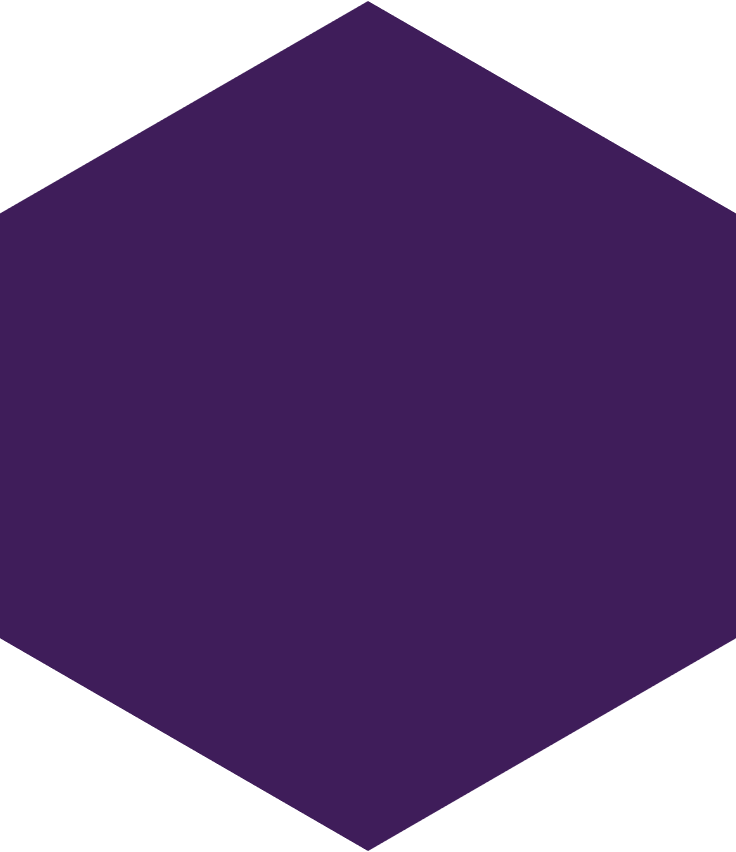
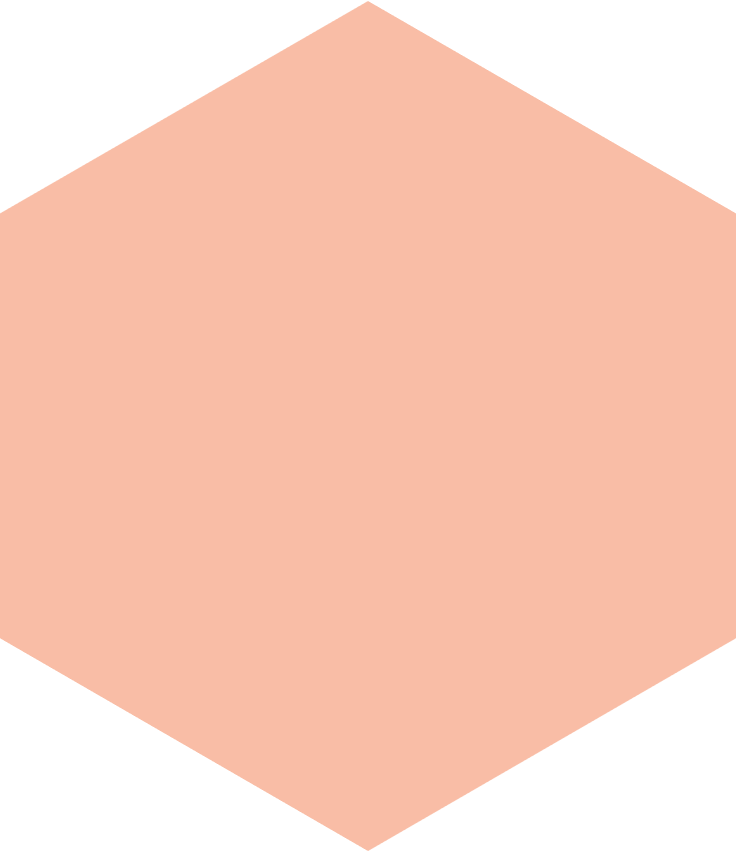


|  |
| --- |
| **Desktop Messenger Application** |
| FOR DOGE FINANCIAL |
| Aleksy RUSZALA @1902370 |
|  |





Contents

[Introduction 2](#_Toc24866751)

[Propose 2](#_Toc24866752)

[Scope 2](#_Toc24866753)

[Research 2](#_Toc24866754)

[Specific requirements 3](#_Toc24866755)

[Functional requirements 3](#_Toc24866756)

[Non-functional requirements 3](#_Toc24866757)

[Security 5](#_Toc24866758)

[Patterns 6](#_Toc24866759)

[Development methodology 7](#_Toc24866760)

[Project Timeline 8](#_Toc24866761)

[UML Diagrams 9](#_Toc24866762)

[User Guide 16](#_Toc24866763)

[Testing 17](#_Toc24866764)

[Resources 18](#_Toc24866765)

[Evaluation (and Improvements) 19](#_Toc24866766)

# Introduction

## Propose

The purpose of this document is to provide a detailed specification of details about creating project. This specification is intended to be presented and approved by the client and as a reference point for the developers to create applications to meet customer requirements.

## Scope

The project is to create an application for DOGE FINANCIAL, used for communication between company users. The program is to be characterized by a high standard of security, as the data transmitted is sensitive and requires special treatment and protection against third parties. Also, the program is to send data between users as soon as possible - in real time. The only place to store data is to be the database that is hosted on the company's server. Only verified employees can use the program, they will can access with the appropriate login and password.

## Research

Mainly research was focus to all application security issues available for windowed applications also real-time connection between programs. As a search result, I decided to use the SignalR framework to implement the connection in real time and implement rest-API. In order to secure the application, I chose client verification by token. Separating application layers into client-server-databases to isolate the database from direct requests. For data security, HTTPS protocol and various types of encryption were used.

# Specific requirements

## Functional requirements

#### Login to the application

The application should be available only to users who have a user account. Such a user must be able to log into the program with a username and password.

#### Sending messages

The user should be able to send messages to selected users. The message should be sent only to the selected user.

#### Receiving messages

The user should be able to receive messages sent by other users.

#### Starting a conversation

The user has the option to choose a user from the list with whom he wants to contact and start a conversation with him in a new window. The user must be able to talk to each other user individually.

#### Viewing the user list

The program should show a list of all available users in the system.

#### Message encryption

All messages should be secure and encrypted to prevent other users from accessing confidential information.

#### Minimal latency

Messages should arrive as quickly as possible.

#### Microsoft Windows 7 compatibility

The software should be compatible with Windows 7 and run smoothly.

#### Copy of messages

The program cannot store any information on the machine from which the goat user. They cannot be stored on disk.

#### Message sound

When receiving a new message, the program should communicate this by playing a sound or inform the user visually.

#### Remove old messages

Messages older than six months should be permanently deleted from the system.

## Non-functional requirements

#### Offloading the database

The program should not unnecessarily call the database with queries. Should use it only when it downloads data for the first time, or in case the data has changed.

#### Connection security

The connection should be secure and preferably encrypted. The program should only exchange information in this way. This reduces the chances of receiving valuable information by third parties.

#### Professional application template

The application should look professional and reflect the best possible user experience.

#### User management

The system administrator should be able to manage users through the program.

#### Work in a remote network

As the company grows, users should be able to communicate between branches that are away from each other and that use other networks.

#### Preventing multi-logging

The user should be able to log in only once at the same time, otherwise the previous login session should be deleted, and the program should log out user on last PC

#### Session

Users should be automatically logged out when the server shuts down, this also applies to blocking users who want to resume sessions after restarting the server. The user should be forced to log in again.

# Security

#### Password encryption

The user's password is encrypted to prevent intercept by third party. The password is already encrypted at the stage of sending the request to the server. The password in the database itself is also not stored in a pure form, so that anyone hacking into the database could not steal it or restrict administrators from reading passwords. Password checking is based on a comparison of the encrypted password in the database with that in the client request.

#### Message encryption

Like the password, messages are also encrypted to protect them from being read by unauthorized persons. Compared to the password encoding used, the messages are hashed with a more advanced script that uses salt. In this case, messages received can be decoded by the customer who has the appropriate salt. Messages are also stored in coded version.

#### Tokens

The server uses external tokens to verify connections. The token is generated by the server when the user logs in and his lifetime lasts 5 minutes to increase security. Along with the token, the client also receives a key to refresh the token, which after it expires will generate a new token and a new key. Each connection requires a valid key to be provided by the client, then the server checks its correct and allows it to use authorized queries. If the token is not valid, the server will reject the connection.

#### Session

The server monitors the issued keys during the session if the program does not have a key in memory. Rejects the call. The same happens with SignalR connections. All session memory is reset when the server is restarted, which prevents the use of old keys or previously established connections.

#### Hold data in memory

The client does not write data to disk but only stores current data in memory during the window's life. When the window closes, all data downloaded from the server is lost. This requires downloading fresh data from the server when reopening the message window or the main window.

#### Application Layers

Dividing the program into layers allowed to increase security e.g. prevent direct queries to the database that is stored locally.

# Patterns

#### Dependency injection

using this pattern will make the code less associated with other elements, and thus be more open to extensions.

#### MVC/MVP

although WinForms is not designed for this pattern, its use a benefit here. Separation of application responsibility between three units; model, view and controller. These three units work together to support the entire system and its logic. The main advantage is the fact that each of the units can operate independently of the other which gives modularity and reusability.

#### Repository pattern + Entity Framework

Using this pattern following benefits can be mentioned: decouples an application from the persistent framework and promotes testability.

#### Singleton

This this program provide access to the same instance of class for example in case need to access to same set of data globally.

#### Real-Time

Not directly but using the SignalR framework server and client have the option of real-time invoke methods with each other

#### Application Layers

The main advantage of layered architecture is the division of logic between components. Specific layers apply only to the logic that applies to this layer. It also increases safety. Access to the database is behind server layer, that means no one from outside the device (in network) can access it.

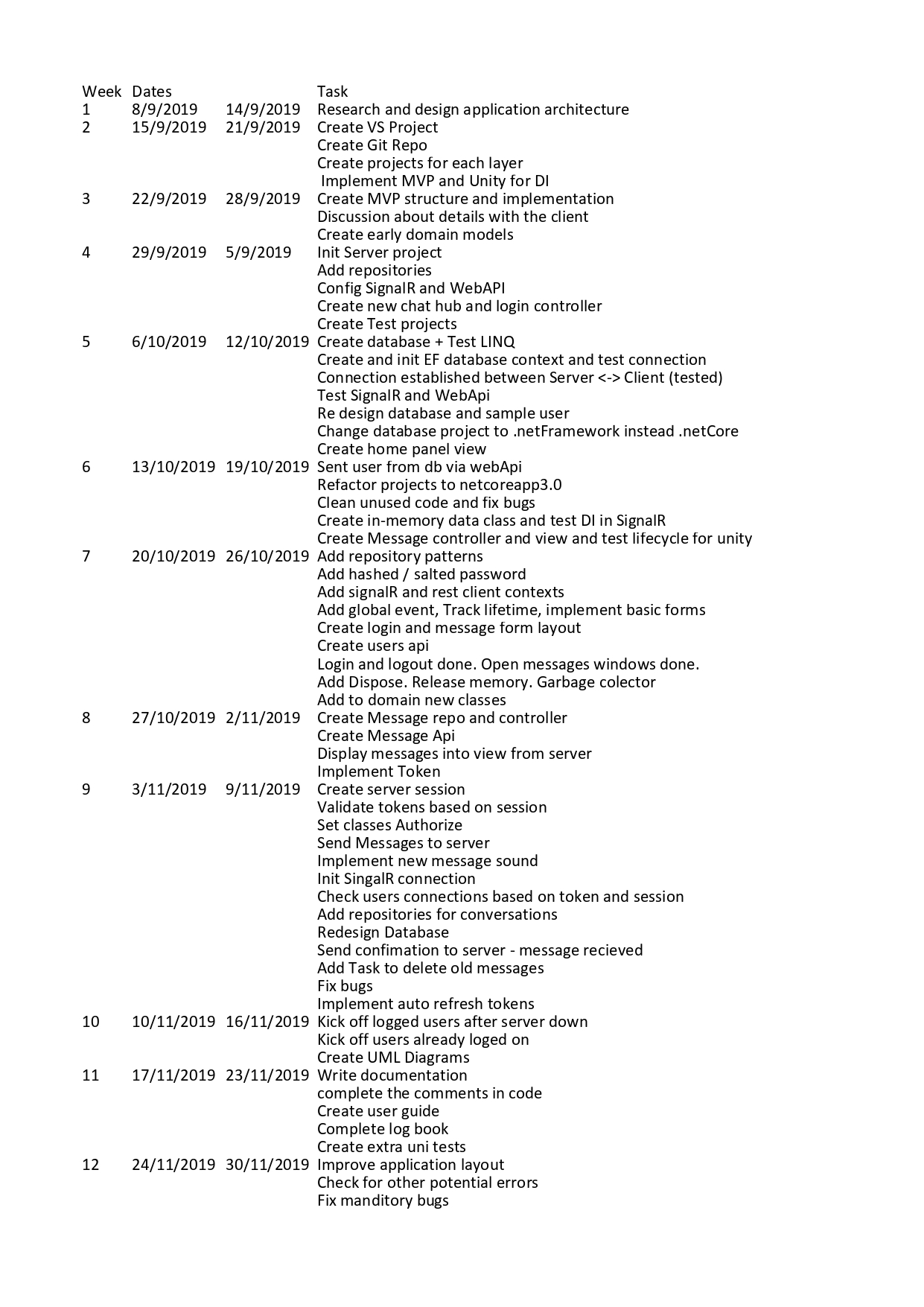
#### Others

There are also several other patterns and approaches to be found in the project.

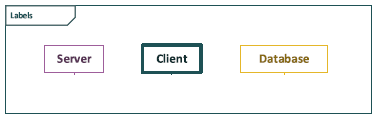
# Development methodology

The project is created using the Agile methodology, which will allow us to implement subsequent application modules and collect early feedback from the client. The whole process was broken down by me into smaller tasks and into weekly sprints which, depending on the priority, were implemented into the project. When creating the application, I was in constant contact with the client, which is why details could be agreed on a regular basis. The task was clearly defined and in case of problems I could freely navigate them to get next working versions of prototypes, which in turn could be tested regularly and potential errors or future failures can be detected. Thanks to the comments from the client, it was possible to bring the application to the right track and provide subsequent versions in line with the client's expectations, meeting further requirements. Currently, most requirements are met, and applications run independently without problems. This does not mean that the applications cannot be further developed to meet all expectations. Using this methodology, there was a risk that the product would not work well enough to allow use, although the risk was quite low due to well defined tasks in the backlog. Despite the changes in requirements during the project, the changes were implemented by modifying several tasks and refactoring several previous lines of code created. With each iteration, the program more and more reminded the target program to as a result give its first full working version.

# Project Timeline



# UML Diagrams

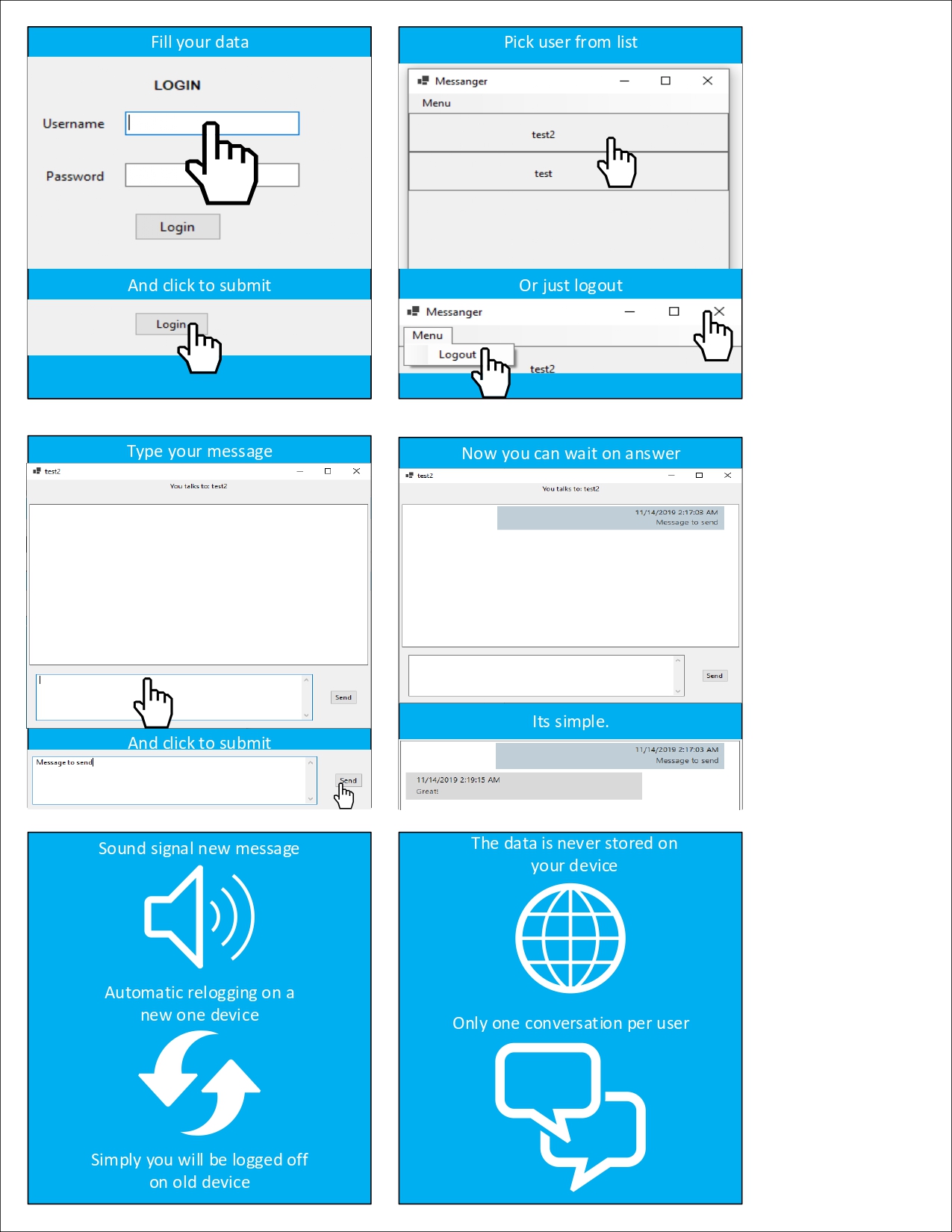


#### Client class diagramA screenshot of class diagram for client application.

#### Server class diagramA screenshot of class diagram for server application.

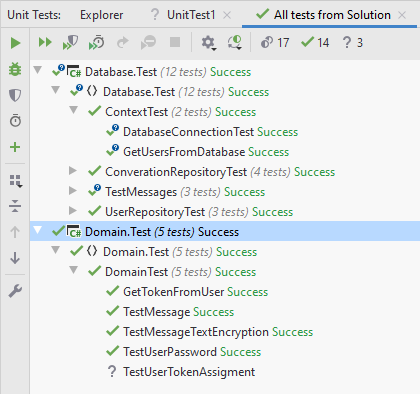
#### Use case diagram

# User Guide

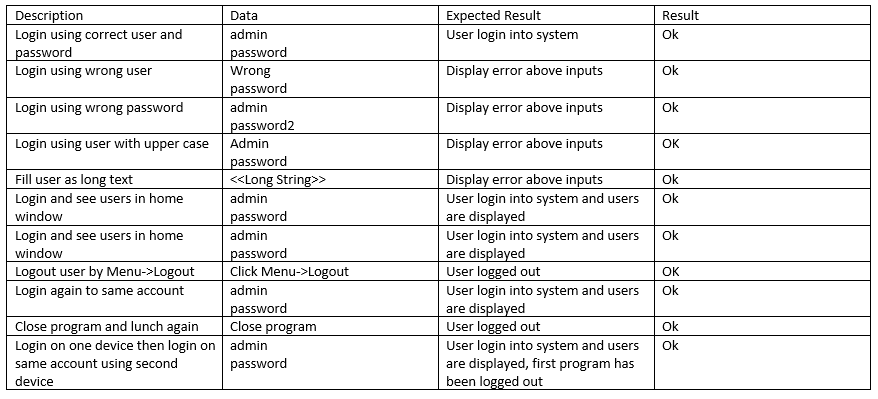


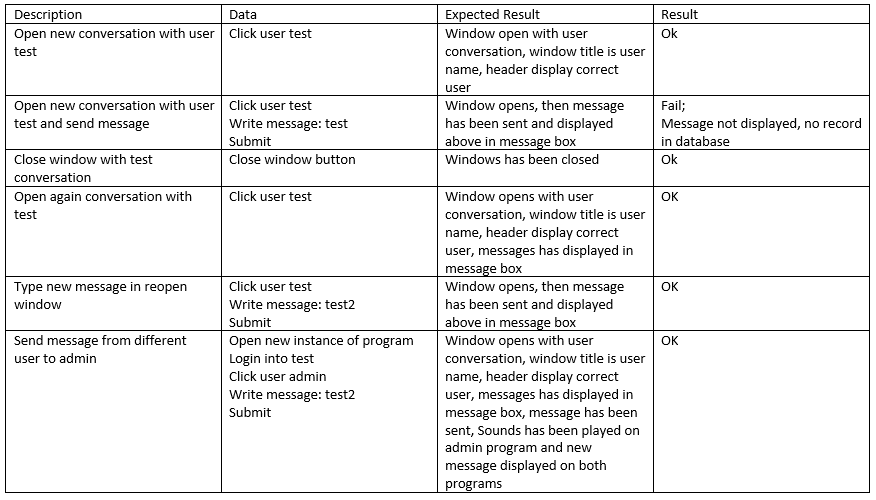
# Testing

On the one hand, the test object was the application code, on the other hand the application itself on the user's side. To this end, techniques called white and black box tests were used. Testing the white box allowed to test the program based on the internal structure of the application. To achieve this, applications were tested using unit tests where they could be used.



Because the application has a complex structure, it uses several frameworks and the logic has been divided between the server and the client. Therefore, the program was also tested as a product. For this purpose, we will use a technique called Black Box. Testing consisted of testing individual functions of the application by following specific steps and using various types of input data (normal, exceptional and extreme); making sure that the data received during the tests are as expected.





# Evaluation

The program meets all functional requirements and most non-functional ones. The program has many security features that will prevent attacks and attempts to steal sensitive data. The architecture of the program allows for further extension and improvement of the program in the long term, also to easier maintain application. Currently, the application can operate on the local network and on the Internet after appropriate configuration. Using the appropriate frameworks, it was possible to relieve the database of unnecessary queries (pulling). The main assumption was to focus more on the security and structure of the application than over its appearance because the view of application is independent of the entire application, therefore the application template needs improvement or purchase a ready-made template. Now, the template is rendering rather slowly and is noticeable to the user. Messages in conversations are currently not copiable for security reasons, which should be noted (requirements) when upgrading the template.

Some of the tests operate on the base database, therefore they should not be run after application release. Those tests should be converted to work on in-memory database. The application currently has several known and detected bugs inside the server code. These errors do not significantly affect the operation of the program and their repair is not quite complicated and can be implemented at the next iteration. One of them is the error that appears after the newly added user that user want to contact. The error appears because there is a problem creating a conversation when for first open conversation the window. Currently, to continue the conversation, all is need that reopening window.

The customer also has the option of implementing the connection certificate validation in the appropriate place in the program. This has not been configured yet. Configuration also requires the appropriate setting of ports and IP addresses for the proper connections of the application. Correct server-side firewall configuration will also be required.

Currently, users can enjoy shared communication in a secure environment after accessing the program. It is also important that login data should be manually set by administrator and there is no tool to do it.