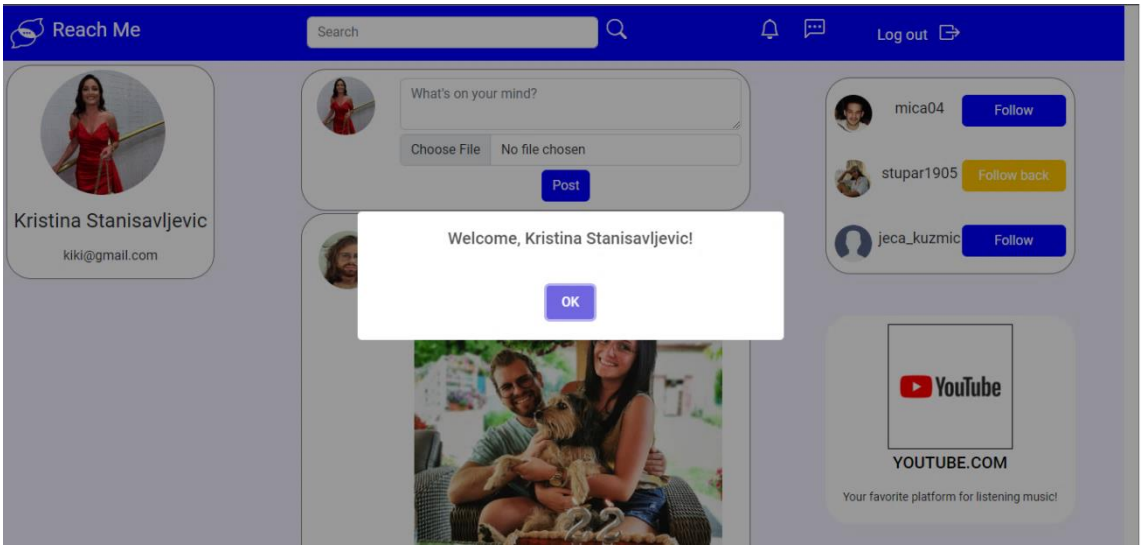
2. Корисник **контролише** да ли је коректно унео корисничко име и лозинку. (АНСО)

3. Корисник **позива** систем да га пријави. (АПСО)

Опис акције: Кликом на дугме „Log in“ корисник позива системску операцију LogIn(UserDto).

4. Систем **проверава** податке о кориснику. (СО)

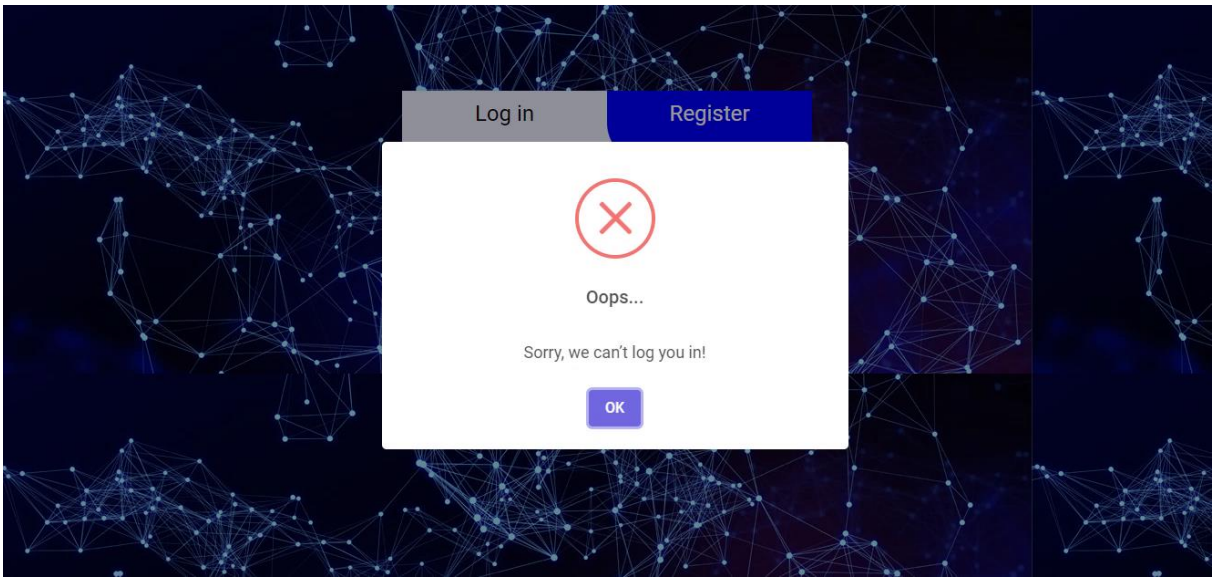
5. Систем **приказује** кориснику почетну страну и поруку: “Welcome, ime prezime!“. (ИА)



Слика 128. Успешна пријава на систем

Алтернативна сценарија

5.1. Уколико систем не може да нађе корисника, он **приказује** кориснику поруку: “Sorry, we can’t log you in!”. (ИА)



Слика 129. Неуспешна пријава

СК2: Случај коришћења – Креирање корисничког налога

Назив СК

Креирање корисничког налога

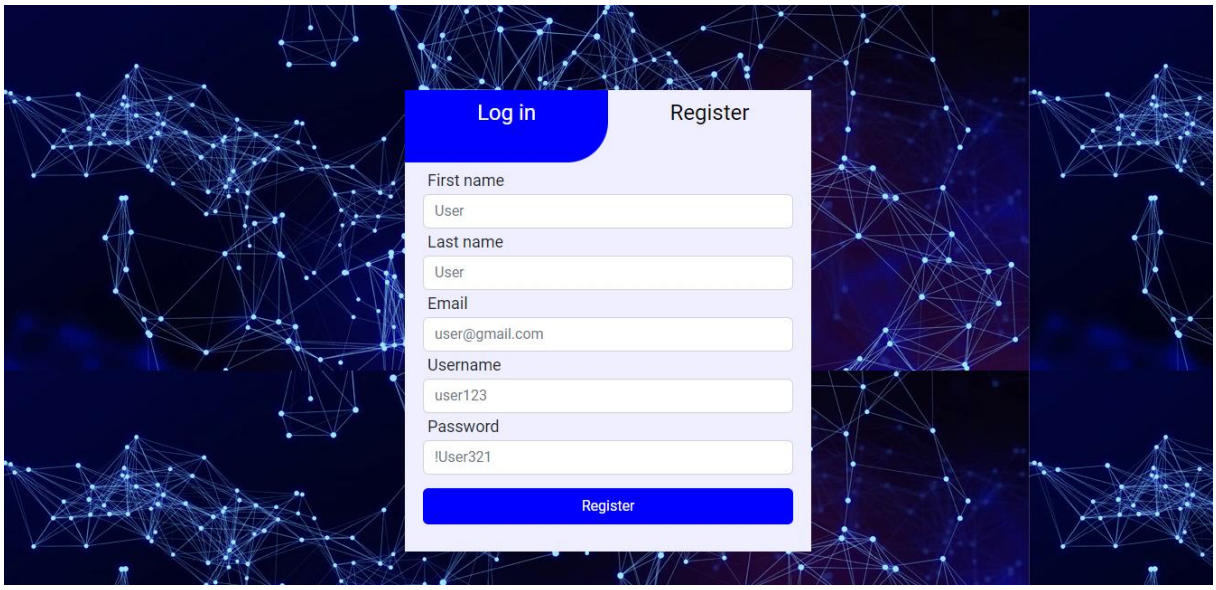
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

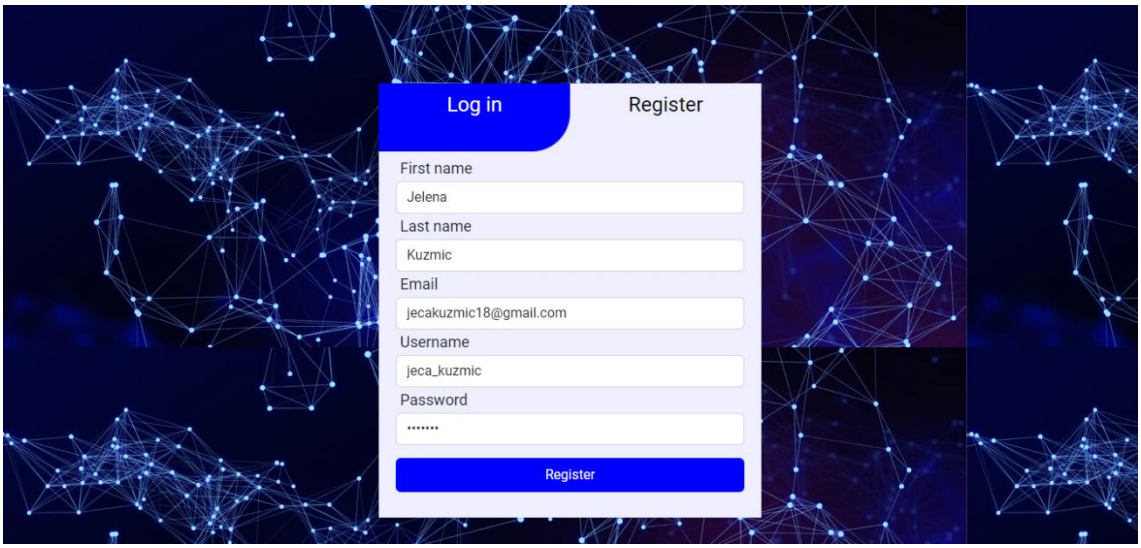
Предуслов: Систем је укључен и приказује форму за регистрацију на систем.



Слика 130. Форма за регистрацију

Основни сценарио СК

1. Корисник **уноси** основне податке за креирање корисничког налога. (АПУСО)



Слика 131. Попуњена форма за регистрацију

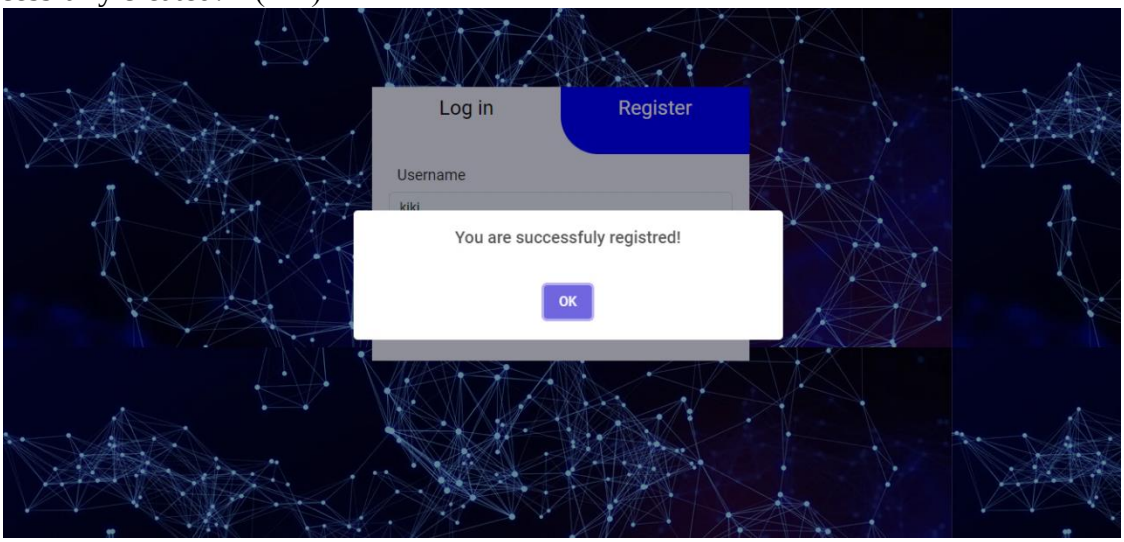
2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)

3. Корисник **позива** систем да га региструје. (АПСО)

Опис акције: Кликом на дугме „Register“ корисник позива системску операцију Register(RegisterDto).

4. Систем **памти** податке о кориснику. (СО)

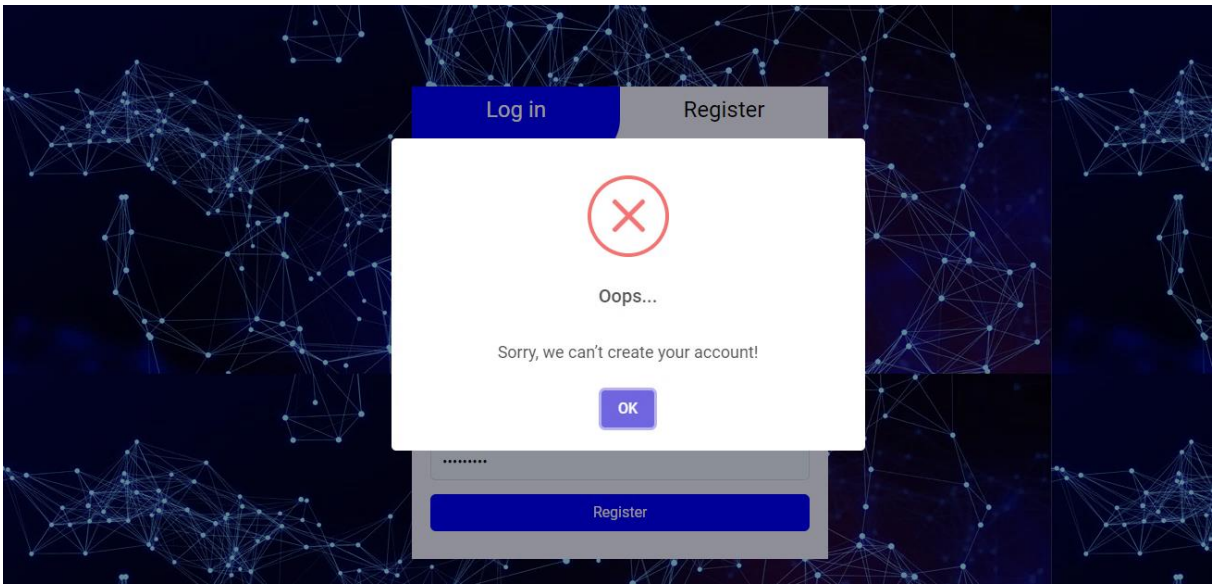
5. Систем **приказује** кориснику страницу за пријаву и поруку: “Your account is successfully created!“. (ИА)



Слика 132. Успешна регистрација

Алтернативна сценарија

5.1. Уколико систем не може да региструје корисника, он **приказује** кориснику поруку: “Sorry, we can’t create your account!”. (ИА)



Слика 133. Неуспешна регистрација

СКЗ: Случај коришћења – Креирање нове објаве

Назив СК

Креирање нове објаве

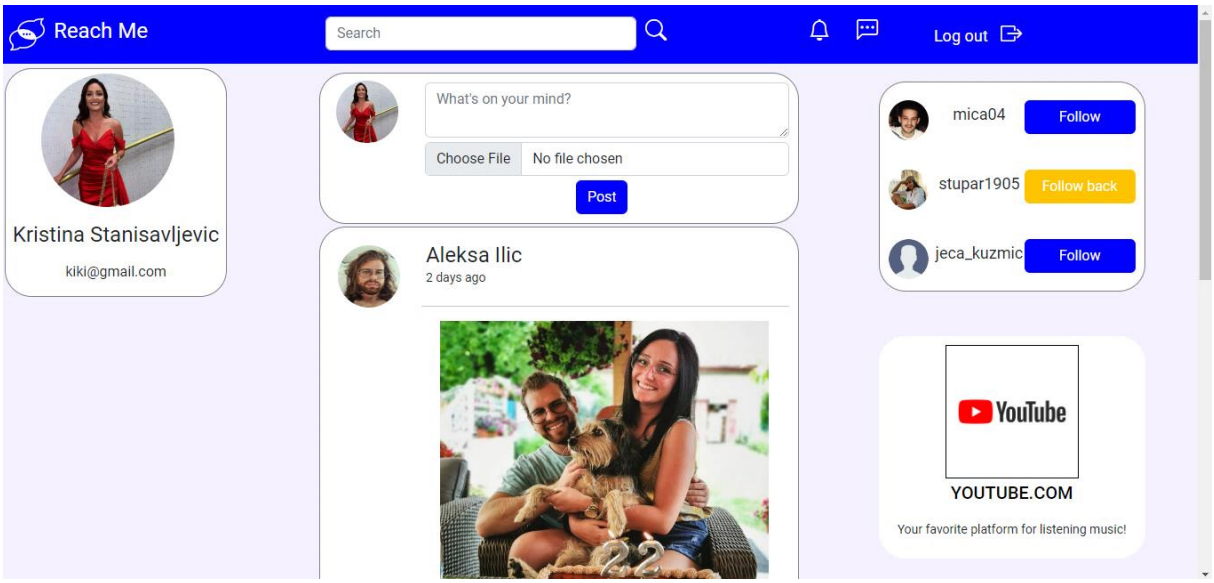
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

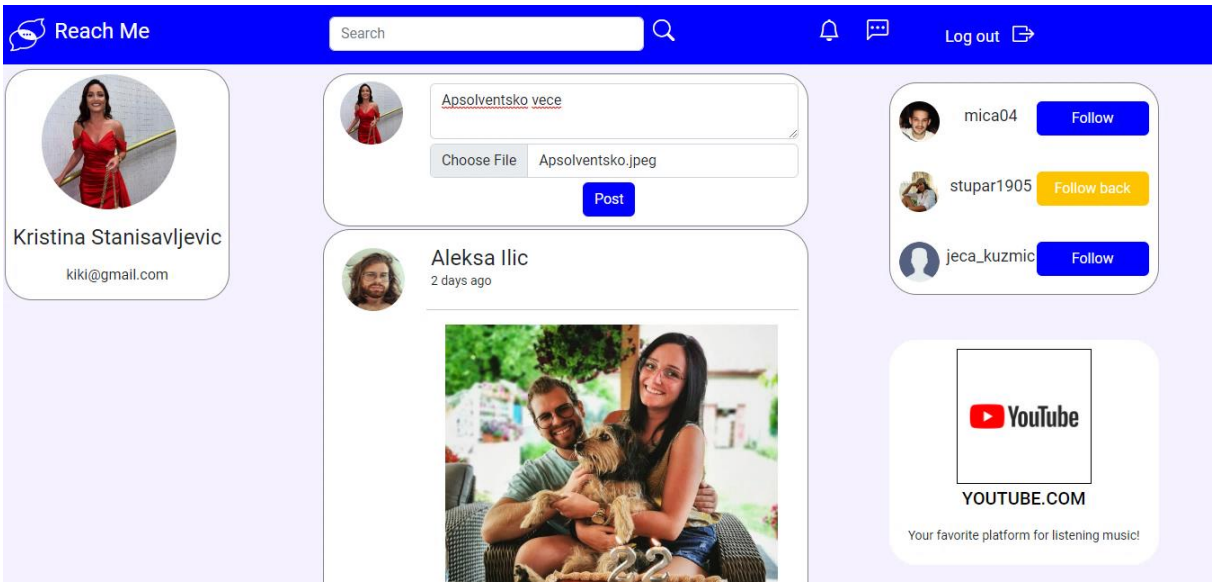
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника, на којој се налази форма за унос нове објаве.



Слика 134. Почетна страница на којој се налази форма за креирање нове објаве

Основни сценарио СК

1. Корисник **уноси** податке о објави. (АПУСО)



Слика 135. Унети подаци о новој објави

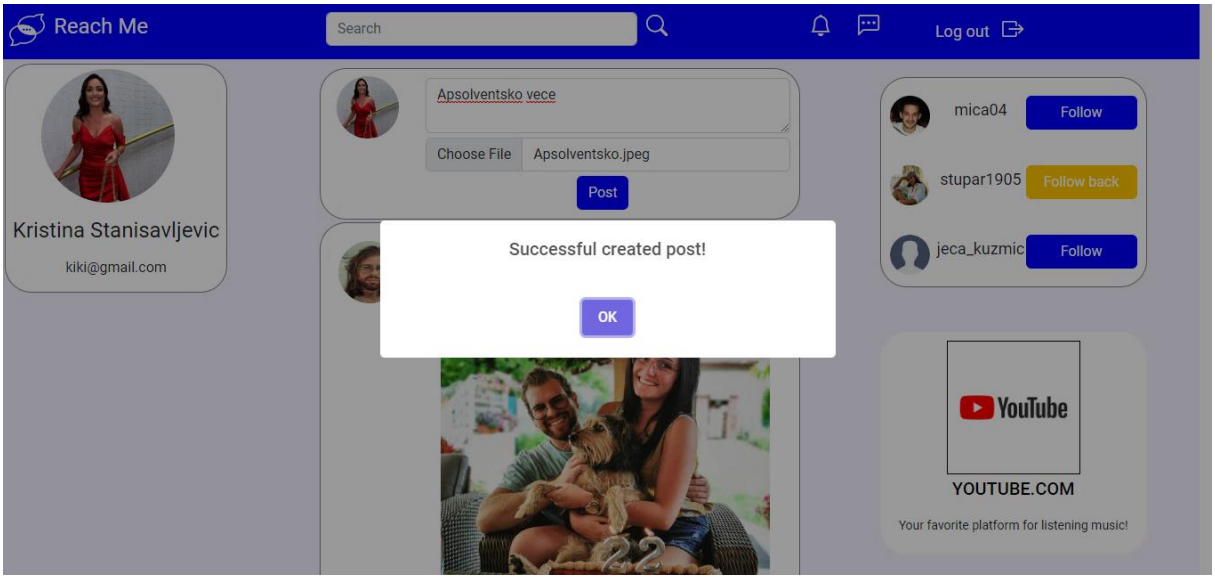
2. Корисник **контролише** да ли је коректно унео своје податке. (АНСО)

3. Корисник **позива** систем да запамти податке о објави. (АПСО)

Опис акције: Кликом на дугме „Post“ корисник позива системску операцију Create(PostRequest).

4. Систем **памти** податке о објави. (СО)

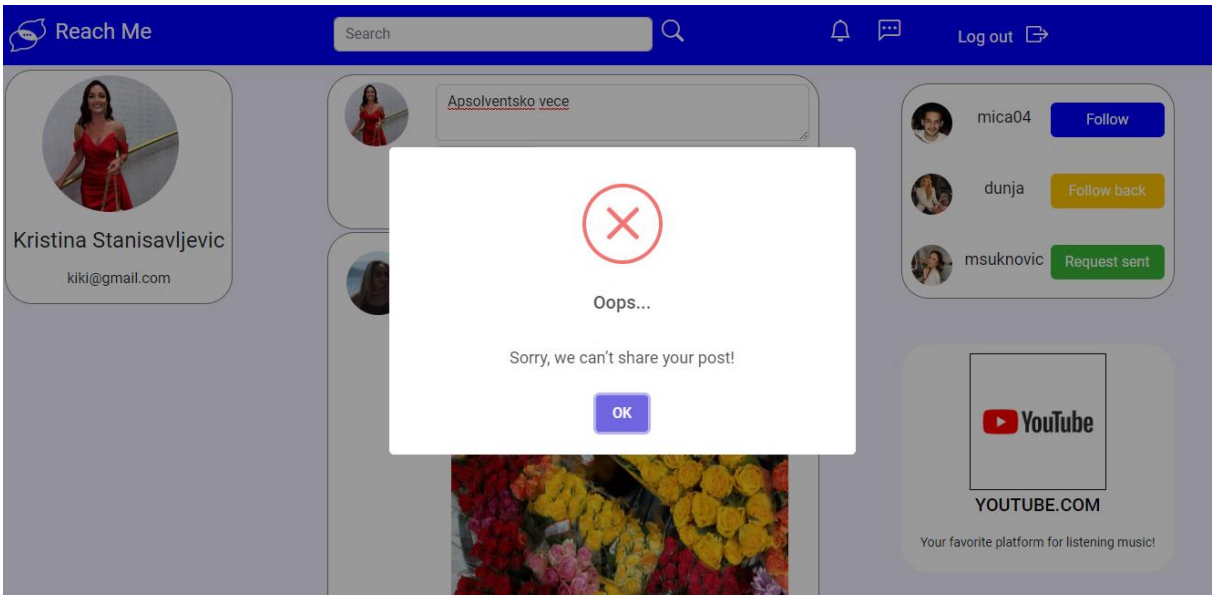
5. Систем **приказује** кориснику поруку: “Your post has been shared!”. (ИА)



Слика 165. Успешно креирана објава

Алтернативна сценарија

5.1. Уколико систем не може да запамти податке о објави, он **приказује** кориснику поруку: “Sorry, we can’t share your post!”. (ИА)



Слика 137. Неуспешно креирана објава

СК4: Случај коришћења – Реаговање на објаву

Назив СК

Реаговање на објаву

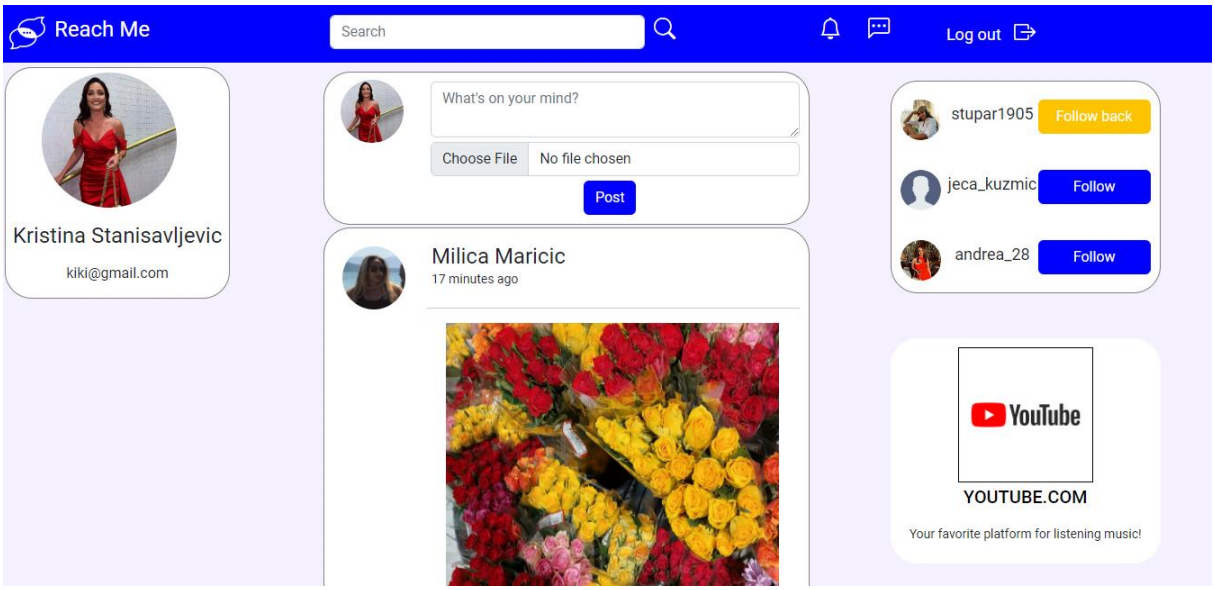
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

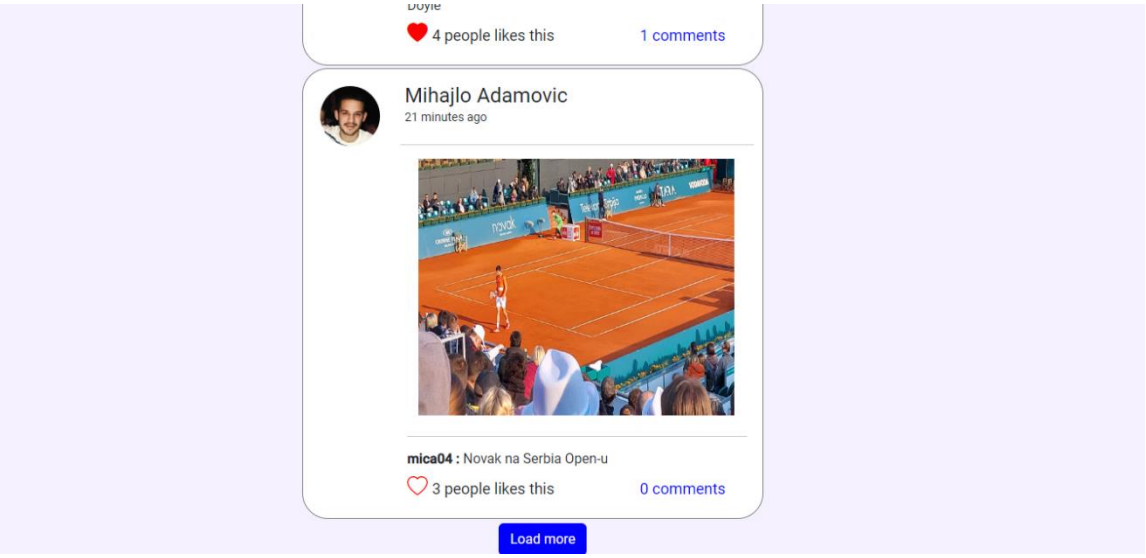
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих објава особа које корисник прати.



Слика 138. Почетна страница

Основни сценарио СК

1. Корисник **бира** објаву на коју жели да реагује. (АПУСО)



Слика 139. Изабрана објава

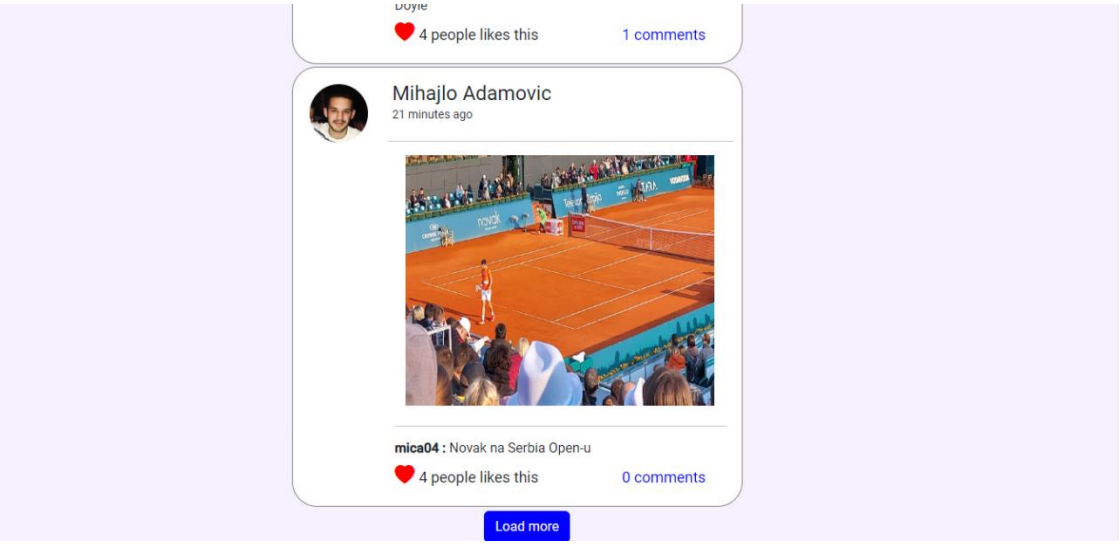
2. Корисник **позива** систем да реагује на одабрану објаву. (АПСО)

Опис акције: Кликом на дугме корисник позива системску операцију

LikeIt(postId, userId).

3. Систем **памти** реакцију за одабрану објаву. (СО)

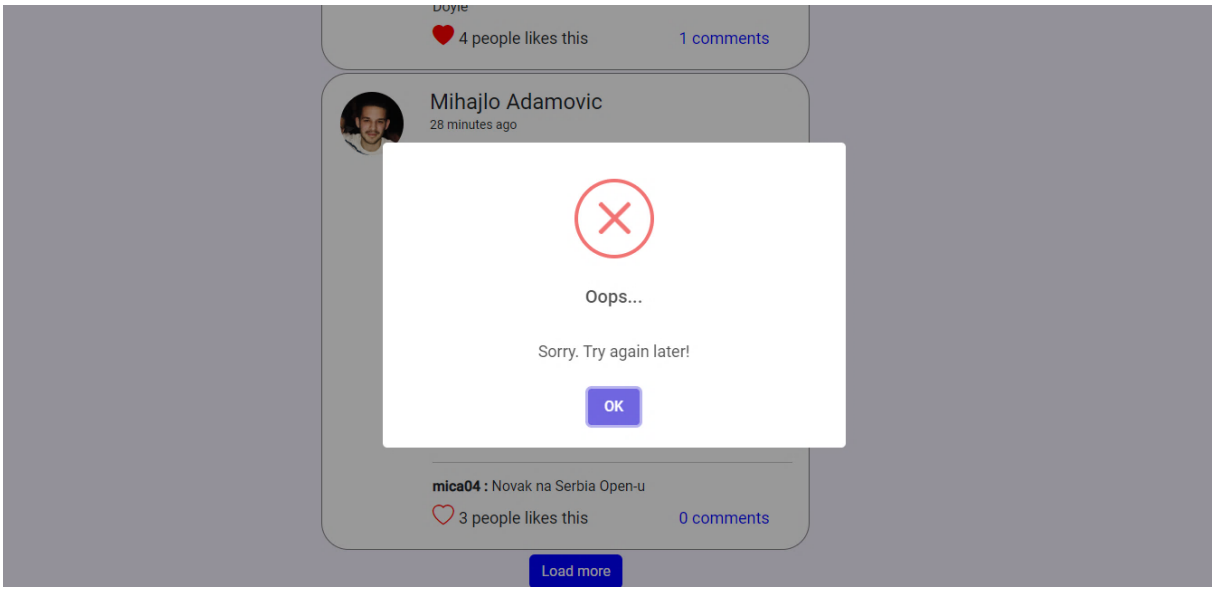
4. Систем **приказује** кориснику да је успешно реаговао на објаву. (ИА)



Слика 140. Успешно реагована објава

Алтернативна сценарија

4.1. Уколико систем не може да запамти реакцију за одабрану објаву, он **приказује** кориснику поруку: “Sorry. Try again later!”. (ИА)



Слика 141. Корисник не може да реагује на објаву

СК5: Случај коришћења – Коментарисање објаве

Назив СК

Коментарисање објаве

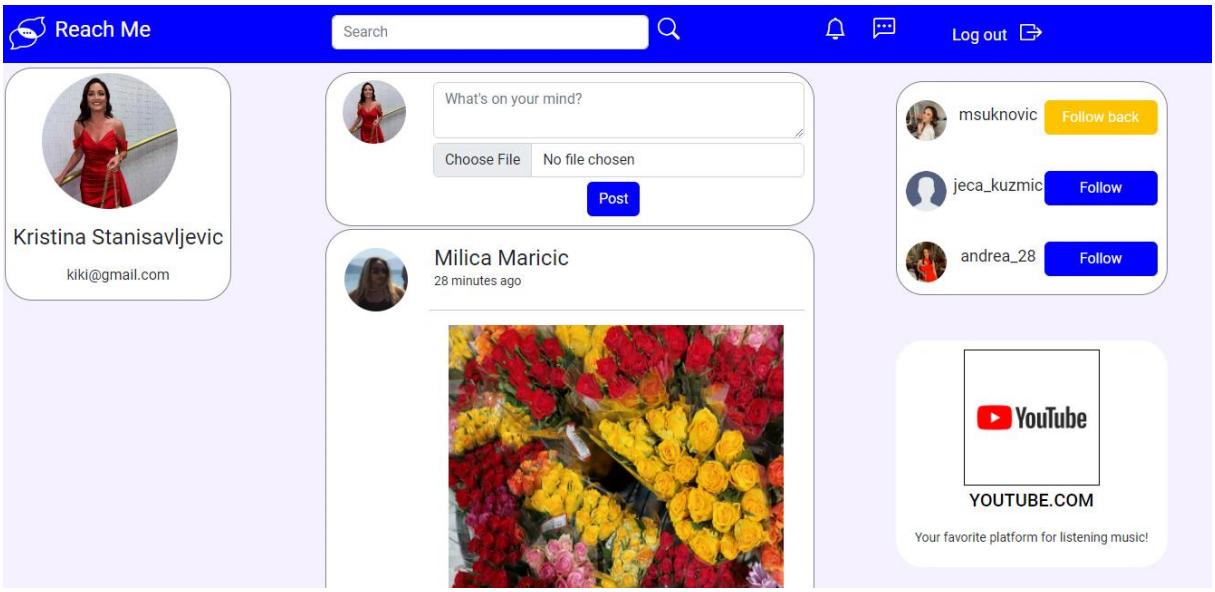
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

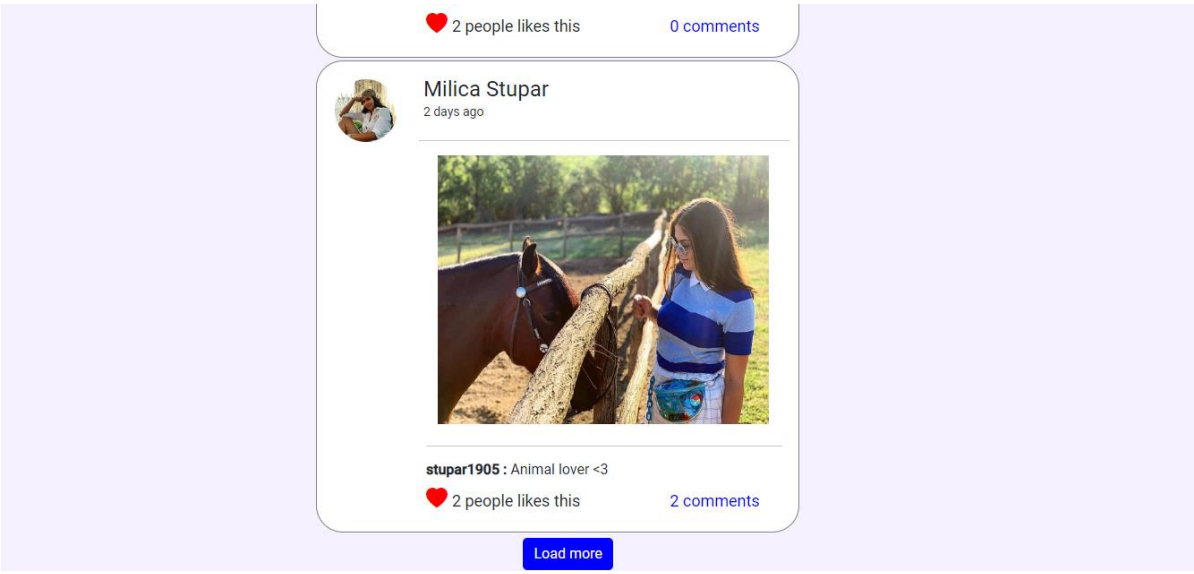
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих објава особа које корисник прати.



Слика 142. Почетна страна

Основни сценарио СК

1. Корисник **бира** објаву коју жели да коментарише. (АПУСО)



Слика 143. Изабрана објава

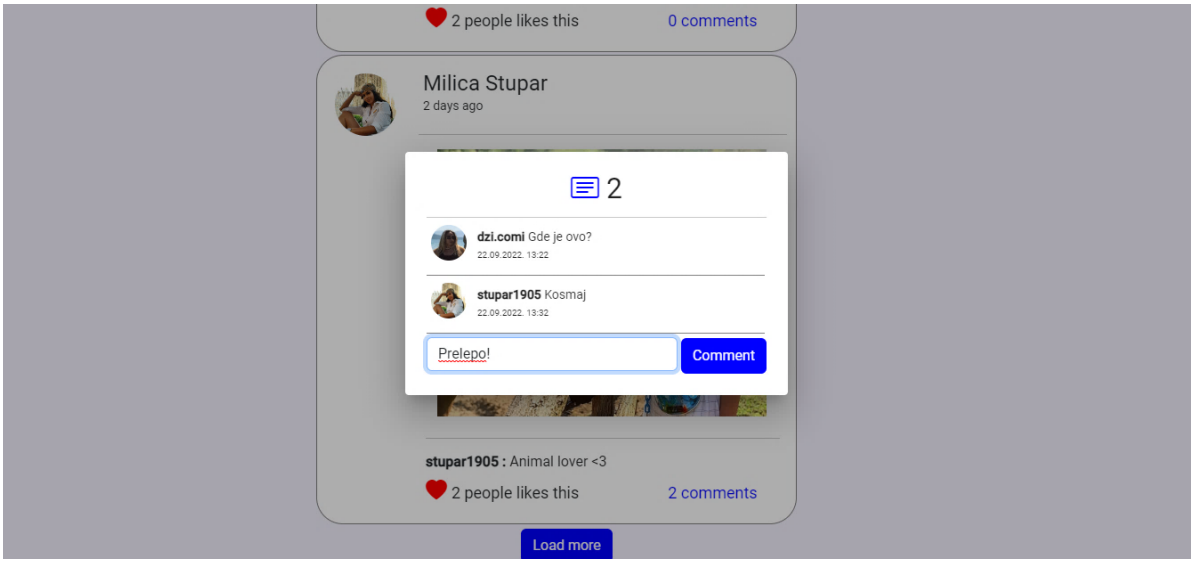
2. Корисник **позива** систем да учита све коментаре за изабрану објаву. (АПСО)

Опис акције: Кликом на дугме „Comments“ корисник позива системску операцију GetComments(postId).

3. Систем **учитава** коментаре за одабрану објаву. (СО)

4. Систем **приказује** кориснику све коментаре за изабрану објаву. (ИА)

5. Корисник **уноси** нови коментар за изабрану објаву. (АПУСО)



Слика 144. Унет коментар

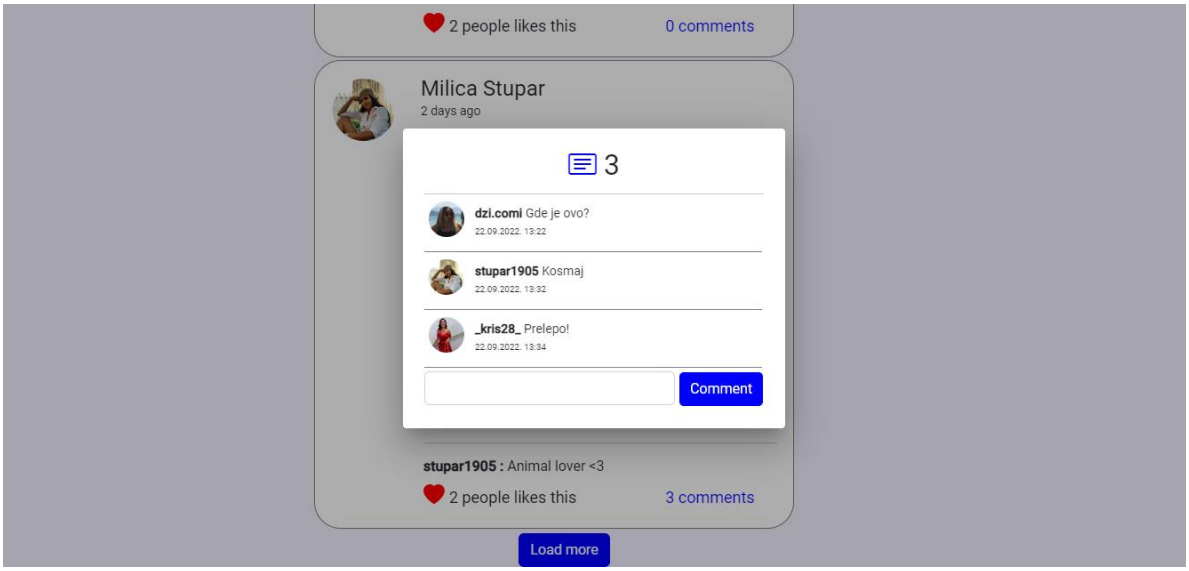
6. Корисник **контролише** да ли је коректно унео нови коментар. (АНСО)

7. Корисник **позива** систем да запамти коментар за изабрану објаву. (АПСО)

Опис акције: Кликом на дугме „Comment“ корисник позива системску операцију PostComment(CommentRequest).

8. Систем **памти** коментар за објаву. (СО)

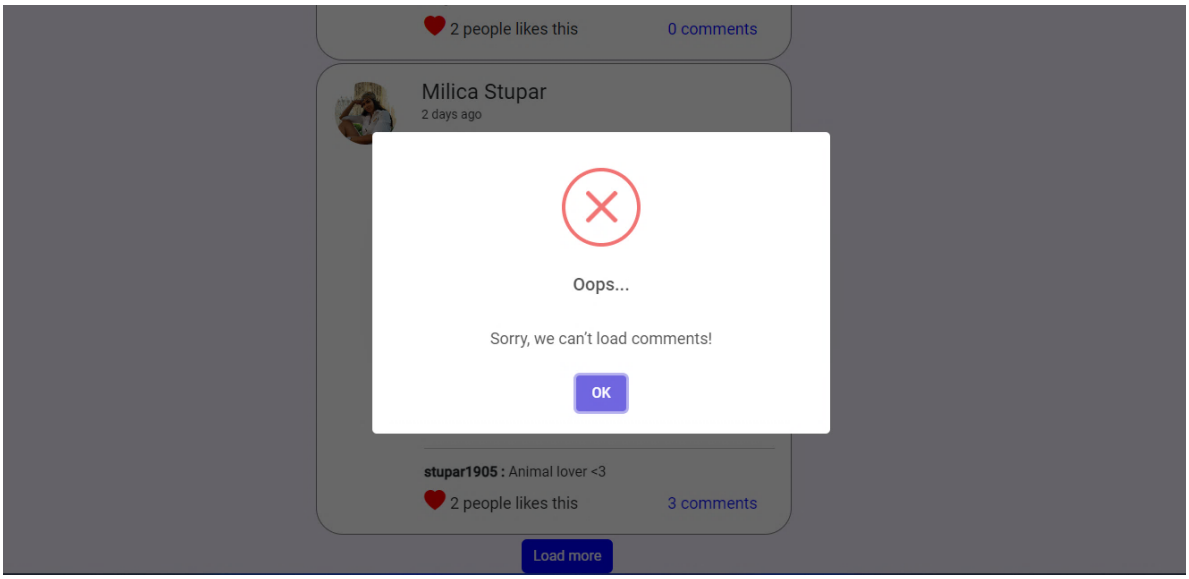
9. Систем **приказује** кориснику запамћен коментар. (ИА)



Слика 145. Коментар је запамћен

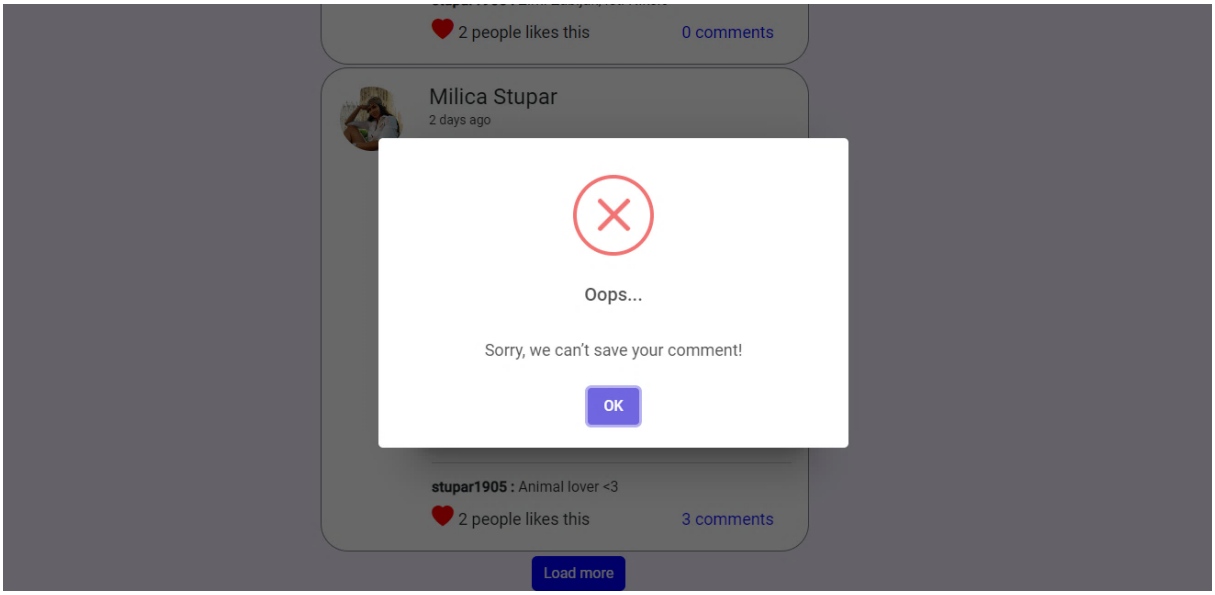
Алтернативна сценарија

4.1. Уколико систем не може да прикаже све коментаре за одабрану објаву, он **приказује** кориснику поруку: “Sorry, we can’t load comments!”. (ИА)



Слика 145. Систем не може да учита коментаре

9.1. Уколико систем не може да запамти коментар, он **приказује** кориснику поруку: “Sorry, we can’t save your comment!”. (ИА)



Слика 146. Систем не може да запамти коментар

СК6: Случај коришћења – Претраживање корисника

Назив СК

Претраживање корисника

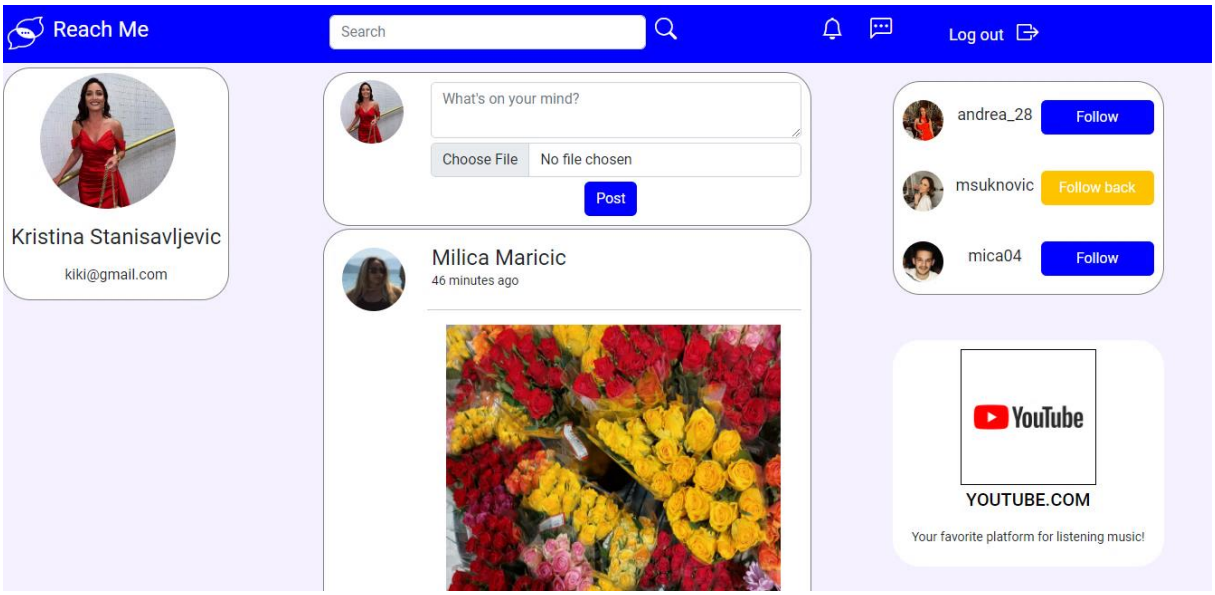
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.



Слика 147. Почетна страна на којој се налази форма за претрагу

Основни сценарио СК

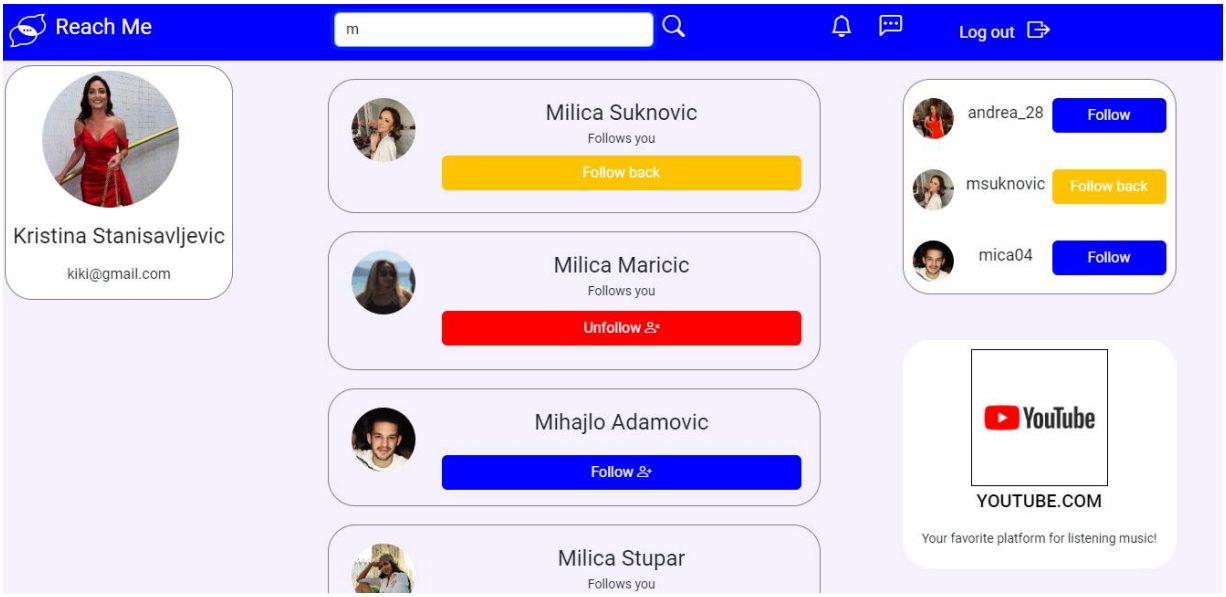
1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију Search(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)



Слика 149. Пронађени корисници

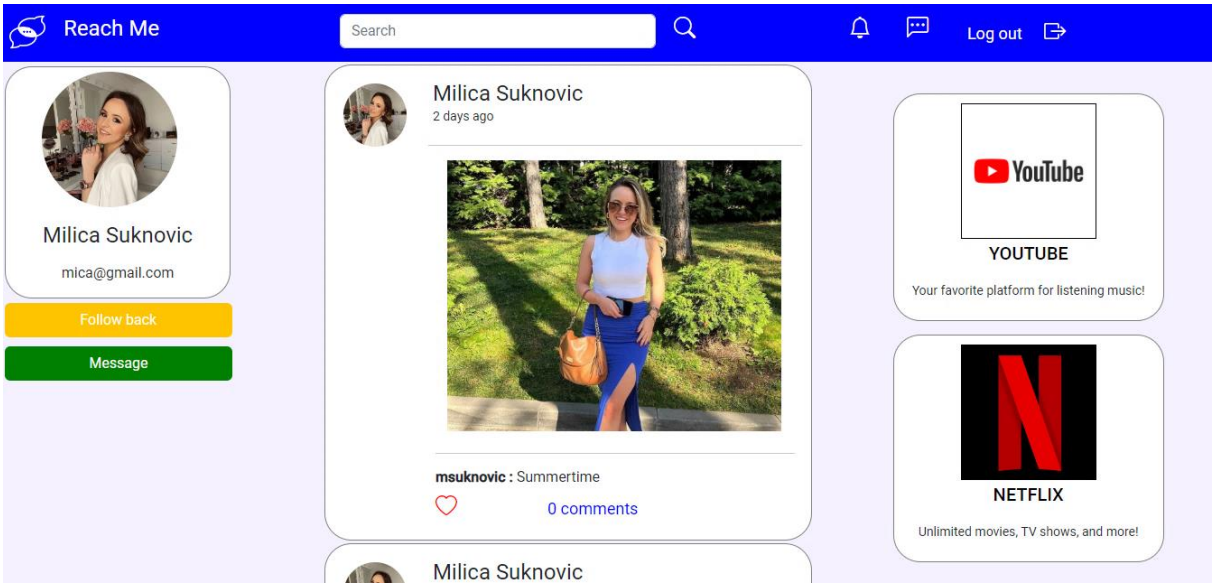
5. Корисник **бира** корисника. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију UcitajUsera(userId, username).

7. Систем **учитава** профил изабраног корисника. (СО)

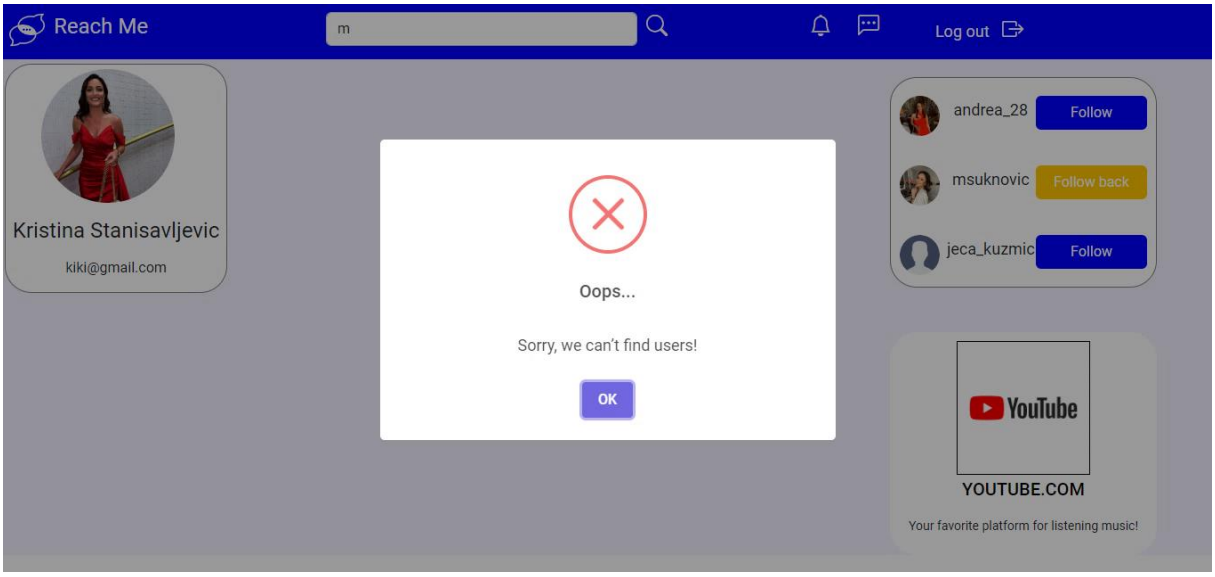
8. Систем **приказује** кориснику профил изабраног корисника. (ИА)



Слика 150. Профил траженог корисника

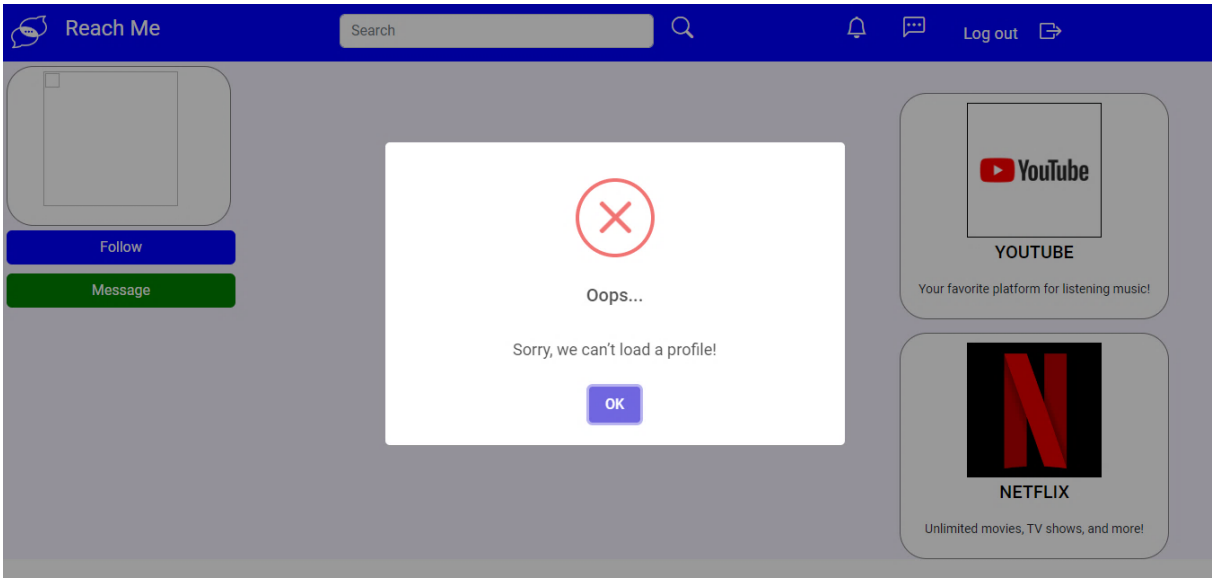
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



Слика 151. Систем не може да пронађе кориснике

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)



Слика 152. Систем не може да учита профил траженог корисника

СК7: Случај коришћења – Слање захтева за праћење

Назив СК

Слање захтева за праћење

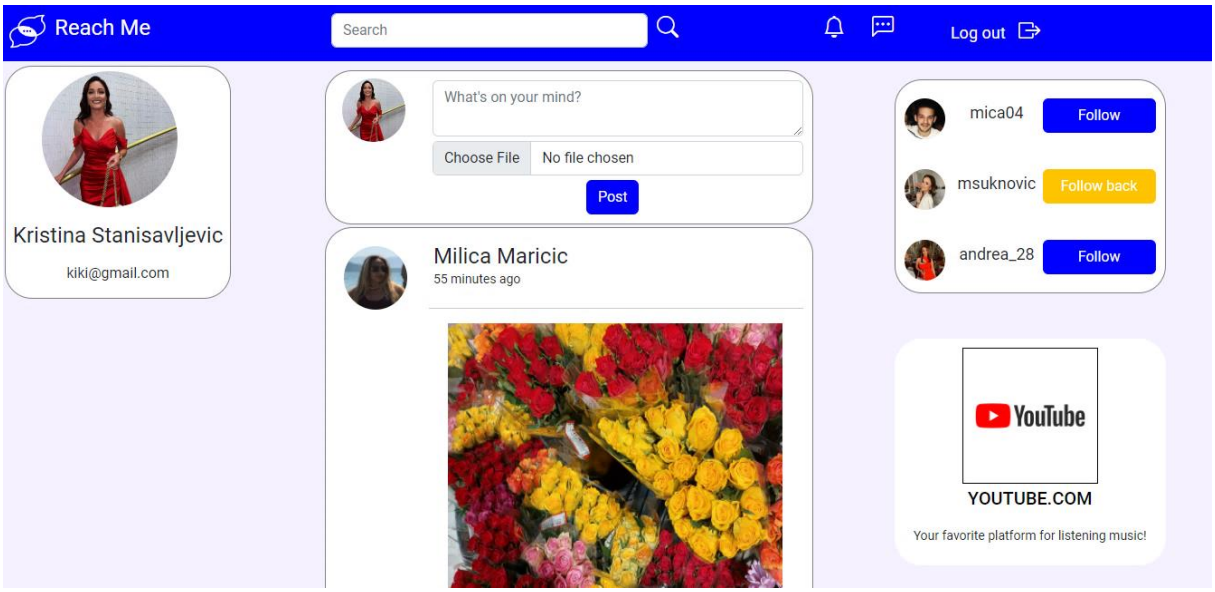
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.



Слика 153. Почетна страница на којој се налази форма за претрагу

Основни сценарио СК

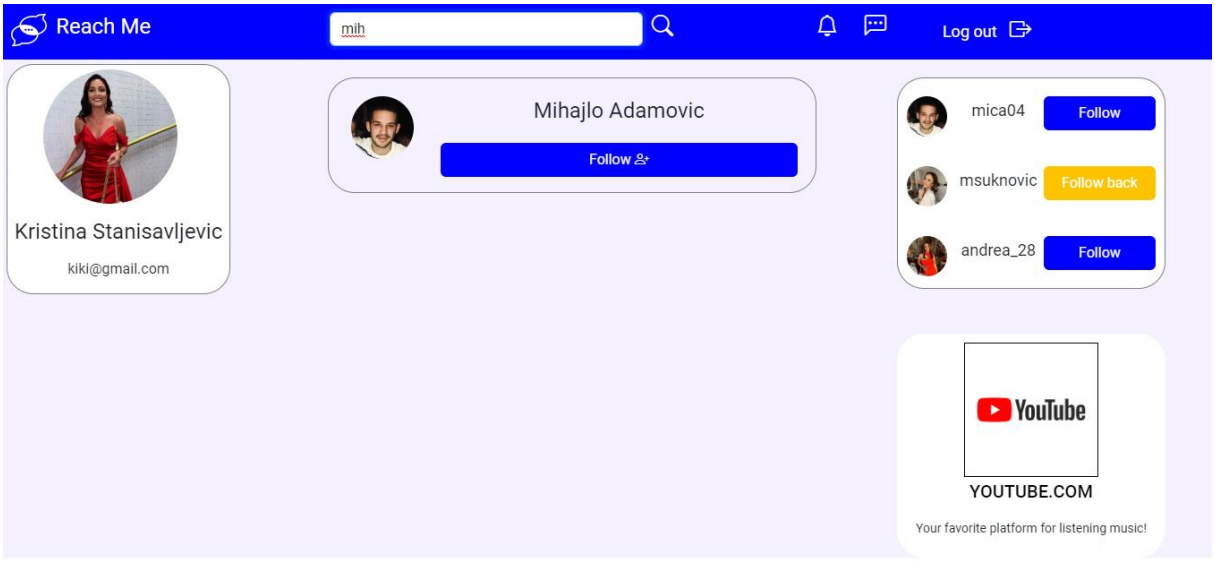
1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију Search(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)



Слика 154. Пронађени корисници

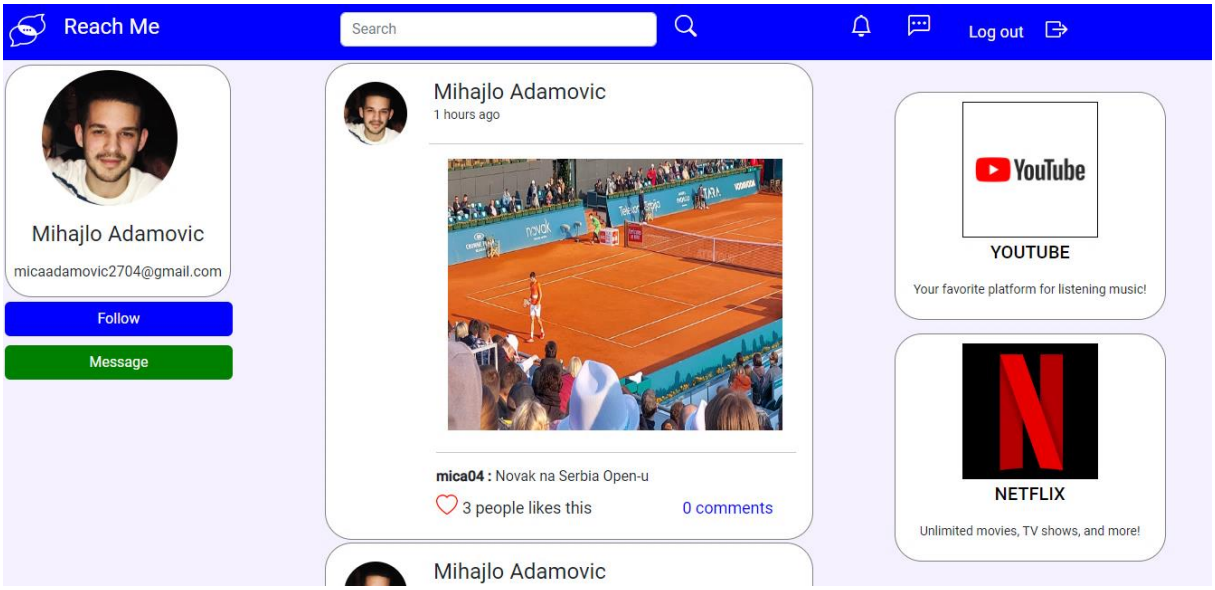
5. Корисник **бира** корисника којем жели да пошаље захтев. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију UcitajUsera(userId, username).

7. Систем **учитава** профил изабраног корисника. (СО)

8. Систем **приказује** кориснику профил изабраног корисника. (ИА)



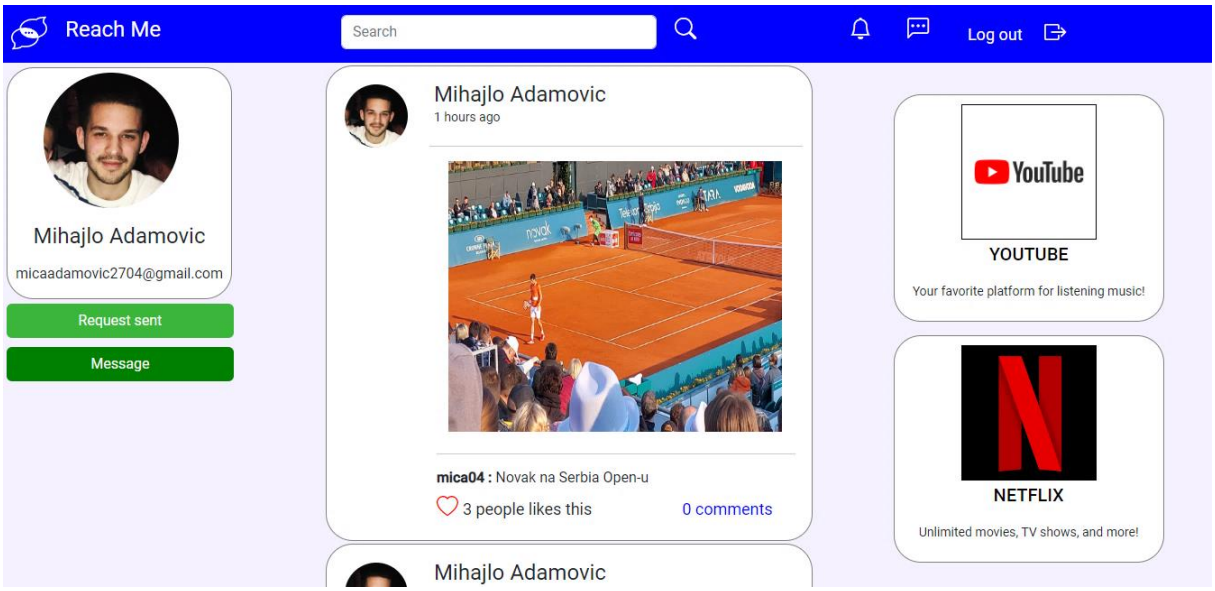
Слика 155. Профил траженог корисника

9. Корисник **позива** систем да пошаље захтев изабраном кориснику. (АПСО)

Опис акције: Кликом на дугме „Follow“ корисник позива системску операцију SendRequest(userId, followId).

10. Систем **памти** захтев за праћење. (СО)

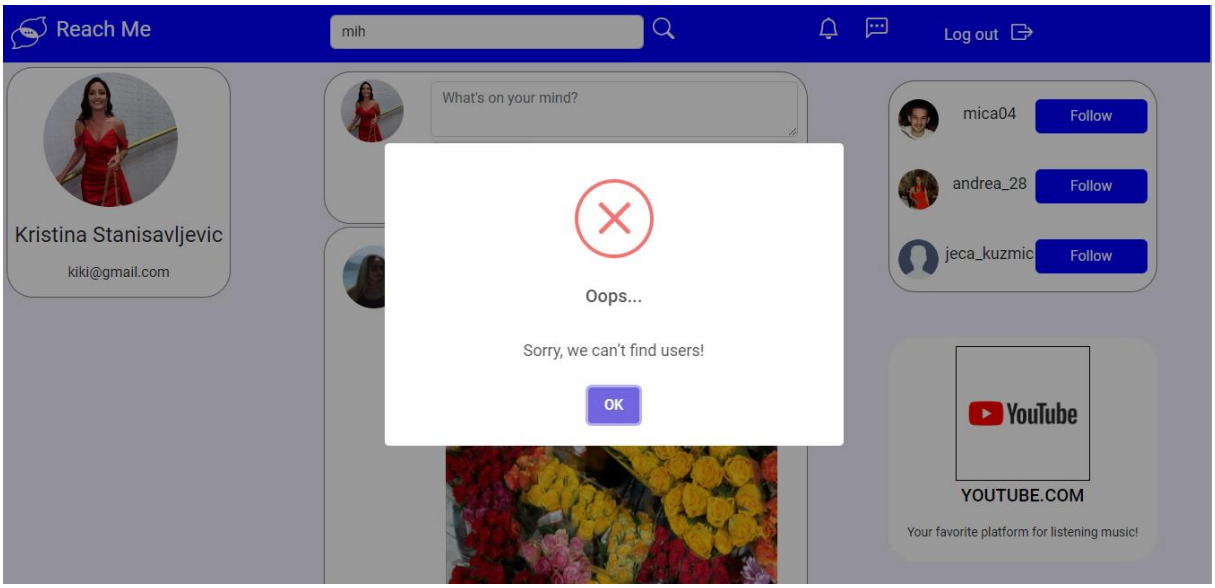
11. Систем **приказује** кориснику да је захтев успешно послат. (ИА)



Слика 156. Захтев за праћење је послат

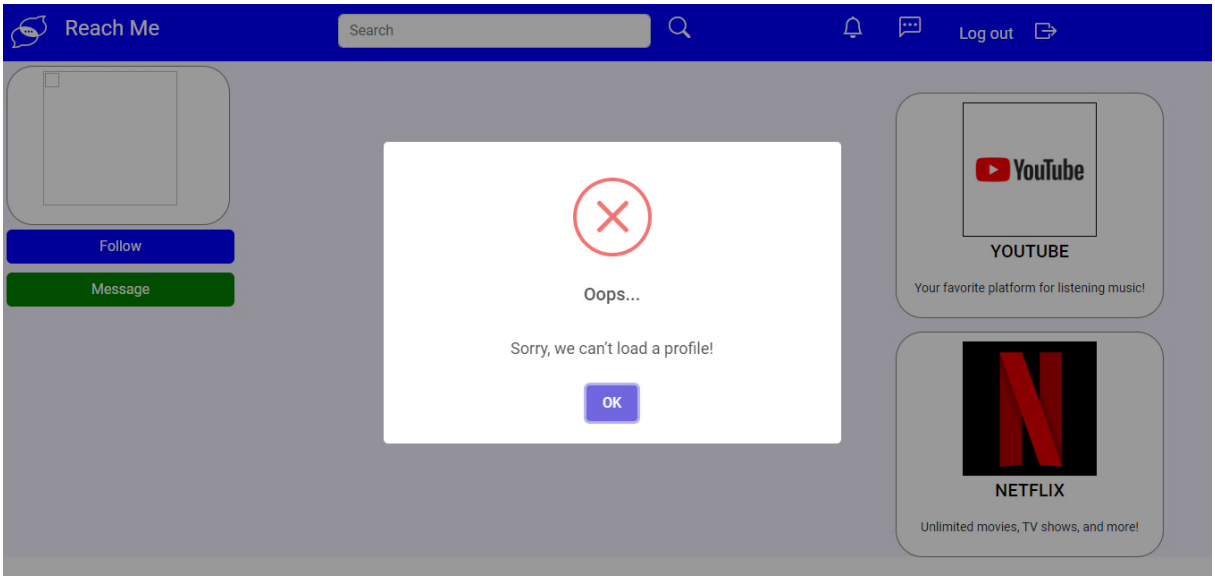
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



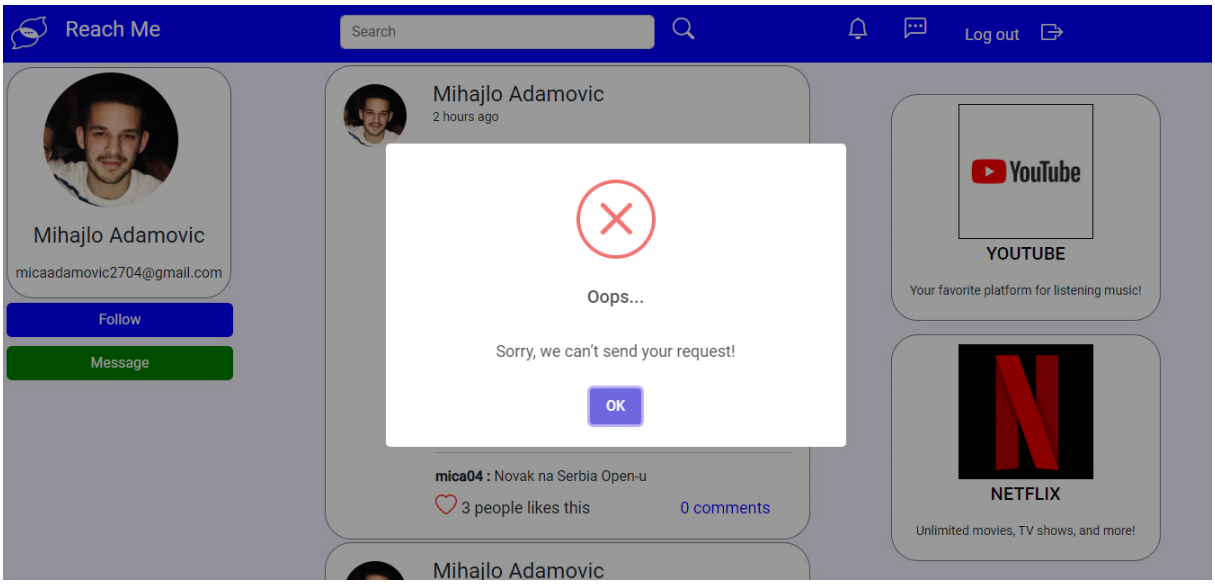
Слика 157. Систем не може да пронађе кориснике

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”. (ИА)



Слика 158. Систем не може да учита профил траженог корисника

10.1. Уколико систем не може да запамти захтев за праћење, он **приказује** кориснику поруку: “Sorry, we can’t send your request!”. (ИА)



Слика 159. Систем не може да пошаље захтев кориснику

СК8: Случај коришћења – Отпраћивање

Назив СК

Отпраћивање

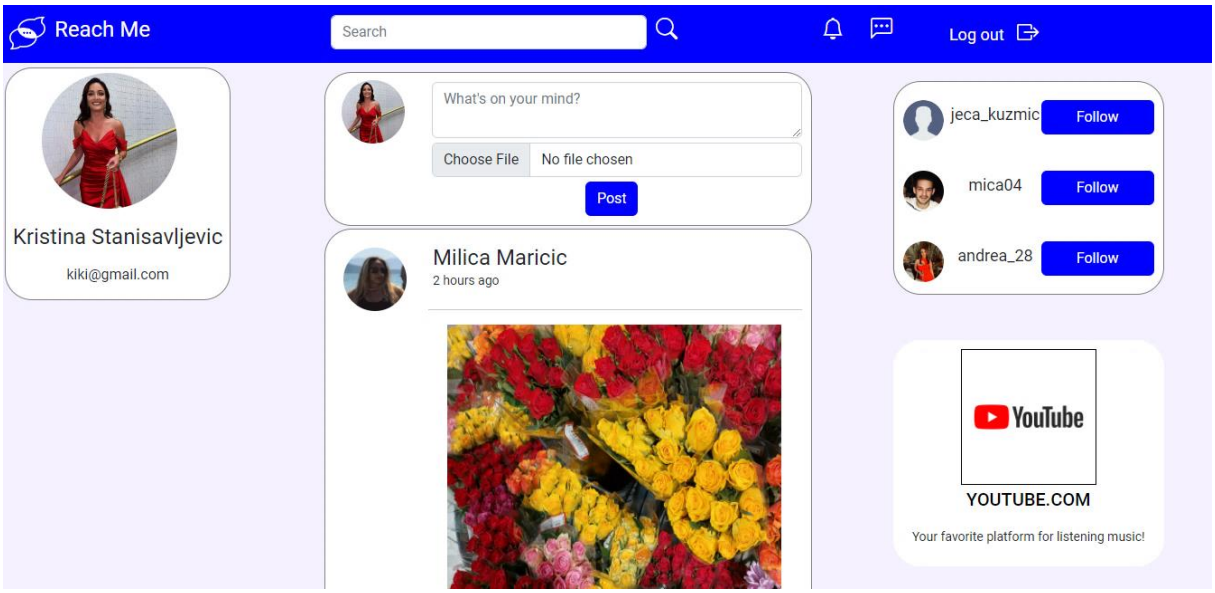
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника на којој се налази форма за претрагу.



Слика 160. Почетна страна на којој се налази форма за претрагу

Основни сценарио СК

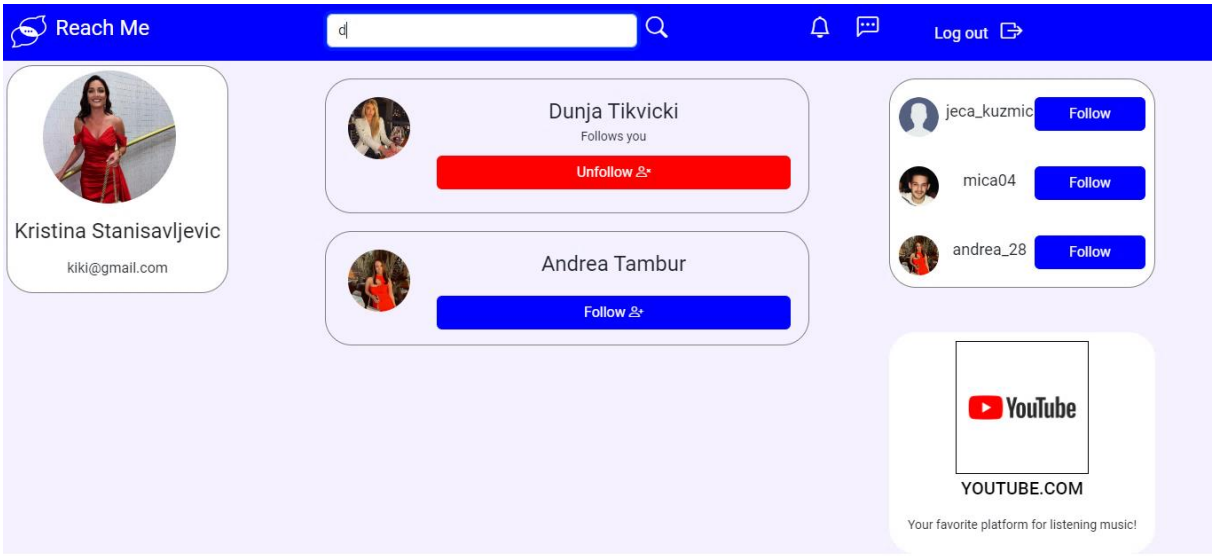
1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију Search(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)



Слика 161. Пронађени корисници

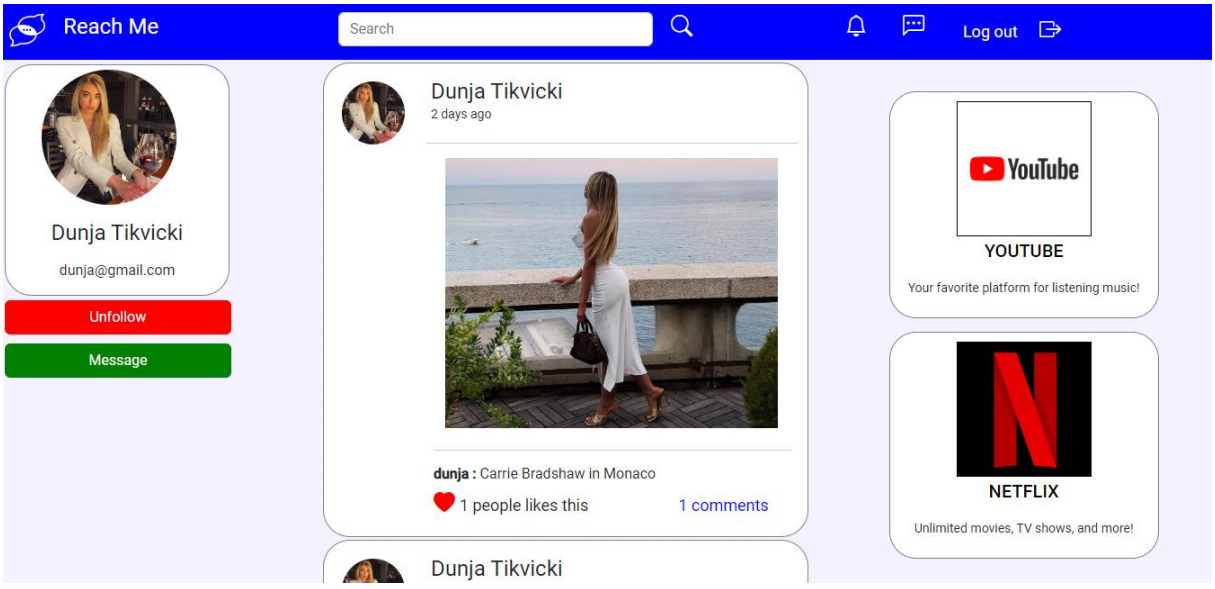
5. Корисник **бира** корисника којег жели да отпрати. (АПУСО)

6. Корисник **позива** систем да учита профил изабраног корисника. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију UcitajUsera(userId, username).

7. Систем **учитава** профил изабраног корисника. (СО)

8. Систем **приказује** кориснику профил изабраног корисника. (ИА)



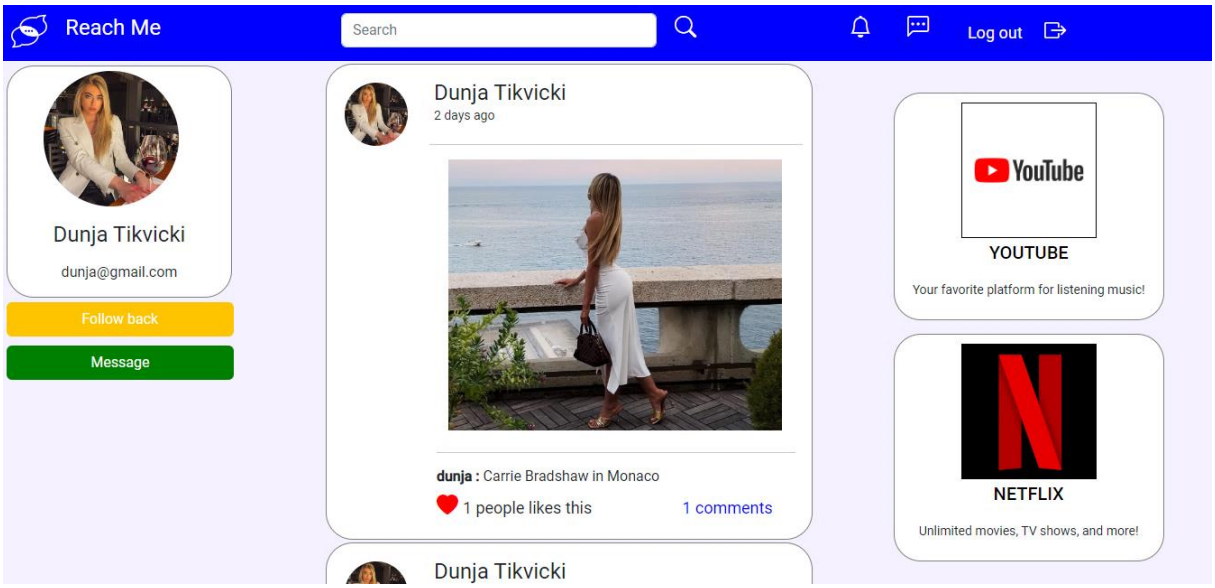
Слика 162. Профил траженог корисника

9. Корисник **позива** систем да отпрати изабраног корисника. (АПСО)

Опис акције: Кликом на дугме „Unfollow“ корисник позива системску операцију Unfollow(username, userId).

10. Систем **отпрађује** изабраног корисника. (СО)

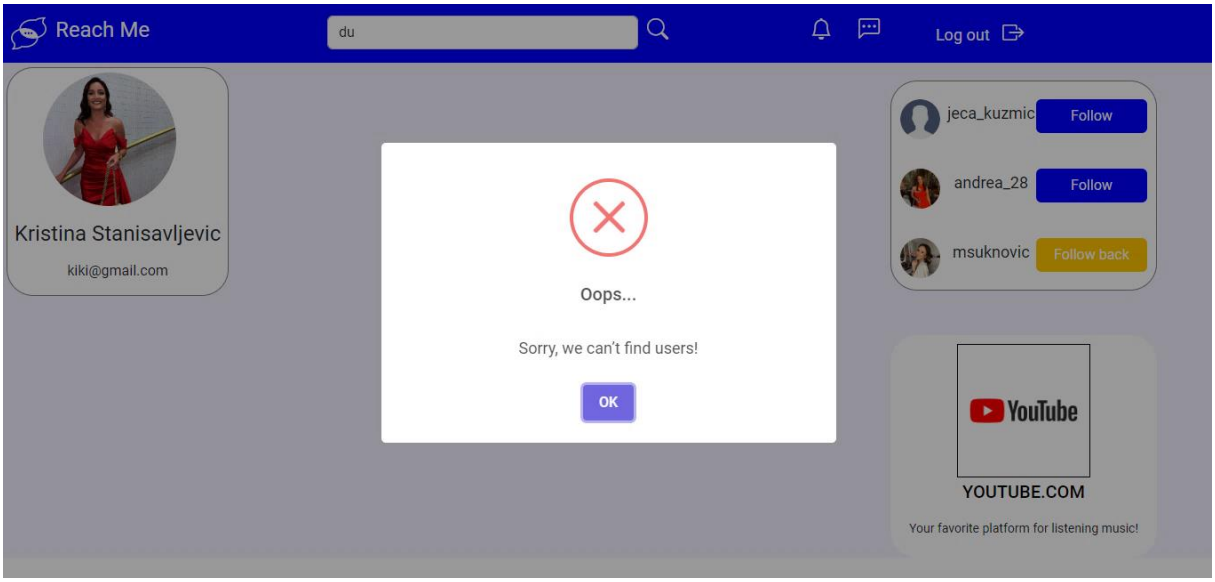
11. Систем **приказује** кориснику да је изабрани корисник успешно отпраћен. (ИА)



Слика 163. Корисник је отпраћен

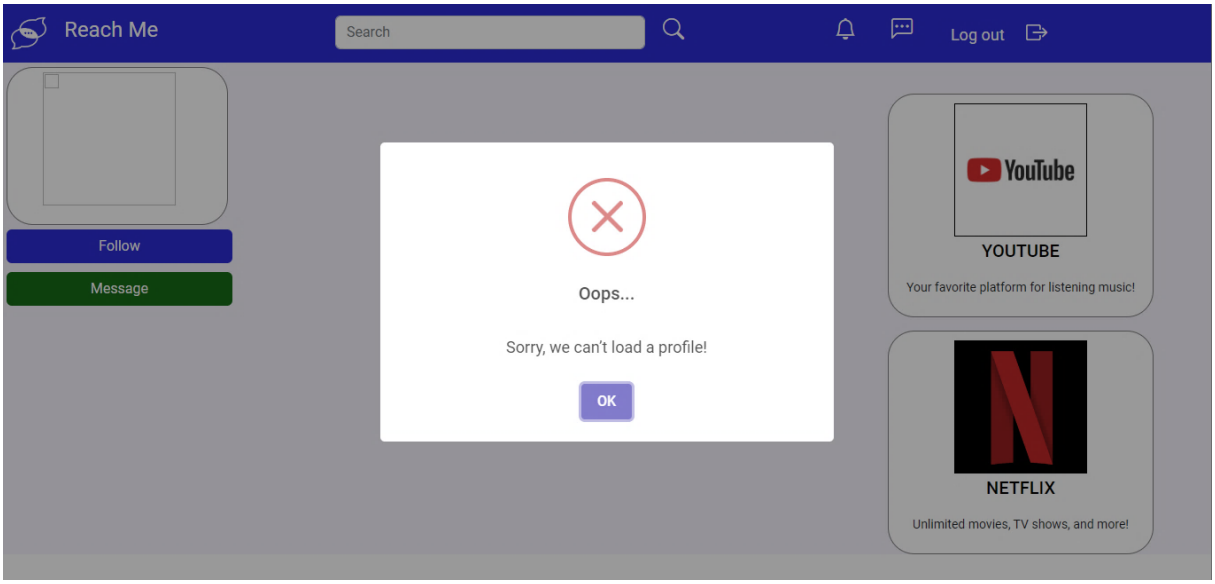
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



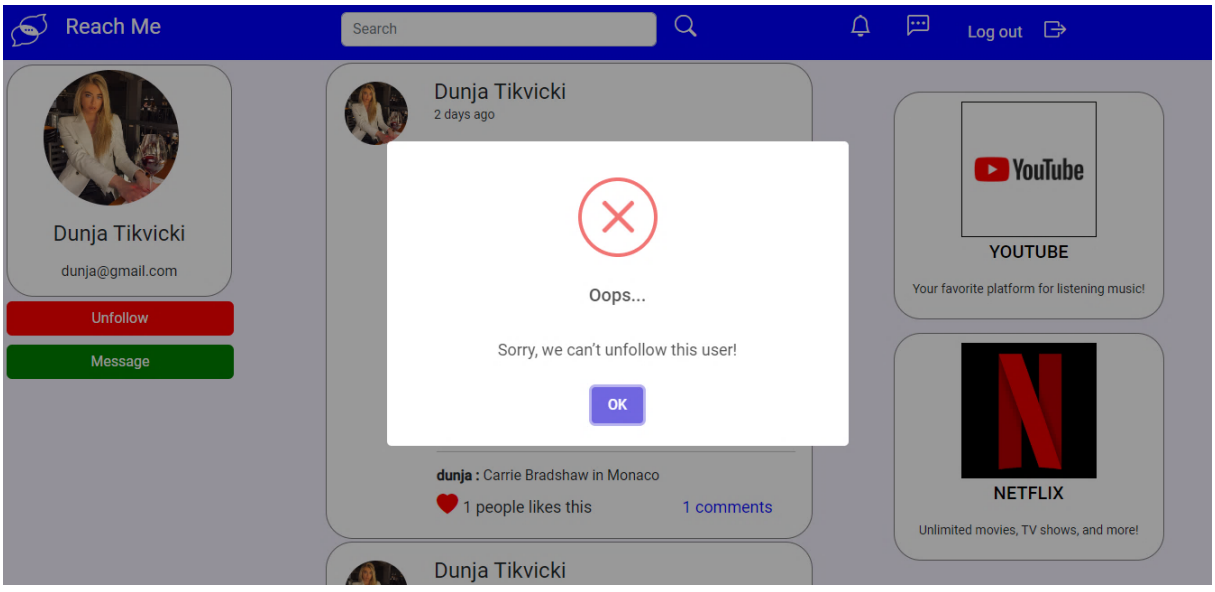
Слика 164. Систем не може да пронађе кориснике

8.1. Уколико систем не може да приказе кориснику профил изабраног корисника, он **приказује** кориснику поруку: “Sorry, we can’t load a profile!”(ИА)



Слика 165. Систем не може да учита профил траженог корисника

10.1. Уколико систем не може да отпрати корисника, он **приказује** кориснику поруку: “Sorry, we can’t unfollow this user!”. (ИА)



Слика 166. Систем не може да отпрати корисника

СК9: Случај коришћења – Измена профила

Назив СК

Измена профила

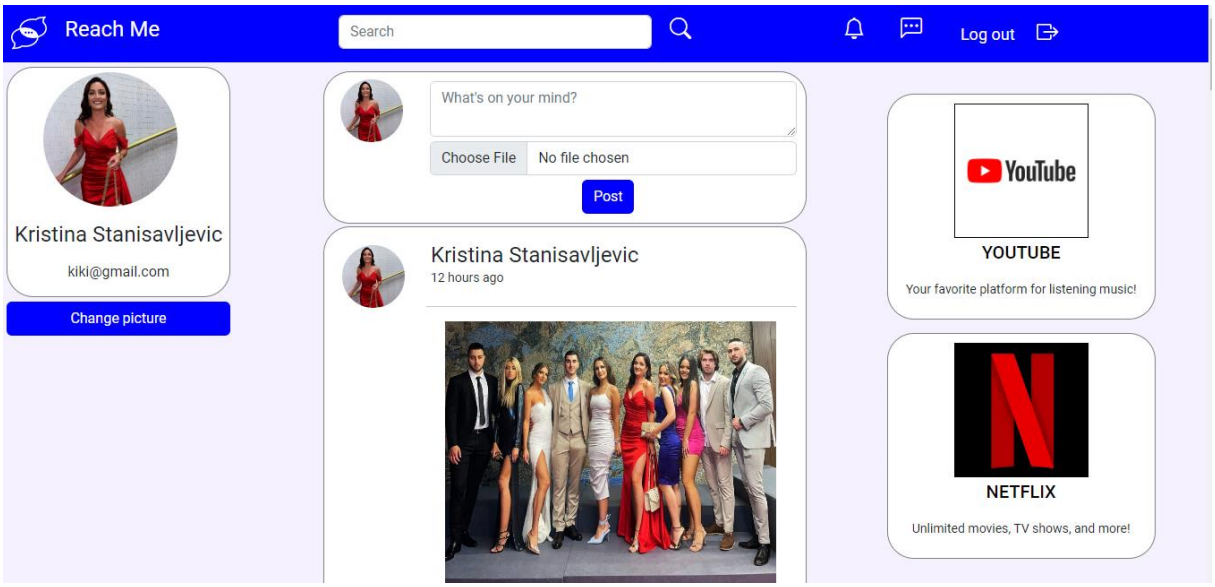
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

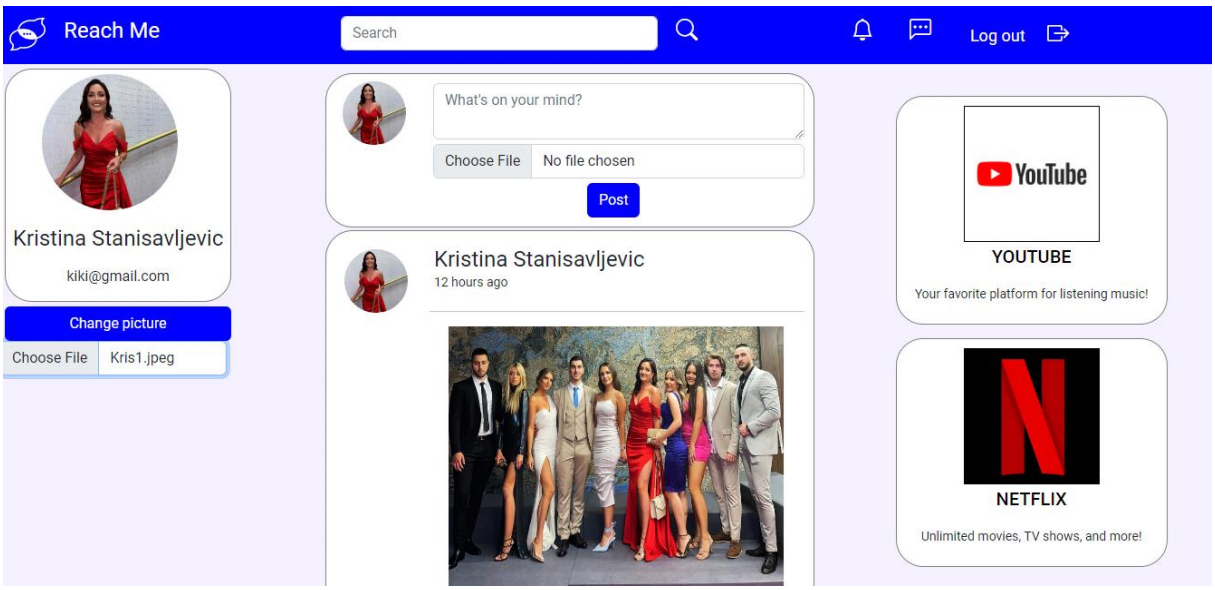
Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује профил улогованог корисника. Учитани су подаци о улогованом кориснику као и листа свих објава улогованог корисника.



Слика 167. Профил тренутно улогованог корисника

Основни сценарио СК

1. Корисник **уноси (мења)** своју профилну слику. (АПУСО)



Слика 168. Унета нова профилна слика

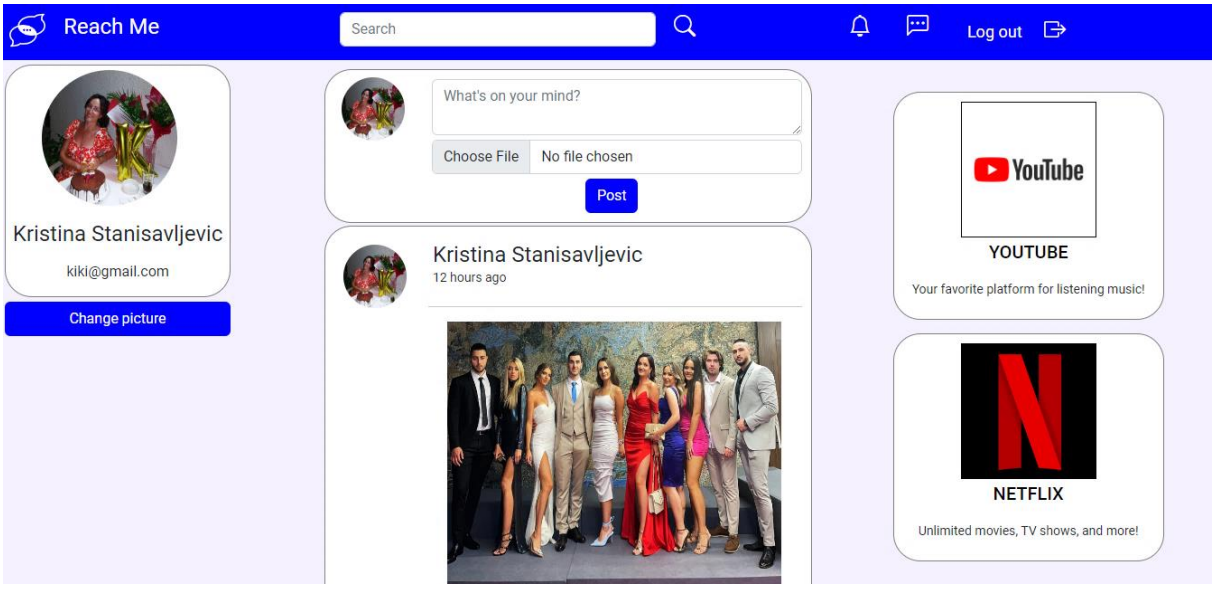
2. Корисник **контролише** да ли је коректно унео нову профилну слику. (АНСО)

3. Корисник **позива** систем да запамти податке о кориснику. (АПСО)

Опис акције: Кликом на дугме „Change picture“ корисник позива системску операцију ChangePicture(UserDto).

4. Систем **памти** податке о кориснику. (СО)

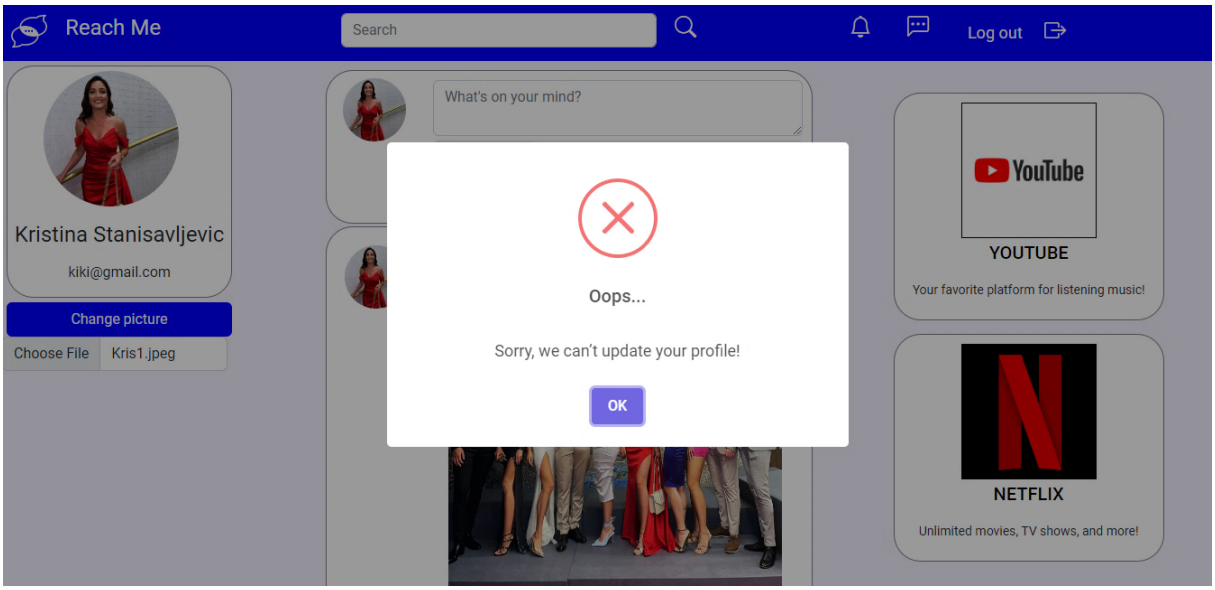
5. Систем **приказује** кориснику да је запамтио нове податке. (ИА)



Слика 169. Успешно промењена профилна слика

Алтернативна сценарија

5.1. Уколико систем не може да запамти податке о кориснику, он **приказује** кориснику поруку: “Sorry, we can’t update your profile!”. (ИА)



Слика 170. Систем не може да измени профилну слику

СК10: Случај коришћења – Слање поруке

Назив СК

Слање поруке

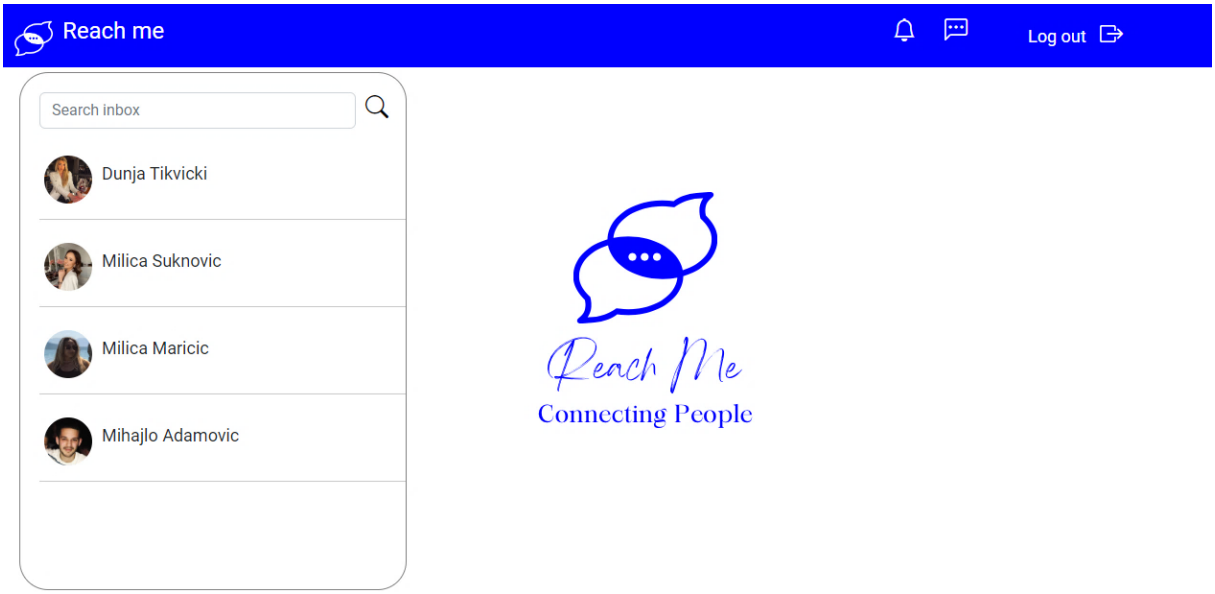
Актори СК

Корисник

Учесници СК

Корисник и систем (програм)

Предуслов: Систем је укључен и корисник је пријављен на систем под својом шифром. Систем приказује почетну страницу за корисника. Учитана је листа свих корисника са којима улоговани корисник има поруке.



Слика 171. Инбох тренутно улогованог корисника

Основни сценарио СК

1. Корисник **уноси** вредност по којој претражује кориснике. (АПУСО)

2. Корисник **позива** систем да нађе кориснике по задатој вредности. (АПСО)

Опис акције: Корисник притиском типке тастатуре позива системску операцију FindInboxUsers(kriterijum, userId).

3. Систем **тражи** кориснике по задатој вредности. (СО)

4. Систем **приказује** кориснику пронађене кориснике. (ИА)

Слика 172. Пронађени инбох корисници

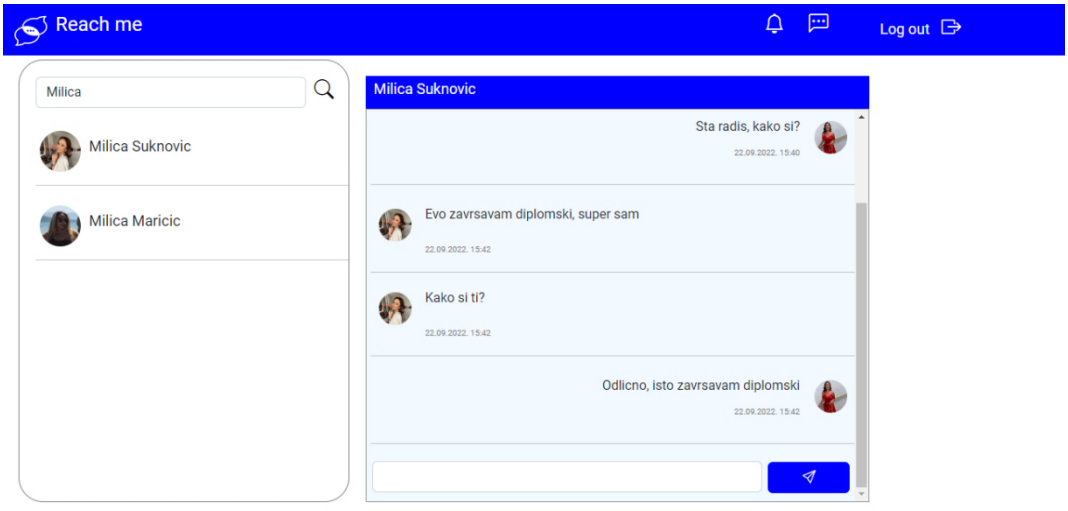
5. Корисник **бира** корисника којем жели да пошаље поруку. (АПУСО)

6. Корисник **позива** систем да учита chat са изабраним корисником. (АПСО)

Опис акције: Кликом на име и презиме корисник позива системску операцију GetChat(fromId, forId).

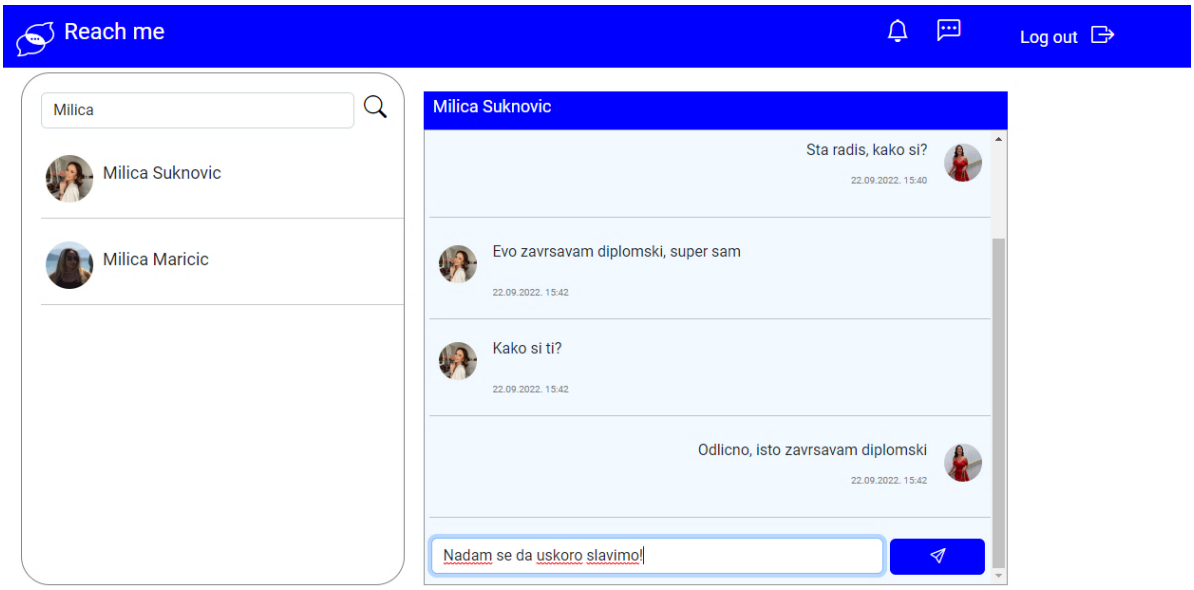
7. Систем **учитава** листу свих порука са изабраним корисником. (СО)

8. Систем **приказује** кориснику поруке. (ИА)



Слика 173. Све поруке са изабраним корисником

9. Корисник **уноси** нову поруку за изабраног корисника. (АПУСО)



Слика 174. Унета порука за слање кориснику

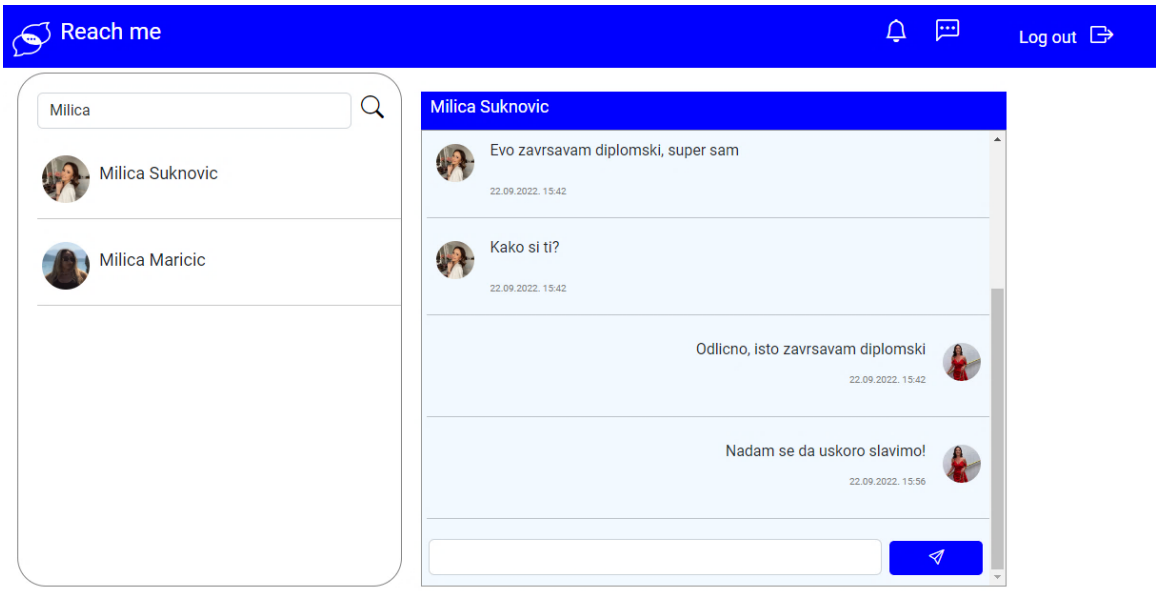
10. Корисник **контролише** да ли је коректно унео нову поруку. (АНСО)

11. Корисник **позива** систем да пошаље поруку. (АПСО)

Опис акције: Кликом на дугме корисник позива системску операцију SendMessage(MessageDto).

12. Систем **памти** поруку за изабраног корисника. (СО)

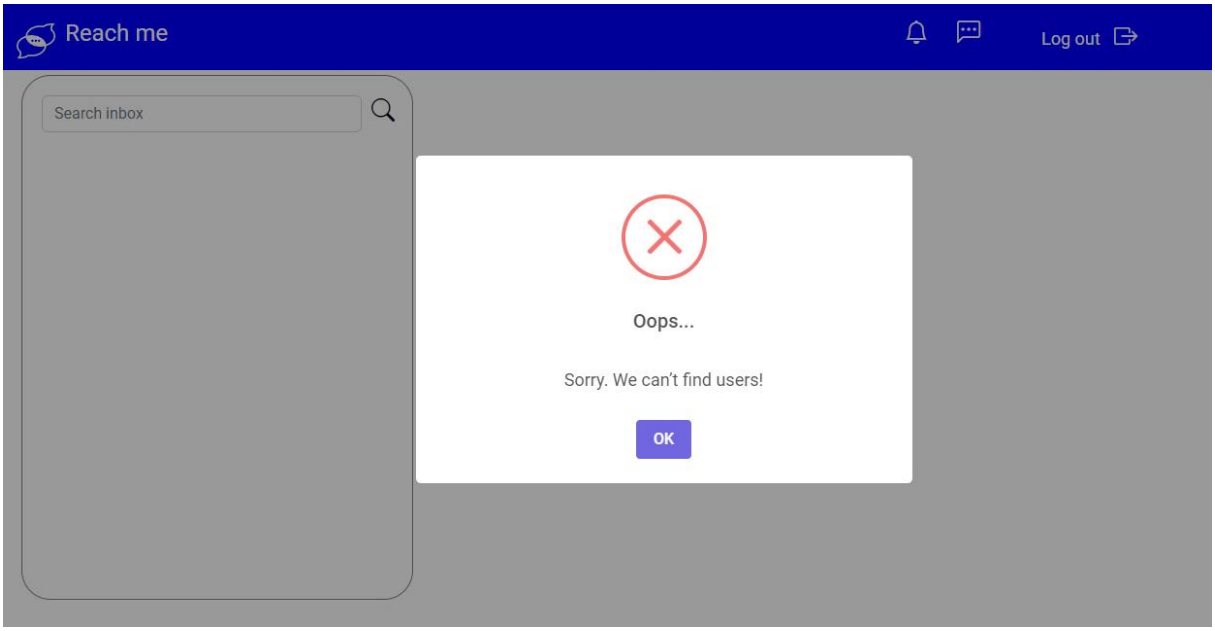
13. Систем **приказује** кориснику послату поруку. (ИА)



Слика 175. Успешно послата порука

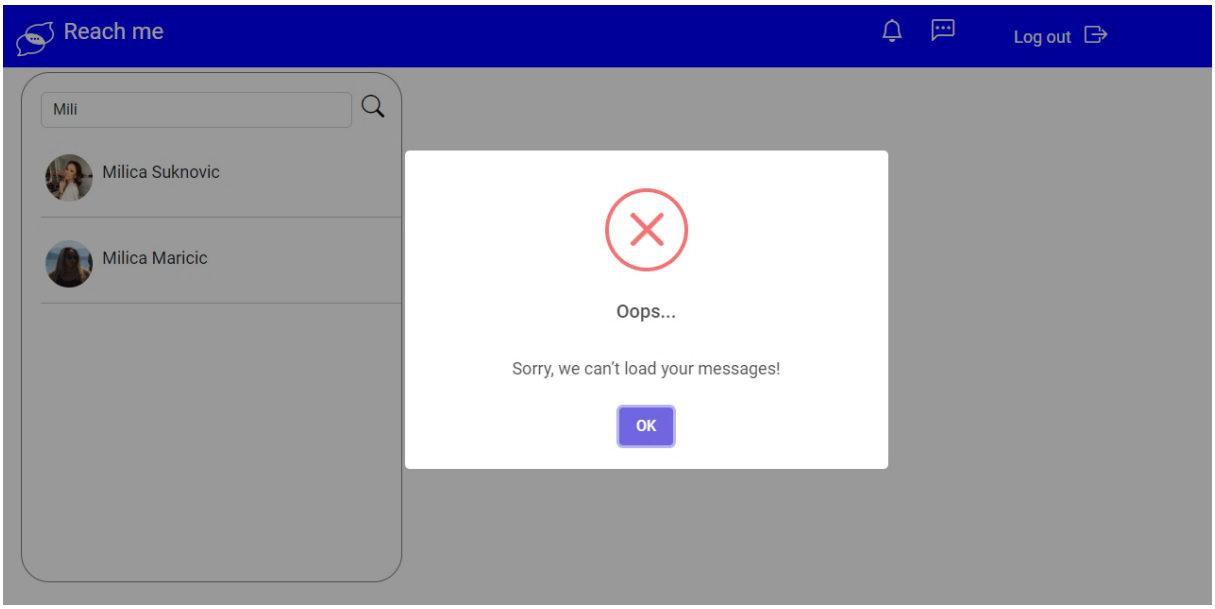
Алтернативна сценарија

4.1. Уколико систем не може да пронађе кориснике, он **приказује** кориснику поруку: “Sorry, we can’t find users!”. (ИА)



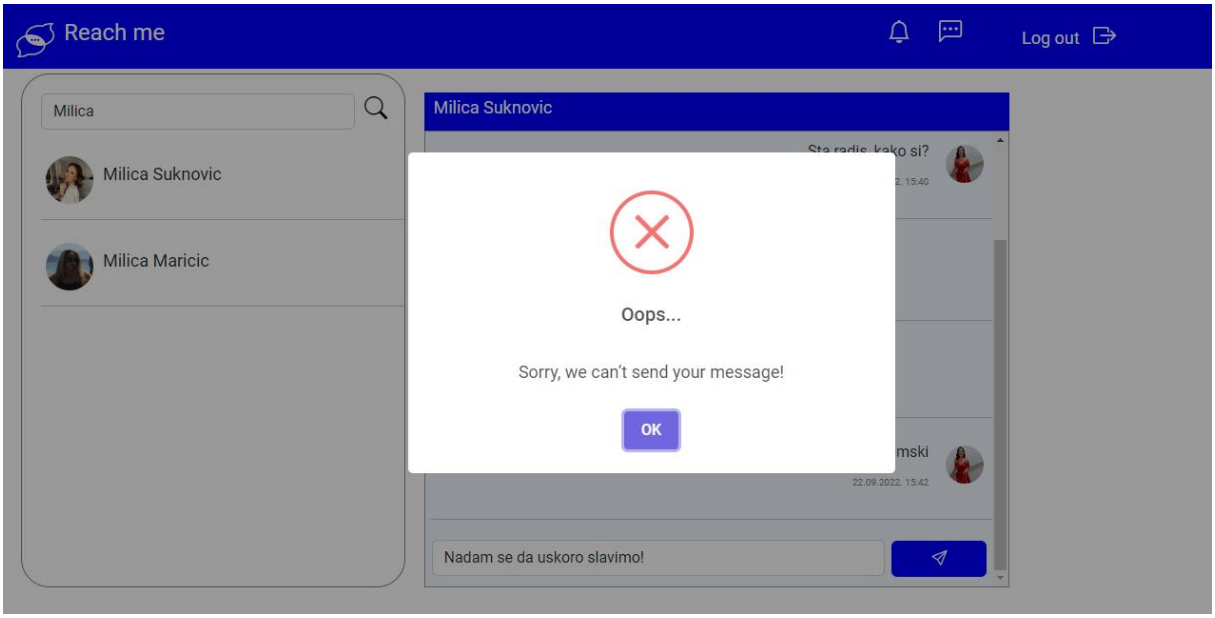
Слика 176. Систем не може да пронађе инбоџ кориснике

8.1. Уколико систем не може да приказе кориснику поруке са изабраним корисником, он **приказује** кориснику поруку: “Sorry, we can’t load your messages!”. (ИА)



Слика 177. Систем не може да учита поруке са изабраним корисником

13.1. Уколико систем не може да запамти поруку, он **приказује** кориснику поруку: “Sorry, we can’t send your message!”. (ИА)



Слика 178. Систем не може да пошаље поруку кориснику

11.3. Имплементација пословне логике

11.3.1. Комуникација са клијентом

Полазну тачку у комуникацији за клијентом представља Program.cs класа. Метода CreateBuilder() је одговорна за конфигурацију ASP.Net-а.

Комуникација са клијентом се врши путем HTTP захтева. Контролери су задужени за прихватање тих захтева и прослеђивање захтева до одговарајућих сервис класа у којима су имплементиране системске операције. Помоћу инстанце UnitOfWork се приступа складишту података и извршавају одређене операције над подацима. Контролери такође и шаљу обрађене захтеве назад клијенту.

Потребно је и да се контролери, сервиси као и класе које омогућавају приступ подацима регистују и да се омогуће потребни middleware-и (посредници).

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
builder.Services.AddScoped<JwtAuthentification>();

builder.Services.AddScoped<IUserService, UserService>();
builder.Services.AddScoped<IPostService, PostService>();
builder.Services.AddScoped<IFollowNotificationService, FollowNotificationService>();
builder.Services.AddScoped<ILikeNotificationService, LikeNotificationService>();
builder.Services.AddScoped<ICommentNotificationService, CommentNotificationService>();
builder.Services.AddScoped<IMessageService, MessageService>();
builder.Services.AddScoped<IReactionService, ReactionService>();
builder.Services.AddScoped<ICommentService, CommentService>();
builder.Services.AddScoped<IAuthentificationService, AuthentificationService>();
builder.Services.AddScoped<JwtAuthentification>();

builder.Services.AddScoped<IUnitOfWork, UnitOfWork>();
builder.Services.AddDbContext<UserContext>(options=>
{
    options.UseSqlServer(builder.Configuration.GetConnectionString("baza"));
});
builder.Services.AddIdentity<User, IdentityRole<int>>().AddEntityFrameworkStores<UserContext>();
builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSignalR();

builder.Services.AddAuthentication(options=> {
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    })
    .AddJwtBearer(options =>
    {
```

Слика 179. Регистрација потребних контролера, сервиса и класа

11.3.2. Имплементација пословне логике

Доменске класе направљене на основу концептуалног модела налазе се у пројекту Domain. Постоји десет доменских класа, у наставку ће бити приказа једна као пример.

```
namespace Domain
{
    public class User : IdentityUser<int>, MyEntity
    {
        [Required]
        public String FirstName { get; set; }
        [Required]
        public String LastName { get; set; }
        [Required]
        [EmailAddress]
        public String Email { get; set; }
        public String ProfilePicture { get; set; }
        public List<Post> Posts { get; set; }
        public List<User> Followers { get; set; }
        public List<User> Following { get; set; }
        [JsonIgnore]
        public List<Reaction> Reactions { get; set; }
        [JsonIgnore]
        public List<Message> Received { get; set; }
        [JsonIgnore]
        public List<Message> Send { get; set; }
        [JsonIgnore]
        public List<Comment> Comments { get; set; }
        [JsonIgnore]
        public List<Notification> MyNotifications { get; set; }
        [JsonIgnore]
        public List<Notification> NotificationsFromMe { get; set; }
    }
}
```

Системске операције имплементиране су у оквиру одговарајућих сервис класа унутар пројекта BusinesLogicLayer.

1. **Уговор УГ1:** LogIn

Операција: LogIn(LoginDto, UserDto):signal;

Веза са СК: СК1

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је улогован.

```
public async Task<UserDto> LogIn(LoginDto dto)
{
      return await jwt.AuthentificationAsync(dto.Username, dto.Password);
}
```

```
public async Task<UserDto> AuthentificationAsync(String username, String password)
        {

            var user = await manager.FindByNameAsync(username);

            if(user == null) { return null;  }

            if (await manager.CheckPasswordAsync(user, password))
            {
                List<Claim> claims = new List<Claim>();

                claims.Add(new Claim(ClaimTypes.Name, user.FirstName));
                claims.Add(new Claim(ClaimTypes.NameIdentifier, user.UserName));


                ClaimsIdentity identity = new ClaimsIdentity(claims);
                var signCredentials = new SigningCredentials(new
SymmetricSecurityKey(Encoding.ASCII.GetBytes(KEY)),
SecurityAlgorithms.HmacSha256);

                var tokenDescriptor = new SecurityTokenDescriptor()
                {
                    Expires = DateTime.UtcNow.AddHours(1),
                    Subject = identity,
                    SigningCredentials = signCredentials
                };

                var tokenHandler = new JwtSecurityTokenHandler();
                var token =
tokenHandler.WriteToken(tokenHandler.CreateToken(tokenDescriptor));
                UserDto userNew = new UserDto()
                {
                    FirstName = user.FirstName,
                    LastName = user.LastName,
                    Id = user.Id,
                    Username=user.UserName,
                    Token = token
                };
                return userNew;
            }
            return null;
        }
```

2. **Уговор УГ2:** Register

Операција: Register(RegisterDto):signal;

Веза са СК: СК2

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је регистрован.

```
public async Task<IdentityResult> Register(RegisterDto dto)
        {
            var newUser = new User
            {
                FirstName = dto.FirstName,
                LastName = dto.LastName,
                Email = dto.Email,
                UserName = dto.Username,
                ProfilePicture = dto.ProfilePicture
            };

            return await manager.CreateAsync(newUser, dto.Password);
        }
```

3. **Уговор УГ3:** Create

Операција: Create(PostRequest):signal;

Веза са СК: СК3

Предуслови: Вредносна и структурна ограничења над објектом **Post** морају бити задовољена.

Постуслови: Објава је запамћена.

```
public bool Create(PostRequest entity)
        {
            User u = new User() { Id = entity.UserId };
            u = unit.UserRepository.SearchById(u);
            if (u != null)
            {
                try
                {
                    Post p = requestMapper.toEntity(entity);
                    DateTime now = DateTime.Now;
                    p.Date = now;
                    unit.PostRepository.Add(p);
                    unit.Save();
                    return true;
                }
                catch (Exception)
                {
                    return false;
                }
            };
            return false;
```

4. **Уговор УГ4:** GetAllForHome

Операција: GetAllForHome(userId, numOfPosts, List<PostResponse>):signal;

Веза са СК: СК4

Предуслови:

Постуслови:

```
public List<PostResponse> GetAllForHome(int i, int numOfPosts)
        {
            int skip = numOfPosts / 5;
            skip = (skip - 1) * 5;
            List<Post> posts=unit.PostRepository.GetAllForHome(i, skip, 5);
            List<PostResponse> response = new List<PostResponse>();
            foreach(Post p in posts)
            {
                PostResponse pr = responseMapper.toDto(p);
                p.Reactions.ForEach(r =>
                {
                   if (r.UserId == i)
                   {
                       pr.ILiked = true;
                   }
                });
                pr.NumberOfLikes = p.Reactions.Count();
                pr.NumberOfComments = p.Comments.Count();
                response.Add(pr);
            }
            return response;
              }
```

5. **Уговор УГ5:** LikeIt

Операција: LikeIt(postId, username):signal;

Веза са СК: СК4

Предуслови: Вредносна и структурна ограничења над објектом **Reaction** морају бити задовољена.

Постуслови: Реакција је запамћена.

```
public void LikeIt(int postId, string username)
        {
            User u = new User { UserName = username };
            u = unit.UserRepository.SearchByUsername(u);
            Reaction r = new Reaction
            {
                PostId = postId,
                UserId = u.Id
            };
            unit.ReactionRepository.Add(r);
            unit.Save();
        }
```

6. **Уговор УГ6:** GetComments

Операција: GetComments(postId, List<CommentResponse>):signal;

Веза са СК: СК5

Предуслови:

Постуслови:

```
public List<CommentResponse> GetComments(int postId)
        {
            List<Comment> comments = unit.CommentRepository.GetComments(postId);
            List<CommentResponse> commentsDto = new List<CommentResponse>();
            comments.ForEach(c =>
            {
                commentsDto.Add(mapper.toDto(c));
            });
            return commentsDto;
        }
```

7. **Уговор УГ7:** PostComment

Операција: PostComment(CommentRequest, CommentResponse):signal;

Веза са СК: СК5

Предуслови: Вредносна и структурна ограничења над објектом **Comment** морају бити задовољена.

Постуслови: Коментар је запамћен.

```
public CommentResponse PostComment(CommentRequest dto)
        {
            User u = new User { UserName = dto.Username };
            u = unit.UserRepository.SearchByUsername(u);
            Comment c = new Comment();
            c.CommentText = dto.CommentText;
            c.PostId = dto.PostId;
            c.User = u;
            c.DatumVreme = DateTime.Now;
            Comment result = unit.CommentRepository.PostComment(c);
            if (result != null)
            {
                unit.Save();
                return mapper.toDto(c);
            }
            return null;
        }
```

8. **Уговор УГ8:** Search

Операција: Search(kriterijum, userId, List<UserDto>):signal;

Веза са СК: СК6

Предуслови:

Постуслови:

```
public List<UserDto> Search(string kriterijum, int id)
        {
            List<User> users=unit.UserRepository.Search(kriterijum, id);
            List<UserDto> usersDto = new List<UserDto>();
            if(users.Count()==0)
            {
                return usersDto;
            }
            users.ForEach(u => {
                UserDto dto = mapper.toDto(u);
                Notification not = new Notification { FromWhoId = id, ForWhoId =
      dto.Id };
                bool exist =
      unit.FollowNotificationRepository.ExistActiveFollow(not);
                if(exist)
                {
                    dto.RequestSent = true;
                }
                User pronadjen=u.Followers.Find(u => u.Id == id);
                if(pronadjen!=null)  dto.IFollow = true;
                User pronadjen2 = u.Following.Find(u => u.Id == id);
                if (pronadjen2 != null) dto.FollowingMe = true;
                usersDto.Add(dto);
            });
            return usersDto;
        }
```

9. **Уговор УГ9:** UcitajUsera

Операција: UcitajUsera(userId, username, UserDto):signal;

Веза са СК: СК6

Предуслови:

Постуслови:

```
public UserDto UcitajUsera(int id, string username)
        {
            User u = new User { Id = id };
            User ja = new User { UserName = username };
            ja = unit.UserRepository.SearchByUsername(ja);
            u=unit.UserRepository.SearchById(u);
            if(u!=null)
            {
```

```
                bool not = unit.FollowNotificationRepository.ExistActiveFollow(new
Notification { FromWhoId = ja.Id, ForWhoId = id });
                UserDto user=mapper.toDto(u);
                 if(not)
                 {
                     user.RequestSent = true;
                 }
                 ja.Followers.ForEach(f =>
                 {
                     if (f.Id == u.Id)
                     {
                         user.FollowingMe = true;
                     }
                 });
                 if (!user.RequestSent)
                 {
                     ja.Following.ForEach(f =>
                     {
                         if (f.Id == u.Id)
                         {
                             user.IFollow = true;
                         }
                     });
                 }
                 return user;
             }
            return null;
               }
```

10. Уговор УГ10: CreateFollow

Операција: CreateFollow(userId, username):signal;

Веза са СК: СК7

Предуслови: Вредносна и структурна ограничења над објектом **FollowNotification** морају бити задовољена.

Постуслови: Захтев је запамћен.

```
public FollowNotificationDto CreateFollow(int userId, string username)
        {
            User following = new User { UserName = username };
            User followed = new User { Id = id };
            following=unit.UserRepository.SearchByUsername(following);
            followed = unit.UserRepository.SearchById(followed);
            FollowNotification not = new FollowNotification {
                Date=DateTime.Now,
                ForWho=followed,
                FromWho=following,
                Status=FollowStatus.Waiting
            };
            unit.FollowNotificationRepository.Add(not);
            unit.Save();
            return mapper.toDto(not);
        }
```

11. Уговор УГ11: Unfollow

Операција: Unfollow(username, userId):signal;

Веза са СК: СК8

Предуслови: Структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Корисник је отпраћен.

```
public bool Unfollow(string username, int userId)
        {
            bool result=unit.UserRepository.Unfollow(username, id);
            if(result)
            {
                unit.Save();
            }
            return result;
        }
```

12. Уговор УГ12: ChangePicture

Операција: ChangePicture(UserDto):signal;

Веза са СК: СК9

Предуслови: Вредносна и структурна ограничења над објектом **User** морају бити задовољена.

Постуслови: Профилна слика корисника је промењена.

```
public bool ChangePicture(UserDto user)
        {
            User u = mapper.toEntity(user);
            if (u.ProfilePicture == null || u.ProfilePicture == "")
            {
                return false;
            }
            bool result = unit.UserRepository.ChangePicture(u);
            if (result)
            {
                unit.Save();
            }
            return result;
        }
```

13. Уговор УГ13: GetAllMyPosts

Операција: GetAllMyPosts(userId, username, List<PostResponse>):signal;

Веза са СК: СК9

Предуслови:

Постуслови:

```
public List<PostResponse> GetAllMyPosts(int userId, string username)
        {
            try
            {
                List<Post> posts=unit.PostRepository.GetMyPosts(userId);
             User u = unit.UserRepository.SearchByUsername(new User { UserName =
             username });
                List<PostResponse> responses = new List<PostResponse>();
                posts.ForEach(p =>
                {
                    PostResponse response = responseMapper.toDto(p);
                    p.Reactions.ForEach(r =>
                    {
                        if (r.UserId == u.Id)
                        {
                            response.ILiked = true;
                        }
                    });
                    response.NumberOfLikes = p.Reactions.Count();
                    response.NumberOfComments = p.Comments.Count();
                    responses.Add(response);
                });
                return responses;
            }
            catch (Exception ex)
            {
                return null;
            }
           }
```

14. Уговор УГ14: GetInboxUsers

Операција: GetInboxUsers(userId, List<UserDto>):signal;

Веза са СК: СК10

Предуслови:

Постуслови:

```
public List<UserDto> GetInboxUsers(int userId)
     {
         List<User> users= unit.MessageRepository.GetInboxUsers(userId);
         List<UserDto> usersDto = new List<UserDto>();
         if(users.Count()==0)
         {
             return usersDto;
         }
         users.ForEach(u =>
         {
             usersDto.Add(userMapper.toDto(u));
         });
         return usersDto;
           }
```

15. Уговор УГ15: FindInboxUsers

Операција: FindInboxUsers(kriterijum, userId, List<UserDto>):signal;

Веза са СК: СК10

Предуслови:

Постуслови:

```
public List<UserDto> FindInboxUsers(int userId, string kriterijum)
        {
            List<User> users = unit.MessageRepository.GetInboxUsers(userId);
            users = users.Where(u => u.FirstName.Contains(kriterijum)).ToList();
            List<UserDto> usersDto = new List<UserDto>();
            users.ForEach(u =>
            {
                usersDto.Add(userMapper.toDto(u));
            });
            return usersDto;
        }
```

16. Уговор УГ16: GetChat

Операција: GetChat(fromId, forId, List<Message>):signal;

Веза са СК: СК10

Предуслови:

Постуслови:

```
public List<Message> GetChat(int fromId, int forId)
        {
            Message m = new Message { ForId = forId, FromId = fromId };
            return unit.MessageRepository.GetChat(m);
           }
```

17. Уговор УГ17: SendMessage

Операција: SendMessage(MessageDto, MessageDto):signal;

Веза са СК: СК10

Предуслови: Вредносна и структурна ограничења над објектом **Message** морају бити задовољена.

Постуслови: Порука је запамћена.

```
public MessageDto SendMessage(MessageDto mess)
        {
            Message m = messageMapper.toEntity(mess);
            m = unit.MessageRepository.Send(m);
            if(m!=null)
            {
                unit.Save();
                return messageMapper.toDto(m);
            } else
            {
                return null;
            }
        }
```

11.4. Имплементација слоја приступа подацима

У сврху објектно-релационог мапирања коришћен је Entity Framework Core. Како би се омогућио рад са базом података, неопходно је направити модел који мапира ентитете и односе дефинисане у моделу у табеле базе. Класа која се користи за интеракцију са базом је DbContext.

Креирана је класа UserContext која наслеђује класу DbContext. У оквиру доменских класа већ су дефинисани атрибути и одговарајући спољни кључеви, али за рад са базом то није довољно, односно потребно је експлицитно дефинисати везе између класа. Те везе дефинисане су у оквиру методе *OnModelCreating*. Табеле у бази дефинишу се помоћу пропертија DbSet<EntityType>.

```
39 references
public class UserContext : IdentityDbContext<User, IdentityRole<int>, int>
{

    0 references
    public UserContext([NotNull] DbContextOptions options) : base(options)
    {

    }

    11 references
    public DbSet<User> Users { get; set; }
    4 references
    public DbSet<Post> Posts { get; set; }
    6 references
    public DbSet<Message> Messages { get; set; }
    2 references
    public DbSet<Reaction> Reactions { get; set; }
    2 references
    public DbSet<Comment> Comments { get; set; }
    1 reference
    public DbSet<Notification> Notifications { get; set; }
    3 references
    public DbSet<LikeNotification> LikeNotifications { get; set; }
    1 reference
    public DbSet<CommentNotification> CommentNotification { get; set; }
    4 references
    public DbSet<FollowNotification> FollowNotifications { get; set; }
```

Слика 180. Класа UserContext

```
0 references
protected override void OnModelCreating(ModelBuilder builder)
{
    base.OnModelCreating(builder);
    builder.Entity<User>().HasMany(s => s.Posts).WithOne(p => p.User);

    builder.Entity<User>().HasMany(u => u.Followers).WithMany(u => u.Following);

    builder.Entity<Reaction>().HasKey(r => new { r.PostId, r.UserId });
    builder.Entity<Reaction>().HasOne(r => r.User).WithMany(u => u.Reactions);
    builder.Entity<Reaction>().HasOne(r => r.Post).WithMany(p => p.Reactions);
    builder.Entity<Reaction>().ToTable("Reactions");

    builder.Entity<Message>().HasKey(m => m.MessageId);
    builder.Entity<Message>().HasOne(m => m.FromUser).WithMany(s => s.Received).HasForeignKey(m => m.FromId);
    builder.Entity<Message>().HasOne(m => m.ForUser).WithMany(s => s.Send).HasForeignKey(m => m.ForId);

    builder.Entity<Comment>().HasKey(c => new { c.UserId, c.PostId, c.DatumVreme });
    builder.Entity<Comment>().HasOne(c => c.Post).WithMany(p => p.Comments);
    builder.Entity<Comment>().HasOne(c => c.User).WithMany(u => u.Comments);
    builder.Entity<Comment>().ToTable("Comments");

    builder.Entity<Notification>().HasKey(n => new { n.ForWhoId, n.FromWhoId, n.Date });

    builder.Entity<LikeNotification>().HasOne(c => c.Post).WithMany(p => p.LikeNotifications);
    builder.Entity<CommentNotification>().HasOne(c => c.Post).WithMany(p => p.CommentNotifications);
    builder.Entity<Notification>().HasOne(n => n.FromWho).WithMany(u => u.NotificationsFromMe);
    builder.Entity<Notification>().HasOne(n => n.ForWho).WithMany(u => u.MyNotifications);

    builder.Entity<LikeNotification>().HasBaseType<Notification>().ToTable("LikeNotifications");
    builder.Entity<CommentNotification>().HasBaseType<Notification>().ToTable("CommentNotifications");
    builder.Entity<FollowNotification>().HasBaseType<Notification>().ToTable("FollowNotifications");
```

Слика 181. Дефинисање веза између ентитета

У наставку је приказа имплементација Unit of Work патерна у студијском примеру.

```
public class UnitOfWork : IUnitOfWork
{
    private readonly UserContext context;
    public UnitOfWork(UserContext context)
    {
        this.UserRepository = new UserRepository(context);
        this.PostRepository = new PostRepository(context);
        this.MessageRepository = new MessageRepository(context);
        this.LikeNotificationsRepository = new
      LikeNotificationRepository(context);
        this.CommentNotificationRepository = new
      CommentNotificationRepository(context);
        this.FollowNotificationRepository = new
      FollowNotificationRepository(context);
             this.ReactionRepository = new ReactionRepository(context);
             this.CommentRepository = new CommentRepository(context);
             this.context = context;
     }
    public IUserRepository UserRepository { get; set; }
    public IPostRepository PostRepository { get; set; }
    public IMessageRepository MessageRepository { get; set;  }
    public INotificationRepository NotificationRepository { get; set; }
    public ILikeNotificationsRepository LikeNotificationsRepository { get; set; }
    public ICommentNotificationRepository CommentNotificationRepository {
      get; set; }
    public IFollowNotificationRepository FollowNotificationRepository { get;
      set; }
    public IReactionRepository ReactionRepository { get; set; }
    public ICommentRepository CommentRepository { get; set; }
    public void Save()
    {
       context.SaveChanges();
    }}
```

У наставку је приказана имплементација Repository патерна на у студијском примеру.
Генерички репозиторијум:

```
public interface IRepository<TEntity> where TEntity : class
    {
        public void Add(TEntity entity);
        public TEntity SearchById(TEntity entity);
    }
```

Интерфејс за доменски објекат Post:

```
public interface IPostRepository : IRepository<Post>
    {
        public List<Post> GetAllForHome(int id, int skip, int take);
        public List<Post> GetMyPosts(int id);
    }
```

Конкретна имплементација интерфејса за доменски објекат Post:

```
public class PostRepository : IPostRepository
    {
        private readonly UserContext context;
        public PostRepository(UserContext context)
        {
            this.context = context;
        }
        public void Add(Post entity)
        {
            context.Add(entity);
        }
        public List<Post> GetAllForHome(int i, int skip, int take)
        {
            User user = context.Users.Include(u=> u.Following).SingleOrDefault(u
      => u.Id == i);
            List<Post> posts = new List<Post>();
            user.Following.ForEach(u => {
                List<Post> p = context.Posts.Include(p =>
      p.Reactions).Include(p=>p.Comments).Where(m => m.UserId == u.Id).ToList();
                posts.AddRange(p);
            });
            posts=posts.OrderByDescending(s => s.Date).ToList();
            return posts.Skip(skip).Take(take).ToList();

        }
        public List<Post> GetMyPosts(int id)
        {
            List<Post> posts=context.Posts.Include(p => p.Reactions).Include(p=>
      p.User).Include(p => p.Comments).Where(p => p.UserId == id).ToList();
            posts= posts.OrderByDescending(s => s.Date).ToList();
            return posts;
        }
        public Post SearchById(Post entity)
        {
            return context.Posts.SingleOrDefault(s => s.PostId == entity.PostId);
        }
    }
```

11.5. Имплементација складишта података

На основу доменских класа, пројектоване су табеле релационог система за управљање базом података.

kristy\LOCALDB#F2...- dbo.AspNetUsers

Column Name	Data Type	Allow Nulls
Id	int	☐
FirstName	nvarchar(MAX)	☐
LastName	nvarchar(MAX)	☐
Email	nvarchar(256)	☐
UserName	nvarchar(256)	☑
NormalizedUserName	nvarchar(256)	☑
NormalizedEmail	nvarchar(256)	☑
EmailConfirmed	bit	☐
PasswordHash	nvarchar(MAX)	☑
SecurityStamp	nvarchar(MAX)	☑
ConcurrencyStamp	nvarchar(MAX)	☑
PhoneNumber	nvarchar(MAX)	☑
PhoneNumberConfirmed	bit	☐
TwoFactorEnabled	bit	☐
LockoutEnd	datetimeoffset(7)	☑
LockoutEnabled	bit	☐
AccessFailedCount	int	☐
ProfilePicture	nvarchar(MAX)	☐
		☐

Слика 182.. Табела User

kristy\LOCALDB#F2...ork - dbo.UserUser

Column Name	Data Type	Allow Nulls
FollowersId	int	☐
FollowingId	int	☐
		☐

Слика 183. Табела UserUser

kristy\LOCALDB#F2...etwork - dbo.Posts

Column Name	Data Type	Allow Nulls
PostId	int	☐
UserId	int	☐
Date	datetime2(7)	☐
Description	nvarchar(MAX)	☐
ImagePath	nvarchar(MAX)	☐
		☐

Слика 184. Табела Post

kristy\LOCALDB#F2...rk - dbo.Messages

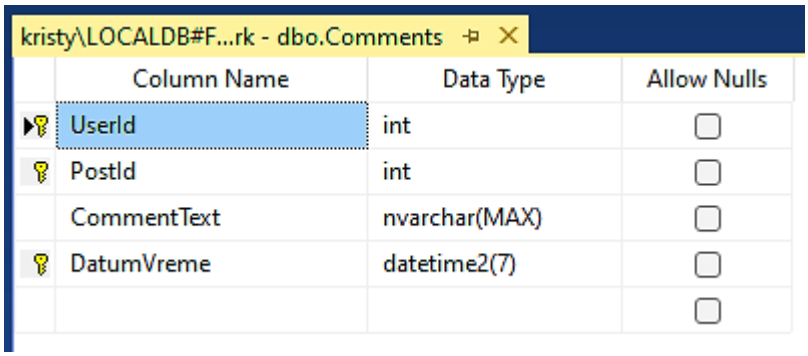
Column Name	Data Type	Allow Nulls
FromId	int	☐
ForId	int	☐
MessageText	nvarchar(MAX)	☐
MessageId	int	☐
Time	datetime2(7)	☐
		☐

Слика 185. Табела Message

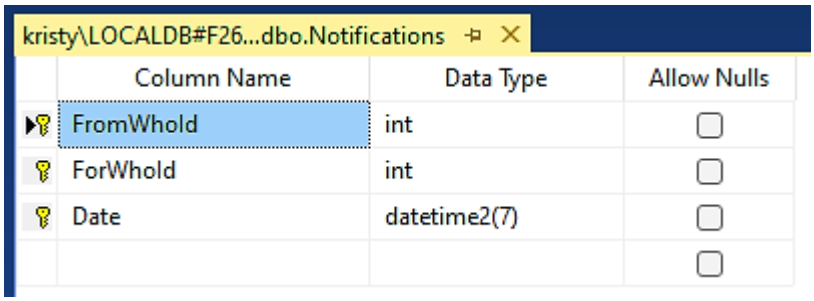
kristy\LOCALDB#F26...rk - dbo.Reactions

Column Name	Data Type	Allow Nulls
UserId	int	☐
PostId	int	☐
		☐

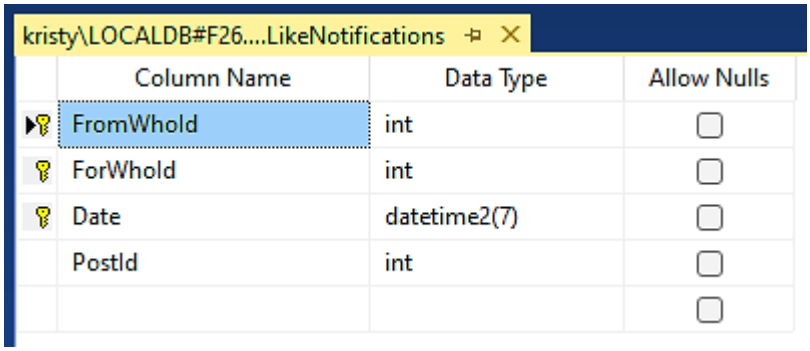
Слика 186. Табела Reactions



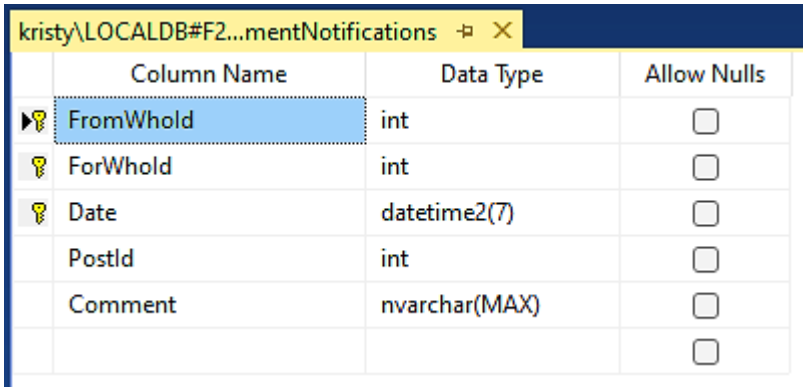
Слика 187. Табела Comments



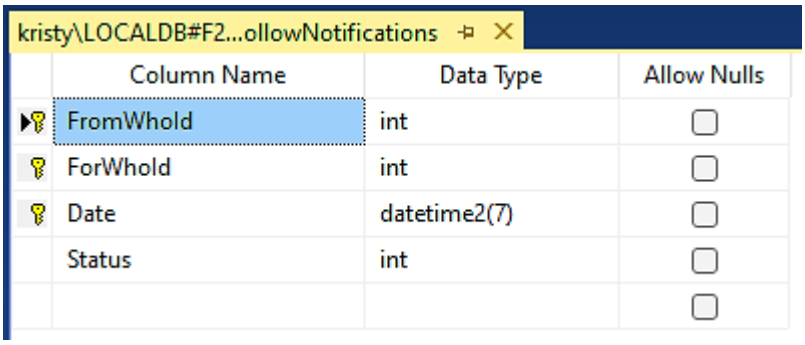
Слика 188. Табела Notifications



Слика 189. Табела LikeNotifications



Слика 190. Табела CommentNotifications



Слика 191. Табела FollowNotifications

12. Тестирање

Тестирање је извршено мануелно, нису коришћени никакви алати за тестирање. Тестирање је вршено покретањем апликације и уношењен неисправних вредности, али такође и правилних како би се осигурало правилно функционисање апликације. Фазом тестирања утврђено је да софтвер правилно функционише и да је спреман за коришћење.

13. Закључак

У раду *Развој друштвене мреже применом ASP.NET Core и SignalR* оквира описана је комплетна архитектура и поступак за креирање Web апликације.

Објашњене су технологије које су коришћене за развој софтверског система. За серверски део апликације то су *ASP.NET Core* технологије, *Entity Framework Core* за рад са подацима из базе, *SignalR* који омогућава асинхрону комуникацију између клијента и сервера и релациона база података *MySQL*. За клијентски део апликације коришћен је *JavaScript* библиотека *Angular*, која својим могућностима и компонентама олакшава развој корисничког интерфејса. Поред ових технологија, имплементирани су и патерни, *Unit of Work* и *Repository* патерн, који омогућавају независност слоја приступа подацима.

Фазом тестирања утврђено је да апликација потпуно исправно функционише. Такође, утврђене су и могућности за побољшање апликације, као што су приказ тренутно доступних (пријављених) корисника, приказ податка који говори да ли је корисник прочитао поруку, приказ броја пратиоца, могућност да профил корисника буде приватан... Ове и још других ствари, биће имплементиране у будућности.

Када сам се први пут сусрела са .NET технологијама на предмету Напредне .NET технологије већ сам имала основно знање о програмском језику C#. На предмету Напредне .NET технологије то знање сам значајно унапредила и проширила. Упознала сам се са многим битним и данас јако популарним концептима који ће ми значити у мом даљем раду и учењу. Прављењем ове апликација схватила сам да је софтверско инжењерство област у којој желим да радим и усавршавам се, посебно у области .NET технологија.

14. Литература

[1] Влајић, С. – 2020. *Пројектовање софтвера (скрипта).* Београд, Србија: ФОН

[2] What is .NET? (.) Преузето са: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

[3] Entity Framework Core (2021, мај, 25). Преузето са: Overview of Entity Framework Core - EF Core | Microsoft Learn

[4] Entity Framework Core (.). Преузето са: Entity Framework Core Tutorials (entityframeworktutorial.net)

[5] .NET Core, .NET Framework, Xamarin (2016, јун, 27) Преузето са: .NET Core, .NET Framework, Xamarin – The "WHAT and WHEN to use it" - Cesar de la Torre (microsoft.com)

[6] ASP.NET Core (2022, септембар, 21) Преузето са: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

[7] Увод у LINQ (2018, април, 17). Преузето са: MANUEL RADOVANOVIĆ : Uvod u LINQ (manuelradovanovic.com)

[8] Hubs in SignalR (2022, јул, 15). Преузето са: Use hubs in ASP.NET Core SignalR | Microsoft Learn

[9] Introduction to SignalR (2020, октобар, 9). Преузето са: Introduction to SignalR | Microsoft Learn

[10] WebSockets (2022, август, 26). Преузето са: https://javascript.info/websocket

[11] Authorization in ASP.NET Core (2022, јун, 3). Преузето са: Introduction to authorization in ASP.NET Core | Microsoft Learn

[12] Authentication in ASP.NET Core (2022, јун, 3). Преузето са: Overview of ASP.NET Core Authentication | Microsoft Learn

[13] Linq-to-Entities Query (.). Преузето са: https://www.entityframeworktutorial.net/querying-entity-graph-in-entity-framework.aspx

[14] Шта су Web сервиси? (.). Преузето са: https://www.webprogramiranje.org/web-servisi-osnove/

[15] An overview of HTTP (.). Преузето са: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[16] API Architecture: The HTTP Protocol and Its Importance (2021, април, 7). Преузето са: https://medium.com/api-world/api-architecture-the-http-protocol-and-its-importance-aeba0fe46f91

[17] Introducing JSON (.). Преузето са: https://www.json.org/json-en.html

[18] TypeScript vs JavaScript (2022, септембар, 5). Преузето са: https://radixweb.com/blog/typescript-vs-javascript

[19] Introduction to Angular concepts (.). Преузето са: https://angular.io/guide/architecture

[20] Understanding the Purpose and Use of the Selector in Angular (2020, јануар, 31). Преузето са: https://www.pluralsight.com/guides/understanding-the-purpose-and-use-of-the-selector-in-angular

[21] What is JSON Web Token? (.). Преузето са: https://jwt.io/introduction

[22] The WebSocket API (.). Преузето са: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[23] Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application (2022, јануар, 07). Преузето са: Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application (9 of 10) | Microsoft Learn

[24] Overview of .NET Framework (2021, септембар, 15). Преузето са: Overview of .NET Framework - .NET Framework | Microsoft Learn

[25] Sta je Bootstrap – osobine, prednosti i mane (2015, април, 22). Преузето са: https://falcon-tech.rs/blog/sta-je-bootstrap/

[26] Uvod u Angular i TypeScript (2018, децембар). Преузето са: https://rti.etf.bg.ac.rs/rti/ir4pia/materijali/predavanja/si4pia_Angular7-P1-P3.pdf

[27] Overview of ASP.NET Core SignalR (2022, септембар, 21). Преузето са: Overview of ASP.NET Core SignalR | Microsoft Learn