Funded by EU's Horizon 2020

D 2.3

VOIP Telephony and Messaging Architectures Specification

Supporting Active Ageing through Multimodal coaching

DISCLAIMER

This document reflects the opinion of the authors only and not the opinion of the European Commission. The European Commission is not responsible for any use that may be made of the information it contains.

AKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No.769661.

Supporting Active Ageing through Multimodal coaching

DELIVERABLE DOCUMENTATION SHEET

Deliverable:	*2.3*
WP №	*2*
Title:	*VOIP Telephony and Messaging Architecture Specifications*
Editor(s):	*Dimitar Skripkin*
Contributor(s):	*Dimo Kapnilov, Irina Stoyanova*
Type:	*Report/Documentation*
Version:	*V1.0*
Submission Due Date:	*30.09.18*
Dissemination level (CO/PU):	*Public*
Copyright:	*©SAAM Project - All rights reserved*

☒ Approved by the WP Leader

☒ Approved by the Technical/Exploitation Manager[1]

☒ Approved by the Coordinator

☒ Approved by the PSC

[1] Choose Technical Manager for Deliverables in WP1-7,10 and Exploitation Manager in WP 8-10

PUBLISHABLE SUMMARY:

The deliverable aims to describe the main functionalities and components of the VoIP and Messaging Module of the SAAM system, as well as the module's integration with the entire SAAM system.

The document is to be used as an initial baseline for the development team responsible for producing the module, but also as a placeholder for technical decisions taken during the implementation stage and the functional and non-functional requirements for the module groomed during the implementation, making it a live document to update periodically.

The document also includes definitions and explanations of the most crucial for the module technologies and tech terms, in order to make it more comprehensive for Junior Software Engineers.

QUALITY CONTROL ASSESSMENT SHEET

Version	Date	Comment	Name of author/reviewer/contributor
V0.1	*30.10.18*	*First Draft*	*Dimitar Skripkin, Dimo Kapnilov*
	16.11.18	*Contributions*	*Irina Stoyanova*
V0.2	*29.11.18*	*Second Draft*	*Dimitar Skripkin*
	30.11.18	*1st Peer review*[2]	*Martin Žnidaršič*
	30.11.18	*2nd Peer review*	*Bernard Ženko*
	01.12.18	*WP Leader approval*	
	03.12.18	*Coordinator approval*	
		EAB Review	*n.a.*
	10.12.2018	*PSC approval*	
V1.0	*10.12.2018*	*Submission to EC*	

HISTORY OF CHANGES

For updating the Deliverable after submission to the EC if applicable

Version	Date	Change
V2.0		

[2] Each Deliverable to have two experts (partners) assigned for Peer review. The experts should be different from the Experts, contributing to its creation. List of Deliverables and Peer review experts to be decided at Kick-off.

Supporting Active Ageing through Multimodal coaching

PROJECT DOCUMENTATION SHEET

Project Acronym:	*SAAM*
Project Full Title:	*Supporting Active Ageing through Multimodal coaching*
Grant Agreement:	*GA № 769661*
Call identifier:	*H2020-SC1-2017-CNECT-1*
Topic:	*Personalised coaching for well-being and care of people as they age*
Action:	*Research and Innovation Action*
Project Duration:	*36 months (1 October 2017 – 30 September 2020)*
Project Officer:	*Jose Albacete VALVERDE*
Coordinator:	*Balkan Institute for Labour and Social Policy (BILSP)*
Consortium partners:	*Jožef Stefan Institute (JSI)* *University of Edinburgh (UEDIN)* *Paris-Lodron Universität Salzburg (PLUS)* *Scale Focus AD (SCALE)* *Interactive Wear AG (IAW)* *Univerzitetni rehabilitacijski inštitut Republike Slovenije (SOČA)* *Nacionalna Katolicheska Federacia CARITAS Bulgaria (CARITAS)* *Bulgarian Red Cross (BRC)* *Eurag Osterreich (EURAG)*
website:	*saam2020.eu*
social media:	*#saam2020, #saamproject*

ABBREVIATIONS

SAAM	Supporting Active Ageing through Multimodal Coaching
NAT	Network Address Translation
STUN	Session Traversal Utilities for NAT
TURN	Traversal Using Relays around NAT
VoIP	Voice over Internet Protocol

CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

The Central VOIP solution will be integrated with the SAAM system to provide functionalities for notifying users by either sending messages or initiating calls. It will also provide the functionality to send automated notifications based on schedules or pre-defined thresholds. VOIP and messaging functionality is an important part of the coaching pipeline as it provides communication channels for reaching users' social circles and initiating communication between them. The VoIP and Messaging Solution of SAAM will be accessible for the Primary and Secondary users though a web-browser based interface and an android application on their mobile devices (mobile phone / tablet (with Android OS)) and PCs .

In their simplest form, Voice over IP protocols simply enable two (or more) devices to transmit and receive real-time audio traffic that allows their respective users to communicate. In general, VoIP architectures are partitioned in two main components: signalling and media transfer. Signalling covers both abstract notions, such as endpoint naming and addressing, and concrete protocol functions such as parameter negotiation, access control, billing, proxying, and NAT traversal. Depending on the architecture, quality of service (QoS) and device configuration/management may also be part of the signalling protocol (or protocol family). The media transfer aspect of VoIP systems generally includes a comparatively simpler protocol for encapsulating data, with support for multiple codecs and (often, but not always) content security.

SAAM project will have internal messaging system based on SignalR in order to add support for user to user and user to caregiver text communication. Based on SignalR we will also add support for user notifications and reminders, which will be used as a part of the coaching system.

2 VOIP AND MESSAGING COMMUNICATION ARCHITECTURE AND SCALABILITY

2.1 Architecture Overview

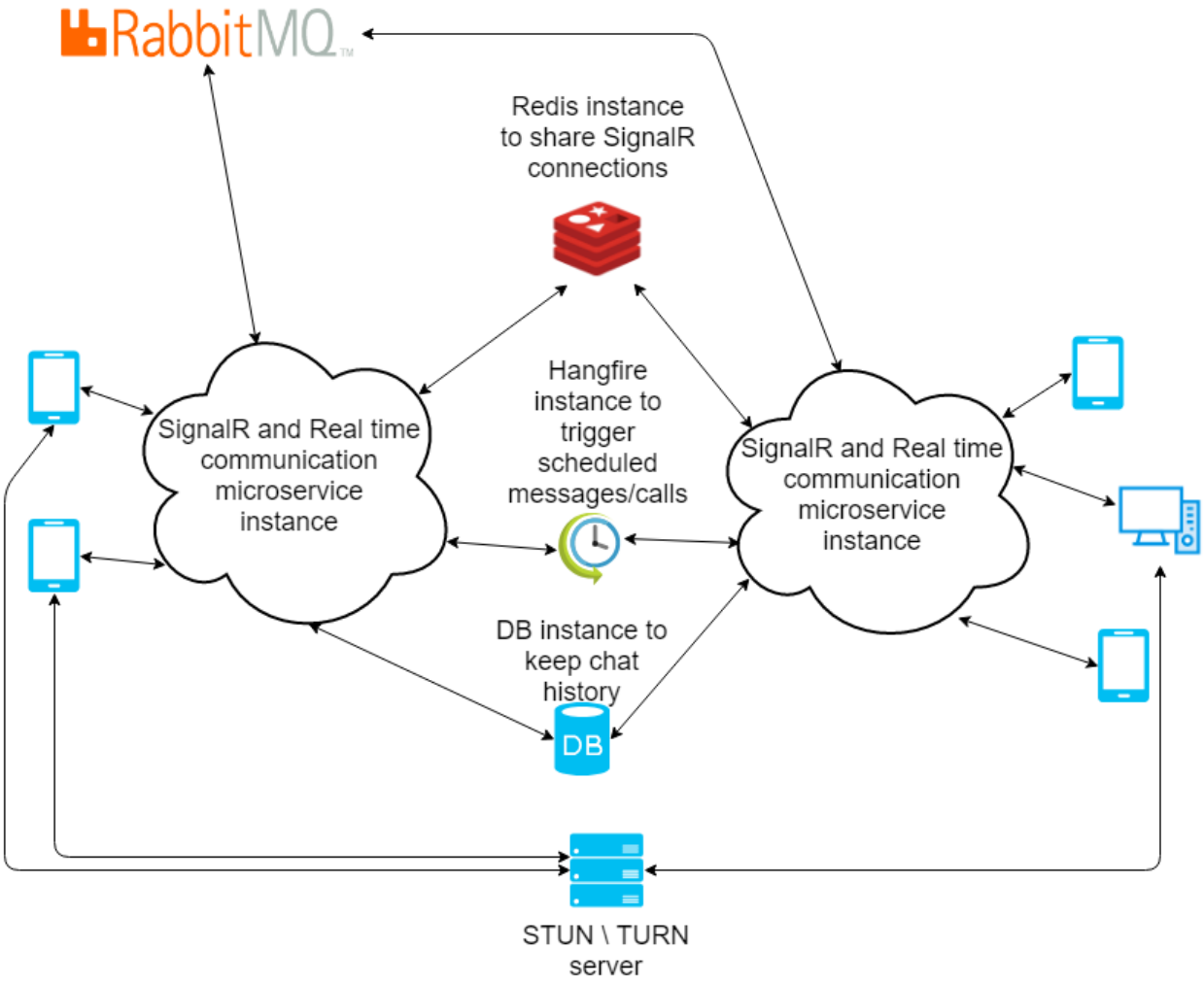

Figure 1 VoIP and Messaging Module Architecture

The current diagram in ***Error! Reference source not found.*** *depicts the communication architecture of he solution describing the VoIP, Massaging and Notification Functionalities*.

Every time the SAAM platform is being used by a logged-in user, the client will connect to SignalR chat hub with Microsoft Provided Libraries. The only requirement for the connection to be established is for the user to have internet access. The SignalR hub will relay the messages between the relevant platform users or system messages in order to establish VoIP connection when necessary and push a copy to

database so we can retrieve the chat history later. The SignalR will also be used to forward the messages (e.g. 'sent', 'delivered', read', etc. statuses) and user status information (e.g. 'on-line', off-line', etc.).

When a user logs on-line to the SAAM platform the chat history for the last few messages or for the last few days will be retrieved via a dedicated microservice- that is to say that the user will be able to see any unread messages or missed calls. In the context of SAAM coaching a message will be pushed to the RabbitMQ and read by a notification microservice and sent via SignalR to the proper user. For example: The system has detected that user has not left the apartment for two consecutive days and has triggered a message to members of her or his social circle.

We will use Redis to distribute messages across a SignalR application that is deployed on two separate instances – when the volume of users grows to a point where numerous SignalR Hubs are required for their management, this Redis functionality will allow the connection establishment between users handled by several separate SignalR Instances. Redis is an in-memory key-value store. It also supports a messaging system with a publish/subscribe model. The SignalR Redis backplane uses the pub/sub feature to forward messages to other SignalR hubs. The scheduled messages will be triggered from Hangfire instance. Hangfire is an open-source framework that helps you to create, process and manage your background jobs, i.e. operations you don't want to put in your request processing pipeline.

One of the functionalities the SAAM system is planning to provide to its users is a scheduler with a reminder option. E.g. a user has to take different medications with various regularity. The SAAM system can serve as a pill intake reminder and allow the user to schedule reoccurring reminders for different medications they have been prescribed. The notification portion of the service will be implemented via a notification microservice and sent through SignalR to the proper user. At this point, it still needs to be decided where the actual scheduling will be implemented.

The current Tech Stack decided for the Module is the following (in bold are the solutions specific to the current module and the rest, being part of the entire SAAM platform Tech Stack):

- **WebRTC;**

For the implementation of the module the primary criteria for choosing a solution were for it to be standardized and an Open Source. Standardized, means to be easy to implement through all major browsers, since the alternative to not using a standardized solution would be to script one form scratch or use dedicated custom plug-ins for each browser (WebRTC covers this criteria). With respect to being Open Source, currently on the market there's no open source alternative to WebRTC in a browser for the purposes of voice/video, chat/calling (e.g Red5 is an excellent solution, but it is subscription based). Another benefit of the chosen solution is that WebRTC has a vibrant and growing ecosystem making it well supported and fast developing technology.

- **TURN Server;**

The TURN server is part of the tech stack, because the WebRTC needs at least a STUN server to collect connection information and signaling server to initiate connection and send connection information. The TURN server provides all functionalities of the STUN but in addition allows for two clients which

cannot connect to each other directly to establish connection via rerouting the entire conversation through the solution.

- **SignalR;**

The top alternative considered to the chosen SignalR is Socket.IO. Both solutions are very similar (e.g both are Open Source, both support selective subscription fan-out, both form a logical connection between a browser and a pool of servers if sticky sessions are enabled). The tip of the scale while making the decision is based on the fact that our entire tech stack is based on C#, and the code base would be easier for maintenance. SignalR has also out of the box solution for authentication and scaling. The .Net Core is a multithreaded solution which leads to optimisation when using the solution for sending files, images etc. Additionally if in the future we decide to track the system for uncensored language and rude behaviour on certain chat channels the solutions allows for this function to be performed in parallel with the rest of its main functions without this affecting the performance.

- RabbitMQ;
- Reddis;
- Angular;
- IdentityServer4.

2.2 Solutions Used

The VoIP will use WebRTC for transfer of video and audio. WebRTC (Web Real-Time Communication) has been chosen as a part of the project's tech stack since it is a free, open-source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). Another significant advantage of the chosen solution is that it is supported by Google, Microsoft, Mozilla, and Opera. It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps and is being standardized through the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). WebRTC also provides open libraries that can be used by a native app if necessary during the implementation of the Project.

The messaging part of the module will be based on SignalR, which is a library that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.

WebRTC benefits can be summarized as follows:

- **It's free.** WebRTC is an open-source application programming interface.
- **Platform and device independent.** Any WebRTC-enabled browser with any operating system and a web services application can direct the browser to create a real-time voice or video connection to another WebRTC device or to a WebRTC media server.

- **Advanced voice and video quality.** WebRTC uses the Opus audio codec that produces high fidelity voice. The Opus codec is based on Skype's SILK codec technology. The VP8 codec is used for video. These selections ensure interoperability and avoid the need for codec downloads that may contain malicious code.
- **Secure voice and video.** Implementations use secure protocols such as DTLS and SRTP. Encryption is mandatory for all WebRTC components, including signalling mechanisms.

2.3 WebRTC Integration with SAAM Architecture

2.3.1 Definitions

Let us first provide a list of technologies used and their brief descriptions.

- **NAT** - Network address translation (NAT) is a method for remapping one IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device. The technique was originally used as a shortcut to avoid the need to readdress every host when a network was moved. It has become a popular and essential tool in conserving global address space in the face of IPv4 address exhaustion. One Internet-routable IP address of a NAT gateway can be used for an entire private network.
- **STUN** - Session Traversal Utilities for NAT (STUN) is a standardized set of methods, including a network protocol, for traversal of network address translator (NAT) gateways in applications of real-time voice, video, messaging, and other interactive communications. STUN is a tool used by other protocols, such as Interactive Connectivity Establishment (ICE), the Session Initiation Protocol (SIP), or WebRTC. It provides a tool for hosts to discover the presence of a network address translator, and to discover the mapped, usually public, Internet Protocol (IP) address and port number that the NAT has allocated for the application's User Datagram Protocol (UDP) flows to remote hosts. The protocol requires assistance from a third-party network server (STUN server) located on the opposing (public) side of the NAT, usually the public Internet.
- **TURN** - Traversal Using Relays around NAT (TURN) is a protocol that assists in traversal of network address translators (NAT) or firewalls for multimedia applications. It may be used with the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). It is most useful for clients on networks masqueraded by symmetric NAT devices. TURN does not aid in running servers on well-known ports in the private network through a NAT; it supports the connection of a user behind a NAT to only a single peer, as in telephony, for example.

2.3.2 WebRTC connection initiation

The configuration of an endpoint on a WebRTC connection is called a session description. The description includes information about the kind of media being sent, its format, the transfer protocol being used, the endpoint's IP address and port, and other information needed to describe a media

transfer endpoint. This information is exchanged and stored using Session Description Protocol (SDP); more details on the format of SDP data are available in RFC 2327.

When a user starts a WebRTC call to another user, a special description is created called an offer. This description includes all the information about the caller's proposed configuration for the call. The recipient then responds with an answer, which is a description of their end of the call. In this way, both devices share with one another the information needed in order to exchange media data. This exchange is handled using Interactive Connectivity Establishment (ICE, a protocol which lets two devices use an intermediary to exchange offers and answers even if the two devices are separated by Network Address Translation (NAT).

Each peer, then, keeps two descriptions on hand: the local description, describing itself, and the remote description, describing the other end of the call. The offer/answer process is performed both when a call is first established, but also any time the call's format or other configuration needs to change. Regardless of whether it's a new call, or reconfiguring an existing one.

As well as exchanging information about the media (discussed above in Offer/Answer and SDP), peers must exchange information about the network connection. This is known as an ICE candidate and details the available methods the peer is able to communicate (directly or through a TURN server). Typically, each peer will propose its best candidates first, making their way down the line toward their worse candidates. Ideally, candidates are UDP (since it's faster, and media streams are able to recover from interruptions relatively easily), but the ICE standard does allow TCP candidates as well.

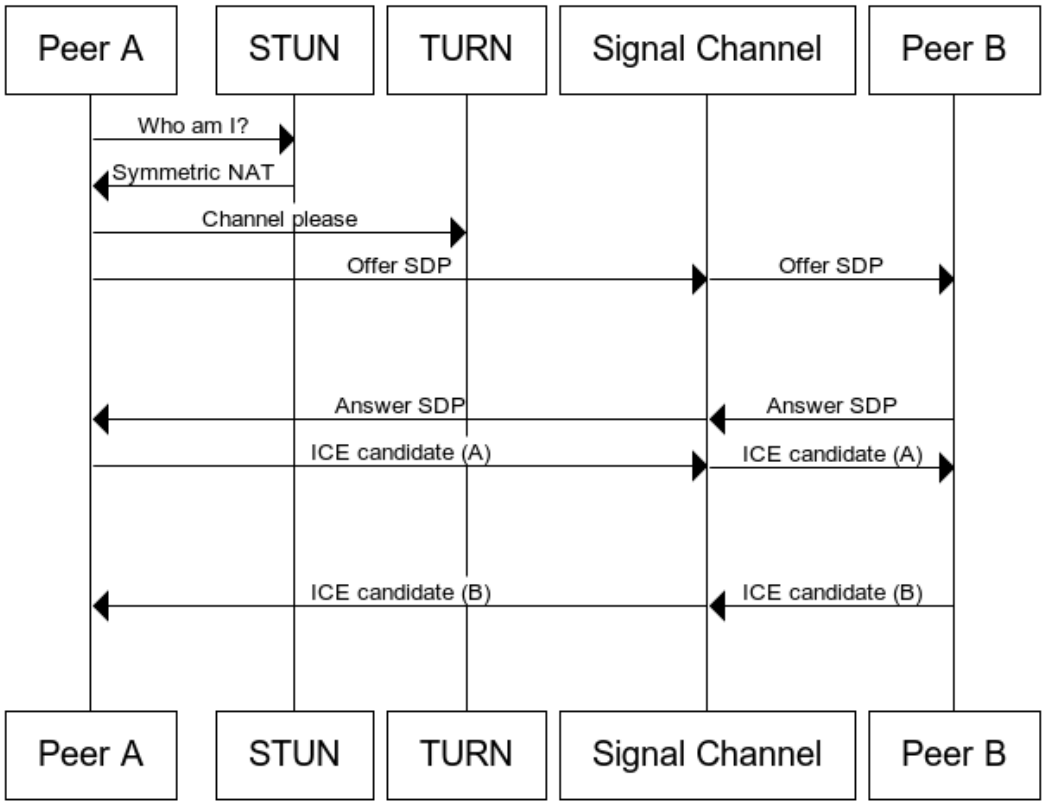

Figure 2 - WebRTC connection initiation

WebRTC connection

WebRTC needs at least a STUN server to collect connection information and signalling server to initiate connection and send connection information. The connection is established as follows.

- Peer clients will get public connection information from the STUN server.
- Then the connection information is send through signalling server.
- If peer-to-peer connection is not established, then the connection is routed to a TURN server that will forward the data between the clients.

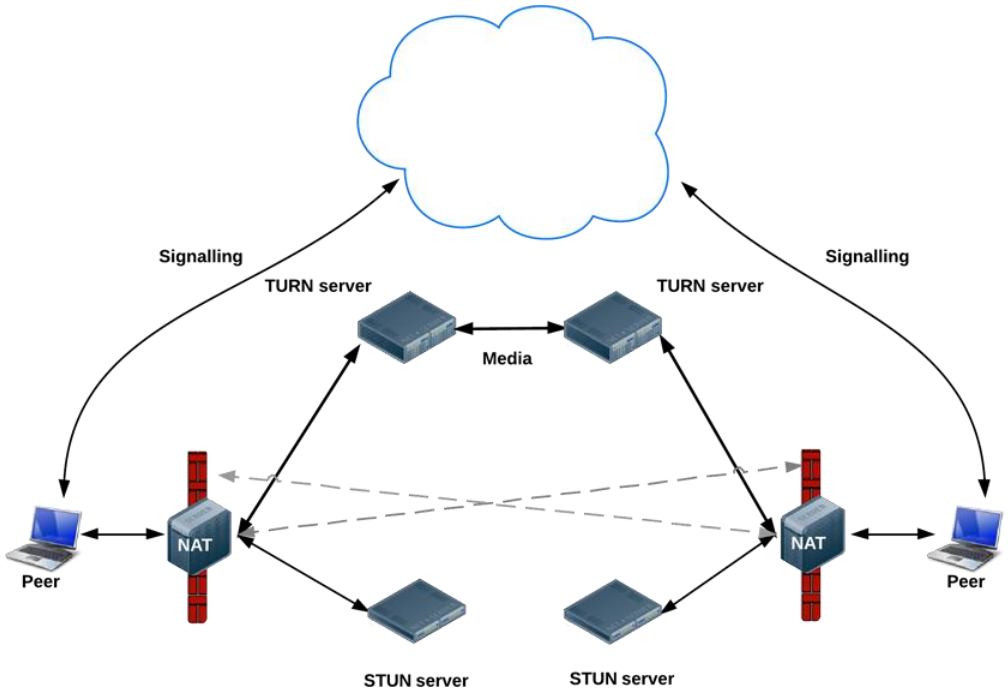

Figure 3 - WebRTC connection

In our case the signalling server and the TURN server will be hosted by ScaleFocus.

Signalling Server

The signalling server will be containerized application that is responsible for:

- Connection exchanges between chat/lobby participants
- Data exchange in order to send text messages
- User Status Change Information
- Connection Status Management

3 MODULE FUNCTIONALITIES

The purpose of this section of the document is to list the main functionalities, planned for the Module and their hierarchical connection (Figure 4 - VoIP and Messaging Functionalities).

1. **Chat Related Functionalities**
   - 1.1. Group Chats
     - Maximum number of People: 10
     - Mentions will be available with the use of the '@' symbol
     - A group chat can be joined via the group participant
     - Leave Group Chat Option
     - Turn off Notifications for Group Chat Option
   - 1.2. Individual Chats
     - Have favourites chats sitting at the top of the module panel for ease of access
   - 1.3. Message Related Actions
     - Edit Message
     - Delete Message
     - Copy Message
     - Resend message
     - Send Emoticons
   - 1.4. History with an option to load chat history for the past 30 days, 3 months, 6 months, 12 months)
   - 1.5. Different Status indications in chat activity
     - Typing
     - Message Sent
     - Message Delivered
     - Message not sent (with an option of resending said message)
   - 1.6. Coaching and Consulting
     - Set by the users (e.g. by Social Workers with reminders for forthcoming visits, doctor's check-ups and other related commitments) one-time or reoccurring messages that will be send by the system. The sending will be triggered by Hangfire.
     - Messages Initiated by the System as Coaching triggers, serving as notifications;
     - Messages Initiated by the System in case of emergency, triggered by predefined events.
     - Predefined emoticon-like invitation icons at a convenient for the user place on the UI, helping reaching out to their Contacts in the Family and Friends Circle.
2. **Contacts Management**
   - 2.1. Already in the user's Circles
     - Initiate Call/Chat
     - Invite the user in a group chat
   - 2.2. Not in Circles

- Send an invite

3. Calls

3.1. Call Types
- Video and Audio Calls
- Individual and Group Calls

3.2. Calls related Actions
- Take a Call
- Mute
- Hang up

3.3. Coaching and Consulting
- Emergency Call Initiated by the System based on predefined triggering events
- Scheduling calls with social workers or Friends and Family members with a preliminary reminder suggesting to initiate the planned call.

3.4. History
- Log of past calls – duration, time, date

4. Report

4.1. Option to report unwanted activities within the system's module will be available.
- Report an inappropriate Message
- Report a User

5. **User Authentication, Settings, Contacts and User's Profile**

Since the module will be part of the entire SAAM system, all up listed functionalities will be handled in a centralised manner. An overview of all functionalities is presented in Figure 4

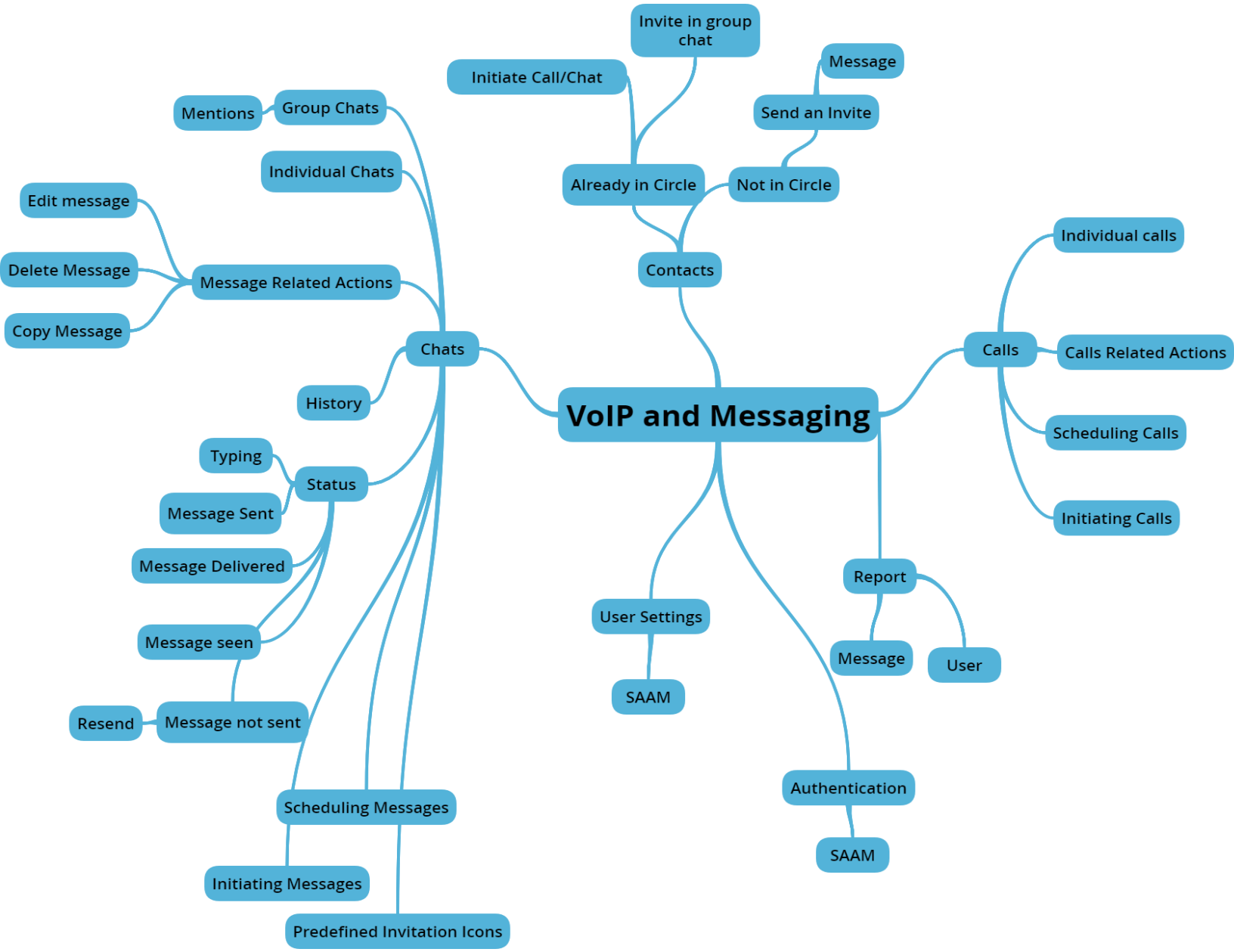

Figure 4 - VoIP and Messaging functionalities