



Messenger Requirements Definition

Version 2.0

Revision History

Date	Version	Description	Author
2004-11-15	1.0	Initial Draft	Jonas Wadsten
2004-11-23	1.1	Almost complete, sent in	Jonas + Zahid
2004-11-24	1.2	Updated the table of contents	Jonas
2004-11-25	1.3	Revise due to Rikard's comments	Zdenek + Zahid
2004-12-8	1.4	Revise due to Rikard's comments	Zahid
2004-12-11	1.5	Last touch with formatting and the last try of change for the intro text	Jonas
2004-12-15	1.6	Changed Introduction	Tihana
2004-12-16	1.61	Changed Introduction	Marko
2004-12-16	1.7	Checked grammar and made some changes in the text (marked KiM)	Marko
2004-12-17	2.0	Did the last formatting of the doc. Deleted info not needed about related documents.	Jonas

Table of Contents

1.	Introduction	3
1.1	Purpose of this document	3
1.2	Intended Audience	4
1.3	Scope	4
1.4	Definitions and acronyms	4
1.4.1	Acronyms and abbreviations	4
2.	Requirements Description	5
2.1	General requirements	5
2.2	Specific requirements	5
3.	Use Case Models	5
3.1	Use case model 1	5
3.1.1	Use case Start-up process for the messenger client	6
3.1.2	Use case Start recording	7
3.1.3	Use case Stop recording	7
3.1.4	Use case Search	8
3.1.5	Use case Save to database	8
3.1.6	Use case Search local file	9
3.1.7	Use case Search database	9
3.2	Use case model 2	10
3.2.1	Use case Login as User	10
3.2.2	Use case change password	11
3.2.3	Use case Search	11
3.2.4	Use case View results	12
3.2.5	Use case View the conversation	12
3.2.6	Use case Login as Administrator	12
3.2.7	Use case Add project	13
3.2.8	Use case Add project member's name	13
3.2.9	Use case Login as super administrator	13
3.2.10	Use case Delete project	14
3.2.11	Use case Delete member	14
4.	Requirements Definition	15
4.1	Requirement Group Definitions	15
4.2	Requirement Sources	15
4.3	Requirements definitions	16
4.3.1	Change Log	17
4.4	Privacy and non Intrusiveness Requirement Definitions	17
4.5	Implementation Requirement	18
4.5.1	Basic requirements	18
4.5.2	Normal requirements	19
4.5.3	Timeline	20
4.5.4	Detailed schedule	21
4.5.5	Advanced phase	25
4.6	Error handling	25
5.	Future Development	26
5.1	General Overview	26
5.2	User friendliness	26
5.3	Portability	26
5.4	Storing saved chat sessions.	26

1. Introduction

Some communication between members in a software project leaves traces (such as emails or protocols from meetings) that make it possible to retrieve information later on. Some other communication channels leave no trace. One of such ways of communication, often used in distributed development projects, is formal communication through various Instant messengers. Though they sometimes do have an option to save an conversation, this option is limited to individual saving. That means the information is available only to the participant of this kind of conversation who actually saved the conversation. The other participants can't retrieve it unless they also saved it. This is however not formal enough, because every team member should have accesses to the same relevant information concerning the meetings, and this information should be secure. When each team member saves the conversation him/herself locally in plain text format, data manipulations are possible. Consider a case of some unscrupulous team member that is capable of changing the text of conversations. This hypothetical team member gives promises he can't deliver and then later changes the text of conversations in order to avoid responsibility for his actions. This hypothetical case can cause much harm to team work in distributed software development process. The main goal of this project is therefore to store relevant chat sessions in one place, regardless of the Instant messenger participant's use, so that this information can be easily retrieved later by all the participants. The information stored should be secure, because of the formal character of meetings.

1.1 Purpose of this document

The purpose of this document is to describe the requirements and the forthcoming of this project. The document describes the project functional and nonfunctional requirements of the system, and about the user of the system and the services provided by the system with respect to the specific user. The document describes the outcome of the project. This document is related to other documents that give detailed information of some issues listed below.

Project description - in project description document you can find more information on intended audience (members of steering group and customers) and project management (listed in section 2). There can also be found some constraints (technological, environmental, interpersonal) we are bounded by (listed in section 3. subsections 3.1. 3.2. and 3.3.), deliverables (section 4.), project risks (section 5.), means of communication (section 7.) and project plan that consists of time schedule and activity plan (section 9. subsections 9.1. 9.2.). This file can be found in CVS repository on orson.rasip.fer.hr server. Server path is var/cvsdsd/Messenger/Documents.

Design description - in design description document you can find more information on software architecture, Messenger client and Web interface (section 2. subsections 2.1. 2.3. 2.4.). There can be found detailed description of program interface (buttons, tabs) and web interface (login, search, add new member, add new project...). You can also find, in this document, details of software design like CRD-class relation diagram, OSD-object sequence diagrams and database design description (section 3. subsections 3.1. 3.2. 3.3.). This file can be found in CVS repository on orson.rasip.fer.hr server. Server path is var/cvsdsd/Messenger/Documents.

1.2 Intended Audience

Our intended audience is:

- Steering Group
- Project staff
- Class fellows and customer

1.3 Scope

This project will identify the possibilities of making messenger programs context aware, and define an architecture that supports a number of instant messenger tools, and implement a general framework, database and specific components needed for different messengers. The task of this project is to store relevant chat sessions in a way so that this information can be easily retrieved later. Any popular chat program: ICQ, AIM, MSN messenger, Yahoo messenger, Miranda Instant Messenger and so on. Client program saves chat sessions in a central project database: MySQL, PostgreSQL and so on.

1.4 Definitions and acronyms

1.4.1 Acronyms and abbreviations

Acronym or Abbreviation	Definitions
CVS	Concurrent Version System
ICQ/AIM/MSN	Different messenger programs for chat sessions
I/M	Information and Member

2. Requirements Description

2.1 General requirements

The program should:

- *Be robust* – the program should not crash or hang
- *Be non-intrusive* – the program should not be a nuisance to users, but should be as “invisible” to users as
- *Honor privacy* – the users must be aware that (some of) their chat sessions are stored, but still feel that the
- Information will not be used for other purposes than strictly project management.
- *Be secure* – the program must be as secure (regarding hostile intrusions etc.) as the messengers they use.
- *Allow easy integration of new messenger.*

2.2 Specific requirements

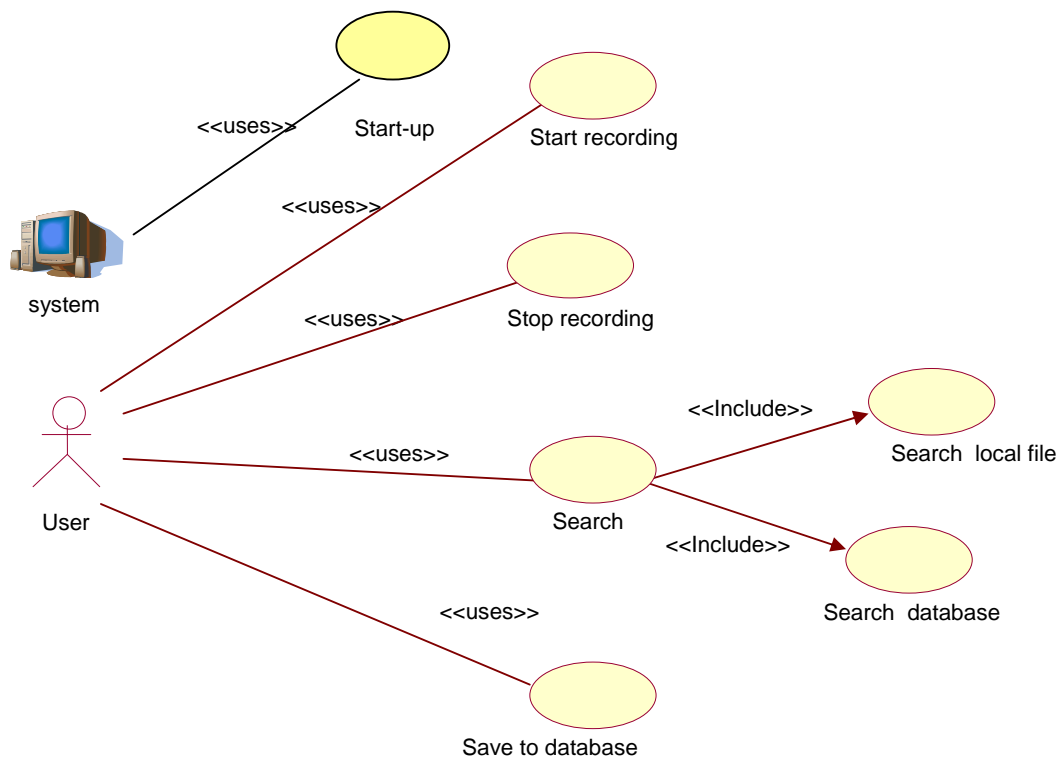
- Build upon existing messenger tools.
- Capture different start-up events, different messengers.
- When conversation is finished, save it to central database.
- Store chat sessions for later retrieval and any related information such as date, time, project members involved.
- Login to all pages, different login for administrate? Allow searches based on user names, keywords, dates, etc. in a user-friendly user interface. The database and retrieval mechanisms must be carefully designed and implemented to make it easy to rapidly find the relevant information.
- Allow easy integration with new/other/new version of messengers.

3. Use Case Models

In this document there are two use case models. First use case model describes features that are accessible from the client program. The second use case model represents the services provided by the system for administration purposes. The administration uses these features to accomplish different task like team member searches, adding new members or discussions of particular project. This is al done through the web interface.

3.1 Use case model 1

The first use case model contains one use case initiated from the system, and for initiated by user. There are also two use cases initiated from the program, when the user calls search function. This is because the user calls the search procedure, and depending on the given parameters system can search local file, database, or both. This is to be clear, that searching the database differs from searching the local file (searching the database requires a database connection to be present, and it also requires for user to log in)



3.1.1 Use case Start-up process for the messenger client

Initiator: Computer system

Goal: Start-up the messenger client via autorun when computer-system starts-up.

Main Scenario:

1. Whenever client machine starts, messenger software is activated through its autorun method.
2. Messenger program then connects to its local database where it gets different messengers information and then scans the client machine for different chat messengers (e.g. yahoo) whether running or not.
3. If any chat messenger is running check who (either team member or not) is logged into chat messenger.
4. If team member is logged into the chat messenger CheckChat method checks whether team member is chatting with other team member(s) or not. CheckChat method gets other team members information from centralized database.
5. If team member is not logged into the chat messenger keep checking else keep looking for chat sessions.

3.1.2 Use case Start recording

Initiator: User

Goal: Start recording of one of the currently active chat sessions that are not already being recorded

Main Scenario:

1. A list of currently active chat sessions that are not already being recorded is displayed
2. User marks one chat session and presses "Start Recording"
3. A record confirmation is requested from all other participants in the selected chat session
4. System initiates the recording process (information about chat session is stored into control file, new file is opened in which the chat session will be saved, messages are being captured from IMM and saved into that file)
5. The list of currently active chat sessions that are not already being recorded is refreshed

Extensions: If user clicks on close button the window is closed.

3.1.3 Use case Stop recording

Initiator: User

Goal: Stop recording of one of the currently active chat sessions that are being recorded

Main Scenario:

1. A list of currently active chat sessions that are being recorded is displayed
2. Actor marks one/more chat session/s and presses "Stop Recording"
3. System stops the recording process (file in which the chat session has been saved is closed, and system stops messages capture from IMM)
4. List of currently active chat sessions that are being recorded is refreshed

Extensions: If user clicks on close button the window is closed

3.1.4 Use case Search

Search function searches through database only, local file only, or both. Some of the chat sessions stored locally will eventually be sent to the database and deleted locally. However if user wishes to store some conversation locally only (and these are private conversations, possibly between team members), he is given an option not to send to database every conversation stored locally. Because of this, program also searches locally stored files. The other reason why local files are searched is database availability. When database is not available, user can still search through conversations that are stored locally (not sent to database).

Initiator: *User*

Goal: *Get the parameters for search process*

Main Scenario:

1. *A search form dialog is displayed*
2. *Actor marks/enters search parameters in corresponding fields and clicks "Search" Button*
3. *System calls search functionality with given parameters and displays list of conversations (can be assembled from one or more chat sessions) to user*
4. *Actor marks one/more result/s and presses the "View" button*
5. *System decrypts each of the selected conversations chat sessions found locally.*
6. *Marked conversation/s is/are being displayed in separate window/s*

Extensions: *If user clicks on close button the window is closed*

3.1.5 Use case Save to database

Initiator: *User*

Goal: *Store local files to the database*

Main Scenario:

1. *System initiates "log in to the database" process*
2. *A list of currently available chat sessions stored locally that are no longer being recorded and can be saved to the database is displayed*
3. *Actor marks one/more/all chat sessions listed and presses the button "Send"*
4. *System sends the data to database*

Extensions: *If user clicks on close button the window is closed*

3.1.6 Use case Search local file

Initiator: System

Goal: Search local file(s) according to existing search parameters

Main Scenario:

1. System searches local control file according to existing search parameters
2. System generates a list of results with parameters that are necessary to access the data itself

3.1.7 Use case Search database

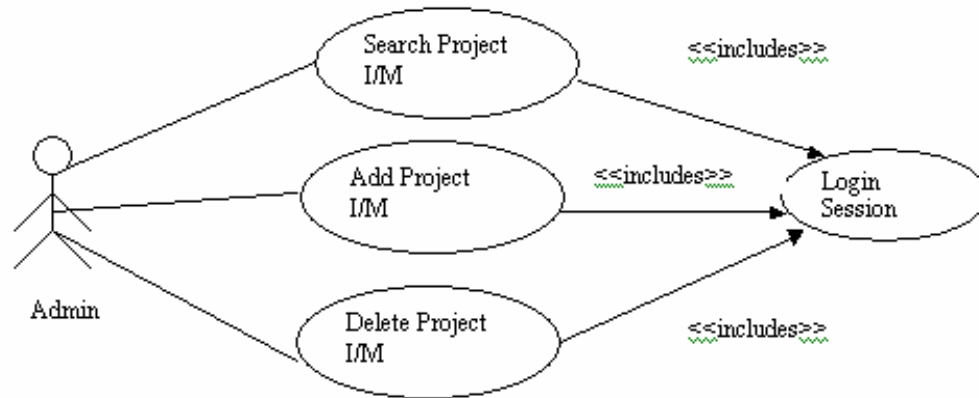
Initiator: System

Goal: Search database according to existing search parameters

Main Scenario:

1. System initiates "log in to the database" process
2. System generates a list of results with parameters that are necessary to access the data itself

3.2 Use case model 2



3.2.1 Use case Login as User

Initiator: User

Goal: Login as User to the database with purpose to search for conversation made within a project.

Main Scenario:

1. Actor types user name
2. Actor types password than presses OK.
3. System compares the user name and the password with information stored in the database. If comparison is OK than user is transferred to search page.
4. If it is his/her first login time than s/he is transferred to change-password page.

Extensions: If user types wrong user name or password than s/he is not allowed to enter.

3.2.2 Use case change password

Initiator: System

Goal: Change user's password

Main Scenario:

1. A password page is displayed
2. Actor must type his/her old password
3. Actor must type his/her new password and confirm it, than pres OK.
4. System compares old password with database, if it is the same than system compares new password with confirmation password. If it is same than system sends the new password to database.
5. System displays dialog box that password was changed

Extensions: If user types wrong confirmation password than exception raises. User must type it again or if old password is wrong.

3.2.3 Use case Search

Initiator: User

Goal: Get the parameters for search process

Main Scenario:

1. A search page dialog is displayed
2. Actor marks/enters the search parameters in corresponding fields and clicks "Search" Button
3. System calls search method with given parameters and displays result on result page to user
4. User can choose the type of search. By project name, member involved in conversation, keyword or date. The date-search is already set but can be changed. From today's date to yesterday's date.

Extensions: If user clicks on close button the window is closed

3.2.4 Use case View results

Initiator: *User*

Goal: *Display the results of the search*

Main Scenario:

1. *System displays the results of the search in order:*
 - *Name of project*
 - *Participants in conversation*
 - *Date/s of the conversation/s*
2. *The dates are hyperlinks. Using hyperlinks user can access the conversation*
3. *Actor clicks to the date of conversation and the conversation page is displayed with the chosen conversation*

3.2.5 Use case View the conversation

Initiator: *System*

Goal: *Displays the chosen conversation*

Main Scenario:

1. *System displays the chosen conversation*

3.2.6 Use case Login as Administrator

Initiator: *User*

Goal: *Login as Administrator to the database to search for conversation made within a project or to add project name or project member.*

Main Scenario:

1. *Administrator types administrator name*
2. *Administrator types password than presses OK.*
3. *System compares the administrator name and the password with information stored in the database. If comparison is OK than user is transferred to the search page.*
4. *If it is his/her first login time than s/he is transferred to change-password page.*

Extensions: *If user types wrong administrator name or password than s/he is not allowed to enter.*

3.2.7 Use case Add project

Initiator: Administrator

Goal: Creates new project name.

Main Scenario:

1. Actor types new project name
2. Presses ADD
3. System creates new project name and writes it to database.
4. Project list is refreshed

Extensions: No special characters, but underscore, are allowed

3.2.8 Use case Add project member's name

Initiator: Administrator

Goal: Creates new project member's name

Main Scenario:

1. Administrator chooses the project to which the member's name will be added
2. Administrator types new project member's name
3. Administrator ADD
4. System creates new member's name and writes it to database.
5. Member's list is refreshed

Extensions: No special characters, but underscore, are allowed

3.2.9 Use case Login as super administrator

Initiator: Super administrator

Goal: Login as super administrator to the database to search for conversation made within a project or to add project name or project member. Also erasure of project name and project members is possible.

Main Scenario:

1. Administrator types super administrator name
2. Administrator types password then presses OK.
3. System compares the super administrator name and the password with information stored in the database. If comparison is OK than user is transferred to the search page.
4. If it is his/her first login time than s/he is transferred to change-password page.

Extensions: If user types wrong administrator name or password than s/he is not allowed to enter.

3.2.10 Use case Delete project

Initiator: *Super administrator*

Goal: *Delete project name from the database with all the members*

Main Scenario:

1. *Super administrator chooses project name from the list.*
2. *Presses DELETE*
3. *System raises alert dialog box with "Do you want to DELETE project _____ with all its members? YES / NO*
4. *If super administrator clicks YES than system marks the project as closed and no more changes are allowed. The project will be stored in the database for further searches.*
5. *List of projects is refreshed.*
6. *If No button is pressed than system closes the alert dialog box.*

3.2.11 Use case Delete member

Initiator: *Super administrator*

Goal: *Delete member from the database*

Main Scenario:

1. *Super administrator chooses member's name from the list.*
2. *Presses DELETE*
3. *System raises alert dialog box with "Do you want to DELETE member _____ from the project _____? YES / NO*
4. *If super administrator clicks YES than system changes the status of the user to closed. No more changed are possible. User is not removed from the database, only his/her records are not anymore changeable.*
5. *List of members is refreshed*
6. *If No button is pressed than system closes the alert dialog box.*

4. Requirements Definition

4.1 Requirement Group Definitions

Identification	Requirement Group	Rem.
ADM	System Administration	
HR	Human Resources	
DX	Data exchange	

4.2 Requirement Sources

Source	Description	Rem.
Adm	Administrator or Manager	
Mbm	Member	
Sys	System	
Gst	Guest	

4.3 Requirements definitions

Id	Status	Priority	Description	Source
M-1	A	2	<p>Definition: The web interface of the system has two parts: User pages and Administrator pages.</p> <p>Motivation: To keep the user to its own section.</p>	Mbm
M-2	I	1	<p>Search functionality:</p> <ul style="list-style-type: none"> Definition: Web page will allow manager to search project information from the saved chat database. Allow searches based on user names, keywords, dates, etc. in a user-friendly user interface. The database and retrieval mechanisms must be carefully designed and implemented to make it easy to rapidly find relevant information. <p>Motivation: A manager can search required information by using user-friendly web interface.</p>	Sys
M-3	A	2	<p>Administrator part:</p> <p>Definition: Administrator part consists of project information and facility to add new project member and search from the database.</p>	Adm
M-4	I	1	<p>Add/Delete information:</p> <p>Definition: This web page allows administrator to delete the information by using user-friendly interface.</p> <p>Motivation: Easy to delete information</p>	Adm
M-5	I	1	<p>Start/Stop recording:</p> <p>Definition: Member can choose start/stop recording while chatting to other members or guests.</p> <p>Motivation: A member can easily cancel his/her recording.</p>	Mbm
M-6	I	1	<p>Update project/member information:</p> <p>Definition: This application allows administrator to update information by using user-friendly web interface.</p> <p>Motivation: Easy to update information.</p>	Adm
M-7	I	1	<p>Security implementation:</p> <p>Definition: The system will be reliable and secured from intruders.</p> <p>Motivation: The goal of the system has to provide full freedom with trust and reliability to chat without any hesitation.</p>	Sys
M-8	H	2	<p>Java-client</p> <p>Definition: A client to channel data through the firewall at the server in Zagreb</p> <p>Motivation: To be able to use the database down in Zagreb. If the database in Sweden works, we don't need this java-client.</p>	Sys

Requirement status:

I = initial (this requirement has been identified at the beginning of the project),
D = dropped (this requirement has been deleted from the requirement definitions),

H = on hold (decision to be implemented or dropped will be made later),

A = additional (this requirement was introduced during the project course).

Messenger Client	Version: 2.0
Requirements Definition	Date: 2004-12-17

4.3.1 Change Log

Identity	Action	Date	Comments
M-8	A	11 Nov 2004	We might need to build a java-client to be able to communicate with the database in Zagreb.
M-8	H	28 Nov 2004	We have postponed this requirement to keep the work hours down but we will do it if it is necessary.

Requirement status:

D = dropped (this requirement has been deleted from the requirement definitions),

H = on hold (decision to be implemented or dropped will be made later),

A = added (this requirement was introduced during the project course).

R = resurrected (dropped or on hold requirement was reactivated)

4.4 Privacy and non Intrusiveness Requirement Definitions

Chatting mode	Description	Solution
Mbm + Mbm	Two project members chatting, this situation is considered to be default case	Chat is automatically saved
Mbm + Gst	One member is chatting with a guest	System gives member two options - save the chat or not. If chat should be stored, system first adds the member and then records the conversation. If chat should not be stored conversation is not monitored.
Mbm + Mbm	Two members are chatting on any private topic	They should be notified that chat should not be saved

4.5 Implementation Requirement

Implementation requirements for this years project are stated in this document. Every requirement has its own marking written between square brackets in a format:

[REQ <B/N/A><roman number> <letter>] (for example [REQ BIII c])

- REQ** The first part of the marking, the "REQ" means that it is a requirement marking.
- <B/N/A>** Second part of the marking can be B if it marks a basic requirement; N if it stands for a normal requirement, and A if it is an advanced requirement. The basic requirement is a requirement that is crucial to the program functionality. A normal requirements stands for a requirement that benefits the usability of the program, from a user point of view. It provides everything that a programm must have, but it is not really crucial to get the job done. The advanced requirement is "would be nice to have one day" but is not necessary to implement now.
- <roman number>** Third part is the roman number associated to a certain module as follows:
- I. Local storage module
 - II. Database communication client
 - III. User interface module
 - IV. Main module
 - V. Messenger specific modules, Lurker module, Stargate module
- <letter>** just order letter in a group of requirements associated with a certain module, to help distinct them within the group

4.5.1 Basic requirements

Local storage module

Simply realized with saving data into files without encryption. Each conversation (chat session) is stored in its own file. Each chat session is also saved in one control file, where all the relevant data about the chat session is stored, including name of the file where the chat itself is stored. This file should provide faster searching. [REQ BI a]

User interface module

Simple user interface that should implement:

- Simple dialog box with following buttons: "Start Recording", "Stop Recording", "Search" [REQ BII a]
- Start recording dialog - list of currently available chat sessions [REQ BIII b]
- Search dialog - "search database" option disabled [REQ BIII c]
- "More Details" window - part of main dialog, should display any relevant information about what the program is doing at the moment [REQ BIII d]

Main module

Implement following functions:

- Start Recording [REQ BIV a]
- Stop Recording [REQ BIV b]
- Search Local File [REQ BIV c]

Lurker, Stargate and Yahoo modules

Test the old modules with new ones, and correct problems found:

- LURKER MODULE to MAIN MODULE [REQ BV a]
- STARGATE MODULE to MAIN MODULE [REQ BV b]
- YAHOO MESSENGER SPECIFIC MODULE to STARGATE MODULE [REQ BV c]

4.5.2 Normal requirements

A. Local storage module

Implement:

- Encryption [REQ NI a]
- Decryption [REQ NI b]

B. Database communication client

Implement:

- TCP/IP connection through windows socket (change the old implementation) [REQ NII a]
- Communication through that socket towards server side database communication daemon (It is required, prior this implementation, that the server side daemon is well defined if not already implemented) [REQ NII b]

C. User interface module

- Add "Send to Server Now" button [REQ NIII a]
- Implement "Send to Server Now" dialog (choice of conversations stored locally, default: "Send All") [REQ NIII b]
- Enable "Search Database" in Search dialog [REQ NIII c]
- Implement options dialog [REQ NIII d]
- Implement "Database Server" option tab [REQ NIII e]
- Implement "Predefined Research" option tab [REQ NIII f]
- Implement standard "About" dialog [REQ NIII g]
- Implement "Help" dialog [REQ NIII h]
- Implement Error message box [REQ NIII i]
- Implement inactive buttons when operation for which they are designed is not available (no active messenger -> inactive "Start Recording", "Stop Recording"; no data in the search file -> "Send to Server Now" inactive) [REQ NIII j]
- Implement system tray menu [REQ NIII k]
- Add "More Details / Less Details" button [REQ NIII l]

D. Main module

Implement:

- "Send to Server Now" function [REQ NIV a]
- "Search database" function [REQ NIV b]
- "Save Options" function [REQ NIV c]
- Error reporting function [REQ NIV d]
- System tray icon and menu [REQ NIV e]
- Implement "More Details" window hiding [REQ NIV f]
- Implement main window is hidden by default [REQ NIV g]

E. Messenger specific modules

Implement:

- MSN Messenger Specific Module [REQ NV a]
- ICQ Messenger Specific Module [REQ NV b]

4.5.3 Timeline

4.5.3.1 Optimistic timeline

	46	47	48	49	50	51	52	53	1	2
BASIC										
NORMAL										
ADVANCED										

4.5.3.2 Realistic timeline

	46	47	48	49	50	51	52	53	1	2	
BASIC											
NORMAL											
ADVANCED											

4.5.4 Detailed schedule

Detailed schedule gives each source code team member strict development tasks (requirement implementation) divided in three phases. For each phase there is a table, which contains these tasks in order they should be developed, and each task is associated with a deadline. Each phase has it's own deadline, which is always the same as the deadline of the last task on the list. In each phase table, this final deadline is bolded and printed in different color. Phase deadlines should be respected, and the task deadlines are given for orientation purposes only.

There are two kinds of detailed schedules for each source code team member: optimistic and realistic. Each developer should try to keep the deadlines, from the optimistic schedule, thus we can also call it "developer schedule". The realistic schedule is here for official documentation purposes, so we can call it "official schedule". It is imperative that the deadlines from the official schedule are kept. The primary goal of having two schedules is to finish with the project before official deadline. That is therefore the main reason, why source code team should stick to developer schedule.

4.5.4.1 Optimistic detailed schedule

- Irfan
 - A. PHASE 1: Basic requirement implementation

Requirement	Deadline
REQ BI a	18.11.2004
REQ BIV a	20.11.2004
REQ BIV b	22.11.2004
REQ BIV c	24.11.2004
REQ BV a	26.11.2004
REQ BV b	28.11.2004

- B. PHASE 2: 90% Normal requirement implementation

Requirement	Deadline
REQ NI a	2.12.2004
REQ NI b	2.12.2004
REQ NII a	8.12.2004
REQ NII b	12.12.2004
REQ NIV a	14.12.2004
REQ NIV b	16.12.2004

- C. PHASE 3: Messenger specific modules implementation

Requirement	Deadline
REQ NV a	22.12.2004

- **Tihana**

A. PHASE 1: Basic requirement implementation

Requirement	Deadline
REQ BIII a	18.11.2004
REQ BIII b	20.11.2004
REQ BIII c	21.11.2004
REQ BIII d	23.11.2004
REQ BV c	25.11.2004

B. PHASE 2: 90% Normal requirement implementation

Requirement	Deadline
REQ NIV d	26.11.2004
REQ NIV f	27.11.2004
REQ NIII a	28.11.2004
REQ NIII b	29.11.2004
REQ NIII c	30.11.2004
REQ NIII d	1.12.2004
REQ NIII e	3.12.2004
REQ NIII f	5.12.2004
REQ NIII g	6.12.2004
REQ NIII h	7.12.2004
REQ NIII i	8.12.2004
REQ NIII j	10.12.2004
REQ NIII k	11.12.2004
REQ NIII l	11.12.2004
REQ NIV c	12.12.2004
REQ NIV e	15.12.2004
REQ NIV g	16.12.2004

C. PHASE 3: Messenger specific modules implementation

Requirement	Deadline
REQ NV b	22.12.2004

4.5.4.2 Realistic detailed schedule

- Irfan

A. PHASE 1: Basic requirement implementation

Requirement	Deadline
REQ BI a	19.11.2004
REQ BIII a	23.11.2004
REQ BIII b	25.11.2004
REQ BIII c	29.11.2004
REQ BIV a	3.12.2004
REQ BIV b	5.12.2004

B. PHASE 2: 90% Normal requirement implementation

Requirement	Deadline
REQ NI a	10.12.2004
REQ NI b	10.12.2004
REQ NII a	18.12.2004
REQ NII b	22.12.2004
REQ NIV a	3.1.2004
REQ NIV b	5.1.2004

C. PHASE 3: Messenger specific modules implementation

Requirement	Deadline
REQ NV a	9.1.2004

- **Tihana**

- A. PHASE 1: Basic requirement implementation

Requirement	Deadline
REQ BII a	20.11.2004
REQ BII b	24.11.2004
REQ BII c	25.11.2004
REQ BII d	28.11.2004
REQ BIV c	2.12.2004

- B. PHASE 2: 90% Normal requirement implementation

Requirement	Deadline
REQ NIV d	4.12.2004
REQ NIV f	5.12.2004
REQ NIII a	6.12.2004
REQ NIII b	7.12.2004
REQ NIII c	8.12.2004
REQ NIII d	9.12.2004
REQ NIII e	13.12.2004
REQ NIII f	15.12.2004
REQ NIII g	16.12.2004
REQ NIII h	17.12.2004
REQ NIII i	18.12.2004
REQ NIII j	20.12.2004
REQ NIII k	21.12.2004
REQ NIII l	21.12.2004
REQ NIV c	22.12.2004
REQ NIV e	3.1.2004
REQ NIV g	5.1.2004

- C. PHASE 3: Messenger specific modules implementation

Requirement	Deadline
REQ NV b	9.1.2004

4.5.5 Advanced phase

We should start with this phase only after normal requirements are met and project is completely done. These requirements should be considered as a bonus if we have any time left. They don't have to be implemented at all if, after completing the normal project, we don't have time to implement it.

A. User Interface Module

- Implement menus and operations keyboard shortcuts [REQ AI a]

B. Messenger specific Modules

C. Implement:

- AIM Messenger Specific Module [REQ AII a]
- Miranda Instant Messenger Specific Module [REQ AII b]
- GAIN Messenger Specific Module [REQ AII c]

4.6 Error handling

Error	Action
Database disconnected	Save the conversation encrypted locally, restore database connection when possible, send the conversation saved locally and that are finished. Start record locally the chat sessions.
System crash	Restore database connection at restart, send the conversation saved locally and that are finished. Start record locally the chat sessions.
Client can not connect to database	Save the conversations locally in encrypted. On next successful connection it will upload conversation and delete it locally.
Client crashes during chat session	Restore database connection at restart, send the conversation saved locally and that are finished. Start record locally the chat session.
Messenger program crash during recording	Treat it is as a finished chat-session and save it to central database, if the sessions starts-up again with the same users, creates a new chat-session with the same chat-id.
During chat-session user accidentally close the chat-window	Treat it is as a finished chat-session and save it to central database, if the sessions starts-up again with the same users, creates a new chat-session with the same chat-id.

5. Future Development

This document is related to design description document (as described in introduction of this document). In design description document there can also be found some plans of future development

5.1 General Overview

To create useful and widely adopted software it has to be intuitive and easy to use. This software has to be as user friendly as possible. It should also allow users to use it on any platform and also be able to communicate with other software regardless of the manufacturer. How do these general guidelines apply to our software? This question is answered in subsections below.

The Messenger software is very useful from management as well as development point of view. This software can be used by management as a reference and team members can remind themselves about different topics they have discussed. Currently, Messenger software supports only few chat programs, which limits its use. This issue should be corrected as soon as possible.

5.2 User friendliness

User friendliness of software we will deliver can be improved by implementing following guidelines:

- Implement menus and operations keyboard shortcuts
- Reducing users time spent to configure messenger
- Automatically start recording predefined conversations
- Automatically enable/disable recording of predefined conversations

5.3 Portability

In this project modules for ICQ and MSN messengers should be written. Other messengers should also be supported:

- AIM Messenger support
- Miranda Instant Messenger support
- GAIM Messenger support

This program should also work with messengers that work under Linux operating system.

5.4 Storing saved chat sessions.

In our project we will store chat sessions in form of text documents, one chat file for one chat session. However, big improvement would be if we would be saving chat sessions in XML format. There are some features of XML that make it particularly appropriate for use in our project:

- simultaneously human and machine readable format
- strict syntax makes the necessary parsing algorithms simple, fast and efficient which can improve search speed (e.g. when searching through chat session for keyword)
- plain text files