BLEKINGE TEKNISKA HÖGSKOLA
KARLSKRONA, SWEDEN
May 14, 2015

Lets Talk
swat kats production

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 V1.1

Group: SWAT Kats

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

Publication Date: May 14, 2015 **Version Number: V1.1**

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 v1.1 on 2015-05-14

- Modified section 4, added interaction of login module with other modules

- Modified section 5, added interaction of settings module with other modules

- Modified section 6.1, changed the UML diagram and used UML ALT.

- Modified section 6.2, added different test cases for add, remove, block, unblock according to user requirements in SRS.

- Modified section 7, separated encryption and messaging

- Modified section 8, encryption keys are added only during connection not for every message.

- Modified section 9, split the sequence diagram into 3 subsections for 3 admin tools

Release v1.0 on 2015-04-30

- Initial release

2. GLOSSARY AND ABBREVIATIONS

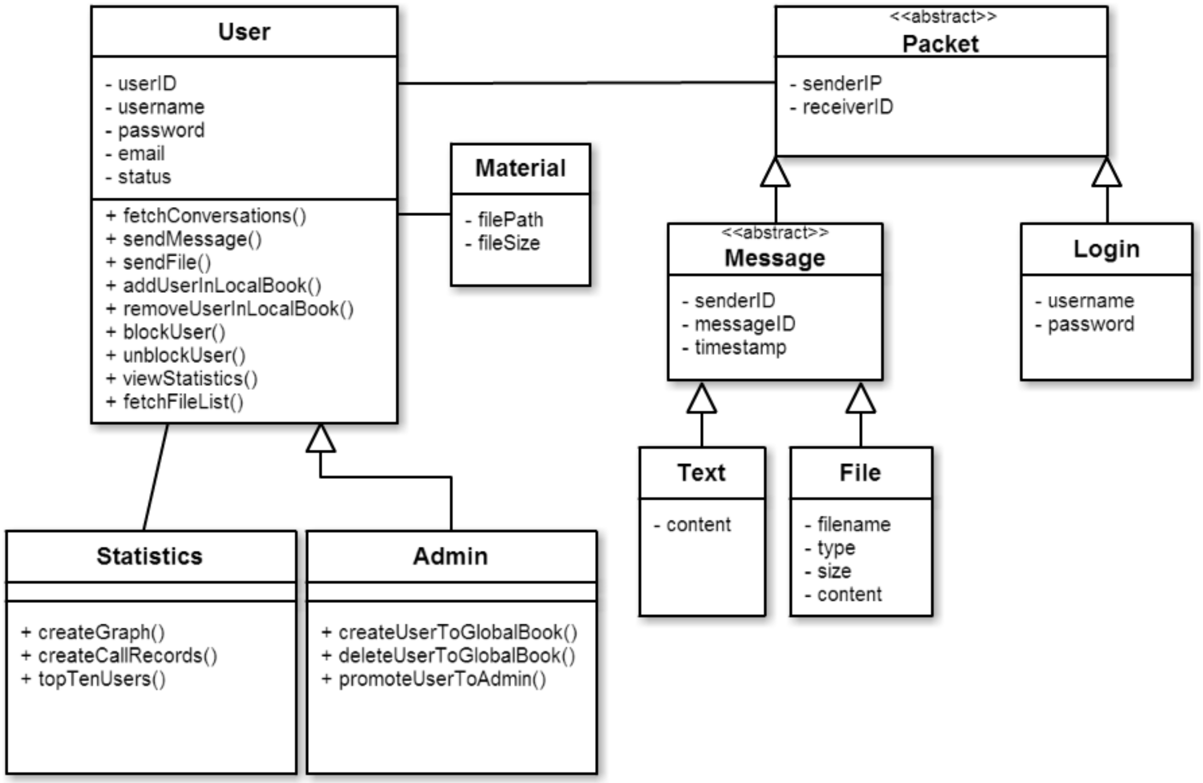
In this section are defined technical terms that are in this document in alphabetical order.

- **REQ**: requirement

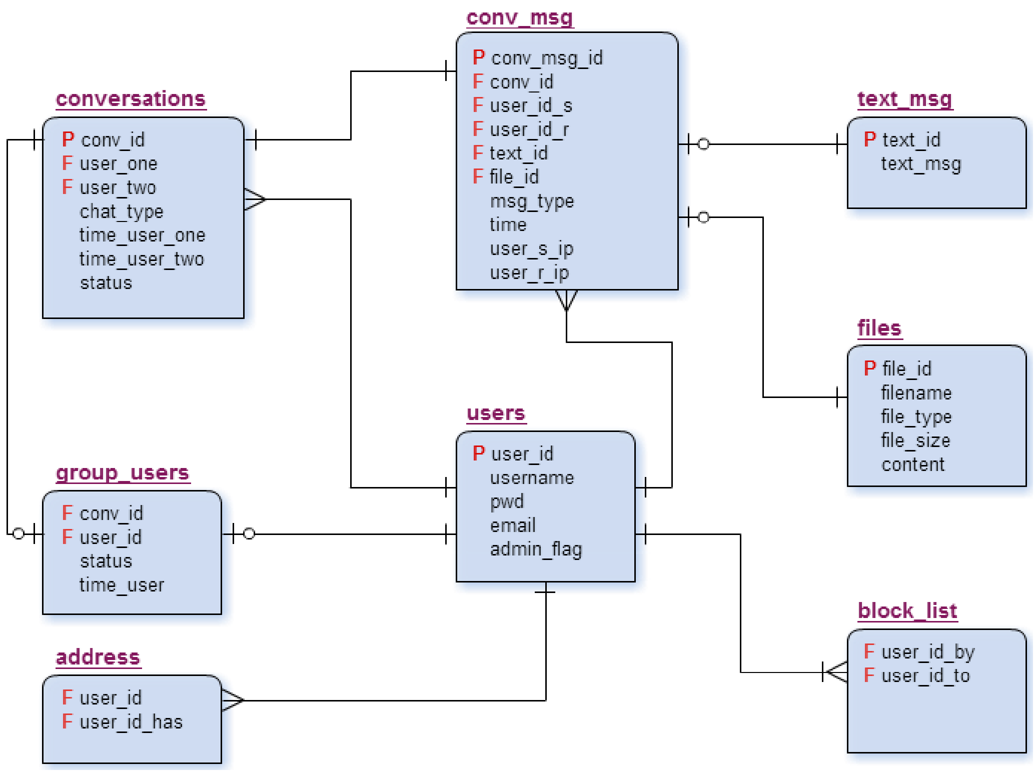
3. DIAGRAMS

3.1 CLASS DIAGRAM

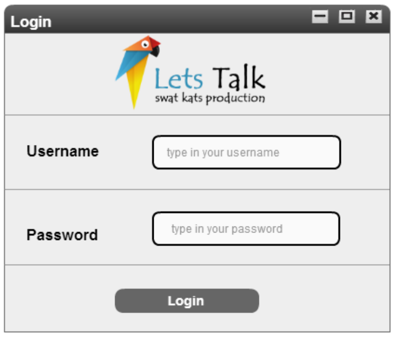
Here you can take an overview of the classes that are going to be used in the implementation.



3.2 DATABASE ER DIAGRAM



4. LOGIN

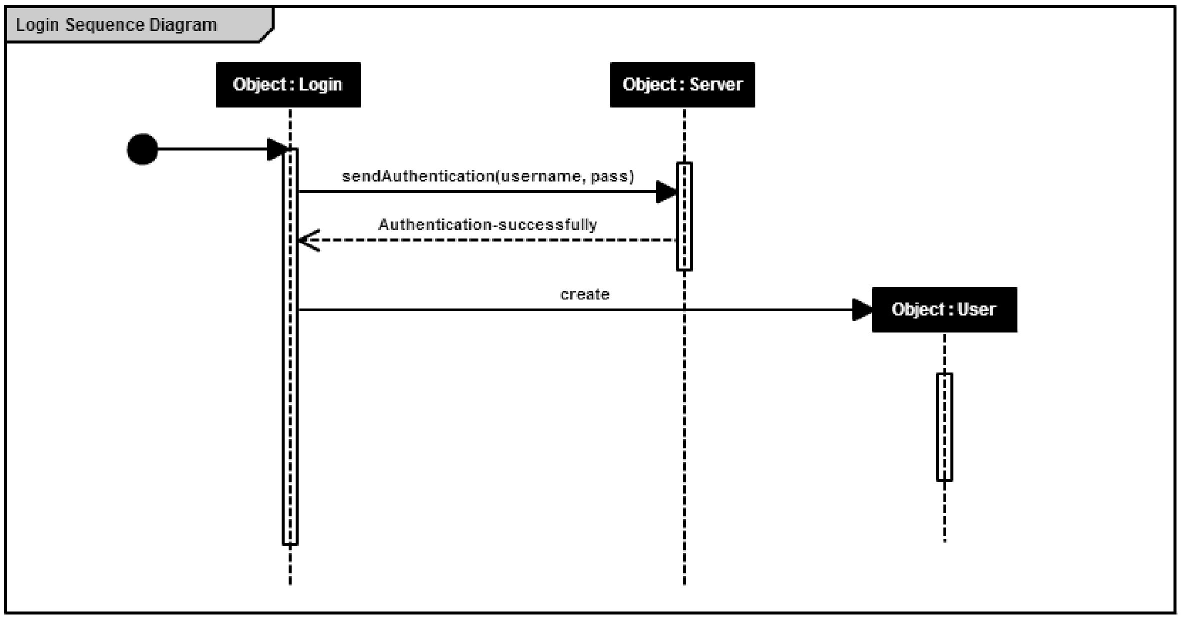


Login module is self-defined. It covers the authentication of the user in our application. It is highly important as we want only authorized users of the company to access it and letting a third-party access our highly secure and encrypted system will be catastrophic. Login happens by providing the username 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 interacts with all other modules. That is why once the user logins in our application his details is stored into a User object (after fetching them from database, using his username and password as credentials) and though that User object which defines who uses our application, all functionalities are being affected and run. Of course everything is connected through the GUI provided.

4.1 DETAILED DESIGN

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



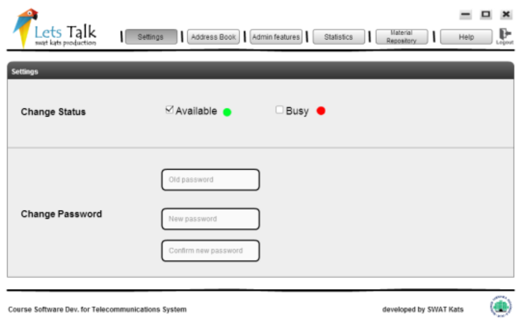
4.2 UNIT TEST PLAN

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

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



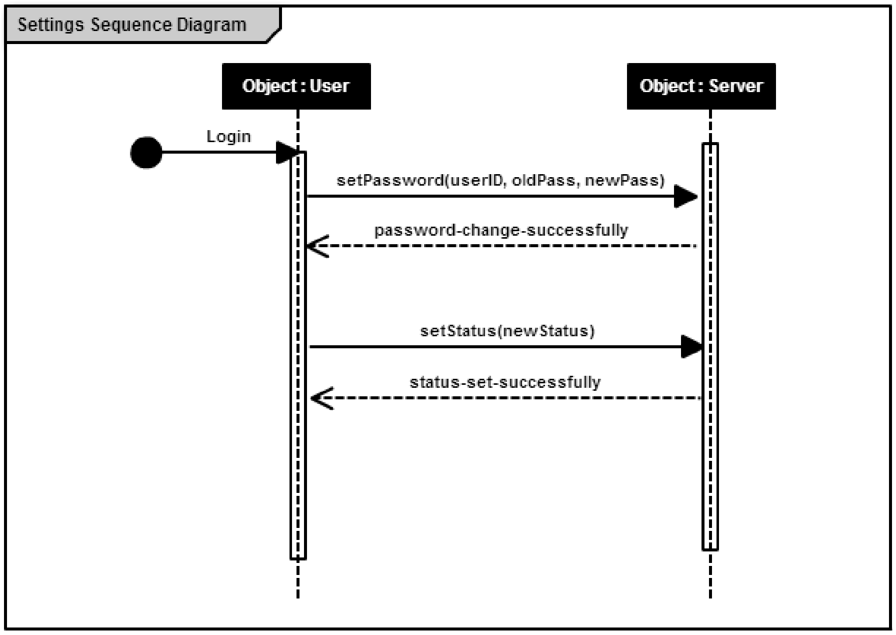
Change Status [✓ Available] ● Busy ●

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 interacts with login modules, from which it fetched the User object that defines who is using the application (which user). Of course GUI provided the appropriate framework to execute settings requests. 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.


Settings Sequence Diagram
Object : User
Object : Server
Login
setPassword(userID, oldPass, newPass)
password-change-successfully
setStatus(newStatus)
status-set-successfully

5.2 UNIT TEST PLAN

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

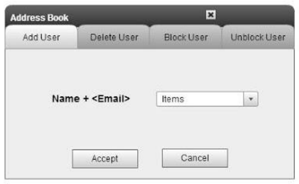
6. **Test**: TEST_SETTINGS_STATUS

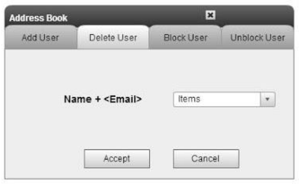
Purpose: It will test that the changing of user's status functions properly
Requirements: REQ_USR_6
Environment: user must be logged in our application
Operation: run the executable providing as parameters a valid username-password pair and the desired status (Available/Busy).
Expected result: status is changed/set according to user's desire
Result: success/failure

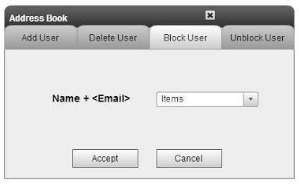
7. **Test**: TEST_SETTINGS_CHANGE_PWD
   Purpose: It will test that the changing of user's password functions properly
   Requirements: REQ_USR_7
   Environment: user must be logged in our application
   Operation: run the executable providing as parameters a valid username-password pair and the desired new password.
   Expected result: password is changed according to user's desire
   Result: success/failure

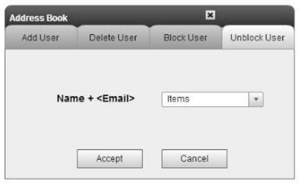
6. ADDRESS BOOK

XtremeSecurity has all its employees' list in a central database, i.e. global address book. This module is all about adding user, removing user, blocking and unblocking user from global address book.







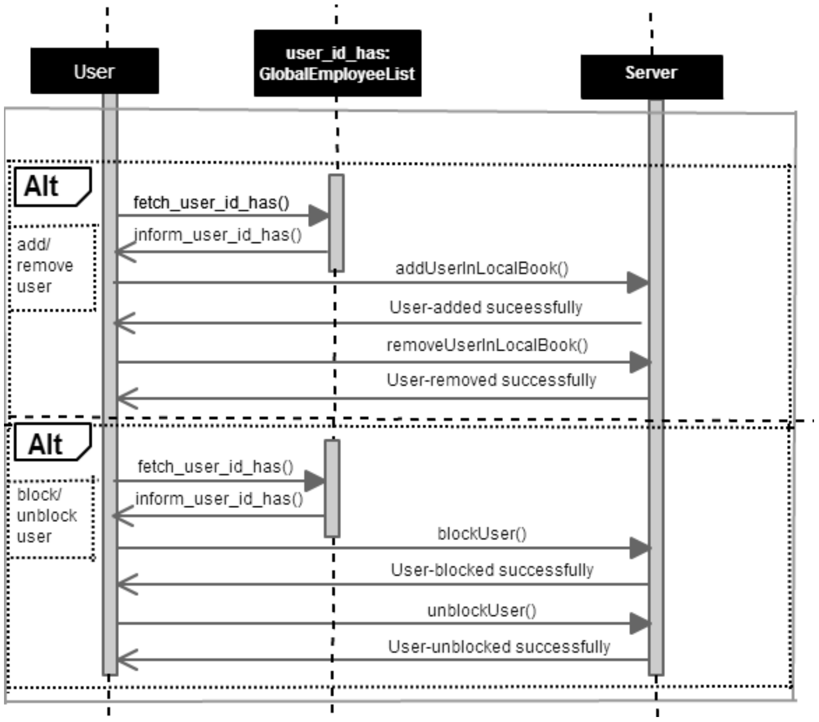


This module satisfies user requirement REQ_USR_2, REQ_USR_3, REQ_USR_4, REQ_USR_5 while it doesn't interact with other modules. After a user clicks on address book, where above window opens up, a user_id is selected to add/remove/block/unblock and database updates accordingly.
If a user:

i. **Adds** another user to local address book then he will be able to chat with him.
ii. **Removes** a user from local address book 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.
iv. **Unblocks** a user then both users will be able to chat with each other but he will have to add him again.

6.1 DETAILED DESIGN

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



6.2 UNIT TEST PLAN

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

2. **Test**: TEST_ADDUSR
   Purpose: Registered user must be able to add other registered users in his address book by selecting users from global address book.
   Requirements: REQ_USR_2
   Environment: user must be logged in our application
   Operation: run the executable providing parameter as a valid username from global address book and choosing an option of add.
   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.
   Result: success/failure

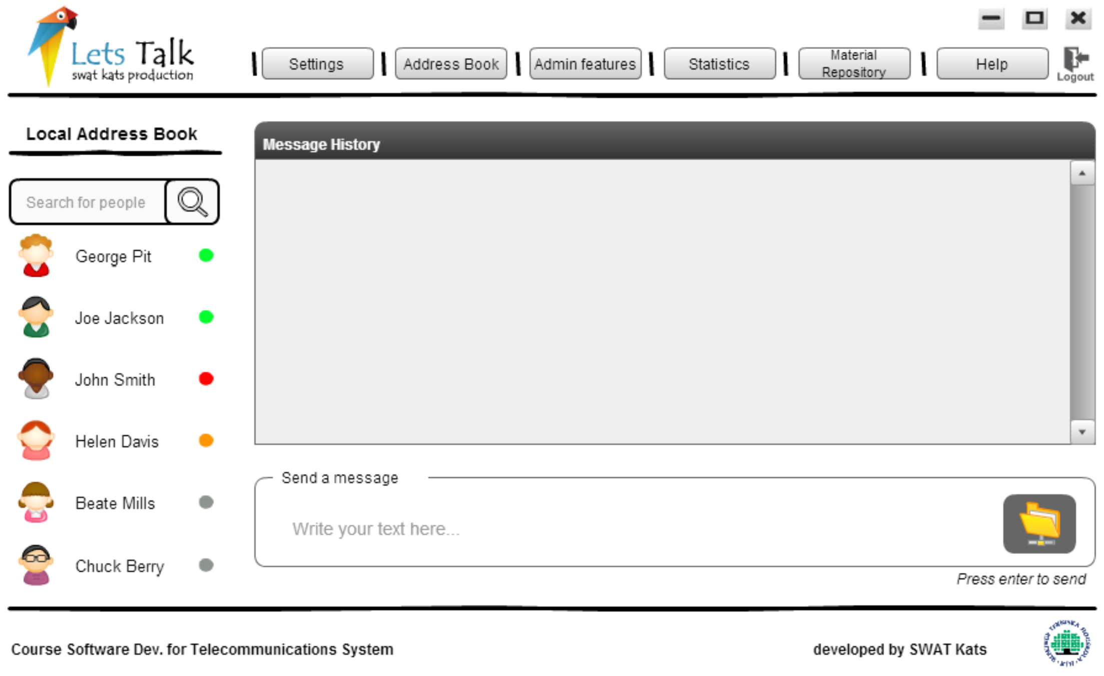
3. **Test**: TEST_BLCKUSR
   Purpose: Registered user must be able to block users in his address book
   Requirements: REQ_USR_3
   Environment: user must be logged in our application
   Operation: run the executable providing parameter as a valid username and choosing an option of block.
   Expected result: If a contact is blocked then it should not appear in local address book (if he was added first) and they should not be able to chat with each other.
   Result: success/failure

4. **Test**: TEST_REMVUSR
   Purpose: Registered user must be able to remove other registered users from his local address book.
   Requirements: REQ_USR_4
   Environment: user must be logged in our application and user exists in local address book.
   Operation: run the executable providing parameter as a valid username from local address book and choosing an option of remove.
   Expected result: If a contact is removed, then it should not appear in user's address book.
   Result: success/failure

5. **Test**: TEST_UNBLCKUSR
   Purpose: Registered user must be able to unblock users from his block list and user is in blocked list.
   Requirements: REQ_USR_5
   Environment: user must be logged in our application and user exists in blocked list..
   Operation: run the executable providing parameter as a valid username from blocking list and choosing an option of unblock.
   Expected result: If a contact is unblocked then it should not appear in address book and they should be able to chat with each other (If any of them adds another one).
   Result: success/failure

7. TEXTING & ENCRYPTION

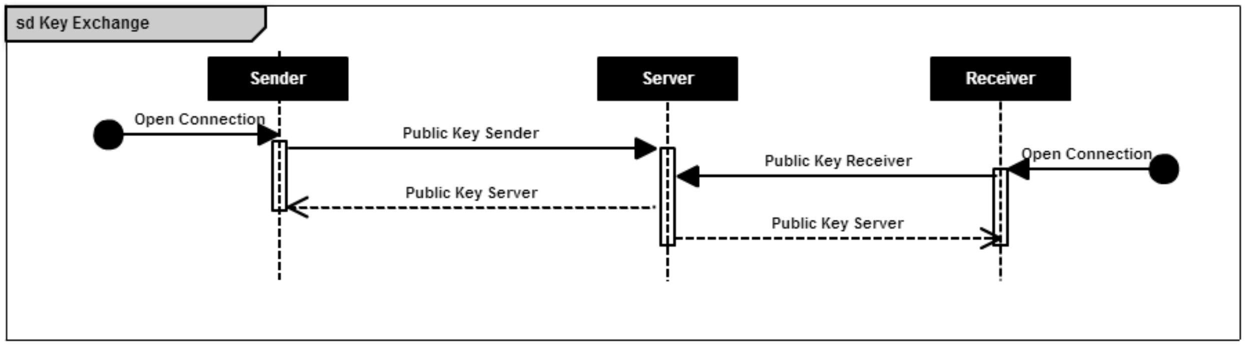
The support of a chat feature is the main purpose of **LetsTalk** as well as the proper encryption to guarantee privacy and security of the conversations.
The Chat window is positioned directly on the main page and enables chatting and text messaging with people in the local address book. See the following picture for the first draft of the GUI.



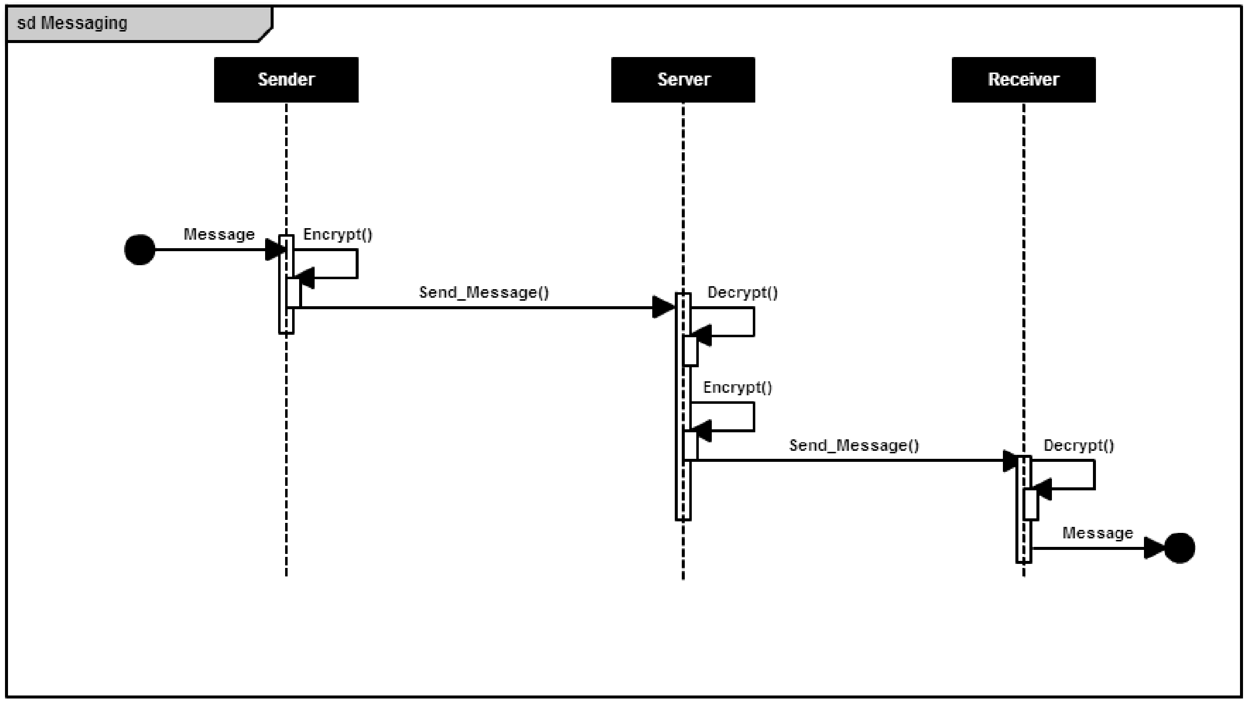
A User selects a contact in the address book and can type messages in the shown text area. If the selected receiver is online, he/she will receive the typed text message in the message history panel and will be able to respond by typing an answer in his/her own message field. These messages have to be encrypted. Therefore a key exchange has to be made beforehand, e.g. when the user goes online. The key exchange however, is not noticeable by the user and is done automatically at connection.

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

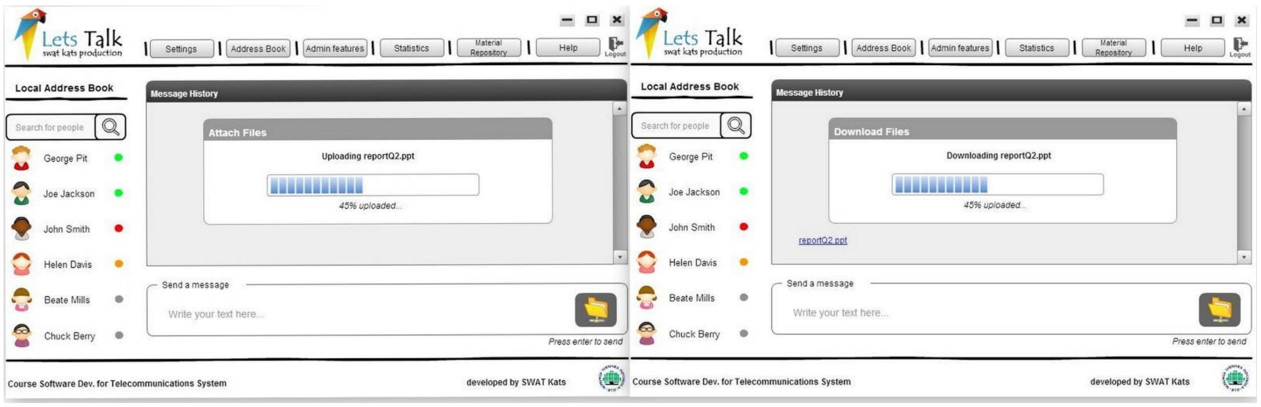
The module's functionality can be fully tested using the following unit test.

2. **Test**: TEST_MESSAGE
   Purpose: Successfully transmit a message from the sender to the receiver.
   Requirements: REQ_USR_8, REQ_SYS_1
   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.

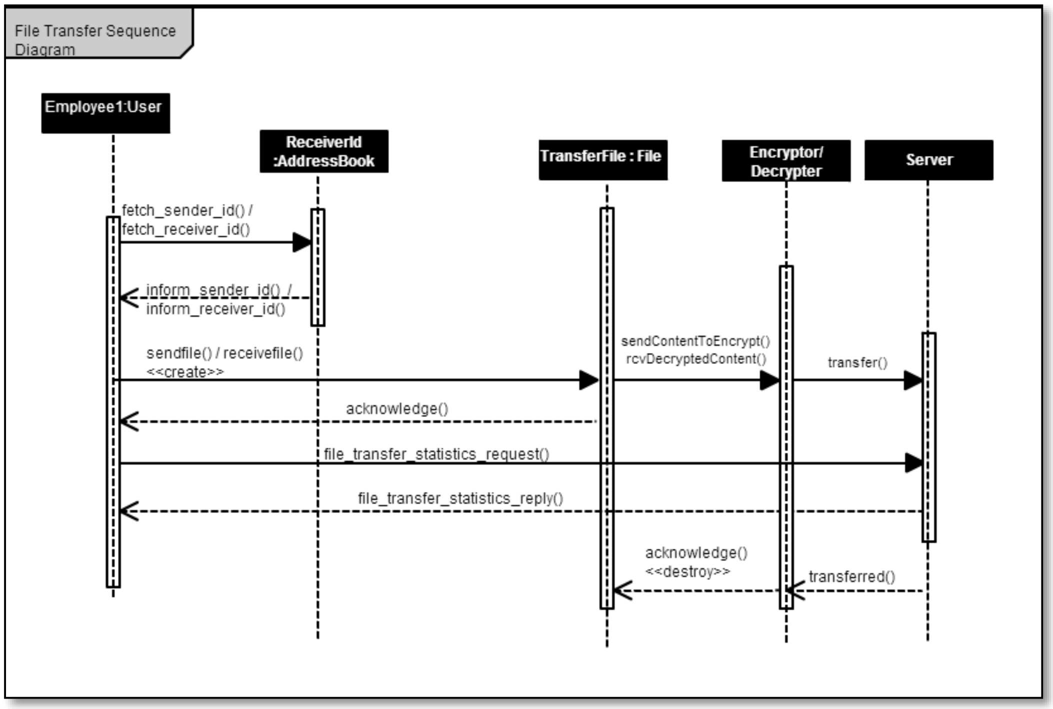
The material repository defines the accessibility of global database of the company's personal files (MOUs, brochures etc.) to all the users (on site or in office). GUI for this module will be same as for the file transfer. So, any user can download the files from the central database on any hourly need.

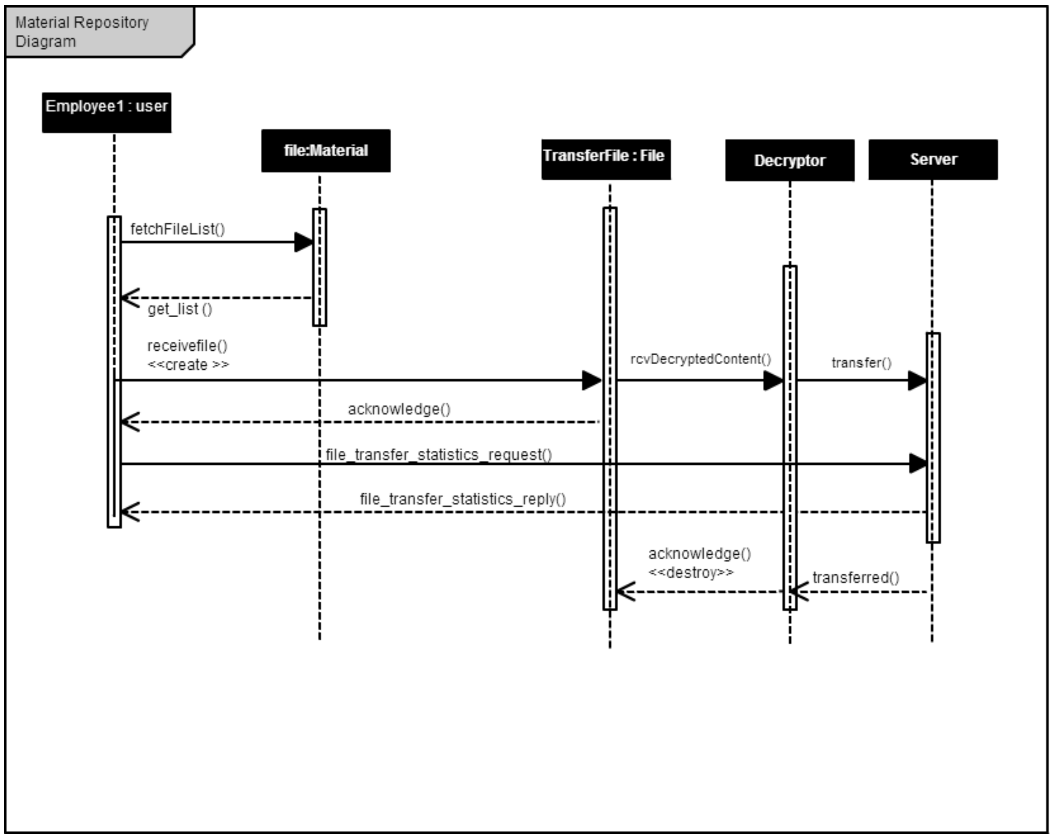


-This module satisfies user requirement 5 (REQ_USR_5). It interacts with address book module to select the receiver, encrypts the file and uploads it to server to be delivered to the selected user.

8.1 DETAILED DESIGN

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


File Transfer Sequence Diagram
Employee1:User
ReceiverId :AddressBook
TransferFile : File
Encryptor/ Decrypter
Server
fetch_sender_id() / fetch_receiver_id()
inform_sender_id() / inform_receiver_id()
sendfile() / receivefile() <<create>>
sendContentToEncrypt() rcvDecryptedContent()
transfer()
acknowledge()
file_transfer_statistics_request()
file_transfer_statistics_reply()
acknowledge() <<destroy>>
transferred()



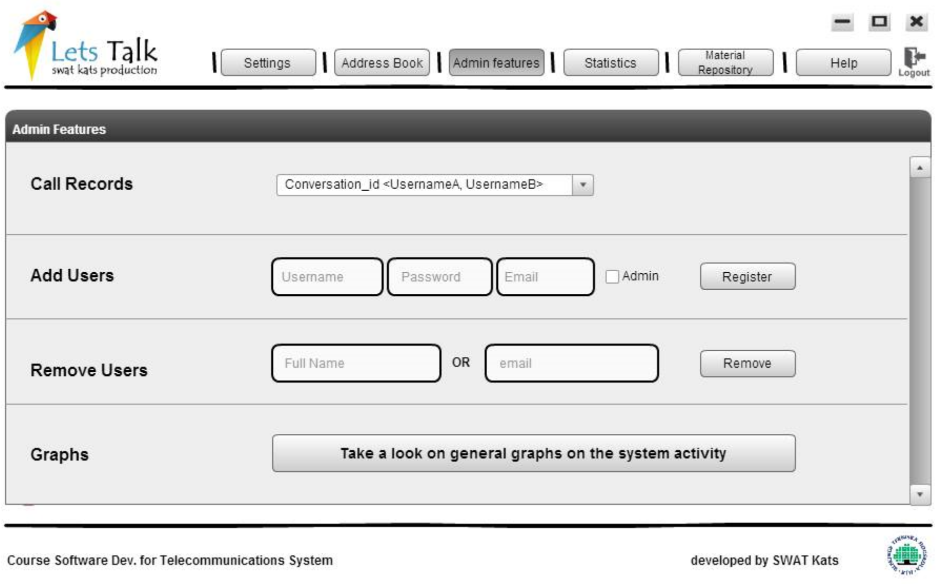
8.2 UNIT TEST PLAN

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

8. **Test**: TEST_FILE_TRANSFER
   Purpose: System shall allow a user to send files
   Requirements: REQ_USR_8
   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

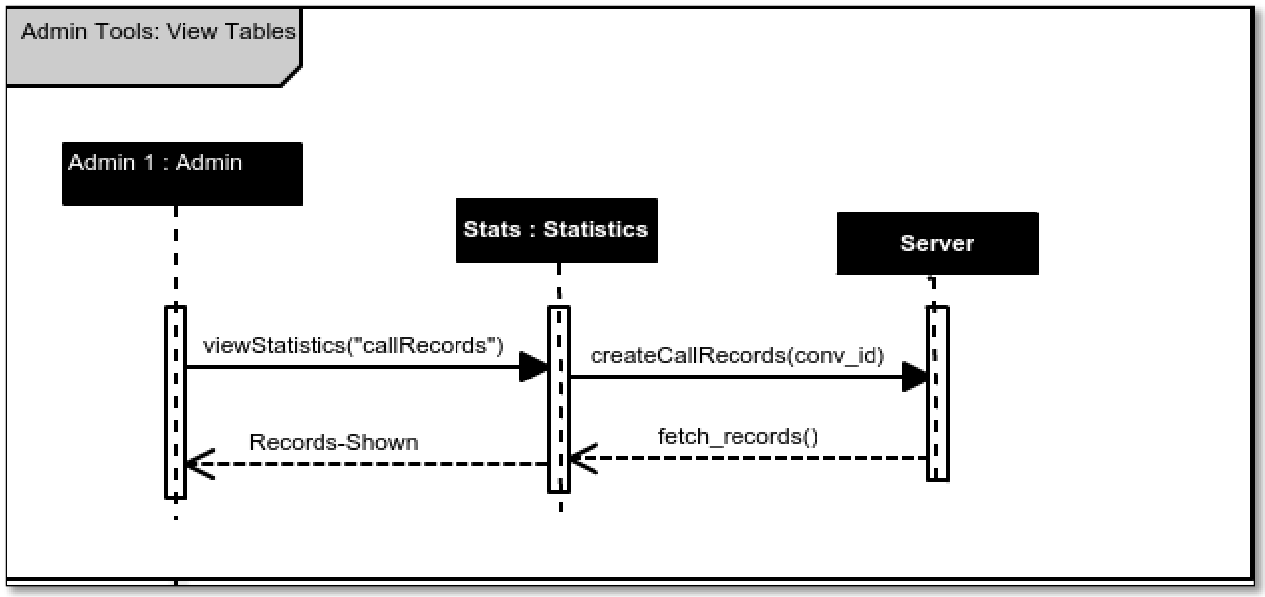
Admin tools module is about the functions that an admin has. The Admin is a normal user with some extra features at hand. The admin can Add/Remove a user GLOBALLY - Change the users' list in the main database or can take a general idea of the chat statistics by viewing the graphs and call records. The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

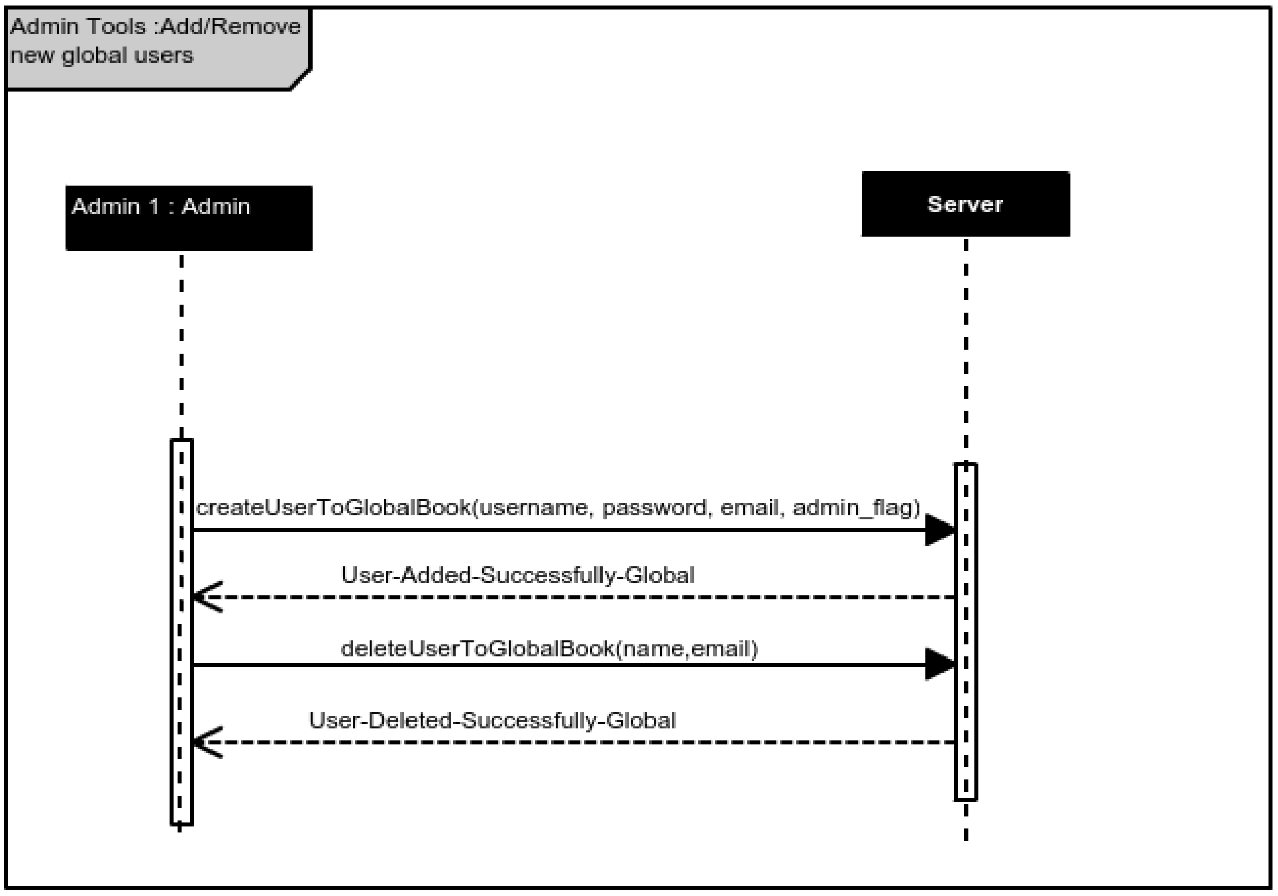


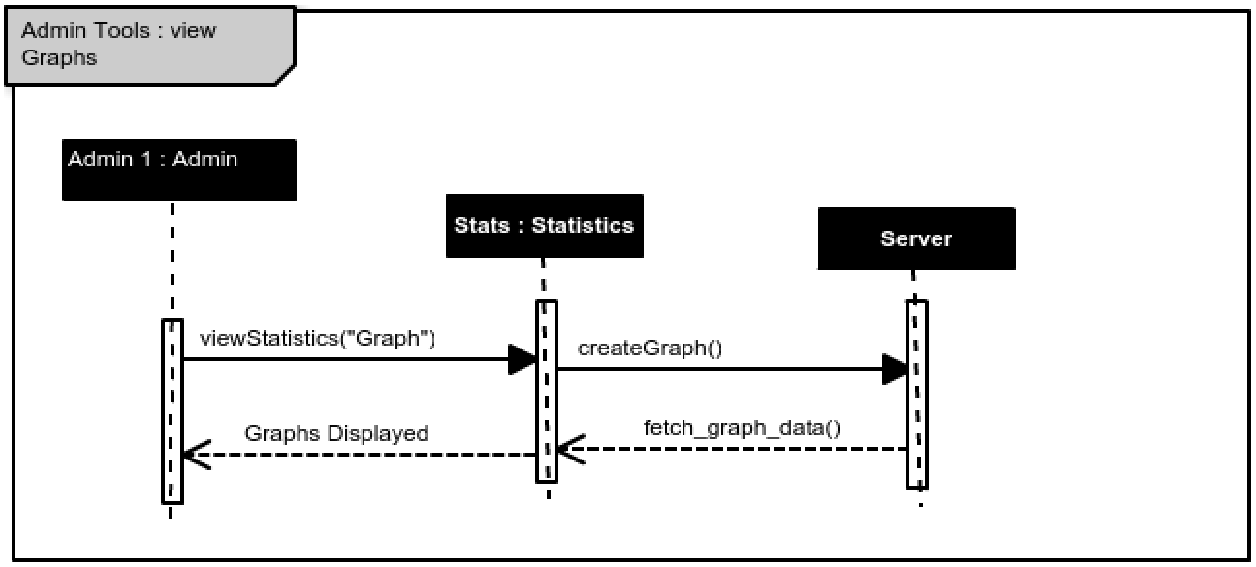
This module satisfies user requirement REQ_USR_6. A user can also be made admin by any other current admin. This module interacts with the "Address Book" module -- when a new user is added, others users can add the new user to local their address book, and when a user is deleted, it is removed from the local address books of all the users who had added the "deleted user".

9.1 DETAILED DESIGN

The following sequence diagrams shows the 3 basic functionality of admin module into 3 subparts.


Admin Tools: View Tables
Admin 1 : Admin
Stats : Statistics
Server
viewStatistics("callRecords")
createCallRecords(conv_id)
Records-Shown
fetch_records()


Admin Tools :Add/Remove new global users
Admin 1 : Admin
Server
createUserToGlobalBook(username, password, email, admin_flag)
User-Added-Successfully-Global
deleteUserToGlobalBook(name,email)
User-Deleted-Successfully-Global



9.2 UNIT TEST PLAN

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

6. **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