Requirements and Analysis Document for CEYMChat

Carl Östling, Yazan Ghafir, Erik Gunnarsson, Mohamad Almasri

date
version

1 Introduction

This document will analyze and discuss the project ”CEYMChat” created by group 21. The problem that the ”CEYMChat” application tries to solve lies within human communication. By creating a simple chat application with an understandable Graphical User Interface (GUI), human interaction and communication will flourish. Anyone who wishes to communicate via online chat with friends or other users can easily do so using ”CEYMChat”. The application requires less personal data than other existing chat applications on the market. ”CEYMChat” is a lightweight chat application developed using the java.net library. It requires no installation of software on the users device and requires minimal personal information during sign-up. No files need to be saved on the users device.

1.1 Definitions, acronyms and abbreviations

”Group 21” refers to Erik Gunnarsson, Carl stling, Yazan Ghafir and Mohamad Almasri.
”CEYMChat” refers to the application built by group 21 during this project.
”GUI” refers to a Graphical User Interface.
”.net” refers to the java.net library used for network connections in the project.
”Multithreading” refers to the usage of multiple threads in a program in order to execute several tasks at the same time.

2 Requirements

2.1 User Stories

2.1.1 Story Identifier: CEYMCHAT001

Story Name: Send a message

Description
As a user, I would like to be able to send a message to another user of choice in order to communicate.

Confirmation
- Message is sent to the server.
- Message is received by another user.
- Message is received by the correct user.
- Code well documented.
- Tests passed.

2.1.2 Story Identifier: CEYMCHAT002

Story Name: Server handles a coming message

Description
As a server, I would like to receive messages to distribute them to the appropriate recipient.

Confirmation
- received to the Server-
- Redistribution of a message.
- Message is sent to the correct client/recipient.
- Code well documented.
- Tests passed.

2.1.3 Story Identifier: CEYMCHAT003

Story Name: Chat on a GUI

Description
As a user, I want to have a GUI so that I can use the program outside a command line

Confirmation

- The client has got a working GUI interface.
- Code well documented.
- Tests passed.

2.1.4 Story Identifier: CEYMCHAT004

Story Name: Register as a user

Description
As a user, I want to register as a user with my e-mail account so I can have my own profile.

Confirmation
- Client recognizes a user object.
- Client recognizes data in a user object in order to change its appearance.
- Code well documented.
- Tests passed.

2.1.5 Story Identifier: CEYMCHAT005

Story name: Server receives commands
Description
As a server, I want to receive commands and execute a corresponding method.

Confirmation
- Server receives messages containing commands.
- Server executes commands it has received.
- Code well documented
- Tests passed

2.1.6 Story Identifier: CEYMCHAT006

Story name: User can save chat history
Description
As a user, I want to be able to save my chat history so I can access it any time I want.

Confirmation
- Chat history can be exported to some readable file.
- A file containing chat history can be saved locally.
- Code well documented
- Tests passed

2.1.7 Story Identifier: CEYMCHAT007

Story name: User can choose whom to chat with
Description
As a user, I want to be able to click on my online friend in a list of friends so that we can communicate.

Confirmation
- Logged in users are displayed in the GUI
- Users can be clicked to choose who to send a message to
- Code well documented
- Tests passed

2.1.8 Story Identifier: CEYMCHAT008

Story name: User can send files
Description
As a user, I want to be able to send files so I can share them with my friends.

Confirmation
- Users can choose a file in the GUI to send
- Chosen file is sent to the server
- Server redistributes the file to the correct user
- Code well documented
- Tests passed

2.1.9 Story Identifier: CEYMCHAT009

Story name: User can see when other people are online
Description
As a user, I would like to see when my friends are online to decide whom to chat with

Confirmation
- Other users are shown in the GUI
- Users currently online are marked as such
- Code well documented
- Tests passed

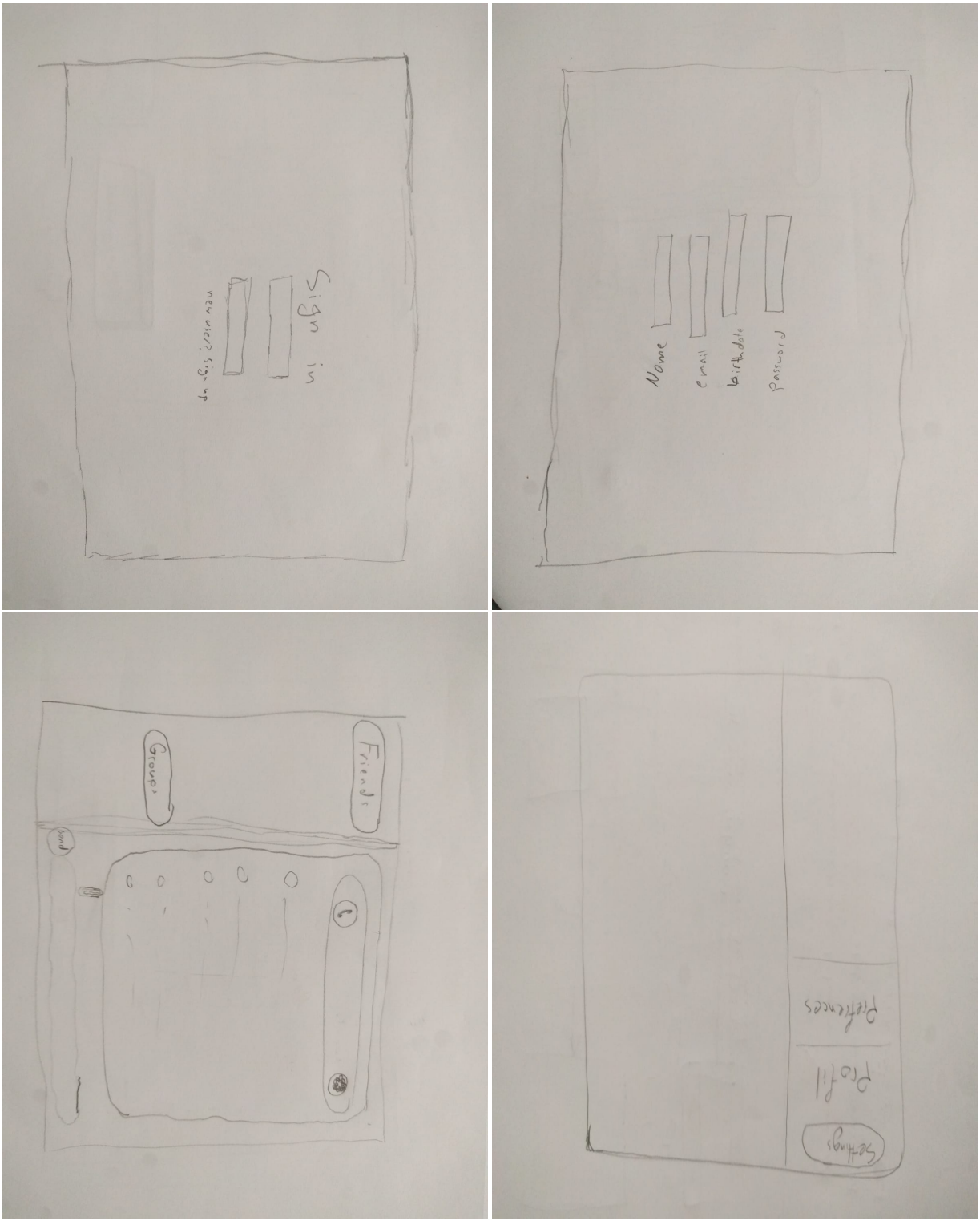
2.1.10 Story Identifier: CEYMCHAT010

Story name: User can add friends to a friendslist

Description
As as user, I want to add another user to a friendslist so I can have quick access to them.
Confirmation

- Users in the current users friendslist are shown in the GUI
- Users can be added to the userlist at run-time
- Code well documented
- Tests passed

2.2 User interface



3 Domain model

Diagram.pdf Diagram.pdf

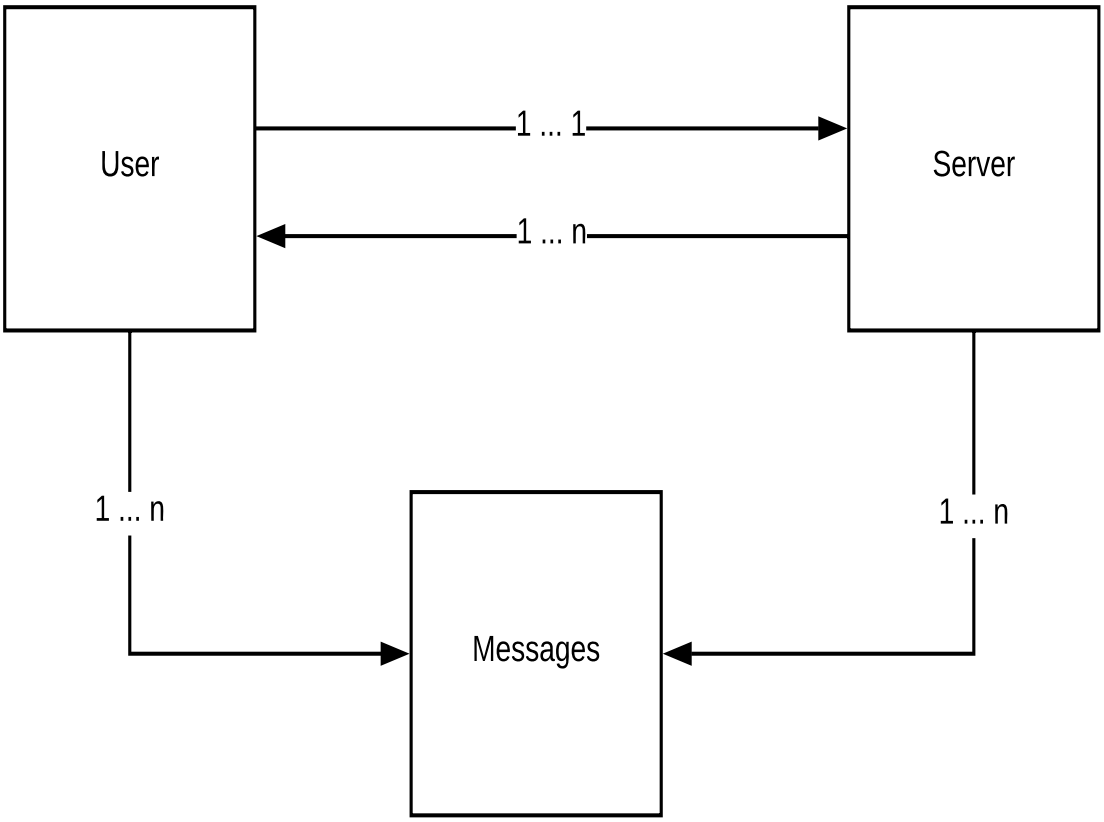


Figure 1: High-level UML-diagram of the application

3.1 Class responsibilities

Name	Module	Summary
ClientMain	Client	This is the runnable class of the Client. It launches the program.
ClientModel	Client	This class contains the model of the Client.
Connection	Client	Establishes a connection with the server and the
ClientController	Client	Controller class to function as input for model and to send updates to GUI.
FriendListItem	Client	Used to create a GUI element that displays a friend.
Message	model-lib	A generic class that encapsulates the data to be sent/received by the actors.
MessageFactory	model-lib	Factory for the Message-class. Narrows the generic element in Message to only be constructed with specified classes.
Command	model-lib	Can be sent by a client to the server to request the server to perform the issues command.
UserDisplayInfo	model-lib	Sent to the client containing information about other users.
SocketHandler	Server	Runs as a Thread to continously accept incoming client connection requests to establish connection between server and client.
ServerMain	Server	This is the runnable class of the server. It launches the server.
ServerModel	Server	This is the model for the server.
User	Server	A user object is created for each user in order to diffrentiate them and store relevant information.
Reader	Server	Every User has a Reader object. A Reader runs as a Thread to continously monitor and proccess incoming Messages from the user.
Writer	Server	Ever User has a Writer object. A Writer is used to send Messages to a Users client.

4 References