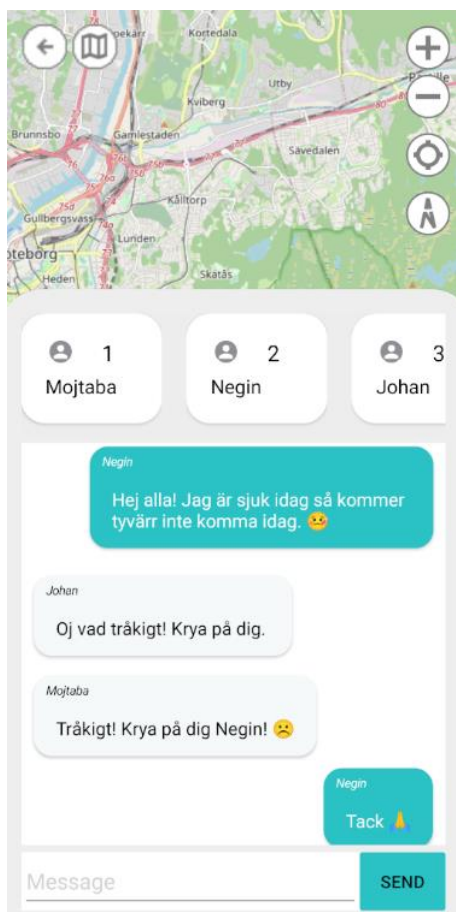




# CHALMERS



## Realtidskommunikation via ett chatsystem

Examensarbete inom Höskoleingenjörsprogrammet Datateknik

**Mojtaba Ataie**

**Negin Bakhtiarirad**

---

Institutionen för Data- och informationsteknik

CHALMERS TEKNISKA HÖGSKOLA

GÖTEBORGS UNIVERSITET

Göteborg, Sverige 2022

# Realtidskommunikation via ett chattsystem

En chattsida som möjliggör realtidskommunikation mellan flera användare i en prototyp

Mojtaba Ataie

Negin Bakhtiarirad

© Mojtaba Ataie, Negin Bakhtiarirad, 2022

Examinator: Jonas Almström Duregård

Handledare: Sakib SisteK

Omslag: Alla figurer i denna rapport är designade av författarna.

Institutionen för Data- och informationsteknik  
CHALMERS TEKNISKA HÖGSKOLA  
GÖTEBORGS UNIVERSITET  
Göteborg, Sverige 2022  
SE-412 96 Göteborg  
Telefon: +46 (0)31-772 1000

## Sammanfattning

Tekniken utvecklas varje dag och realtidskommunikation har blivit ett kritiskt krav för de flesta företag. Företag kan chatta med sina kunder och hålla dem uppdaterade om aktuella händelser med hjälp av ett chattsystem som skickar och tar emot meddelanden i realtid.

I en applikation eller på en hemsida är ett chattsystem oerhört effektivt och användbart för både användare och företag. En helt fungerande chattsida som skickar och tar emot meddelanden i realtid förbättrar användarkommunikationsupplevelse och informationshantering. Ett chattsystem i realtid kommer att implanteras i en befintlig prototyp i detta projekt. Chattsystemet kommer att utvecklas för att vara så användarvänligt som möjligt med hjälp av kunskap om interaktionsdesign och olika metoder som prototypframställning.

Utformningen av chattsidan planeras upp innan implementationen börjar. Chattsystemets användbarhet utvärderas ur användarens synvinkel med hjälp av användartester. Chattsystemets implementering kräver programmeringskunskaper och en grundlig förståelse för realtidskommunikation som används under programmeringsprocessen. Resultatet av detta projekt är en fungerande prototyp som låter användaren kommunicera i realtid. Prototypen har utvecklats för att vara användarvänlig och ha användarupplevelsen i fokus.

**Nyckelord:** Chattsystem, Chattapplikation, SiganlR, Realtidskommunikation, Xamarin.Forms, Interaktionsdesign.

## Abstract

Technology is evolving every day and real-time communication has become a critical requirement for most companies. Companies can chat with their customers and keep them updated on current events using a chat system that sends and receives messages in real time.

In an application or on a website, a chat system is extremely efficient and useful for both users and companies. A fully functional chat site that sends and receives messages in real time improves the user communication experience and information management. A real-time chat system will be implemented in an existing prototype in this project. The chat system will be developed to be as user-friendly as possible with the help of knowledge of interaction design and various methods such as prototyping.

The design of the chat page is planned before the implementation begins. The usability of the chat system is evaluated from the user's point of view using user tests. The implementation of the chat system requires programming skills and a thorough understanding of real-time communication which are used during the programming process. A functional prototype that allows the user to communicate in real time is the result of this project. The prototype has been developed to be user-friendly and have the user experience in focus.

**Keyword:** Chat System, Chat Application, SignalR, Real-time communication, Xamarin.Forms, Interaction design.

## Förord

Denna rapport är en kandidatexamen på Chalmers tekniska högskola inom datateknikprogrammet. Arbetet är en sammanfattning av många kurser som studerats under programmet, samt en hel del nya kunskaper som har uppnåtts under arbetet.

Vi vill tacka vår handledare, Sakib Sistek och vår examinerator, Jonas Almström Duregård som har väglett oss genom processen och hjälpte oss med alla våra frågor. Vi har gjort betydande framsteg i projektet tack vare deras hjälp och stöd.

Ett stort tack till Företaget Elicit för att de gav oss möjlighet att genomföra detta arbete hos dem. Vi vill visa vår tacksamhet till Jenny Forsberg som var med oss under hela arbetet och bistått oss med både tekniska och målsättningsfrågor.

Sist men inte minst, vill vi tacka Cynthia Kozma och Petra Nisan för ett otroligt bra samarbete under projektets gång. Paret skrev kandidatexamen på samma projekt och hanterade databasen för denna prototyp.

Mojtaba Ataie, Negin Bakhtiarirad, Göteborg, Sverige, juni 2022

## Ordlista

**Användarupplevelse (UX):** Användarupplevelse är hur en användare använder en produkt eller tjänst. Det inkluderar personens tankar och åsikter om användbarhet, användarvänlighet och effektivitet.

**API:** API står för Application Program Interface. API är ett protokoll som används för att två applikationer ska kunna kommunicera med varandra och överföra data till den andra.

**Brandvägg:** En brandvägg är en mjukvaru- eller hårdvarubaserad kontrollenhet som avgör om en nätverkstrafik är tillåten eller inte. Om en nätverkstrafik anses vara otillåten och farlig, kommer den trafiken att blockeras.

**CSR-projekt:** CSR-projekt står för corporate social responsibility, vilket betyder att ett företag bidrar till ett bättre samhälle och en välvårdad miljö frivilligt.

**Full-duplex:** Full-duplex beskriver att data kan skickas och tas emot samtidigt och dubbelriktad över en kanal.

**IP adress:** IP står för Internet Protocol. En IP-adress är ett nummer eller en kod som identifierar en enhet och innehåller information som gör att andra parter kan känna igen den.

**Layout:** Layout betyder design och plan. Den är en skiss till uppställning av text och bilder till trycksida.

**Localhost:** Localhost, även känd som lokal server, används för att skapa en IP-anslutning på den specifika datorn. Man kan använda localhost för att komma åt nätverkstjänster på den datorn.

**PaaS:** PaaS står för Platform as a service. Det är en molntjänst som erbjuder en datorplattform och en samling mjukvarusystem som en service.

**Polling:** Polling är ett sätt att skapa webbfunktionalitet i realtid och regelbundet frågar servern om det finns en uppdatering.

**Prototyping:** Att bygga en prototyp är vad prototyping innebär på engelska.

**RDBMS:** RDBMS står för Relational Database Management System och är ett program som används för att underhålla relationsdatabas.

**Remote Procedure Call (RPC):** Fjärrproceduranrop, ett programvarukommunikationsprotokoll som ett program kan använda för att begära en tjänst från ett program som finns på en annan dator.

**Server:** En server kan vara en mjukvara eller hårdvara som över ett nätverk betjänar tjänster till andra datorer, kallad för klienter. Server bearbetar och svarar på nätverksförfrågningar som skickas från klienter.

**WebSocket:** WebSocket är en teknik som möjliggör dubbelriktad kommunikation mellan webbläsare och server.

**Wireframing:** Wireframing är en engelsk term som syftar på skapandet av enkla layouter som visar olika elementens placering och storlekar, utan hänsyn till designdetaljer som färger.

# Innehållsförteckning

1 Inledning .....	1
1.1 Bakgrund .....	1
1.2 Syfte .....	1
1.3 Mål .....	1
1.4 Samarbete .....	2
1.5 Avgränsningar .....	2
2 Teknisk bakgrund.....	3
2.1 Frontend .....	3
2.1.1 XAML.....	3
2.1.2 Interaktionsdesign .....	3
2.1.3 Wireframe .....	3
2.1.4 Designskiss .....	4
2.1.5 Prototypframställning .....	4
2.2 Backend.....	4
2.2.1 .NET.....	4
2.2.2 Programmeringsspråk .....	4
2.2.3 SignalR.....	5
2.2.4 Designmönstret MVVM .....	5
2.2.5 API.....	5
2.2.6 Databas.....	5
2.2.6.1 MySQL .....	6
2.2.6.2 SQL Server Management Studio .....	6
2.2.6.3 DBeaver .....	6
2.2.6.4 Docker.....	6
2.3 Xamarin.Forms.....	6
2.4 Visual Studio .....	7
2.5 Git.....	7
2.6 Azure DevOps .....	7
2.7 Microsoft Teams .....	8
2.8 Discord .....	8



2.9 Miro .....	8
2.10 Figma.....	8
3 Metod .....	9
3.1 Metodik för Agil mjukvaruutveckling .....	9
3.2 Scrum .....	9
3.3 Intervju .....	10
3.4 Observation .....	10
3.5 Användarvänlighet .....	10
4 Genomförande.....	11
4.1 Design.....	11
4.1.1 Primära design .....	11
4.1.2 Designskaiss enligt Pindle-prototypens tema .....	12
4.1.3 Slutlig design .....	13
4.2 SignalR.....	14
4.3 Konfiguration av SignalR.....	15
4.3.1 Skapa Hub.....	15
4.3.2 Skapa Klient.....	16
4.3.2.1 Skapa anslutning till Hub.....	16
4.3.2.2 Skicka meddelande till Hub .....	17
4.3.2.3 Ta emot meddelande från Hub.....	17
4.5 Implementation av chattsidan.....	17
4.6 Implementation av inloggningssida och registreringssida .....	20
5 Resultat .....	21
5.1 Slutresultat.....	21
6 Diskussion.....	24
6.1 Målanalys .....	24
6.2 Några rekommendationer för framtida arbete .....	26
6.3 Etiska frågor och ingenjörens ansvar .....	27
6.4 Miljöpåverkan .....	27
7 Slutsats .....	29
Referenser .....	30

# 1 Inledning

Detta avsnitt introducerar bakgrund, syfte, mål, samarbete och slutligen avgränsningar.

## 1.1 Bakgrund

Detta examensarbete har genomförts i ett IT-företag som heter "Elicit Software". Elicit Software är en del av Elicit Group, ett värderingsstyrkt IT-företag med över 20 års erfarenhet av digitalisering. De skapar hållbara IT-lösningar som förenklar vardagen genom nära samarbeten med såväl medarbetare som kunder. Elicit Software äger sedan många år tillbaka en välutvecklad lerveransplattform, pinDeliver. Plattformen har som syfte att hjälpa företag med sina transporter [1]. Under examensarbetetsgång ska möjligheten att erbjuda ett chattsystem i applikationen som erbjuds från pinDeliver undersökas.

Examensarbetet valdes att genomföras i Elicit för att företaget har en stark vilja att skapa IT-lösningar som verkligen skapar nytta och bidrar till ett mer hållbart samhälle [1]. Under detta examensarbete ville företaget implementera ett chattsystem i Pindle-prototypen som skapades under projektkursen (DAT067) på Chalmers tekniska högskola. Negin Bakhtiairad som jobbar på detta examensarbete, programmerade Pindle-prototypen tillsammans med fem andra Chalmers-studenter under november månad 2021. Ett chattsystem kommer att implementeras i prototypen så att användare kan kommunicera med andra personer som är med i samma gruppchatt. En inloggnings- och registreringssida kommer att implementeras så att avsändaren av varje meddelande på chattsidan kan identifieras.

## 1.2 Syfte

Syftet med arbetet är att prototypens användare ska kunna kommunicera med varandra i realtid via ett chattsystem. Denna funktionalitet ger möjligheten till användarna att hålla sig uppdaterade om aktuella händelser samt få viktiga information.

## 1.3 Mål

Målet med detta projekt är att lägga till ett chattsystem till den befintliga Pindle-prototypen. Detta chattsystem gör det möjligt för användare att kommunicera med andra användare i realtid, skicka viktig information och hålla andra uppdaterade om aktuella händelser. Nyckelaspekten och utmaningen i detta projekt är att kombinera en teknisk lösning med en användarvänlig design som möter företaget Elicits krav. Målen för detta projekt är:

- Designa en gruppchattsida, inloggningssida och registreringssida som prioriterar användarupplevelsen.

- Undersöka biblioteket SignalR för att säkerställa om det är ett lämpligt bibliotek för att implementera ett chattsystem som låter användaren kommunicera i realtid.
- Bygga en prototyp som låter användare kommunicera i realtid samt ger användaren möjlighet att identifiera sig genom att logga in med sin e-postadress eller skapa ett konto.

## 1.4 Samarbete

Pindle-applikationen har delats upp i två kandidatuppsatser av företaget Elicit. Det första är ett realtidschattsystem, som är fokus för detta arbete, och det andra är Pindle-prototypens databas. På grund av detta beslut fick paret samarbeta med databasparet för att få en slutlig fungerande prototyp. Ett möte planerades veckovis för att slå samman koden med det andra teamet och se till att de två programmen fungerade tillsammans. Samtidigt som paret fick en större förståelse för hur faktiskt samarbete i ett företag fungerar, lärde sig alla i båda paren oerhört mycket av varandra. Den tid som ägnas åt att lösa olika problem under samarbetet var lärorikt för båda paren och i slutändan resulterade i en fungerande prototyp.

## 1.5 Avgränsningar

- Fokuset kommer inte ligga på att skapa en applikation i sin helhet utan endast kommunikationens funktionalitet i kombination med Pindle-prototyp kommer att undersökas.
- Pindle är en befintlig prototyp som är utvecklad i Xamarin.Forms. Den kommer att användas som underlag för kommunikation under projektets gång.
- Olika bibliotek/ramverk för realtidskommunikation kommer inte att prövas. En undersökning av biblioteket SignalR kommer att genomföras under projektet.
- Ett möte med företaget Elicit Software utförs varje vecka på kontoret och resten av arbetet kommer utföras både på plats och distansform.

## **2 Teknisk bakgrund**

Det här avsnittet innehåller definition av varje programvara som har använts under projektet.

### **2.1 Frontend**

Frontend är ett grafiskt användargränssnitt för en webbplats eller en applikation så att människor kan se och interagera med den. Med andra ord är frontend den del av applikation eller websida som användaren ser och upplever direkt. En stor del av användarvänligheten och användarupplevelsen av en produkt bestäms av frontend del av den produkten [2].

#### **2.1.1 XAML**

XAML står för Extensible Application Markup Language och är ett XML-baserat språk utvecklat av Microsoft och används för att skapa applikationsgränssnitt. Detta språk liknar HTML men utan fördefinierade taggar därför utvecklaren definierar sina egna taggar specifikt för sitt behov. XAML definierar innehållet på en websida eller applikation [3].

#### **2.1.2 Interaktionsdesign**

Interaktionsdesign är en designprocess som fokuserar på en produkts eller tjänsts användningsegenskaper. Detta innebär att användbarhet, produktens kvalitet och användarupplevelser beaktas. När en produkt utvecklas med hjälp av interaktionsdesign är användbarhet prioritet, och denna funktion värderas högt i denna metod. Interaktionsdesign garanterar att produkten är enkel att använda och att användarna kan få nytta av den när de behöver det. Denna metod är baserad på människa-datorinteraktion och den används ofta för att designa digitala verktyg för produktivt arbete [4].

#### **2.1.3 Wireframe**

Innan den slutliga designprocessen börjar är Wireframes skisser som utformas för en tjänst eller digital produkt. Skisserna illustrerar den grundläggande utseendet, strukturen och till och med navigeringen för en produkt. Färger och former är däremot inte prioriterade i Wireframe-metoden. Detta är en utmärkt metod för företag att presentera alternativa tankar och lösningar för kunden samtidigt som de lyssnar på kundernas krav och synpunkter för att lättare nå kundens designmål. Pappers- och penna metod används ofta för att producera Wireframes, som sedan gradvis utvecklas till digitala skisser [5].

Wireframing har fördelen att designteamet kan testa och hitta brister tidigt i processen, vilket gör att de kan förbättra designen snabbt. Dessutom betonar Wireframes användarvänlighet och grundläggande struktur, vilket innebär att avgörande funktioner som är kundens krav och användarbehov kommer att inkluderas i designen [5].

### **2.1.4 Designskiss**

En designskiss är en mer detaljerad design av slutprodukten, inklusive färger, former och bilder. Designteamet och kunderna fokuserar mer på detaljerade konstruktioner för den färdiga produkten efter Wireframe-processen. Designskisser visar en bättre bild av slutprodukten utseende, men det är ännu inte en klickbar prototyp [5].

Designskisser baseras ofta på kundens aktuella grafiska profil eller grafiska detaljförfrågningar. Däremot med hjälp av designskisser kan olika designval erbjudas som inkluderar rätt färger och former för att ge en optimal design baserad på användarvänlighet [5].

### **2.1.5 Prototypframställning**

En prototyp är en simulering av en produkt eller tjänst som visar både utseendet och funktionaliteten. Prototyper är klickbara och hanterar navigering, vilket ger en tydlig bild av den färdiga produktens funktionalitet och användbarhet. Genom att skapa en prototyp kan man lättare undersöka hur produkten används samtidigt som flera tester kan utföras på prototypen för att hitta eventuella brister. En prototyp och dess funktionalitet kan även programmeras, och den koden återanvänds ofta i slutprodukten, vilket sparar tid [5].

## **2.2 Backend**

Backend tar hand om funktionalitet av en produkt som en hemsida eller applikation. Den hanterar allt som ska lagras, ordnas samt den ser till att allting på Frontend fungerar som den ska. Backend är den del av applikation eller websida som inte syns för användaren [2].

### **2.2.1 .NET**

.NET är en öppen källkod och plattformsoberoende utvecklarplattform som gör det möjligt för utvecklare att bygga många olika typer av applikationer på macOS, Windows och Linux. Flera språk och bibliotek kan användas med .NET för att bygga mobilapplikationer. Som tidigare nämnts Xamarin.Forms används i .NET implementering för att köra applikationer på alla mobila operativsystem [6].

### **2.2.2 Programmeringsspråk**

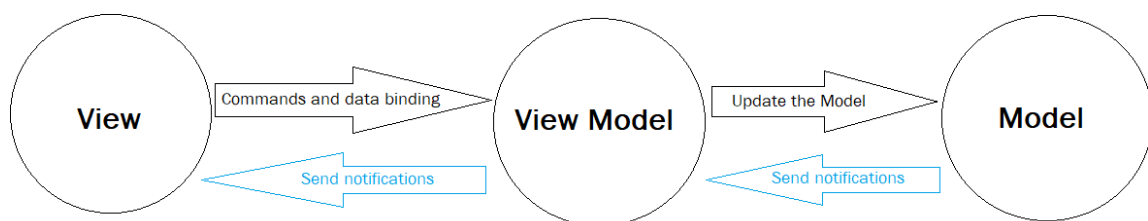
C# är ett modernt, enkelt och objektorienterat programmeringsspråk utvecklat av Microsoft på .NET-plattformen [7]. Som tidigare nämnts eftersom projektet är baserat på Xamarin.Forms, valdes C# för programmeringsspråket för denna prototyp.

### 2.2.3 SignalR

SignalR är ett öppet källkodsbibliotek för ASP.NET utvecklare som möjliggör dubbelriktad webbkommunikation mellan server och klient i realtid. Biblioteket möjliggör också helt nya typer av webbapplikationer som kräver högfrekventa uppdateringar från servern. SignalR tillhandahåller ett enkelt API för att skapa server-till-klient fjärrproceduranrop (RPC) som anropar JavaScript-funktioner i klientwebbläsare från .NET-kod på serversidan [8].

### 2.2.4 Designmönstret MVVM

MVVM är ett designmönster och står för Modell-Vy-VyModell. Detta mönster används oftast av Xamarin.Forms utvecklare där mönstret underlättar att separera affärs och presentationslogik från användargränssnittet. Mönstret stödjer också databindningar som är ett enkelt sätt att förflytta data mellan gränssnitt och datamodellen. I detta kommunikationssätt, kommunicerar Vy:n med VyModellen som i sin tur kommunicerar med Modellen [9].



Figur 2.1: Illustration på hur kommunikation i MVVM mönstret fungerar

### 2.2.5 API

API står för Application Program Interface. Detta protokoll låter två program, såsom en server eller en dator, ansluta och utbyta data. Det gör det också möjligt för två program att dela information och databaser genom att ansluta till samma server [10]. Flera användare kan ansluta till samma server och kommunicera med varandra med samma IP-adress i detta arbete.

### 2.2.6 Databas

Databas är en organiserad samling av information, oftast lagrad elektroniskt i ett datorsystem. Den används för att spara, uppdatera, modifiera och hämta data på ett smidigt sätt. En databas styrs vanligtvis av ett databashanteringssystem (DBMS). De flesta databaser använder SQL (Structured Query Language) för att skriva och begära data [11].

### **2.2.6.1 MySQL**

MySQL är den mest populära öppna källkod relationsdatabashanteringssystem (RDBMS) som huvudsakligt används som databashanterare. Detta utvecklas, distribueras och stöds av Oracle Corporation och är baserat på SQL. MySQL används inom Linux, Mac OS, Windows och andra operativsystem [12].

### **2.2.6.2 SQL Server Management Studio**

SQL Server Management Studio (SSMS) är en integrerad miljö för att hantera vilken SQL-infrastruktur som helst. Den tillhandahåller verktyg för att konfigurera, övervaka och administrera instanser av SQL Server och databaser. Andra användningar av SSMS är bland annat att fråga, designa och hantera databaser i lokala datorer eller i molnet [13]. I detta projekt används SSMS för att kommunicera mellan prototypen och databasen i Windows operativsystem.

### **2.2.6.3 DBeaver**

DBeaver är ett allmänt databasverktyg med öppen källkod för utvecklare, databasadministratörer, analytiker och alla som behöver arbeta med databaser [14]. I detta projekt används DBeaver för att kommunicera mellan applikationen och servern i Mac operativsystem.

### **2.2.6.4 Docker**

Docker är en uppsättning PaaS-produkter och för att leverera programvara i paket använder virtualisering på OS-nivå. Docker är en öppen källkod som gör det möjligt för utvecklare att paketera applikationer i behållare. Docker används under hela utvecklingsprocessen för snabb och portabel applikationsutveckling och moln [15].

## **2.3 Xamarin.Forms**

Xamarin.Forms är ett ramverk för användargränssnitt med öppen källkod. Xamarin.Forms tillåter utvecklare att från en enda baskod kunna bygga Xamarin.Android och Xamarin.iOS, vilket var ett krav för detta projekt. Xamarin.Forms låter att användargränssnitt skapas i XAML med kod i C# [16].

Xamarin.Forms låter utvecklaren att dela användargränssnittslayout, kod över plattformar. Detta används i Visual Studio för applikationer programmerade i C#. Kodens struktur följer Modell-Vy-VyModell (MVVM) design vilket stöds av Xamarin.Forms. .NET Standard-bibliotek är delat i alla Xamarin.Forms applikationer och detta bibliotek innehåller XAML- eller C#-Vyer. Xamarin-plattformen används i Xamarin.Forms för att köra .NET-applikationer över plattformar [16].

## 2.4 Visual Studio

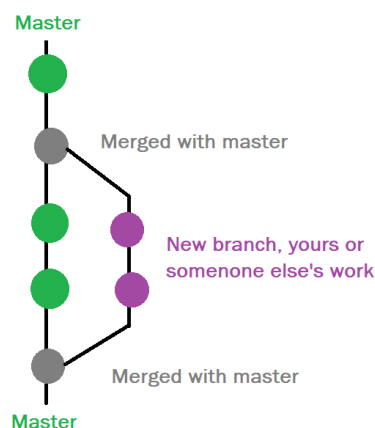
Microsoft Visual Studio är en avancerad mjukvaruutvecklingsmiljö utvecklad av Microsoft som används för att utveckla datorprogram, såväl som webbplatser, webbapplikationer och mobila applikationer. Visual studio innehåller ett antal olika programmeringsspråk, inklusive visual basic, visual C#, visual C++ och visual J#. Visual studio utvecklar plattformsoberoende mobila applikationer med .NET MAUI [17]. Visual Studio 2022 för Windows och macOS användes under hela projektet, den här versionen gjorde programmeringen enklare för paret, och att skapa kod och ändra grenar var enklare.

## 2.5 Git

Git är en mjukvaruplattform för samarbete och versionskontroll. Fler programmerare kan använda Git för att samarbeta i ett gemensamt projekt samtidigt vara helt oberoende av varandra. Git möjliggör också åtkomst till alla nya filer och uppdateringar från ett projekt. Repository, branch, committer och pull-förfrågningar är de mest användbara funktioner i Git. Dessa förmågor gör det lättare att samarbeta med andra programmerare och att arbeta på ett mer strukturerat och säkert sätt som ett team [18].

## 2.6 Azure DevOps

Azure DevOps är en Microsoft-produkt baserad på Azure molnplattform som möjliggör mjukvaruutveckling och distribution. Versionskontroll, testning, rapportering och projekthantering är allt tillgängligt i Azure DevOps. Den hanterar hela processen i programmet och gör allt enklare för programmerare [19]. Flera problem löstes tack vare Azure DevOps, och kommunikationen med databasparet gick smidigare via användning av detta verktyg. Azure DevOps underlättat processen för båda paren och gav de två projekten mer självständighet.



Figur 2.2: Illustration av Azure DevOps



## 2.7 Microsoft Teams

Microsoft Teams är en plattform som möjliggör dokumentdelning och online möte i ett team [20]. Paret har använt detta verktyg för att kommunicera med relevanta anställda på företaget såsom IT-supportrar och handledare angående arbetet och för att hålla kontakt med dem. Den slutliga presentationen av hela prototypen i företag ägde rum i ett hybridformat, och denna teknik användes för att bjuda in personer som inte kunde närvara personligen.

## 2.8 Discord

Discord är ett program som ursprungligen utvecklades för spelgemenskaper, men det används för närvarande också som en social nätverksplattform. Discord är tillgängligt på MacOS, Windows, Linux, Android och iOS, såväl som i webbläsare. De mest användbara funktionerna i detta program inkluderar röstkanaler, textkanaler, videosamtal och skärmdelning [21].

Discord används för kommunikation mellan gruppmedlemmar, dokumentdelning och programmering. Discord gör det lättare för paret att tillsammans lösa ett problem och hjälpa varandra under pandemi och distansmöten. Med hjälp av Discord gick kommunikationen med databasgruppen som tidigare presenterats mycket smidigare.

## 2.9 Miro

Miro är en bra visuell samarbetsplattform. Det är en snabb, gratis och lättanvänd whiteboardtavla online utformad för att göra det möjligt för paret att interagera när som helst, var som helst. Miro används framför allt för att skissa och designa utseende på applikationen och olika chattsidor som paret har ansvar för [22].

Paret kan skissa sidorna i ansökan så realistiskt som möjligt tack vare Miro, vilket ökar kvaliteten på skisserna. Brainstorming var en annan anledning att paret valde använda Miro för att lättare skriva idéer och diskuterar dem efteråt.

## 2.10 Figma

Figma är ett webbaserat verktyg med ett stort antal förbyggda komponentbibliotek som gör processen enklare för användaren. Navigering mellan knappar och sidor som kallas för drag-and drop, rullningsalternativet och en mängd andra funktioner i Figma ger användaren en upplevelse som är väldigt nära en riktig applikation [23]. Att välja Figma för prototyping ger paret en tydlig bild av slutprodukten och väcker nya idéer.

## 3 Metod

Under projektet har paret använt flera metoder för att kunna möta Elicits krav och samtidigt kunna leverera maximalt värde. Några av de metoder som har hjälpt paret att förbättra kvaliteten på arbetsresultatet listas nedan.

### 3.1 Metodik för Agil mjukvaruutveckling

I många branscher har denna teknik blivit användbar och populär. Agila metoder delar upp ett projekt i mindre uppgifter, vilket gör processen enkel och effektiv med utmärkt kvalitet. Genom att använda denna teknik kan nya idéer implementeras i planeringen på ett enkelt sätt, vilket passar bra i teamarbete [24].

Både företaget och paret valde ett agilt arbetssätt för detta projekt för att göra det lättare att integrera nya idéer. Denna metod hjälper också till att sätta mål och ger paret tydliga uppgifter. Agilt arbetssätt möjliggör ett högkvalitativt utvecklingssätt under ett projekt [24]. Arbetet är uppdelat i veckolånga sprinter, där varje sprint inleds med ett veckomöte och ett sprintplaneringspass. Under mötet kommer förändringarna från föregående sprint att gås igenom, följt av en diskussion om vad som fungerade bra och vad som kan förbättras i efterföljande sprint. Under varje möte sätts flera uppgifter och mål för veckan som börjar direkt efter mötet.

### 3.2 Scrum

Scrum beskrivs ofta som ett agilt ramverk för projektledning. Scrum-metoden tillämpas genom att man arrangerar möten, bestämmer vilka roller som används, tilldelar dessa roller till gruppmedlemmar och bestämmer vilka verktyg som ska användas på projektet [25].

För att göra det lättare att sätta kortsiktiga mål för varje uppgift, valde paret Trello-bordet som Scrum Board. Trello har använts under projektet för att underlätta för gruppen att följa processen mer detaljerade och organiserade. Under hela projektet har paret använt agila arbetsmetoder. Varje vecka är en ny sprint, där Trello-tabellen uppdateras och nya mål definieras efter samråd med företagets handledare. Veckans arbete kommer att skrivas under avsnittet "Att göra" och sedan skickas till "Pågår", "Test" och "Kodgranskning" för att kontrollera att koden passar företagets behov och önskemål, och sist till "Klart" när den anses vara helt färdig.



Figur 3.1: Illustration av Trello tabellen. Den visar alla steg för ett veckomål från Todo till Done

### **3.3 Intervju**

En intervju är en metod för att samla in information. Intervjufrågor som är väl utformade kan hjälpa till att mer data och information fås från individen som blir intervjuade. Beroende på syftet med intervjun används olika typer av intervjuer [26].

Under designprocessen av detta projekt används intervjumetoden. Under ett fysiskt möte, genomförs intervjuer för att diskutera prototypens användarvänlighet ur användarens perspektiv. Paret har valt en semistrukturerad intervju som är en typ av intervju med både slutna och öppna frågor [26].

### **3.4 Observation**

Observation är en metod som kan användas under utvecklingsprocessen av en produkt. Denna metod hjälper till att utvärdera en prototyp och analysera hur väl prototypen är designad. Denna metod innebär att vissa uppgifter utförs av vissa individer och deras beteende och reflektion över uppgifterna observeras och analyseras [26].

I detta projekt användes observationsmetoden för att välja den slutliga designen av prototypen. 5 uppgifter skapades för individer att genomföra. Individernas beteende observerades och deras kommentarer noterades under tiden de utförde uppgifterna. Resultaten av denna metod hjälpte paret med utvärdering och datainsamling under denna process.

### **3.5 Användarvänlighet**

Den slutliga designen av prototypen analyserades utifrån Nielsens designprinciper i detta projekt. Dessa principer hjälper till att skapa en användarvänlig design som prioriterar användarupplevelsen [27]. Paret utvärderade den slutliga designen av prototypen och säkerställde att den är användarvänlig utifrån Nielsens tio designprinciperna.

## 4 Genomförande

Det här avsnittet beskriver projektets designprocess, undersökningen av SignalR och implementeringen av chattsystemet.

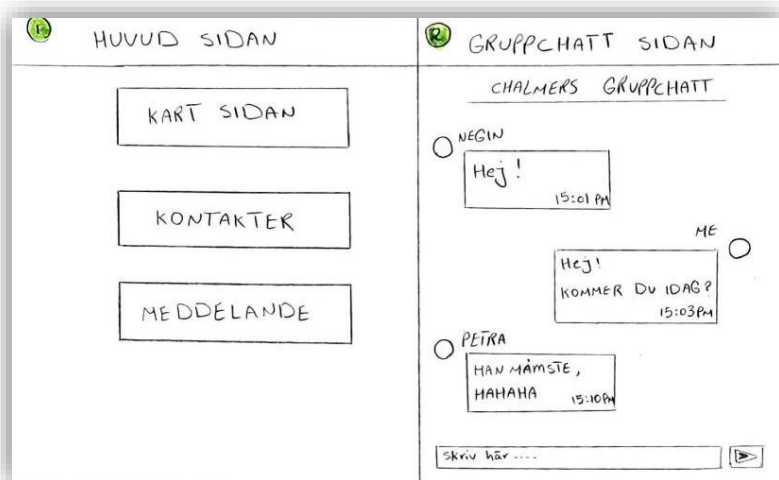
### 4.1 Design

Detta avsnitt visar designprocessen för detta projekt samt de metoder som har använts under tiden.

Processen började med att bestämma hur och var chattsystemet skulle integreras i prototypen. Det är viktigt för Elicit att chattsystemet tillfredsställer kundernas behov och det ska vara begripligt för användarna.

#### 4.1.1 Primära design

Innan chattsidans designprocess, började projektet med några möten med företagets handledare. Målet var att få en bättre förståelse för företagets behov och att få nya idéer innan designprocessen påbörjas. Slutligen valdes en konventionell chattsida, liknande chattsidan på mobiltelefoner. Denna sida borde passa in i prototypens övergripande tema såsom färger och textfont. Designprocessen började med wireframing så att paret kunde få en bild av sina idéer. Eftersom användbarheten av prototypen hade hög prioritet i detta projekt valdes en grundläggande sidlayout för att underlätta användarens användning. Pappers- och penna metoden valdes för detta projekt för att ge paret möjlighet att implementera nya idéer, upptäcka designfel och diskutera förbättringsförslag på ett enkelt sätt. En användarvänlig version av chattsidan skissades i pappersform. En illustration av chattsidan i pappersform visas i figur 4.1 nedan.



Figur 4.1: En illustration av skisser med hjälp av pappers- och penna metod.  
Huvudsidan, med en ny meddelandeknapp, är nummer 1. Nummer 2 visar en konversation i chattsidan.

Inloggningssidan och registreringssidan skissades på samma sätt. Pappers- och penna metoden användes igen för att skissa de två sidorna. En grundläggande, formell och färglös layout valdes för de två sidorna vilket paret kom överens om med företaget. En formell design för inloggningssidan och registreringssidan känns mer bekvämt och bekant för användarna, vilket förbättrar prototypens användbarhet. Figur 4.2 visar skisserna av inloggnings- och registreringssidorna i pappersform.

INLOGGNINGS SIDAN "LOG-IN"	REGISTRERINGSSIDAN "SIGN-UP"
<p>Log-in</p> <p>E-post:</p> <input type="text" value="skriv här..."/> <p>Lösenord:</p> <input type="text" value="skriv här..."/> <p>Log in</p>	<p>Sign-up</p> <p>Ange namn:</p> <input type="text"/> <p>Ange email:</p> <input type="text"/> <p>Ange telefonnummer:</p> <input type="text"/> <p>Ange adress:</p> <input type="text"/> <p>Ange postnummer:</p> <input type="text"/> <p>Ange ort:</p> <input type="text"/> <p><input type="checkbox"/> Privatperson</p> <p><input type="checkbox"/> Företag</p> <p>Skapa</p>

Figur 4.2: En illustration av inloggningssidan och registreringssidan med hjälp av pappers- och penna metoden.

#### 4.1.2 Designskiss enligt Pindle-prototypens tema

Miro användes i detta steg av designprocessen för att skapa chattsidan så nära en verklig applikation som möjligt. Paret fick en tydligare bild av den verkliga versionen av pappersformdesignen genom att använda olika komponenter i Miro. Miro och dess egenskaper studerades under några dagar för att få ett bättre grepp om hur Miro fungerar. Med hjälp av Miro, pappersformskisser och nya idéer skissades en grundläggande digital design för chattsidan. På designen skrevs några kommentarer i enlighet med prototypens designtema.

För att behålla den allmänna designen ändrades färger, text typsnitten och även form och färg på knapparna för att matcha sidan med de andra sidorna i prototypen. Se figur 4.3 för en illustration av designskissen.



Figur 4.3: En illustration av Gruppchatt sidan i Miro med kommentarer för Mer detaljerade design enligt prototypens tema

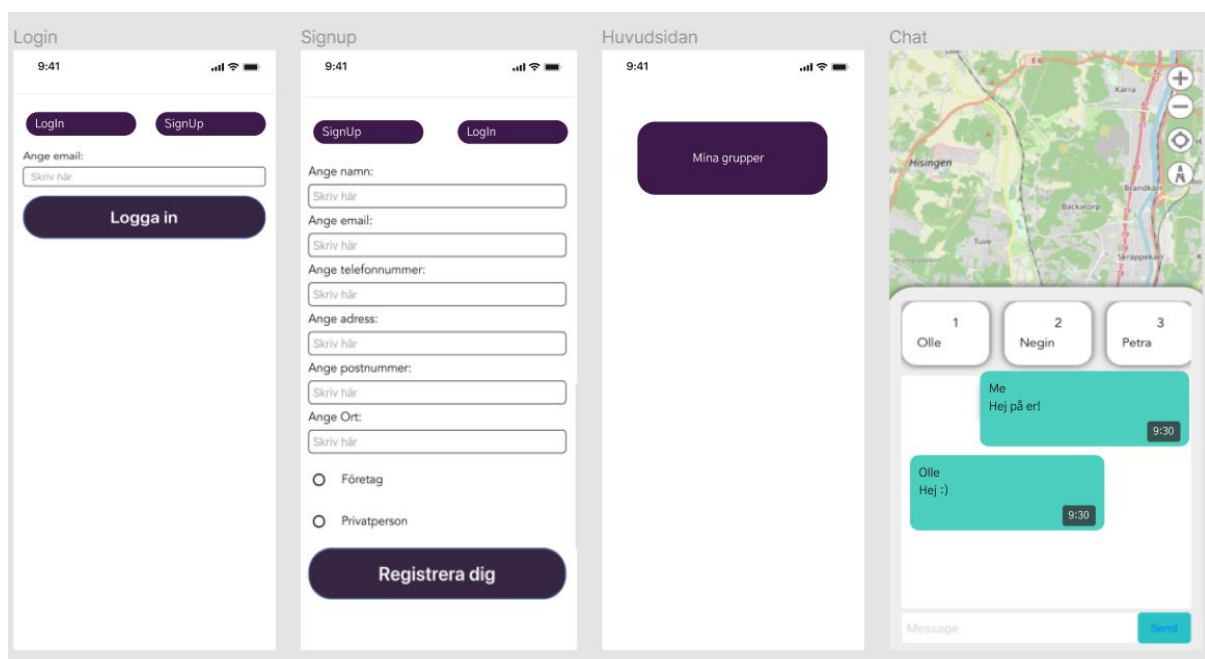
### 4.1.3 Slutlig design

Prototypframställning var den sista fasen i designprocessen. Paret valde den slutliga designen för sidorna och skapade en prototyp i detta steg. Två användartester användes för att utvärdera prototypens användarvänlighet. Det finns olika metoder för att tillämpa användartester. Intervjumetoden och observationsmetoden användes för detta projekt för att utvärdera designen av sidorna. Paret kunde bedöma designen ur användarens perspektiv genom att genomföra intervjuer och observera användarens reflektion kring prototypen genom att be personer att komplettera några uppgifter i prototypen. Med hjälp av de uppgifterna kunde paret analysera användarens reflektioner över användning av prototypen. Sidornas slutliga design diskuterades under ett möte med Elicits UX-designer. Detta möte hjälpte paret med att analysera och förbättra designen, samt implementera företagets preferenser för prototypen. På mötet kom företaget och paret överens om att ta bort lösenordsfältet från inloggningssidan. Borttagningen av lösenordsfältet understryker att detta är en prototyp och varnar användaren om databasens osäkerhet och sårbarhet. Dessutom ledde detta möte till att meddelande knappen togs bort av huvudsidan och gruppchattsidan flyttades till kartsidan som är en redan befintlig sida i prototypen.

Kartsidan i prototypen har olika grupper som innehåller sina egna medlemmar och deras information som adress och namn samt andra faktorer. Denna sida innehåller ett tomt utrymme under en lista med personer som är med i gruppen. Detta lediga utrymme valdes för implementeringen av chattsidan. På så sätt valdes den slutliga layouten för gruppchattsidan,

inloggningssidan och registreringssidan och en prototyp skapades, som visas i figur 4.4. Figma användes för att skapa prototypen och tillämpa uppgifter och intervjufrågor.

För att göra prototypen bekant för användarens ögon och enkel att använda behövdes inga färger eller knappar för att matcha prototypens tema. En enkel och traditionell design av chattsidan, inloggningssidan och registreringssidan valdes för prototypen. Ett möte med Elicits handledare och arbetsgruppen planerades för att utföra vissa uppgifter och ställa intervjufrågor för att analysera prototypen. Enligt svar på intervjufrågorna och reflektioner kring prototypens uppgifter var sidornas design tydliga, begripliga och användarvänliga. Användartester hjälpte paret att välja en användarvänlig slutlig design för dessa sidor. Om nya designidéer uppstår under implementeringen bör de analyseras och diskuteras ur användarens perspektiv innan de implementeras.



Figur 4.4: En illustration av prototypen i Figma

## 4.2 SignalR

Att välja rätt ramverk har stor betydelse inom utveckling av applikationer. Rätt ramverk kan medföra att slutprodukten bli snabb, skalbar pålitlig och säkert. Oavsett om produkten är en webbplats eller en mobilapplikation, ger ramverk en utgångspunkt för att lösa vanliga problem, integrera bästa praxis för programmering och hålla koll på de senaste tekniska alternativen i dagens marknad. Med syftet på att säkerställa om SignalR är ett lämpligt bibliotek till chattsystemet, utfördes en undersökning. Undersökningen genomfördes i form av sökning på nätet och utredning av tidigare chattapplikationer. En faktor som hade stor betydelse i undersökningen var den redan befintliga Pindle-prototypen som utvecklades i Xamarin.Forms och programmerades i .NET och C#. Med hänsyn till alla omständigheter, resulterade undersökningen att SignalR är ett lämpligt bibliotek för detta chattsystem.

SignalR tillåter användaren kommunicera i realtid, vilket är en enorm viktig funktion för ett chattsystem. Realtidsfunktionalitet möjliggör att skicka data från serverkod till anslutna klienter så fort datan blir tillgängligt. Detta befriar klienterna från att göra polling på servern upprepade gånger, samt låta servern vänta på att en klient ska begära nya data. SignalR tar hand om anslutningshantering automatiskt och ger användaren möjligheten att skicka meddelande till specifika personer eller till alla anslutna klienter. Anslutningen i SignalR är beständig till skillnad från den klassiska HTTP-Anslutning där kommunikation mellan server och klient återupprättas [8].

SignalR använder websocket teknik för att skicka data. Websocket är den optimala transporten för SignalR. Anledningen till detta är att den bland annat har den mest effektiva användningen av serverminne och mest underliggande funktionalitet som till exempel full duplexkommunikation mellan server och klient [8].

SignalR använder sig av två olika kommunikationssätt för att kommunicera mellan klient och server. Det första är beständiga anslutningar (Persistent Connections) som ger direkt åtkomst till ett lågnivåkommunikationsprotokoll, där varje anslutning till server identifieras av ett anslutnings-ID. Andra sättet är användning av Hubs. Hubs tillhandhåller ett API på hög nivå som möjliggör att metoder på klientssida kan anropa metoder på serversida och vice versa. Det är möjligt att skapa kommunikation antingen med Hub eller en beständig anslutning, men SignalR-dokumentation rekommenderar Hub.

## 4.3 Konfiguration av SignalR

Detta avsnitt beskriver implementering av ChatHub och klient samt en översiktlig redogörelse på implementering av chattsidan, inloggningssidan och registreringssidan.


Installationen av databasen och den redan befintliga Pindle-prototypen var det första steget efter designprocessen och undersökningen av SignalR. I början av implementeringsfasen var det en utmaning att få den befintliga Pindle-prototypen att fungera som bas för detta arbete och koppla den till servern. För att få kopplingen och kommunikationen mellan de två programmen att fungera, krävdes flera installationer. Paret hade en kämpig start, som löstes efter flera undersökningar och försök.

### 4.3.1 Skapa Hub

Hub är backend-tjänsten som tillåter användaren att skicka och ta emot meddelande. Dess huvuduppgift är att hantera server-klientkommunikation. Klassen ChatHub ärvs från SignalR Hub-klassen och hanterar anslutningar, grupper och meddelanden. Denna klass innehåller en metod som är döpt till *SendMessage*. Denna metod tar 3 parametrar, *chatId* som är en unik siffra på varje gruppchatt, *email* som är mailadress av sändare och *message* som är innehåll av meddelandet. *SendMessage* hanterar alla meddelanden som den tar emot och sedan skickar



meddelanden till alla klienter som aktivt lyssnar på nyckeln *"ReceiveMessage"*. Figuren 4.5 visar en illustration av metoden.



```
1 reference
public class ChatHub : Hub
{
    0 references
    public Task SendMessage(string chatId, string email, string message) =>
        Clients.Others.SendAsync("ReceiveMessage", chatId, email, message);
}
```


Figur 4.5: En bild på *SendMessage*-metod i *ChatHub*

## 4.3.2 Skapa Klient

För integration av *ChatHub* och klient, skapades en service i applikationen. Denna klass döptes till *ChatService.cs* och dess uppgifter är bland annat att ta hand om anslutningen, skicka och ta emot meddelande från backend. Dessa tre är huvuduppgifter av denna klass som kommer att redovisas här nedan.

### 4.3.2.1 Skapa anslutning till Hub

Anslutning till *ChatHub* kan utföras på olika sätt. Det är möjligt att köra hubben lokalt och ansluta via IP-adress eller localhost tillsammans med ett portnummer, vilket användes under projektet. Ett annat alternativ är att konfigurera URL:en där applikationen ska ansluta. Den typen av anslutning kommer att användas senare, när prototypen är redo att ansluta till molnet. Här nedan redovisas hur en anslutning från applikationen till Hubben ser ut. Se figur 4.6 för en illustration av båda alternativen.



```
_hubConnection = new HubConnectionBuilder()
    // .WithUrl("{https://theUrl.com OR ip:port OR localhost:port} + "/chatHub")
    .WithUrl($"https://localhost:5006/chatHub")
    , (opts) =>
    {
        opts.HttpMessageHandlerFactory = (message) =>
        {
            if (message is HttpClientHandler clientHandler)
                // bypass SSL certificate
                clientHandler.ServerCertificateCustomValidationCallback +=
                    (sender, certificate, chain, sslPolicyErrors) => { return true; };
            return message;
        };
    })
    .Build();
```

Figur 4.6: Illustration av anslutning till *ChatHub*

#### 4.3.2.2 Skicka meddelande till Hub

För att kunna skicka meddelande till ChatHub skapades metoden *SendMessage* i *ChatService.cs*. Denna metod innehåller en nyckel som i detta fall är döpt till *"SendMessage"*. Metoden kommer att leta efter ett metodsnamn i ChatHub som överensstämmer med nyckeln. När Hubben tar emot meddelanden, skickar den vidare till alla användare som aktivt lyssnar på *"ReceiveMessage"*. Figur 4.7 visar en bild på hur metoden ser ut.

Meddelanden kan skickas via två olika operationer *SendAsync* och *InvokeAsync* som fungerar ungefär likadant. Den enda skillnaden är att *InvokeAsync* returnerar tillbaka ett svar som beskriver om operationen har lyckats eller misslyckats, men *SendAsync* returnerar inget tillbaka. Med tanken på att svaret inte var intressant i detta projekt, användes *SendAsync* men *InvokeAsync* kan vara användbart i andra situationer.

```
// Send message to the ChatHub
2 references
public async Task SendMessage (string chatId, string email, string message)
{
    await _hubConnection.SendAsync("SendMessage", chatId, email, message);
}
```

Figur 4.7: Illustration av *SendMessage*-metod i *ChatService*

#### 4.3.2.3 Ta emot meddelande från Hub

För att kunna ta emot meddelande från ChatHub, skapades en så kallad *On-method* i konstruktören i *ChatService*. Denna metod innehåller en nyckel som i detta fall är döpt till *"ReceiveMessage"*. Metoden använder sig av en annan hjälpmetod som heter *OnReceiveMessage*. Denna hjälpmetod har en viktig funktion som avgör vilka som ska ta emot meddelande. Det är viktigt att *"ReceiveMessage"* matchar nyckeln som finns i ChatHub, då kan man konstatera att denna klient lyssnar aktivt på alla meddelanden som skickas till ChatHub. Figur 4.8 visar hur en klient aktivt lyssnar på *"ReceiveMessage"*.

```
_hubConnection.On<string, string, string>("ReceiveMessage", (chatId, email, message) => OnReceiveMessage(chatId, email, message));
```

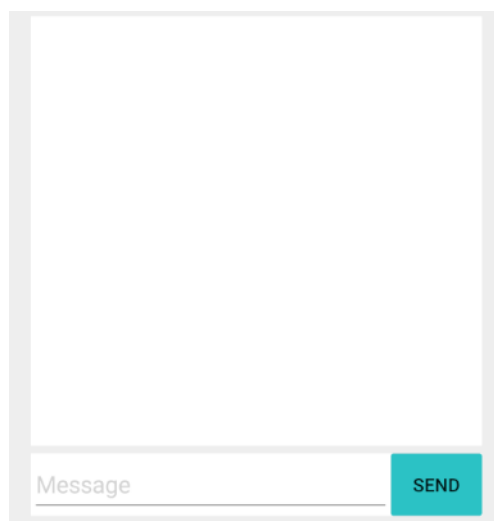
Figur 4.8: Illustration av *ReceiveMessage*-metod i *ChatService*

### 4.5 Implementation av chattsidan

Implementation av chattsidan börjades med en enkel kommunikationstest. Testet utfördes så att ett fast meddelande skulle skickas till servern och samma meddelande skulle fås tillbaka. Därmed kunde man verifiera att prototypen och ChatHubben kommunicerar. Med avsikt att visa meddelandet på skärmen, skapades en ny metod. Denna metod aktiveras via en knapp

som skickar ett meddelande till ChatHubben som i sin tur skickar tillbaka samma meddelande och slutligen visar det på skärmen.

Som första steg till implementation av chattsidan, skapades en rektangulär ram där alla meddelanden skulle synas. En textruta där man skriver meddelandet skapades under ramen och en knapp som är avsedd för att skicka meddelandet placerades bredvid textrutan. Alla komponenter i chattsidan skapades i XAML, men hantering av dessa genomförs via ViewModel:en. Figur 4.9 visar utseendet av den primära chattsidan.



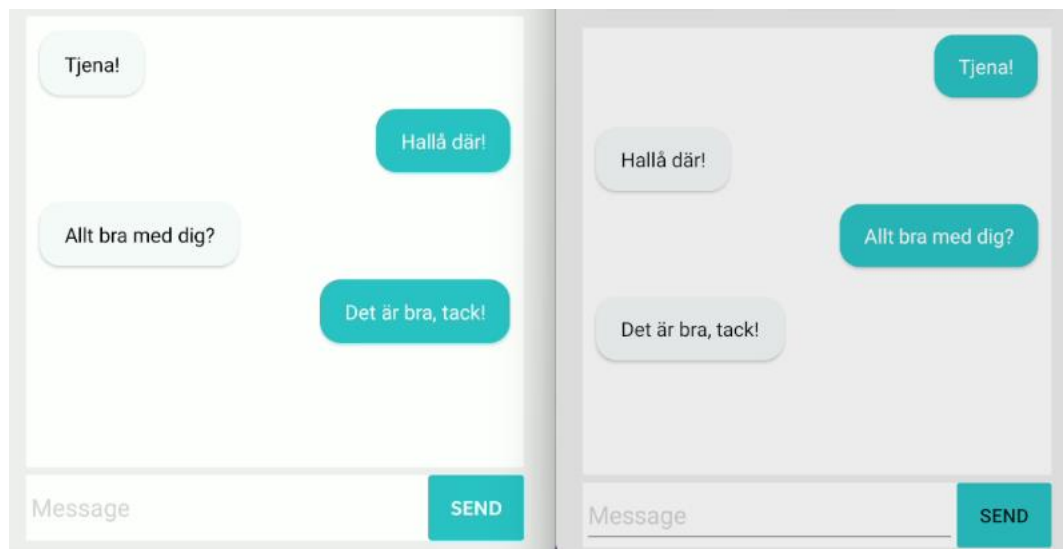
*Figur 4.9: Illustration av den primära chattsidan*

Nästa fas är återigen kommunikation mellan applikationen och servern. Här ska användaren kunna skriva ett godtyckligt meddelande som skickas till servern. När servern tar emot meddelandet, skickar den samma meddelande tillbaka till klienten. Alla meddelanden som skickas från enheten kommer att hamna i högersidan av ramen och allt som tas emot från ChatHubben hamnar i vänstersidan. Figur 4.10 visar en illustration av chattsidan i detta steg.



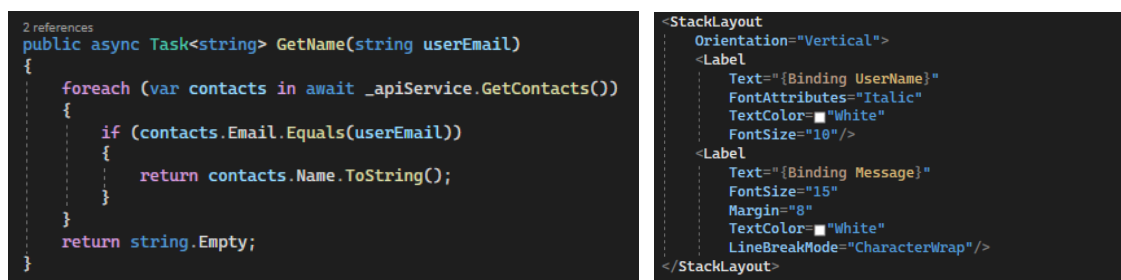
*Figur 4.10: Illustration av kommunikation mellan en klient och Chathubben*

Prototypen behövde modifieras så att klienten inte skulle få tillbaka sitt eget meddelande. Arbetet genomfördes genom att skicka meddelanden till alla klienter som är kopplade till servern förutom själva avsändaren. Figur 4.11 visar hur kommunikationen mellan klienter kan se ut. I nästa steg lades ytterligare funktioner till prototypen i avsikt att meddelanden ska erhållas av utvalda klienter, vilket i detta fall är alla som är med i samma gruppchatt. Metoden *ReceiveMessage* i *ChatService* använder chat-id för att avgöra vilka användare som ska ta emot meddelande.



Figur 4.11: Illustration av kommunikation mellan två olika klienter

I denna version av prototypen behövdes flera klienter eller emulatorer för att testa om prototypen fungerar enligt målen. Det är nästan omöjligt att köra många emulatorer samtidigt på en vanlig dator, då programmen kräver hög prestanda. På grund av dessa svårigheter valdes att lösa problemet genom att avaktivera Windows-brandvägg och låta servern köras i den enheten. Sedan kunde servern delas med andra och på så sätt kunde flera klienter använda samma server. Detta var en stor hjälp under projektet och hela testprocessen. Eftersom applikationen ska användas som en gruppchatt, är det viktigt att kunna identifiera avsändaren av varje meddelande. Detta hanterades genom att implementera en metod som hämtar avsändarens namn för varje meddelande. Metoden utnyttjar användarnas unika e-postadresser för att hitta avsändarens namn. Figur 4.12 visar en illustration av denna metod. Efter implementeringen av denna metod lyckades paret att skicka avsändarens namn med varje meddelande i chattsidan. Den slutliga versionen av chattsidan visas i Figur 4.13.



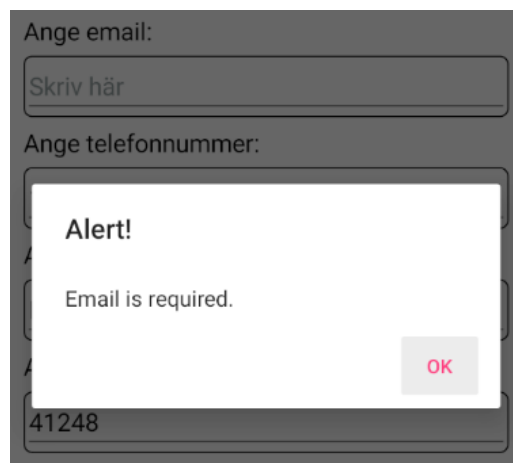
Figur 4.12: Vänstra bilden är en illustration av *GetName*-metoden där den hämtar avsändarens namn. Högra bilden är en illustration av XAML-delen för namn med meddelande.



Figur 4.13: Illustration av chattsidan där varje meddelande är skickat med avsändarens namn

## 4.6 Implementation av inloggningssida och registreringssida

En inloggnings- och registreringssida har implementerats för att avgöra vilken användare som har skickat ett meddelande. Paret kunde identifiera avsändaren av varje meddelande genom att tillåta användaren att skapa ett konto och sedan spara kontot i databasen. För de sidorna skulle en professionell och okomplicerad design väljas och detta skulle vara enkelt att förstå och använda för användaren. Paret programmerade de två sidorna och sedan implementerades felhantering på registreringssidan för den nödvändiga informationen. Felhanteringen varnar användaren för ofylld information på registreringssidan och tillåter inte användaren att skapa ett konto utan att fylla i alla fält vilket visas i figur 4.14.



Figur 4.14: Illustration av felmeddelande, när ett eller flera obligatoriska fält inte är ifyllda.

## 5 Resultat

Detta avsnitt innehåller en översikt över projektets resultat.

### 5.1 Slutresultat

Det är möjligt att köra prototypens databas direkt på webbplatsen genom att välja rätt IP-adress för databasen. Databasen kan köras genom att välja localhost som IP-adress eller datorns IP-adress. Olika klienter måste ansluta till samma server och dela samma databas för att kunna kommunicera med varandra i realtid. Som tidigare nämnts (se 4.3.2.1), en klient kan dela databasen med andra klienter genom att dela enhetens nätverk och IP-adress med andra klienter. På så sätt kan klienter komma åt en gemensam servern och kommunicera med varandra i realtid. Figur 5.1 visar ett exempel på kod vars IP-adress kan ändras manuellt och möjliggör anslutning till olika servrar.

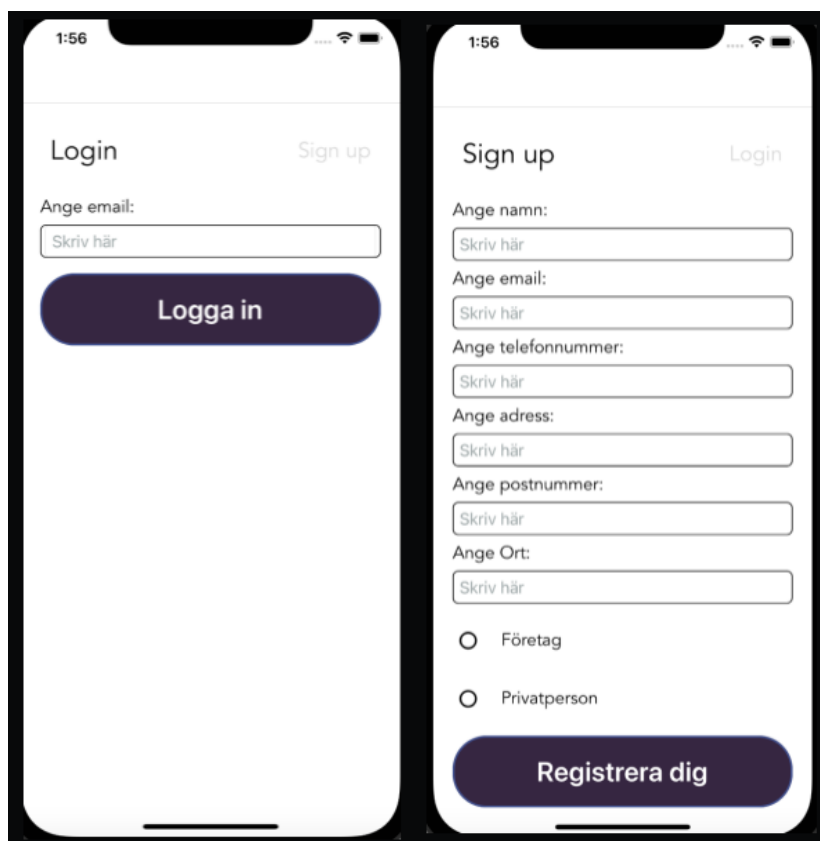
```
var requestUri = "https://192.168.43.189:5006/Contacts";
```

```
var connection = new signalR.HubConnectionBuilder().withUrl("https://localhost:5006/chatHub").build();
```

*Figur 5.1: Exempel på kod i både klienten och databasen där utvecklaren kan manuellt ändra IP-adress och ansluta till olika servrar.*

För att använda prototypen måste den vara ansluten till en server. Därefter kommer användaren till inloggningssidan där e-post fältet behöver fyllas i innan man kan gå vidare till prototypens huvudsida. När en användare inte har ett konto, finns möjligheten att skapa ett konto via registreringssidan som nås genom att klicka på knappen "Sign up".

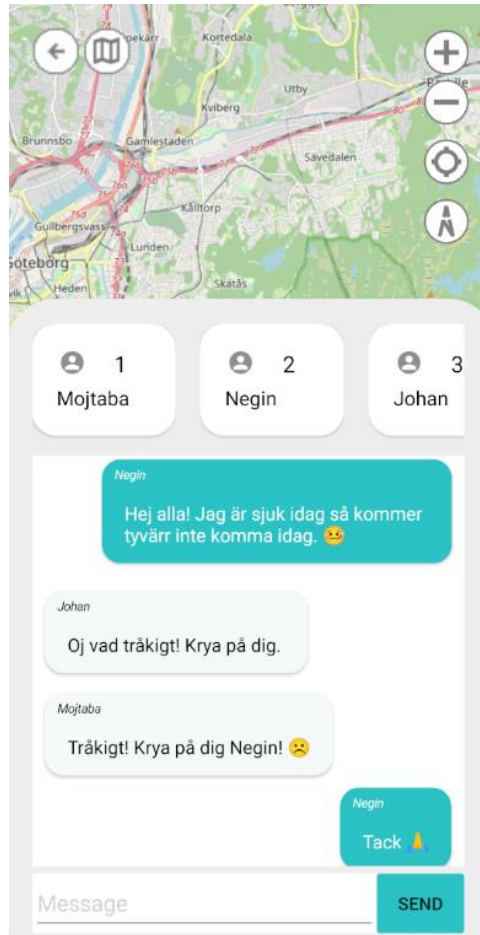
Registreringssidan har sex fält med obligatorisk information. Två klickbara rutor har lagts till så att användare kan välja om den är ett företag eller en privatperson. Om obligatoriska fält är ofyllda i någon av inloggnings- eller registreringssidan, skickas ett varningsmeddelande som visar vilka fält måste fyllas i. Inloggningssidan och registreringssidan illustreras i figur 5.2.



*Figur 5.2: Illustration av slutliga inloggningssidan och registreringssidan*

Efter inloggningssidan, har användaren tillgång till grupperna som den är medlem i. Varje grupp har en chatsida som tillåter realtidskommunikation mellan de användare som är med i gruppen. När en klient skickar ett meddelande, tas det emot av alla andra klienter i gruppchatten. Användarens namn visas bredvid meddelandet för att göra det uppenbart vem som har skickat det. Denna funktion gör att flera klienter kan chatta med varandra i realtid i en gruppchatt.

Figur 5.3 i nästa sida visar den slutliga gruppchattsidan. Bilden är en illustration av konversationen mellan tre användare i en gemensam gruppchattsida. Namnet på varje avsändare syns ovanför varje meddelande.



Figur 5.3: Illustration av den slutliga gruppchattssidan där flera användaren kan chatta med varandra i realtid



## 6 Diskussion

Detta avsnitt innehåller en analys av projektets mål och några rekommendationer för framtida arbete med denna prototyp. Sedan kommer projektets miljöpåverkan och etiska frågor att diskuteras.

### 6.1 Målanalys

Det första delmålet, "designa en gruppchattsida, inloggningssida och registreringssida som prioriterar användarupplevelsen", har uppnåtts. För dessa sidor användes en tydlig och begriplig design som tidigare nämnts (se 4.1.3), och genomfördes två användartester på prototyper för att ge bästa möjliga användarupplevelse. Intervjumetoden och observationsmetoden var de två användartester som användes. Intervjufrågorna och analys av användarnas reflektion av prototypen visades att sidornas design var användarvänlig. Prototypen skapades för att vara enkel att använda samtidigt som den tillhandahåller de väsentliga funktioner som behövs.

Det finns inga checklistor eller regler som definierar vad som är en bra och användarvänlig design [28]. Det finns flera designprinciper och teorier som kan användas under olika designprocesser. Eftersom en bra design inte kan uppnås enbart genom att tillämpa intervjuer och observationer, är bra designprinciper viktiga för att analysera den slutliga designen. Paret har valt att jämföra den slutliga designen med Nielsens designprinciper för att säkerställa att den uppfyller kraven för ett användarvänligt gränssnitt. Expert i ämnet användbarhet, Jakob Nielsen, ger tio designprinciper för en bra design [27].

Följande är Nielsens tio designprinciperna [27]:

1. Synlig systemstatus
2. System överensstämmer med den verkliga världen
3. Användarkontroll och frihet
4. Konsekvens och standard
5. Förhindra fel
6. Igenkänning är bättre än minne
7. Flexibelt och effektivt att använda
8. Estetiskt och minimalistisk design
9. Hjälp användarna att känna igen, diagnosera och återhämta sig från fel
10. Hjälp och dokumentation

Nielsens 10 principer användes för att utvärdera prototypens design. Denna prototyp uppfyller åtta av de tio principerna, vilket visar att den uppfyller en stor del av kraven på en användarvänlig design. Följande är de 8 punkterna:

1. Synlig systemstatus: Användaren behöver inte vänta länge på att något ska hända i denna prototyp. När varje knapp trycks in tas användaren direkt till nästa sida i prototypen.
2. System överensstämmer med den verkliga världen: Prototypen kan översättas till två språk, engelska och svenska, vilket kommer att tilltala en bredare publik. Samtidigt följer prototypen användarens mentala mönster. Till exempel, efter att användaren har tryckt på en knapp, händer det som användaren förväntar sig.
3. Användarkontroll och frihet: I denna prototyp har användaren möjlighet att ångra sig och kunna göra om. Det finns knappar som tar användaren tillbaka till huvudsidan eller föregående sida och låter användaren att ångra sig och kunna göra om.
4. Konsekvens och standard: Korta och informativa ord har valts för knappar och fält i denna prototyp, vilket gör att användaren inte behöver fundera över ord i olika situationer.
5. Förhindra fel: Genom att ta bort lösenordsfältet i inloggningssidan har paret varnat användaren om databasens sårbarhet och hindrat användaren från att göra något fel.
6. Igenkänning är bättre än minne: I prototypen har samma mönster följts på de flesta sidor för att avlasta användarens minne och betona alla viktiga steg med texter och färger.
7. Estetiskt och minimalistisk design: Det finns inga irrelevanta bilder eller texter som skulle förvirra användaren i utformningen av denna prototyp.
8. Hjälp användarna att känna igen, diagnosera och återhämta sig från fel: I denna prototyp har felhantering implementerats. Detta felmeddelande är tillräckligt tydligt och informativt för att användaren ska förstå och kunna göra rätt. Detta meddelande kan även översättas till både engelska och svenska språk.

Prototypens design utvärderades med Nielsens designprinciper, vilket visade att den är en användarvänlig design. Dessutom har resultatet av användartesterna visat att prototypens design är förståelig och användarvänlig. All insamlade data och utvärderingars resultat visar att prototypens design är användarvänlig och att detta delmål har uppnåtts.

Det andra delmålet, ”undersöka biblioteket SignalR för att säkerställa om det är ett lämpligt bibliotek för att implementera ett chattsystem som låter användaren kommunicera i realtid” har uppnåtts. Undersökningen av biblioteket SignalR visade att det omfattar viktiga funktioner som ett chattsystem bör innehålla. Biblioteket SignalR passar in i Xamarin.Forms-

miljö och kan utvecklas i .Net och C# vilket används i den redan befintliga Pindle-prototypen. Dessutom har biblioteket SignalR används under implementation av Pindle-prototypens chattsystem. Resultatet av detta projekt visar att användning av SignalR möjliggör realtidskommunikation.

Projektets sista delmål var att skapa en prototyp som låter användare chatta i realtid samtidigt som de kan identifiera sig genom att logga in med sin e-postadress eller skapa ett konto. Detta mål har uppnåtts och den primära funktionen för dessa sidor har implementerats och dess funktionalitet har testats och godkänts av företaget Elicit. Användaren kan chatta i realtid med andra användare i samma gruppchatt, samt identifiera sig och skapa ett konto i resultatprototypen.

## **6.2 Några rekommendationer för framtida arbete**

En del av resultatet är fortfarande inte fullständiga och det finns utvecklingsmöjligheter. En nödvändighet för framtida arbete med denna prototyp är att inkludera ett lösenord i inloggningssidan, så att varje användare kan logga in med sitt eget lösenord. Eftersom ingen lösenordskryptering har använts, är prototypen för närvarande osäker. Databasen är osäker eftersom allt sparas direkt och alla med databasåtkomst kan se all information och i värsta fall utnyttja de på ett negativt sätt. Genom att använda lösenord krypteringsalgoritmer eller hash-funktioner kan användarens lösenord skyddas.

En annan möjlighet för framtida arbete är chattsystemets databas, som kan vidareutvecklas. Att spara meddelanden i databasen och låta användaren se gamla meddelanden i varje gruppchatt kan vara en funktion som prototypen saknar. Denna funktionalitet ledde till en lång diskussion i Elicit mellan de ansvariga för prototypen. Ur företagets synvinkel var det inte självklart att funktionaliteten skulle implementeras i denna prototyp, med andra ord användbarheten av denna funktion var låg för företaget. Att spara meddelande i chattsidan var låg prioritet, och det tilldelades aldrig som en uppgift för paret.

Ett annat förslag till framtida arbete är att inkludera en privat chattsida. Detta arbete har gett hög prioritet åt gruppchattsidan i en och samma grupp, men att lägga till möjligheten att ha en privat chattkonversation kan vara en användbar förbättring för denna prototyp. Via en privat chattsida kan användare välja själv om dens information ska delas privat eller i en grupp med andra användare. Detta kan förbättra användarupplevelse av prototypen vilket är viktigt för Elicit.

## 6.3 Etiska frågor och ingenjörens ansvar

Flera etiska ramverk kan användas för att diskutera effekterna av ett chattsystem. Ur olika etiska perspektiv kan ingenjörens ansvar, företagets plikt och produktens säkerhet tolkas olika. Den vanligaste och mest känsliga oron i ett chattsystem är användarens information. Detta projekts chattsystem är en chattsida utan begränsningar. Olika användare kan skriva vad de vill och i värsta fall publicera känsliga personliga information i chattsidan. Sådana meddelanden kan få allvarliga konsekvenser om de publiceras på en chattsida. Projektets chattsystem har ett specifikt syfte och tillämpningsområde. Syftet är att tillåta användare att kommunicera med andra i realtid i en gruppchattsida. Inga känsliga eller privata information behöver delas mellan användare i prototypens chattsida eftersom användningsområde är tydligt. Ansvaret ligger i användarnas händer att inte fritt dela privata information i chattsidan.

Den andra etiska frågan som kan diskuteras är hackning och sårbarhet i ett chattsystem. Om systemet hackas, kan användarnas information utnyttjas på många negativa sätt. Denna fara kan orsaka förödande effekter för ett företags verksamhet såväl som för dess användares integritet. Ingenjören och företagen har en moralisk skyldighet att tillhandahålla en säker produkt, vilket kommer att beaktas vid användning av den färdiga applikationen av denna prototyp.

Annonser som tillhandahålls av specifika företag är ett annat etiskt problem. Vissa företag kan använda en chattsida för att skicka erbjudanden eller annonser till användare utan att be om tillåtelse och samtycke. Detta kan vara ett besvärande problem som kan få användaren att överge applikationen. Detta är inget problem i denna prototyp eftersom syftet med denna prototyp är kommunikation i en gruppchatt och endast aktuella information om den gruppen bör delas. Elicit har inga planer eller avsikter att publicera reklam eller erbjudanden genom denna prototyp, följaktligen är det användarnas skyldighet att använda denna prototyp för ett korrekt och lämpligt ändamål.

## 6.4 Miljöpåverkan

Ny teknik kan ha negativa miljökonsekvenser, men dessa konsekvenser kan undvikas om alla företag och ingenjörer tar miljöansvar. Varje företag bör utvärdera och välja det mest ekologiskt vänliga alternativet för varje ny teknik innan projektet startar. Realtidschattsystem som är syftet med detta projekt är ett program utan direkt påverkan på miljön. Men detta chattsystem har implementerats på Pindle-prototypen som är en mobilapplikation och kommer att användas i mobiltelefoner. Mobiltelefoner kan vara skadliga för miljön och dess konsekvenser bör diskuteras utifrån miljöaspekter. Ingenjörer kan ha positiva inverkan på miljön genom att välja att samarbeta med företag som producerar miljövänliga produkter och arbetar på ett hållbart sätt. Alla ingenjörer på arbetsplatsen kan minska negativa

miljöpåverkan genom att försöka utveckla miljövänliga och hållbara lösningar för varje projekt.

Detta företag har ett CSR-projekt utöver sina projekt med syfte att samla in matrester som bara är till nytta för miljön och samhället, vilket är riktigt konstruktivt. Insamling av matrester gynnar miljön och minimera hungriga människor i samhället därför sådana projekten bör lovordas och olika företag bör uppmuntras att delta.

## 7 Slutsats

Detta projekt förbättrade parets kunskap inom design, programmering och projektplanering. Om ett liknande projekt skulle utföras, skulle tillvägagångssättet planeras på ett annat sätt för att påskynda och förenkla processen. Under tiden kunde en bättre planering och användning av andra metoder resultera i en prototyp med bättre design och mer funktionalitet.

Avslutningsvis, arbetet resulterade i ett funktionellt chattsystem i realtid som tillåter användare att kommunicera med andra användare enbart i en och samma gruppchatt. För prototypen implementerades en inloggningssida och en registreringssida som hanterar kontoskapandet genom kommunikationen mellan prototypen och databasen.

Användargränssnitt för dessa sidor är baserat på interaktionsdesign och användartester, vilket gör de sidorna så användarvänliga som möjligt.

# Referenser

- [1] D. Stenberg, "Elicit it's about time," *Om Elicit*, Apr. 2020. [Online].  
Tillgängligt: <https://www.elicit.se/om-elicite>. [Hämtad: Apr. 11, 2022]
- [2] GeeksforGeeks.org, "Frontend vs Backend," Aug. 2021. [Online].  
Tillgängligt: <https://www.geeksforgeeks.org/frontend-vs-backend/>. [Hämtad: Maj. 03, 2022]
- [3] Techlib.se, "XAML." Nov. 2019. [Online].  
Tillgängligt: <https://techlib.se/definition/xaml.html>. [Hämtad: Apr. 10, 2022]
- [4] A. Johansson, "Interaktionsdesign," *Interaktionsdesign – definition, tips på utbildningar & jobb*, Mar.2020. [Online]. Tillgängligt: <https://andreasjohanssonux.se/interaktionsdesign/>. [Hämtad: Apr. 12, 2022]
- [5] Limetta.se, "Wireframe, designskiss och prototyp - vad är skillnaden?," Nov. 2019. [Online].  
Tillgängligt: [Wireframe, designskiss och prototyp - vad är skillnaden? | Limetta](https://limetta.se/wireframe-designskiss-och-prototyp-vad-ar-skillnaden/). [Hämtad: Apr. 10, 2022]
- [6] Microsoft.com, "What is .NET?," Maj. 2021. [Online].  
Tillgängligt: <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet>. [Hämtad: Apr. 15, 2022]
- [7] Microsoft.com, "A tour of the C# language," Mar. 2022. [Online].  
Tillgängligt: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>. [Hämtad: Apr. 15, 2022]
- [8] Microsoft.com, "Introduction to SignalR," Sep. 2020. [Online].  
Tillgängligt: <https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr>. [Hämtad: Maj. 03, 2022]
- [9] Microsoft.com, "The Model-View-ViewModel Pattern," Jul. 2021. [Online].  
Tillgängligt: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>. [Hämtad: Apr. 04, 2022]
- [10] E-Identitet.se, "Vad är API?," Mar. 2021. [Online].  
Tillgängligt: <https://e-identitet.se/news/vad-ar-api/>. [Hämtad: Apr. 15, 2022]
- [11] Oracle.com, "What Is a Database?," Sep. 2021. [Online].  
Tillgängligt: <https://www.oracle.com/database/what-is-database/>. [Hämtad: Apr. 18, 2022]
- [12] Mysql.com, "What is MySQL?," Feb. 2022. [Online].  
Tillgängligt: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>. [Hämtad: Apr. 18, 2022]
- [13] Microsoft.com, "SQL Server Management Studio (SSMS)," Maj. 2022. [Online].  
Tillgängligt: <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>. [Hämtad: Maj. 03, 2022]

- [14] Dbeaver.io, “About,” Dec. 2021. [Online].  
Tillgängligt: <https://dbeaver.io/about/> . [Hämtad: Apr. 18, 2022]
- [15] Docker.com, “Docker overview,” Feb. 2022. [Online].  
Tillgängligt: <https://docs.docker.com/get-started/overview/>. [Hämtad: Apr. 22, 2022]
- [16] Microsoft.com, “What is Xamarin.Forms?,” Jul. 2021. [Online].  
Tillgängligt: <https://docs.microsoft.com/en-gb/xamarin/get-started/what-is-xamarin-forms> .  
[Hämtad: Maj. 05, 2022]
- [17] Microsoft.com, “Code faster Work smarter,” Apr. 2022. [Online].  
Tillgängligt: <https://visualstudio.microsoft.com/vs/>. [Hämtad: Maj. 06, 2022]
- [18] Git-scm.com, “Git--distributed-even-if-your-workflow-isnt,” Oct. 2020. [Online].  
Tillgängligt: <https://git-scm.com>. [Hämtad: Apr. 29, 2022]
- [19] Dynatrace.com, “What is Azure DevOps?,” Mar. 2022. [Online].  
Tillgängligt: <https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/#free-trial>. [Hämtad: Apr. 27, 2022]
- [20] Microsoft.com, “Microsoft Teams,” Jan. 2022. [Online].  
Tillgängligt: <https://www.microsoft.com/sv-se/microsoft-teams/group-chat-software>[https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/- free-trial](https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/-free-trial) . [Hämtad: Maj. 03, 2022]
- [21] Discord.com, “Nybörjarguide till Discord,” Feb. 2021. [Online].  
Tillgängligt: <https://support.discord.com/hc/sv/articles/360045138571-Nyb%C3%B6rjarguide-till-Discord>[https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/- free-trial](https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/-free-trial). [Hämtad: Apr. 27, 2022]
- [22] Miro.com, “Where teams get work done,” Nov. 2021. [Online].  
Tillgängligt: <https://miro.com/index/>  
[https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/- free-trial](https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/-free-trial). [Hämtad: Maj. 07, 2022]
- [23] Figma.com, “A design platform built for scale,” Sep. 2021. [Online].  
Tillgängligt: <https://www.figma.com/?msclkid=98036e52cf0d11ec95e81ba0ea38f021>  
[https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/- free-trial](https://www.dynatrace.com/monitoring/technologies/azure-monitoring/azure-devops/-free-trial). [Hämtad: Maj. 14, 2022]
- [24] J. Steghöfer, Agile Software Project Management “A Primer to Agile Requirements Management,” Mar.2022. [Online].  
Tillgängligt: <https://chalmers.instructure.com/courses/15313/pages/course-pm-week-by-week-breakdown-and-deliverables> [Hämtad: Jun. 19, 2022]
- [25] K. Bitner, “The Home of Scrum,” WHAT IS SCRUM?, Maj.2022. [Online].  
Tillgängligt: <https://www.scrum.org/resources/what-is-scrum>. [Hämtad: Apr. 24, 2022]



- [26] J. Preece, Y. Rogers, H. Sharp, "*Interaction design: beyond human-computer interaction*," 5th ed., Indianapolis, IN: Wiley, 2019. [Online].  
Tillgängligt: <https://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=cat07472a&AN=clec.EBC5746446&site=eds-live&scope=site&custid=s3911979&authtype=sso> [Hämtad: Jun. 17, 2022]
- [27] J. Wilsson, "*Design av mobila gränssnitt genom operationalisering av användbarhetsprinciper*," Kandidatarbete, Linneuniversitet, 2012 [Online].  
Tillgängligt: <http://www.diva-portal.se/smash/get/diva2:534966/FULLTEXT01.pdf>  
[Hämtad: Jun. 18, 2022]
- [28] S. Majetic, A. Rexhepi, "*En studie av designprinciper: Använder företag specifika designprinciper vid utformning av användbara webbsidor?*," Kandidatarbete, Malmö Högskola, 2013 [Online].  
Tillgängligt: <https://www.diva-portal.org/smash/get/diva2:1480174/FULLTEXT01.pdf>  
[Hämtad: Jun. 18, 2022]

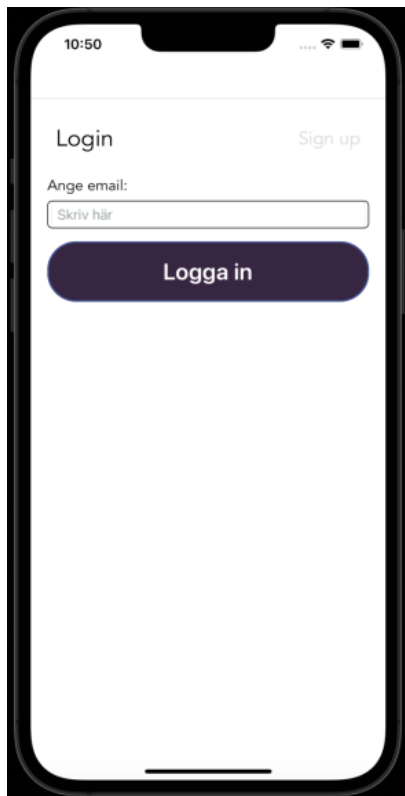
# Bilagor

## Intervjufrågor:

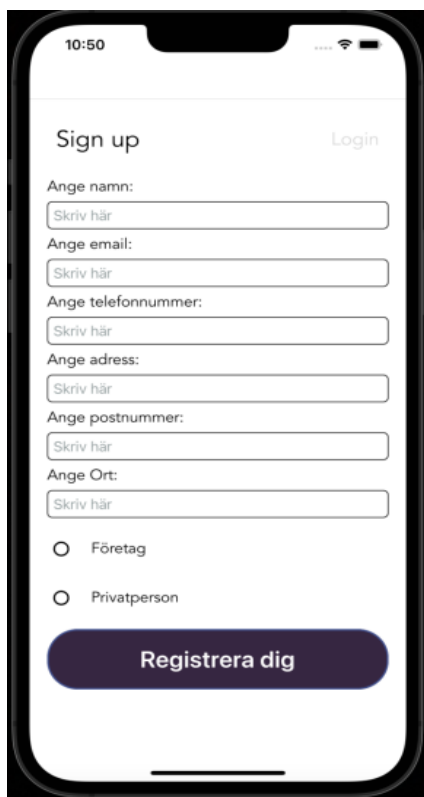
1. Är prototypens sidor tydliga och förståeliga?
2. Är prototypens navigering tydlig?
3. Är placeringen av fälten bra?
4. Är placeringen av knappar bra?
5. Gör knappar det som du förväntar dig?
6. Är texter på knappar tillräckligt informativa om vad de gör?
7. Hade du lätt förstått hur ska du ta dig till gruppchattsidan?
8. Är texter läsbara?
9. Är storlek på texter och knappar bra?
10. Är urvalet av färger bra?
11. Följer alla sidor prototypens tema?
12. Har du några förbättring förslag?
13. Anser du att prototypen är användarvänlig?

## Observationsmetod uppgifter:

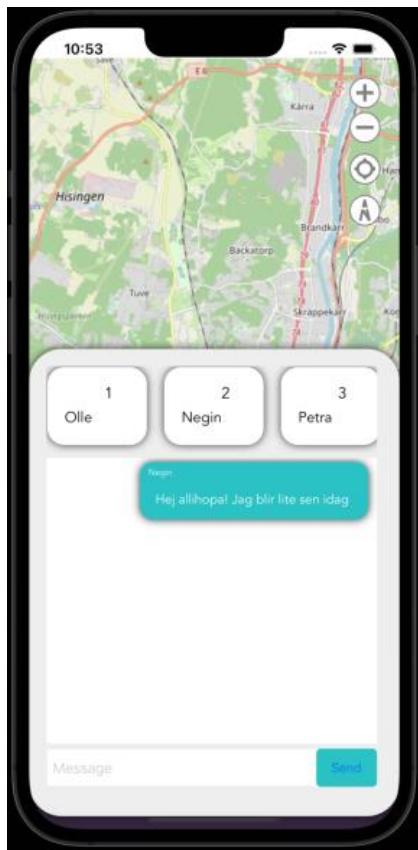
- Logga in dig i prototypen.
- Skapa ett konto i prototypen.
- Hitta kartsidan.
- Ta dig till chattsidan.
- Skicka ett meddelande i gruppen.



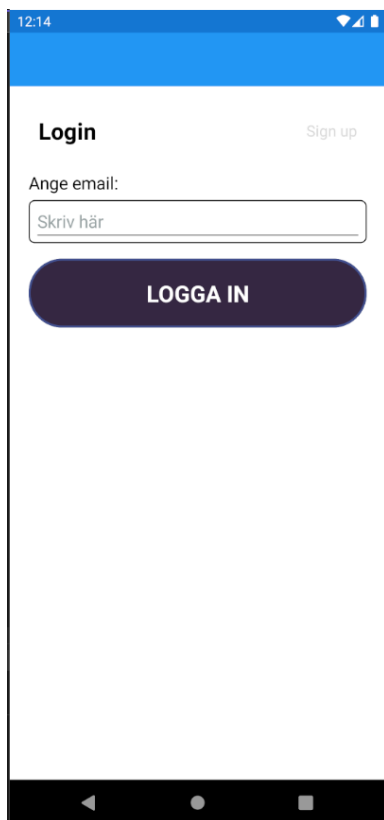
*Prototypens Inloggningssida i iOS emulator*



*Prototypens Registreringssida i iOS emulator*



*Prototypens Gruppchattsida i iOS emulator*



*Prototypens Inloggningssida i Android emulator*

12:25

**Sign up** [Login](#)

Ange namn:

Ange email:

Ange telefonnummer:

Ange adress:

Ange postnummer:

Ange Ort:

☐ Företag  
☐ Privatperson

**REGISTRERA DIG**

*Prototypens Registreringssida i Android emulator*

12:40

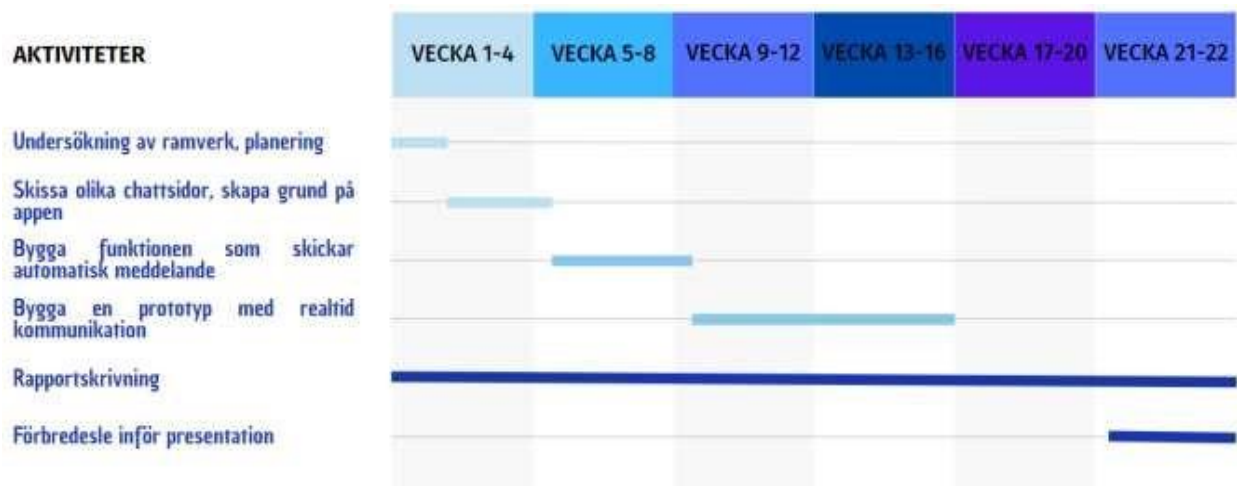
Partille-gr 82 E 2  
 79 70  
 Delsjöområdet naturreservat  
 75 75

1 Mojtaba  
 2 Johan

Mojtaba  
 Hej, tyvärr blir jag lite sen idag! 😞

Message **SEND**

*Prototypens Gruppchattsida i Android emulator*



*En illustration av tidsplan inför projektets process*



**CHALMERS**

