Requirement Model

Confide Instant Messaging Application

Table of Contents

1 Contents

1 Definitions and Acronyms

Definition/Acronym	Description
chat system/system	Confide Instant Messaging Application (CIMA)
user	Employee or operator granted permission to use chat application for Pinnacle Corporation business
address book	List of chat users
account	User chat account
chatroom	A container for multi-user chat sessions
CCRD use case	Critical Core Risky Difficult Use case
normal flow	Intended workflow and/or execution sequence of CIMA
alternate flow	Possible scenario encountered when normal flow fails
exception Flow	Flow may include workarounds to achieve the same outcomes as normal flow
pre-conditions	The operating conditions required for normal flow to begin
post-conditions	Expected outcomes after normal flow concludes.
thread	Sequence or transcript of chat messages between two users or a group of users

2 Introduction

This document lists the functional requirements for the Confide Instant Messaging Application. For more technical details of the implementation of CIMA, please refer to *Final Architecture.docx*. The CCRD use cases for this project are the logistical requirements for two (or more) users to be ready to initiate or receive a chat message, then to actually send messages. This involves the following features:

- Create an account
- Add a user to contacts
- View contacts
- Chat with contacts
- View chats with contacts

3 Requirements

3.1 Functional Requirements[1]

3.1.1 Chats [UC0004 – UC0008,]

a. The chat system will provide users with the ability to send messages to and receive messages from other users. This will be a real-time “instant” messaging service.
b. Users will be able to chat with multiple other users in a single thread (group chat)
c. Users will be able to view the dialogue of current chats and retrieve a transcript of previous chats, on a per user or per group chat basis
d. Users will be able to send/receive chats when one or more intended recipients are currently offline

3.1.2 User Management [UC0001], [UC0010 – UC0011]

a. Users can create an account (username and password) to authenticate with the chat service

[1] Code numbers [UCXXXX] refer to Full Use Case Descriptions (see Section 4)

b. The chat system will require that these details are entered before the user can operate the service.
c. User can logoff (disconnect) from the system

3.1.3 Contact management [UC0002 – UC0004]

a. User “A” will be able to add another user to their contacts by their Jid (id).
b. Users will be able to discover the current status of the other users in their contacts
c. Users will be able to delete or block other users

3.1.4 Notifications [UC0009]

a. Users will receive notifications to alert them to new activity on the chat network. This will include:
    a. New incoming chat message or invitation to add contact
    b. An error has occurred (e.g. loss of network connectivity)
    c. Status update on user’s contacts

3.2 Non-functional Requirements

3.2.1 Performance

a. Users’ perception is that the application can send and receive messages in “real time”
b. Application user interface meets normal user expectation of responsiveness. Application warns user if an operation will take several seconds, or is taking longer than normal

3.2.2 Data Security and Recovery

a. Corporate chat data is hosted in private cloud/data centre. Pinnacle retains control of these servers.
b. Data backup/recovery is outsourced and is provided as part of the hosting package. This is beyond the scope of the current project.

3.2.3 Availability

a. Hosting provider provides guaranteed uptime. This is part of the contractual SLAs to Pinnacle Corporation.

3.2.4 Usability

a. System must behave in a predictable way.
b. Tasks must be intuitive and easy to understand
c. System should be visually pleasing
d. Information should be formatted to aid fast comprehension (justification, colours etc)
e. User experience should be the same on all supported platforms the application can run on.

3.2.5 Maintenance

a. Pinnacle retains full intellectual property rights to the software and source code.
b. The server software and client framework are both open source to facilitate ongoing maintenance and development of new features

3.2.6 Operating Systems

The system should be able to be run on all leading desktop platforms (MacOS, Linux, Windows)

4 Full Use Case Descriptions

4.1 Create an Account [UC0001]

Brief Description:	
When a user Wants to start using the chat system They create an account So that they can continue into the application and use it.	
Trigger:	
A user wants to create an account.	
Actors:	
1. **User**: A user on the client side application, who wants to create an account. 2. **System**: The client side application.	
Pre-Conditions:	
The user has downloaded the client side application. The user has internet connection.	
Post-Conditions:	
The user will have an account, whose information will be stored on the server. The user will be able to log into their account using the information they provided	
Normal Flow	
User	System
1. User opens application.	2. System presents the account screen.
3. User enters their username and password to be into the username and password fields in the create account section..	4. System validates format of inputs, and that the username is new.
5. User selects “Create Account” or presses enter.	6. System creates the account, and notifies user of success.
Alternate Flows:	

Invalid Input If at step 4, any of the input is invalid, then:	
User	System
	4.1 System notifies user of incorrect input.
The flow resumes from step 3 of the normal flow.	
Exceptional Flows:	
User quits process If at any point during the process, the user quits the process, then their account is not created.	
Key Scenarios:	
1. Regular successful account creation The user enters all the correct data and their account is created. **2. Invalid input** The user enters some incorrect data, but eventually they fix it and their account is successfully created. **3. User quits process** The user quits the process at some point before the account has been created, therefore the account is not created.	

4.2 Edit Account Information [UC0012] [not yet implemented]

Brief Description:	
When a user Wants to edit their account information They open the account settings page So that they can change the relevant information.	
Trigger:	
A user wants to edit their account.	
Actors:	
3. **User**: A user on the client side application, who wants to edit their account. 4. **System**: The client side application.	
Pre-Conditions:	
The user has an account on the client side application. The user has internet connection.	
Post-Conditions:	
The user's account details including the parts that have changed, will be stored on the server. The new information will now be presented in relevant dialogues.	
Normal Flow	
User	System
1. User opens account settings.	2. System presents the account information in edit mode.
3. User enters the relevant changes.	4. System validates format of some inputs.
5. User selects "Save Account information"	6. System saves the revised account information, and notifies user of success.
Alternate Flows:	

Exceptional Flows:
User quits process If at any point during the process, the user quits the process, then their account information is not modified.
Key Scenarios:
4. Regular successful account creation The user enters all the correct data and their account is created. **5. Invalid input** The user enters some incorrect data, but eventually they fix it and their account is successfully created. **6. User quits process** The user quits the process at some point before the account has been created, therefore the account is not created.

4.3 Add User To Contacts (Request Contact Add) – [UC0002]

Brief Description:	
When a user Wants to add another user as a contact They search for that user and request to add them as a contact So that they can easily find and interact with them in the system.	
Trigger	
A user wants to add another user as a contact.	
Actors	
Requester: The user requesting the contact-add. **Receiver**: The user receiving the contact-add request. **System**: The Requester's client side application.	
Pre-Conditions:	
The Requester is connected to the internet. Both Requester and Receiver have accounts.	
Post-Conditions:	
A contact-add request will have been sent from the Requester to the Receiver.	
Normal Flow	
<u>Requester</u>	<u>System</u>
1. User opens app and logs in.	2. System displays home screen.
3. User enters the username of the user they wish to add, and presses the "add" button.	4. System displays an alert notifying the user that a contact-add request has been sent to the other user.
Alternate Flows:	
Invalid Format If at step 4 the username of the user to add is invalid, then:	

Requester	System
	4.1. System notifies user of problem.
The flow resumes from step 3 of the normal flow.	
Exceptional Flows:	
NONE	
Key Scenarios:	
1. **Request Contact Add** Requester successfully requests a contact-add with Receiver. 2. **Invalid Format** User entered a username in an invalid format and must try again.	

4.4 Add User To Contacts (Accept Contact Add) – [UC0003]

Brief Description:	
When a user receives a contact-add request, and they Want to become contacts with the requester They select the contact-add request, and accept it So that they can easily find and interact with the requester in the system.	
Trigger:	
Another user requests to add this user as a contact, generating a contact-add request on this users client application.	
Actors:	
Requester: The user requesting the contact-add. **Receiver**: The user receiving the contact-add request. **System**: The Receiver's client side application.	
Pre-Conditions:	
The Receiver is connected to the internet. Both Requester and Receiver have accounts.	
Post-Conditions:	
The Receiver and Requester will now be contacts.	
Normal Flow:	
Receiver	System
	1. System displays the contact request in the users contact list, with a button to accept and a button to decline.
2. User presses the "Accept" button.	3. System notifies user that they have are now contacts with this user.
Alternate Flows:	

Decline Contact Add When at step 4. the Receiver selects “Decline” rather than “Accept”, the contact request is cancelled, and disappears from their contact list.
Exceptional Flows:
NONE
Key Scenarios:
1. **Accept Contact Add** Receiver accepts the contact-add request, and the two users successfully become contacts. 2. **Decline Contact Add** Receiver declines the contact-add request, and the request is cancelled.

4.5 View Contacts – [UC0004]

Brief Description:

When a user
Wants to view their contacts
They select “View Contacts”
So that they can view their contacts and maybe start a chat with one of them.

Trigger:

A user wants to view their contacts.

Actors:

1. **User**: A user on the client side application, who wants to view their contacts.
2. **System**: The client side application.

Pre-Conditions:

The user has downloaded the client side application.
The user has internet connection.
The user has an account.

Post-Conditions:

The user is viewing/viewed their contacts.

Normal Flow

User	System
1. User logs in.	2. System displays the home screen, which contains the contact list.

Alternate Flows:

User Has No Contacts
If at step 1, the user has no contacts, then the contact list will simply be empty.

Exceptional Flows:

NONE

Key Scenarios:

1. **View Contacts**
   The user successfully views their contacts.
2. **User Has No Contacts**

The user has no contacts to view, so their contact list appears empty.

4.6 View Chats – [UC0005]

Brief Description:

When a user
Wants to view their chats
They select “Contacts” to view all their contacts, each of which are associated with a single chat, and then select a specific chat to view it
So that they can see and participate in their chats.

Trigger:

A user wants to see and/or participate in their chats.

Actors:

1. **User**: A user on the client side application, who wants to view their chats.
2. **System**: The client side application.

Pre-Conditions:

The user has downloaded the client side application.
The user has internet connection.
The user has an account.

Post-Conditions:

The user will see/have seen their chats.

Normal Flow

User	System
1. User logs in.	2. System displays home screen.
3. User clicks on a specific contact, to open the chat with that contact.	4. System displays the chat with that contact.

Alternate Flows:

NONE

Exceptional Flows:

NONE

Key Scenarios:

1. **View Chats**
   The user successfully views their chats.

4.7 Chat – [UC0006]

Brief Description:

When a user
Wants to chat with another user
They select login, add the user as a contact (if they aren't already contacts), open their chat with that user, and interact with the chat.
So that they can see and participate in the chat.

Trigger:

A user wants to see and/or participate in a chat.

Actors:

3. **User**: A user on the client side application, who wants to view their chats.
4. **System**: The client side application.

Pre-Conditions:

The user has downloaded the client side application.
The user has internet connection.
The user has an account.

Post-Conditions:

The user will see/have seen their chats.

Normal Flow

User	System
1. User logs in.	2. System displays home screen.
3. User clicks on a specific contact in order to open the chat with that contact.	4. System displays the chat with that contact.
5. User performs sub use cases "send message" or "receive chat" as they wish.	

Alternate Flows:

1. Target User is not a contact
If at step 3, User does not have the user they wish to chat with as a contact, then they add the contact, and the

main flow continues from step 3. **2.Target User is offline (or goes offline)** If at Step 3, the target user is offline (or goes offline) then undelivered chats will be stored on the server. When the target user is next online, they will receive the missing messages.
Exceptional Flows:
NONE
Key Scenarios:
1. **View Chats** The user successfully views their chats.

4.8 Multi-User Chat – [UC0013]

Brief Description:

When a user
Wants to chat with more than one other user
They select login, select the chatroom and start chatting with other users that are in the chatroom.

Trigger:

A user wants to see and/or participate in a multi-user chat.

Actors:

1. **User**: A user on the client side application, who wants to participate in a multi-user chat.
2. **System**: The client side application.

Pre-Conditions:

The user has downloaded the client side application.
The user has internet connection.
The user has the chatroom information.

Post-Conditions:

The user will see/have seen their chats in the chatroom.

Normal Flow

User	System
1. User logs in.	2. System displays home screen.
3. User clicks on the Groups button in the navigation bar.	4. System displays the Groups screen.
5. User clicks on the group chat that they wish to open.	6. System displays the chatroom.
7. User performs sub use cases “send message” or “receive chat” as they wish.	

Alternate Flows:

NONE
Exceptional Flows:
NONE
Key Scenarios:
8. View Chats The user successfully interacts with multiple other users in the chatroom.

SUB USE CASES

4.9 Send Message – [UC0007]		
Brief Description:		
When a user Wants to send a message to one of their contacts They open an existing chat or create a new chat with the contact, type a message, and press send So that they can send a message to a specific contact		
Trigger:		
A user wants to send a message to one of their contacts		
Actors:		
1. **User**: A user on the client side application, who wants to send a message to one of their contacts. **2.** **Contact**: The contact that the message is being sent to. 3. **Client**: The client side application. **4.** **Server**: The server that the client side application is talking to.		
Pre-Conditions:		
The user has downloaded the client side application. The user has internet connection. The user has an account. The user has logged in. The user has completed either of the "View Chats" use case, and is currently viewing a specific chat.		
Post-Conditions:		
User will have sent a message to Contact.		
Normal Flow		
User	Client	Server
1. Type a message and press send.	2. Display the message, and send it to the server.	3. Add the message to the chat, and send the message to the contact.
Alternate Flows:		
No Connection To The Server When Sending Message If at step 2, the client cannot connect to the server, then the client will alert the user and continue to trying to connect to the server. When a connection is established, the flow resumes from step 2 of the normal flow.		

Exceptional Flows:
NONE
Key Scenarios:
1. **Send Message** The user successfully sends a message to the Contact. 2. **No Connection To The Server When Sending Message** The client failed to connect to the server when sending the message, and must continue to try to reconnect before continuing.

4.10 Receive Message – [UC0008]

Brief Description:	
When a user Receives a message They receive a notification So that they will know and can read the message	
Trigger:	
The server sends a message to the client.	
Actors:	
1. **User**: A user on the client side application, who is receiving a message 2. **Client**: The client side application. **3. Server**: The server that the client talks to, and who sent the message.	
Pre-Conditions:	
The user has downloaded the client side application. The user has internet connection. The user has an account. The user has logged in.	
Post-Conditions:	
The user will have received a notification that they have a new message, and they will be able to view this message.	
Normal Flow	
User	Client
	1. The client adds the new message to the corresponding chat, and sets a notification.
2. The user sees the notification, and can view the message.	
Alternate Flows:	
User Has The Relevant Chat Open If at step 1, the user already has the relevant chat open, then the new message will be appended, but the notification will be skipped. And the flow will resume from step 4 of the normal flow.	
Exceptional Flows:	
NONE	

Key Scenarios:
1. **Receive Message** The user successfully received the new message, and a notification of this. 2. **User Has The Relevant Chat Open** The user successfully received the new message, but did not receive a notification as they were already in the relevant chat.

4.11 Notification – [UC0009]

Brief Description:	
When a user Wants to know if they have any new messages or other notifications (e.g. new chat invitations), or possible error with the chat app (e.g. lost connection) They are alerted via visual and/or audible means that a new message has arrived So that they are able to respond to the new activity quickly	
Trigger:	
A new message (chat, invite etc) has arrived	
Actors:	
1. **User**: A user on the client side application, who wants to be alerted to new activity on the app 2. **System**: The client side application.	
Pre-Conditions:	
The user has downloaded the client side application. The user has internet connection. The user has logged into his/her account on the application.	
Post-Conditions:	
The user is alerted to new activity on the chat application and can respond at the earliest convenience.	
Normal Flow	
User	System
1. User has logged into chat app	2. Main chat window is open or minimised
3. User is alerted to new activity	4. Display relevant information/ play sound
5. User processes notification (follow, dismiss, leave)	6. System displays full message (depending on user choice)
Alternate Flows:	
No alternate flows – mechanism is the same for all notification types. User can choose to ignore the notification	

Exceptional Flows:
-
Key Scenarios:
1. **New incoming message** System displays visual and/or audible alert of new message 2. **Error message** System is unable to connect to server, another user has gone offline etc. 3. **Invitation** User receives request to add another user as a contact 4. **Status update** System provides update e.g. user has changed status

4.12 Log In – [UC0010]

Brief Description:

When a user
Wants to log in
They enter their details and press log in
So that they can use the system

Trigger:

A user wants to log in.

Actors:

1. **User**: A user on the client side application, who wants to log in.
2. **Client**: The client side application.
3. **Server**: The server that the client talks to.

Pre-Conditions:

The user has downloaded the client side application.
The user has internet connection.
The user has an account.

Post-Conditions:

The user will be logged in.

Normal Flow

User	Client	Server
1. User enters username and password into the username and password fields in the login section, then presses "Log In".	2. Client validates that input is formatted correctly, then sends it off to the server.	3. Server validates that the input is correct, logs the user in, and alerts the client.
	4. Client notifies the user that they have successfully logged in, by bringing up the home screen.	

Alternate Flows:

Invalid Input Format If at step 2, the client recognises that the format of the input is incorrect, then it notifies the user. The flow resumes from step 1 of the normal flow.
No Matching Details If at step 3, the server fails to find an account that matches the input, then it notifies the client, which notifies the user. The flow resumes from step 1 of the normal flow.
No Connection To Server If at step 2, the client cannot connect to the server, then the client notifies the user, and continues to try to connect. When a connection is established, the flow resumes from step 2 of the normal flow.
No Response From Server If at step 4, the client has not heard back from the server, then it notifies the user, and continues waiting. When a response is received from the server, the flow resumes from step 4 of the normal flow.
Exceptional Flows:
NONE
Key Scenarios:
1. Log In The user successfully logs in. **2. Invalid Input Format** The user entered an invalid format for some of the input, and must try again. **3. No Matching Details** There is no account with the given details, the user must try again. **4. No Connection To Server** The client could not connect to the server to send it the log in data, so it notifies the user and continues trying to connect. **5. No Response From Server** The client has not heard back from the server for an extended period of time, so it notifies the user and continues waiting.

4.13 Log Out – [UC0011]		
Brief Description:		
When a user Wants to log out They select log out So that they will no longer be logged into the system		
Trigger:		
A user wants to log out.		
Actors:		
1. **User**: A user on the client side application, who wants to log out. 2. **Client**: The client side application. **3. Server**: The server that the client is talking to.		
Pre-Conditions:		
The user has downloaded the client side application. The user has internet connection. The user has an account. The user is logged into their account.		
Post-Conditions:		
The user will not be logged into the system on the device used.		
Normal Flow		
User	Client	Server
1. Selects “Log Out” from the navigation bar.	2. Ends the connection to the server. And changes the screen to the account screen.	
Alternate Flows:		
NONE		
Exceptional Flows:		
NONE		
Key Scenarios:		
1. Log Out The user successfully logged out.		