



## ET1446: SOFTWARE DEVELOPMENT FOR TELECOMMUNICATION SYSTEMS

*This document is about the design of our project modules. It is a single document for the whole application, so all parts of it are being analysed here.*

# DESIGN DOCUMENT V1.0

## Group: SWAT Kats

Sokratis Papadopoulos	Jibraan Singh Chahal
Johannes Grohmann	Jyoti
David Alarcón Prada	Akanksha Gupta
Gautam Vij	Rohit Raghav
Inanc Gurkan	

# DESIGN DOCUMENT

## 1. PREFACE

This document is about the design of our project modules. We provide all the information regarding class diagram and both detailed design and unit test plan for every module that we are going to develop in our application, covering all requirements set.

In the following we will first provide a glossary with a list of abbreviation in chapter 2. After that, we present the general class diagram in chapter 3 and later chapters refer to the design of each module in our project.

### VERSION HISTORY OF THIS DOCUMENT:

**Release v1.0 on 2015-04-30**

- Initial release

## 2. GLOSSARY AND ABBREVIATIONS

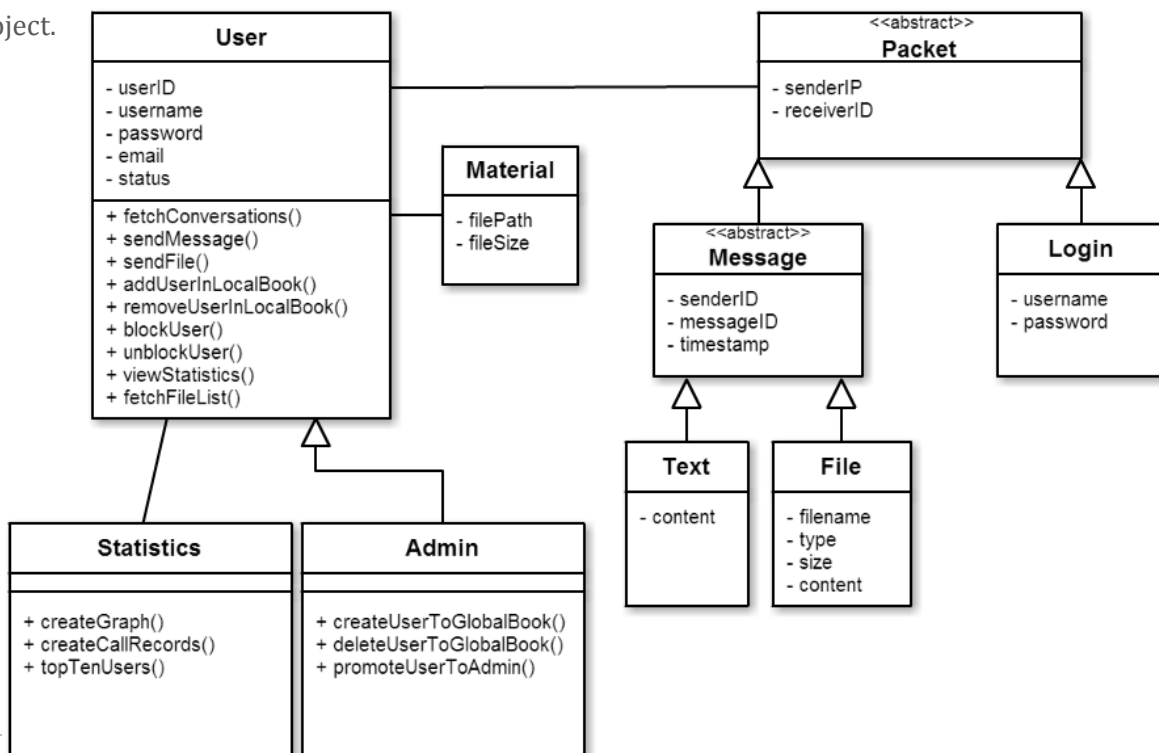
In this section are defined technical terms that are in this document in alphabetical order.

- **REQ:** requirement

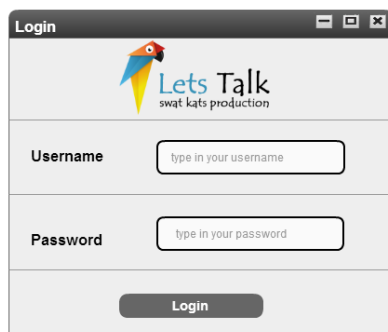
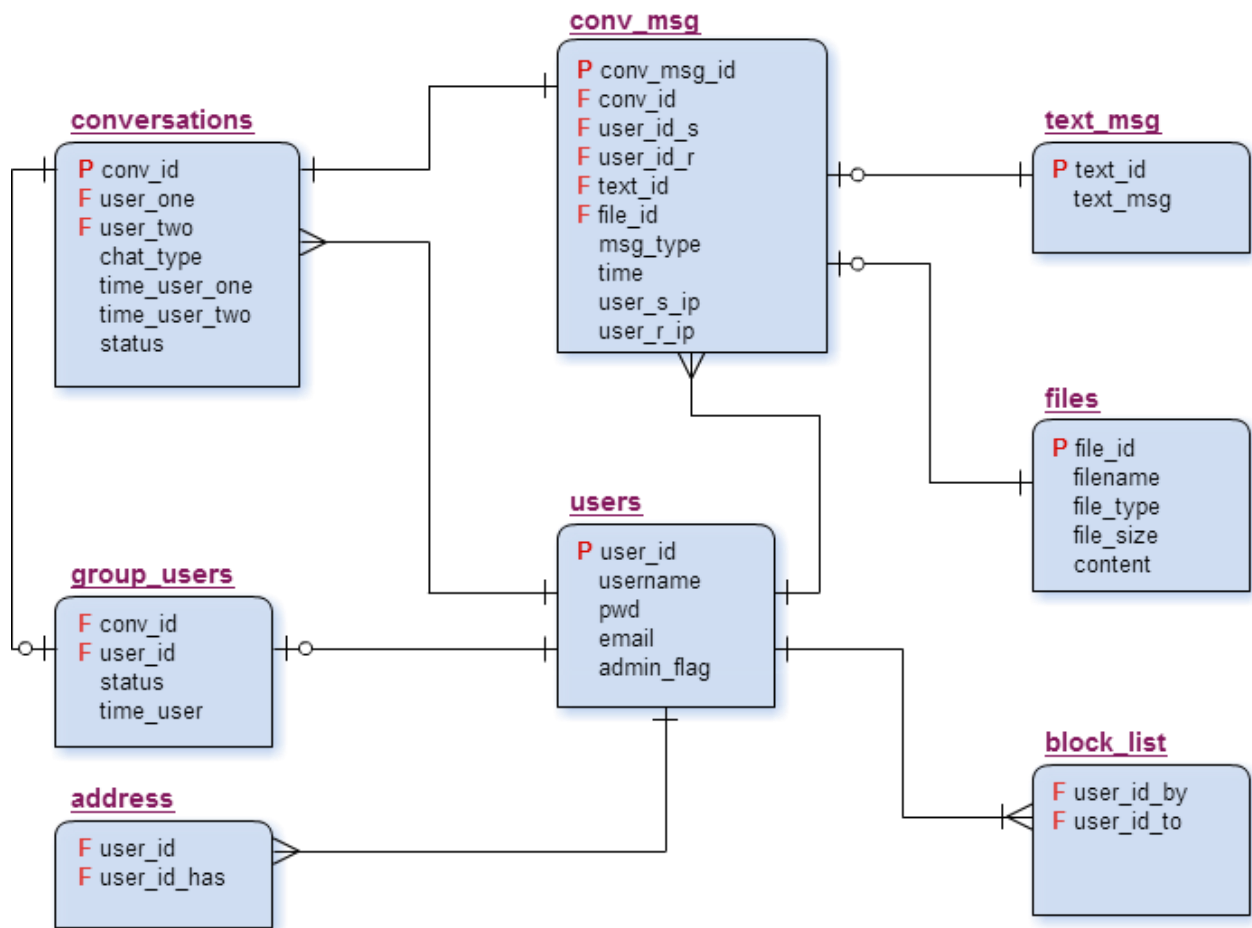
## 3. DIAGRAMS

### 3.1 CLASS DIAGRAM

Here you can take an overview of the classes that are going to be used in the implementation of the project.



### 3.2 ER DIAGRAM



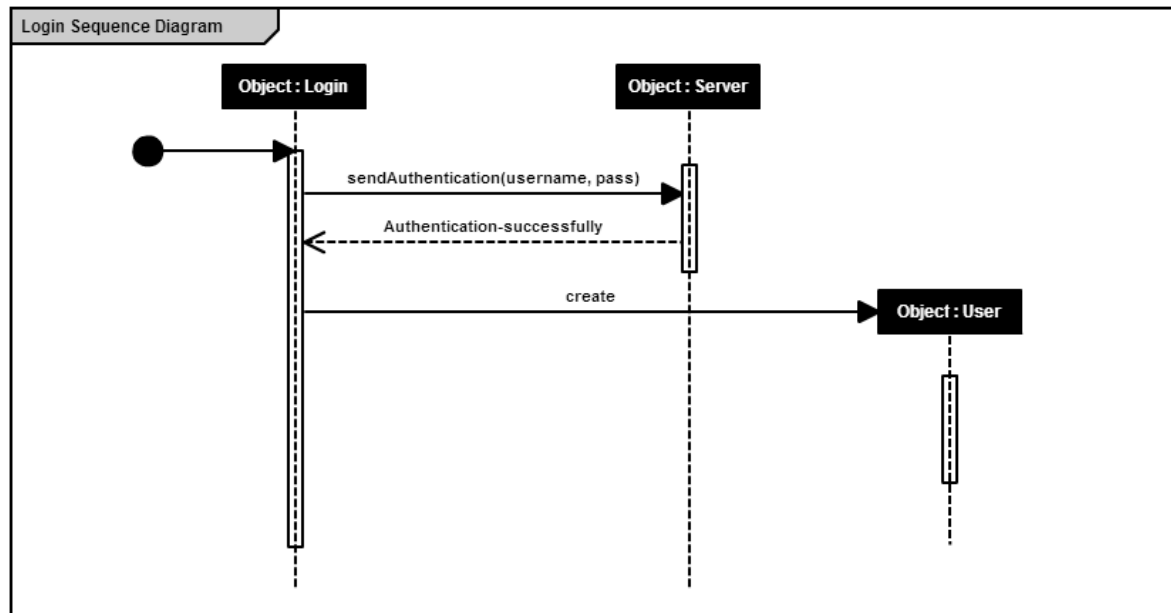
## 4. LOGIN

Login module is self-defined. It covers the authentication of the user in our application. It is highly important as we want only authorized users of the company to access it and letting a third-party access our highly secure and encrypted system will be catastrophic. Login happens by providing the username and password, sending them to server, matching them with the appropriate record -authenticating the user- and after letting him access the application interface with all functionalities (according to his position - admin/non admin). Login page will be the first that shows up, once the application is running.

This module satisfies user requirement 1 (REQ\_USR\_1), while it doesn't interact with other modules, but only with the basic GUI that appears the login window for letting user add the input (username & password).

### 4.1 DETAILED DESIGN

The following sequence diagram shows how the login module is implemented.



## 4.2 UNIT TEST PLAN

The module's functionality can be fully tested using the following various unit tests.

### 1. **Test:** TEST\_LOGIN

**Purpose:** It will test that the login authentication works properly

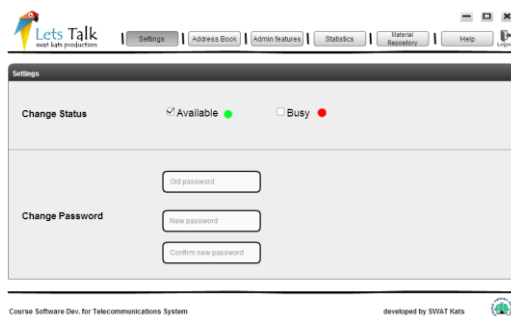
**Requirements:** REQ\_USR\_1

**Environment:** executed the application (login windows appeared)

**Operation:** run the executable providing as parameters a valid username-password. Also run it providing a false username & password.

**Expected result:** login for authorized users / failure message for non-authorized

**Result:** success/failure



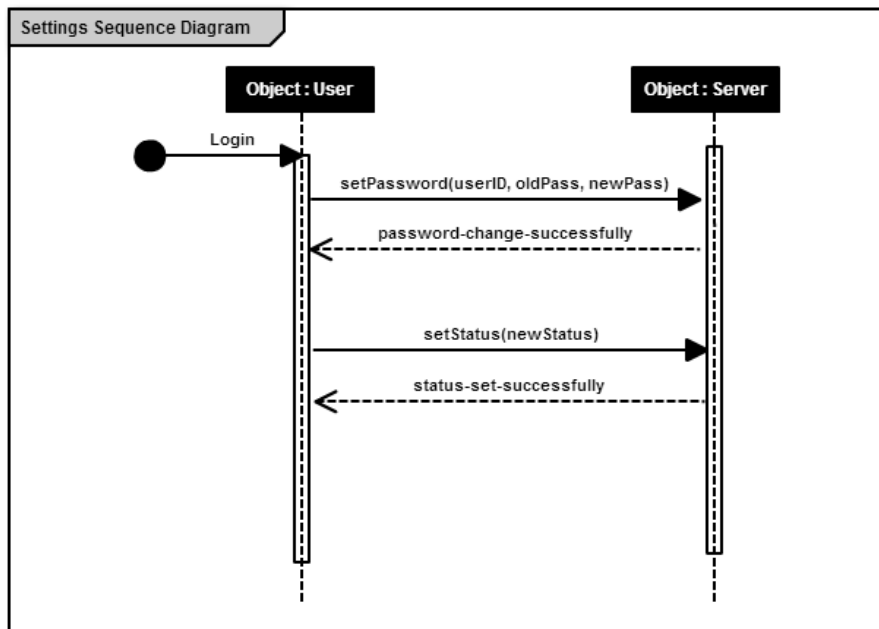
## 5. SETTINGS

Settings module is all about configuration of some user attributes. That includes their current status (Available or Busy), but also their password. The above can be adjusted to user's desires by editing them through our Settings module located at the homepage menu of the application. You can take a look at how it will approximately look like at the following picture.

This module satisfies user requirement 3 (REQ\_USR\_3) and user requirement 4 (REQ\_USR\_4), while it doesn't interact with other modules, but only with the basic GUI that of course includes the settings module. That means that, from the homepage of the application users are able to click to settings on the menu bar and they will be navigated in the above windows where they can configure their settings.

## 5.1 DETAILED DESIGN

The following sequence diagram shows how the settings module is implemented.



## 5.2 UNIT TEST PLAN

The module's functionality can be fully tested using the following various unit tests.

### 2. Test: TEST\_SETTINGS\_STATUS

**Purpose:** It will test that the changing of user's status functions properly

**Requirements:** REQ\_USR\_3

**Environment:** user must be logged in our application

**Operation:** run the executable providing as parameters a valid username-password pair and the desired status (Available/Busy).

**Expected result:** status is changed/set according to user's desire

**Result:** success/failure

### 3. Test: TEST\_SETTINGS\_CHANGE\_PWD

**Purpose:** It will test that the changing of user's password functions properly

**Requirements:** REQ\_USR\_4

**Environment:** user must be logged in our application

**Operation:** run the executable providing as parameters a valid username-password pair and the desired new password.

**Expected result:** password is changed according to user's desire

**Result:** success/failure

## 6. ADDRESS BOOK

XtremeSecurity has all its employees' list in a central database, i.e. global address book. If a user wants to chat

to other user then he should first add him to his local address book. Hence, this module is all about adding user, removing user, blocking and unblocking user from global address book.



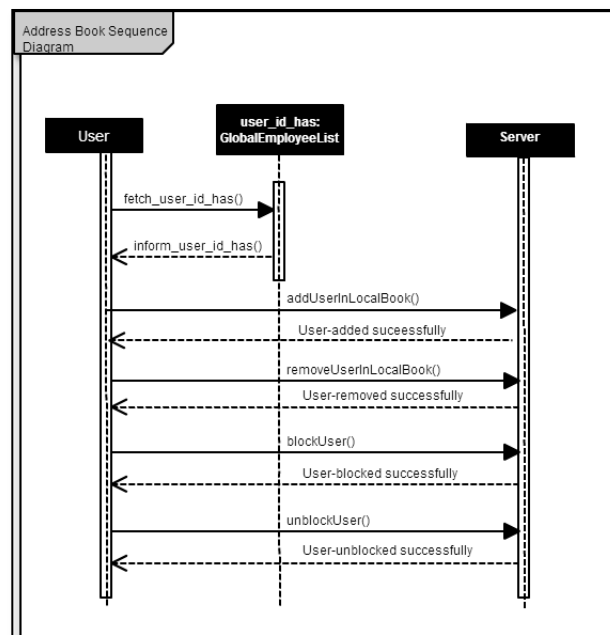
This module satisfies user requirement REQ\_USR\_2, while it doesn't interact with other modules. After a user clicks on address book, where above window opens up, a user\_id is selected to add/remove/block/unblock and database updates accordingly.

If a user:

- i. **Adds** another user then he will be able to chat with him.
- ii. **Removes** a user then he will not be able to chat with him but history will be preserved.
- iii. **Blocks** a user then both users can't chat with each other and can't find each other on address book.
- iv. **Unblocks** a user then both users will be able to chat with each other but he has to add him again.

## 6.1 DETAILED DESIGN

The sequence diagram on the right shows how the login module is implemented.



## 6.2 UNIT TEST PLAN

The module's functionality can be fully tested using the following various unit tests.

### 4. Test: TEST\_ADDBOOK

**Purpose:** Add/remove/block/unblock other users according to user' need.

**Requirements:** REQ\_USR\_2

**Environment:** user must be logged in our application

**Operation:** run the executable providing parameter as a valid username and choosing a valid option among add, remove, block, unblock.

**Expected result:** If a contact is added, then it should appear in user's address book and they should be

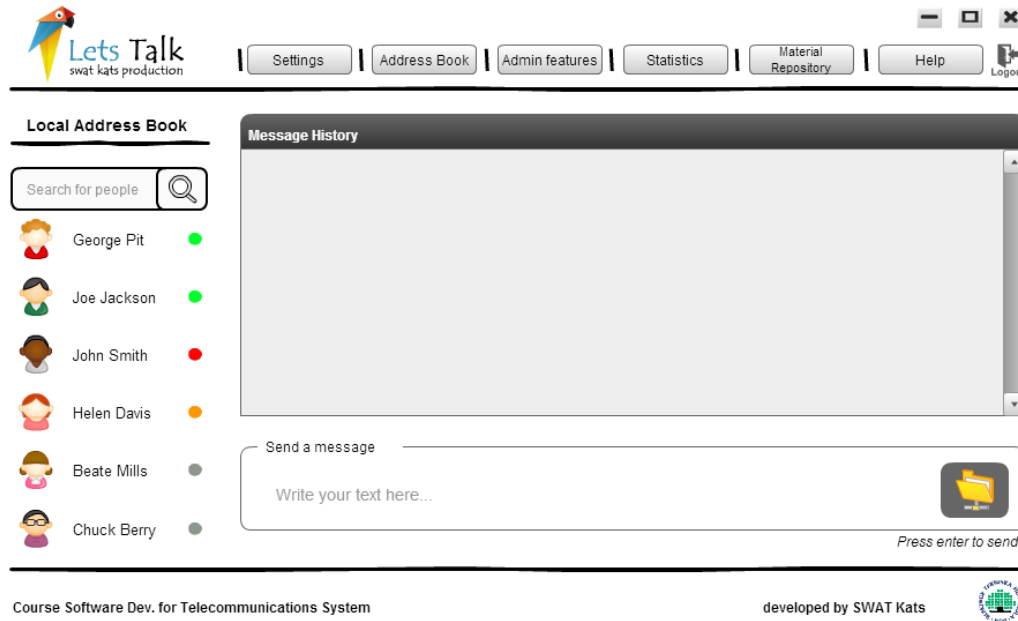
able to chat with each other. If a contact is blocked then it should not appear in address book and they should not be able to chat with each other.

**Result:** success/failure

## 7. TEXTING

The support of a chat feature is the main purpose of **LetsTalk**.

The Chat window is positioned directly on the main page and enables chatting and text messaging with people in the local address book. See the following picture for the first draft of the GUI.

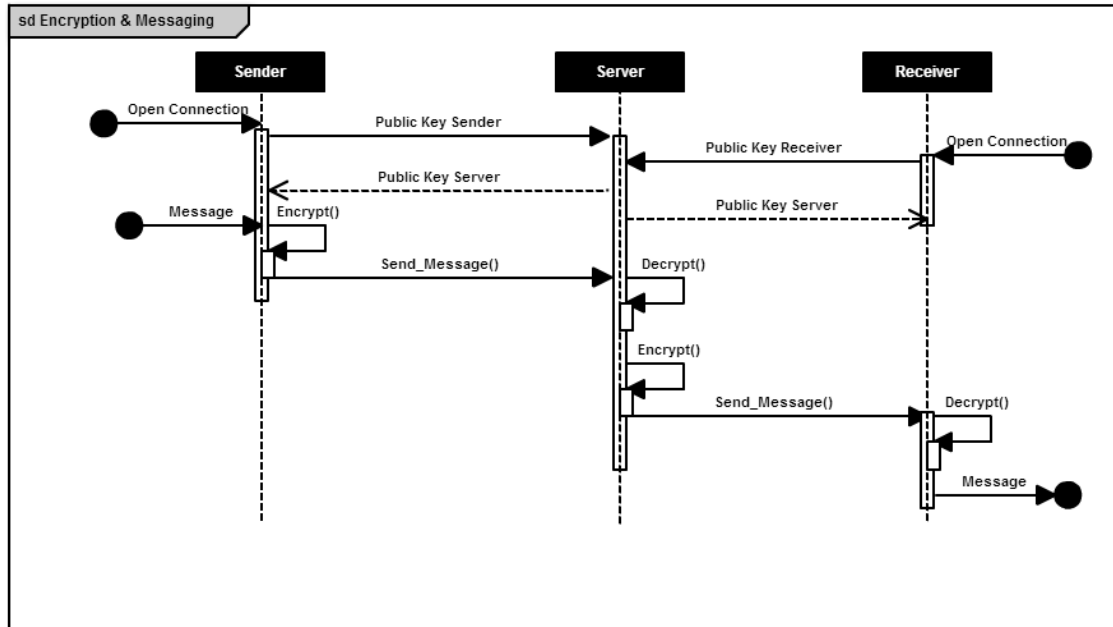


A User selects a contact in the address book and can type messages in the shown text area. If the selected receiver is online, he/she will receive the typed text message in the message history panel and will be able to respond by typing an answer in his/her own message field. These messages have to be encrypted. Therefore a key exchange has to be made beforehand, e.g. when the user goes online.

### 7.1 DETAILED DESIGN

The protocol is handled according to the following sequence diagram. Before exchanging messages, both the sender and the receiver have to exchange keys with the server in order to guarantee secure transmission.

The message can then be decrypted at the server, to store the data for the history and the statistics, and is then forwarded to the actual receiver of the message.



## 7.2 UNIT TEST PLAN

The module's functionality can be fully tested using the following unit test.

### 5. Test: TEST\_MESSAGE

**Purpose:** Successfully transmit a message from the sender to the receiver.

**Requirements:** REQ\_USR\_5

**Environment:** Both sender and receiver have to be online and exchanged keys with the server.

**Operation:** Transmit the message to the required receiver.

**Expected result:** The message has to be received by the receiver. The message shall be the same, as the sender sent it, e.g. no artifact through encryption should occur. If the receiver is an invalid user an exception or a notice should be thrown and the message should not be transmitted. No other users except the specified receiver should get the message.

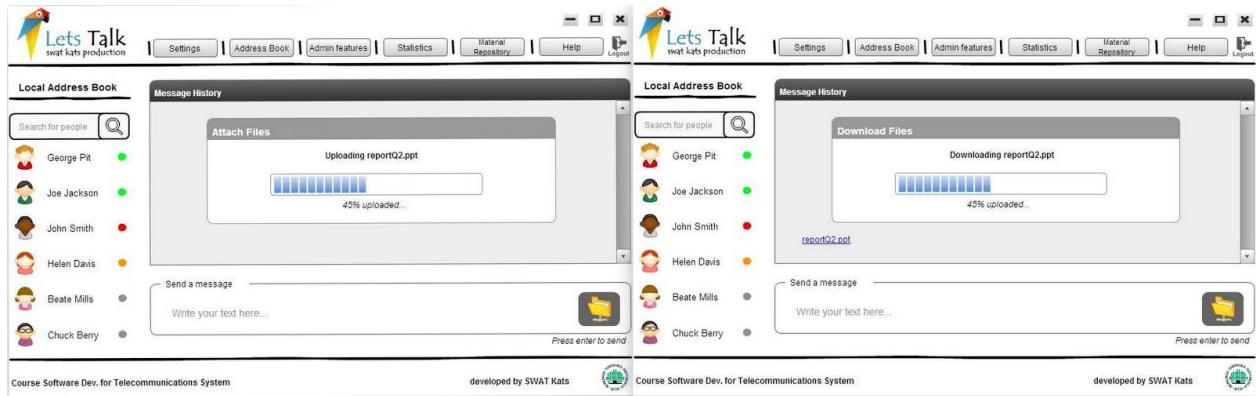
**Result:** success/failure

## 8. FILE TRANSFER AND MATERIAL REPOSITORY

File transfer module is about file sharing among the users. Users can send a file to the other user and download the received file. GUI shows statistics on the used and remaining time for upload/download and a time estimate for completion. GUI for file transfer module will look as shown in the image.

The material repository defines the accessibility of global database of the company's personal files (MOUs, brochures etc.) to all the users (on site or in office). GUI for this module will be same as for the file transfer. So, any user can download the files from the central database on any hourly need.

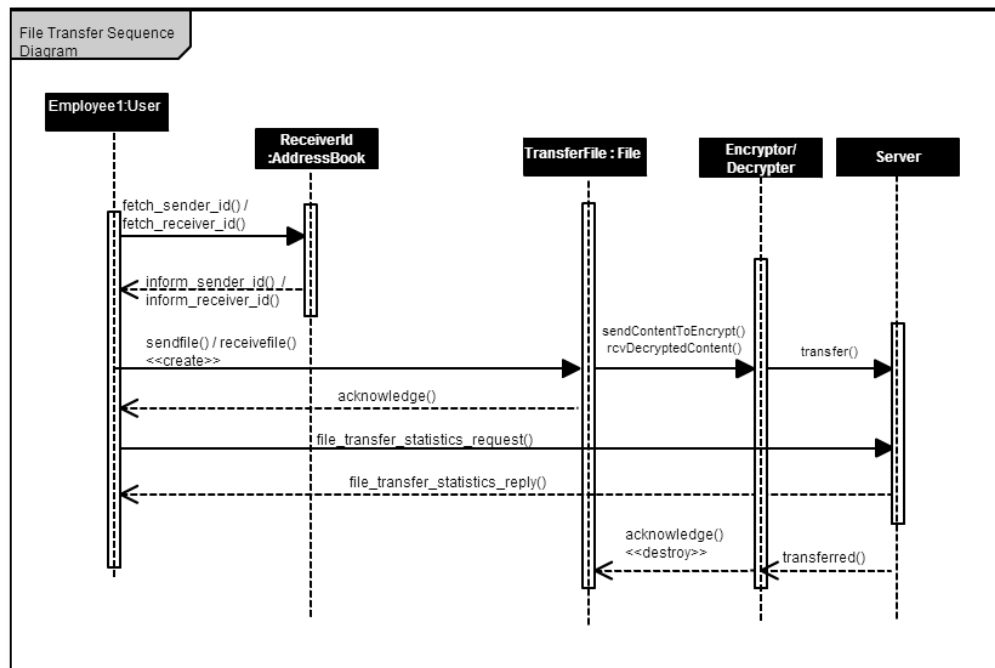


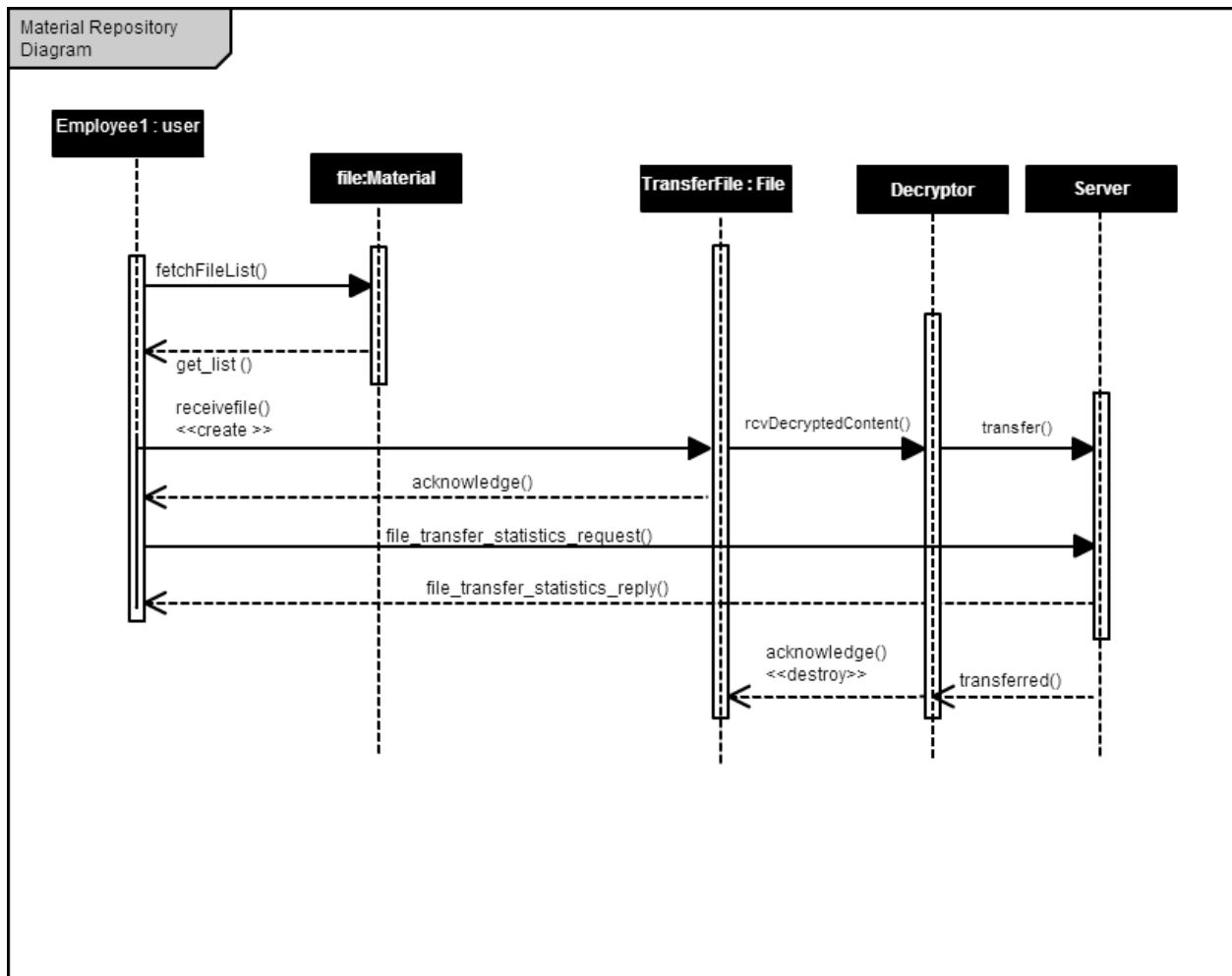


-This module satisfies user requirement 5 (REQ\_USR\_5). It interacts with address book module to select the receiver, encrypts the file and uploads it to server to be delivered to the selected user.

### 8.1 DETAILED DESIGN

The following sequence diagram shows how the file transfer & material repository modules are implemented.





## 8.2 UNIT TEST PLAN

The module's functionality can be fully tested using the following unit tests.

### 6. Test: TEST\_FILE\_TRANSFER

**Purpose:** System shall allow a user to send files

**Requirements:** REQ\_USR\_5

**Environment:** User must be logged in our application

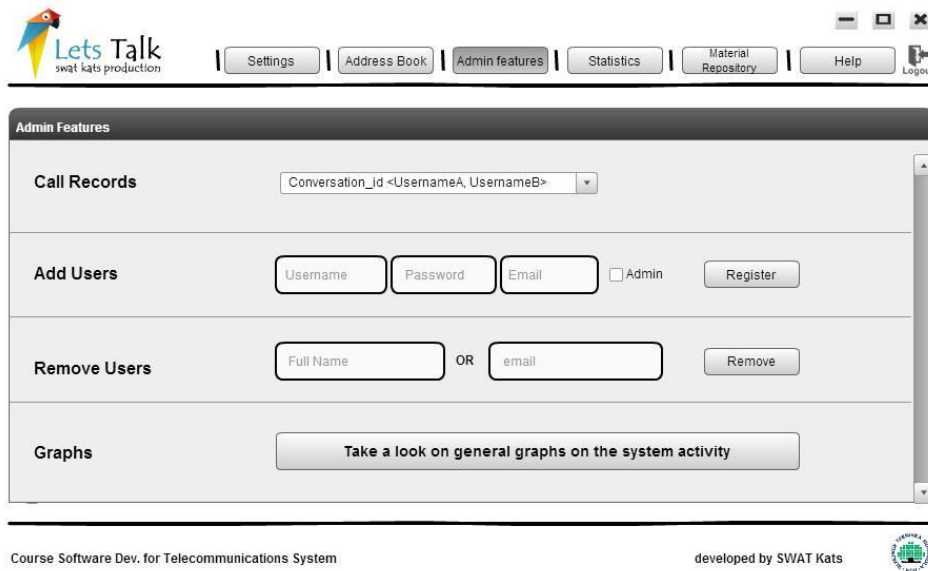
**Operation:** Run the executable providing as parameters a valid username-password pair; file name and the receiver.

**Expected result:** File is received by the receiver and is able to download the file

**Result:** success/failure

## 9. ADMIN TOOLS

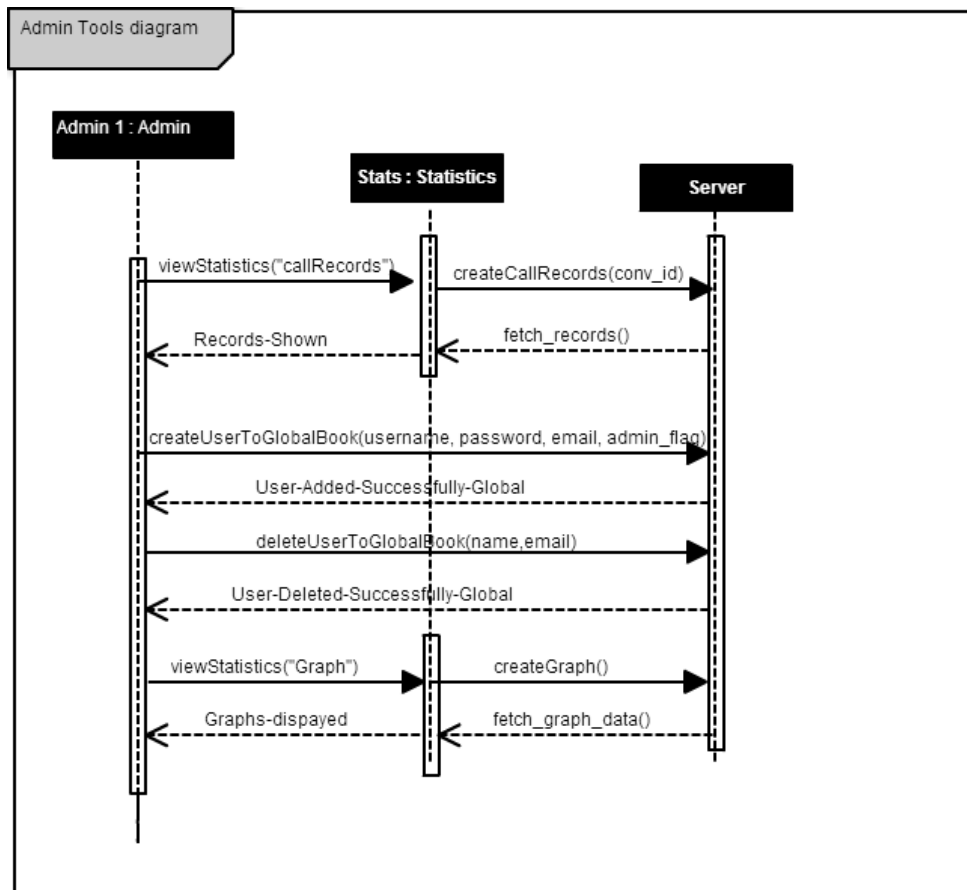
Admin tools module is about the functions that an admin has. The Admin is a normal user with some extra features at hand. The admin can Add/Remove a user GLOBALLY - Change the users' list in the main database or can take a general idea of the chat statistics by viewing the graphs and call records. The following picture represents how the admin tools page will approximately be.



This module satisfies user requirement REQ\_USR\_6. A user can also be made admin by any other current admin. This module interacts with the “Address Book” module -- when a new user is added, others users can add the new user to local their address book, and when a user is deleted, it is removed from the local address books of all the users who had added the “deleted user”.

### 9.1 DETAILED DESIGN

The following sequence diagram shows how the settings module is implemented.



## 9.2 UNIT TEST PLAN

The module's functionality can be fully tested using the following unit tests.

### 7. **Test:** TEST\_ADMIN\_TOOLS

**Purpose:** Add/remove users globally, view the statistics of the chats, or make another user admin

**Requirements:** REQ\_USER\_6

**Environment:** (Admin) User must be logged in our application

**Operation:** Run the executable providing as parameters a valid username-password pair

**Expected result:** Admin can view the conversation call records. If a user is added globally, other users can add the new user in their local address book. If a user is deleted globally, it should be removed from the local address books of every user that has previously added the deleted user. Admin can see a general graph of all the chat activities in LetsTalk

**Result:** success/failure