Echo - Deliverable 1

Table of Contents

Sr.no	Roll.no	Names
1.	23L-0907	Hanzala Ahsan
2.	23L-0718	Muhammad Daniyal
3.	23L-2509	Syed Ali Naqvi
4.	23L-0643	Muhammad Auon Naseer
5.	23L-2528	Moiz

Use Case - 1 (Register/Sign Up)

Identifier	UC-01
Name	Register/Sign Up
Purpose	Allows a new user to register in the LAN chat system with secure credentials
Priority	High
Actors	User, Local Server
Pre-conditions	1. Application deployed and accessible on LAN. 2. User has not yet registered.
Post–conditions	1. New user account created and stored on local server
Dependencies	None

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User opens application and chooses “Register”	**1.1**	System displays a registration form.
2	User enters credentials (username, password, public key)	**2.1**	System encrypts and stores credentials securely on Local Server.
3	User submits the form	**3.1**	System confirms successful registration and shows a login screen.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User enters an existing username.	**1.1**	System shows “User already exists” error and asks for a different username.

Exception:

1. Local Server unreachable —> registration fails, error displayed
2. Missing mandatory fields —> system does not proceed to store credentials

Use Case - 2 (Authenticate/Login)

Identifier	UC-02
Name	Authenticate/Login
Purpose	Enables a registered user to securely log in.
Priority	High
Actors	User, Local Server
Pre-conditions	1. User is already registered. 2. Application running.
Post–conditions	1. User is authenticated and granted access to chat features.
Dependencies	UC-01 (Register / Sign Up)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User launches the application and selects “Login”.	**1.1**	System displays a login screen.
2	User enters username and password.	**2.1**	System verifies credentials with the local server.
3	User clicks “Login”.	**3.1**	System grants access to the main dashboard.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User enters incorrect password	**1.1**	System shows an “Invalid credentials” message.

Exception:

1. Local Server unavailable —> Login fails

Use Case - 3 (Send Text Message)

Identifier	UC-03
Name	Send Text Message
Purpose	Allows a user to send an end-to-end encrypted text message to another user.
Priority	High
Actors	User
Pre-conditions	1. User logged in. 2. Recipient exists.
Post–conditions	1. Encrypted text message sent to Local Server and queued for delivery.
Dependencies	UC-02 (Authenticate/Login)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User types a text message.	**1.1**	System encrypts the message locally.
2	User presses “Send”.	**2.1**	System sends ciphertext to Local Server for delivery.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	AI features enabled —> message checked for toxicity before sending.	**1.1**	AI Module flags harmful content; system asks user to edit or cancel.

Exception:

1. Local Server unreachable —> message stored locally for retry

Use Case - 4 (Send Image)

Identifier	UC-04
Name	Send Image
Purpose	Allows a user to send an end-to-end encrypted image to another user.
Priority	High
Actors	User
Pre-conditions	1. User logged in. 2. Image file selected.
Post–conditions	1. Encrypted image message sent to Local Server.
Dependencies	UC-02 (Authenticate/Login)

Typical Course of Action (Primary Flow)			
S#	**Actor Action**	**S#**	**System Response**
1	User selects an image	**1.1**	System encrypts the image locally
2	User presses “Send”	**2.1**	System sends ciphertext to Local Server.

Alternative Course of Action			
S#	**Actor Action**	**S#**	**System Response**
1	AI features enabled —> system checks metadata or captions for toxicity.	**1.1**	AI Module flags issue and notifies users.

Exception:

1. File too large —> system shows error
2. Local Server unreachable —> system queues image locally

Use Case - 5 (Send Audio)

Identifier	UC-05
Name	Send Audio
Purpose	Allows a user to send an end-to-end encrypted audio clip.
Priority	High
Actors	User
Pre-conditions	1. User logged in. 2. Audio recorded or selected.
Post–conditions	1. Encrypted audio message sent to Local Server.
Dependencies	UC-02 (Authenticate/Login)

Typical Course of Action (Primary Flow)			
S#	**Actor Action**	**S#**	**System Response**
1	User records or selects an audio clip.	**1.1**	System encrypts the audio locally.
2	User presses “Send".	**2.1**	System sends ciphertext to Local Server.

Alternative Course of Action			
S#	**Actor Action**	**S#**	**System Response**
1	AI features enabled —> system analyses transcript for toxicity.	**1.1**	AI module flags and warns user.

Exception:

1. Audio format unsupported —> error shown
2. Local Server unreachable —> audio queued locally

Use Case - 6 (Receive Encrypted Message)

Identifier	UC-06
Name	Receive Encrypted Message
Purpose	Allows a logged-in user to receive encrypted text, image or audio messages from others.
Priority	High
Actors	User
Pre-conditions	1. User logged in and connected to LAN. 2. Local server has messages queued for the user.
Post–conditions	1. Encrypted messages delivered to user's client
Dependencies	UC-02 (Authenticate/Login)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User opens the inbox or waits in the chat window.	**1.1**	System pulls any pending encrypted messages from the Local Server.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User has AI translation enabled.	**1.1**	System offers a "Translate" button for each message.

Exception:

1. No network connection —> messages not received until connection restored

Use Case - 7 (Opt-in/Opt-out of AI Features)

Identifier	UC-07
Name	Opt-in/Opt-out of AI Features
Purpose	Allows a user to enable or disable AI features such as summarization, smart reply, translation, and toxicity filtering. For group chats, the admin selects AI features only once during group creation, and all members will be notified. Members who do not want their data processed by AI may leave the group.
Priority	Medium
Actors	User
Pre-conditions	1. User logged in. 2. For group chats, the group must already be created. 3. Admin must select AI features at the time of group creation.
Post–conditions	1. AI feature settings saved locally for the user. 2. For groups, AI settings applied to the entire group. 3. Members are notified if AI is enabled in the group. 4. Members may leave if they choose not to share data.
Dependencies	1. UC-02 (Authenticate/Login) 2. UC-15/UC-16 (for group creation & management)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User opens settings and toggles AI features.	**1.1**	System saves preference locally.
2	User confirms changes.	**2.1**	System applies new AI settings immediately.
3	Admin selects AI features during group creation.	**3.1**	System applies selected AI features to the group.
4	System notifies all group members: “AI features have been enabled in this group.”	**4.1**	Members may choose to leave the group if they do not want AI processing.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User disables AI features mid-chat.	**1.1**	System stops sending data to the AI Module instantly.

2	A group member does not want AI enabled.	**2.1**	System provides an option to “Leave Group” and removes the member upon confirmation.
Exception: 1. Settings file corrupted —> defaults restored to “AI disabled”. 2. Non-admin users try to change AI features in a group → system displays: “Only the group admin can select AI features during group creation.”			

Use Case - 8 (Handle Encryption)

Identifier	UC-08
Name	Handle Encryption
Purpose	To securely encrypt outgoing content
Priority	High
Actors	User
Pre-conditions	1. User is authenticated and composing a message 2. Encryption keys are available and valid
Post–conditions	1. The content is encrypted and packaged for secure transmission. 2. Encrypted data handed over to Send Text/Send image/Send audio use cases for delivery
Dependencies	1. Valid encryption keys for the current user session.

Typical Course of Action (Primary Flow)			
S#	**Actor Action**	**S#**	**System Response**
1	User selects content to send (text/image/audio)	**1.1**	System retrieves the appropriate encryption key for this session and encrypts the content using the chosen algorithm (end-to-end).
2	-	**2.1**	It then attaches encryption metadata (cipher type, key ID) and passes the encrypted payload back to the calling use case.

Alternative Course of Action			
S#	**Actor Action**	**S#**	**System Response**
1	User attempts to send content but chooses “Send Without Encryption” (if allowed)	**1.1**	System warns user about security risk and either blocks or sends in plain text according to policy

Exception:

1. Encryption service unavailable → content send aborted with error.
2. Invalid or expired keys → prompt user to refresh/re-authenticate.

Use Case - 9 (Verify User Credentials)

Identifier	UC-09
Name	Verify User Credentials
Purpose	To validate the user's submitted credentials.
Priority	High
Actors	User, Local Server
Pre-conditions	1. User has entered credentials
Post–conditions	2. Credentials are marked as valid or invalid.
Dependencies	1. UC-02 (Authenticate/Login)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User enters username and password in the login form	**1.1**	System sends credentials securely to Local Server

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User enters incorrect credentials	**1.1**	Local server returns an error message

Exception:

1. Local Server unreachable or timeout —> displays "server unavailable" ,message
2. User account locked or disabled – >appropriate message returned

Use Case - 10 (Translate Message)

Identifier	UC-10
Name	Translate Message
Purpose	Enables a user to translate a message into another language
Priority	Medium
Actors	User, AI Module
Pre-conditions	1. User is logged in. 2. Message exists.
Post–conditions	1. Translated message displayed to the user.
Dependencies	1. UC-06 (Receive Encrypted Message)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User selects “Translate Message.”	**1.1**	System asks for target language.
2	User chooses a language.	**2.1**	System sends a message to the AI Module for translation.
3	User waits.	**3.1**	System displays translated text.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User changes language mid-way.	**1.1**	System re-requests translation in new language.

Exception:

1. AI Module unavailable → translation fails.
2. Unsupported language → system notifies users.

Use Case - 11 (Summarize Conversation)

Identifier	UC-11
Name	Summarize Conversation
Purpose	Enables a user to request an AI-generated summary of a chat conversation.
Priority	Medium
Actors	User, AI Module
Pre-conditions	1. User is logged in. 2. Conversation exists with sufficient messages
Post–conditions	1. User receives an AI-generated summary of conversation.
Dependencies	1. UC-06 (Receive Encrypted Message)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User selects “Summarize Conversation.”	**1.1**	System sends conversation content to AI Module
2	User waits.	**2.1**	AI Module returns summary; System displays to user.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User selects only part of the conversation to summarize.	**1.1**	System sends only selected parts to the AI Module.

Exception:
1.AI Module unavailable → cannot summarize, error message shown.

Use Case - 12 (Generate Smart Reply)

Identifier	UC-12
Name	Generate Smart Reply
Purpose	Provides context-aware quick reply suggestions for a selected message.
Priority	Medium
Actors	User
Pre-conditions	1. User logged in. 2. Users opted in to AI Features.
Post–conditions	1. Smart reply suggestions shown to user
Dependencies	UC-07 (Opt-in to AI Features)

Typical Course of Action (Primary Flow)			
S#	**Actor Action**	**S#**	**System Response**
1	User selects a message and clicks “Smart Reply”.	**1.1**	System sends context to the AI Module locally.
2	User clicks one suggestion.	**2.1**	System inserts suggestions into the input box ready to send.

Alternative Course of Action			
S#	**Actor Action**	**S#**	**System Response**
1	User edit suggestions before sending.	**1.1**	System updates message accordingly.

Exception:

1. AI Module unavailable —> no suggestion shown

Use Case - 13 (Detect & Filter Toxicity)

Identifier	UC-13
Name	Detect & Filter Toxicity
Purpose	Automatically detects offensive or harmful content in messages and filters them before delivery.
Priority	High
Actors	User, AI Module
Pre-conditions	1. AI Module is enabled for the user or group.
Post–conditions	1. Offensive messages blocked, flagged or modified before delivery.
Dependencies	1. UC-07 (AI Features enabled)

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User sends a message.	**1.1**	System sends plaintext (temporarily decrypted) to the AI Module.
2	-	**2.1**	AI Module scores message for toxicity.
3	-	**3.1**	If safe → encrypt and deliver; if toxic → block or replace with warning.

Alternative Course of Action

S#	Actor Action	S#	System Response
1	Group Admin disables toxicity filter.	**1.1**	System skips this step and sends messages directly.

Exception:

1. AI Module error → default to sending message with warning.

Use Case - 14 (Extract Deadlines)

Identifier	UC-14
Name	Extract Deadlines
Purpose	Automatically identify and extract deadlines, due dates, or time-sensitive information from chat conversations using AI features.
Priority	Medium
Actors	User, AI Module
Pre-conditions	1. User is logged in. 2. Conversation contains text messages with date/time references. 3. AI Features are enabled (UC-07).
Post–conditions	1. Deadlines detected and highlighted in conversation. 2. Optionally, deadlines can be stored in a local “Reminders” list.
Dependencies	1. UC-07 (Opt-in/Opt-out of AI Features). 2. UC-06 (Receive Encrypted Message) – message must first be available in decrypted form for AI analysis.

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	User opens a conversation and enables “Extract Deadlines.”	**1.1**	System scans conversation messages for potential deadlines (e.g., *“Submit by Friday”*, *“Meeting at 3 PM”*).
2	-	**2.1**	AI Module highlights detected deadlines and displays them in a structured list (date/time + context message)..
3	User selects a detected deadline.	**3.1**	System offers options: “Set Reminder,” “Ignore,” or “View in Chat.”

Alternative Course of Action

S#	Actor Action	S#	System Response
1	User extracts deadlines for only selected messages.	**1.1**	System analyzes only those selected messages for deadlines.

2	User edits detected the deadline manually.	**2.1**	System updates deadline record accordingly.

Exception:

1. AI Module unavailable → system displays error and skips deadline extraction.
2. Ambiguous or unsupported date/time format → system asks user to confirm or edit manually.
3. No deadlines found → system shows “No deadlines detected.”

Use Case - 15 (Create Group Chat)

Identifier	UC-15
Name	Create Group Chat
Purpose	To allow a Group Admin to create a new group chat
Priority	High
Actors	Group Admin
Pre-conditions	1. Group Admin is logged in. 2. Necessary permissions to create a group exist.
Post–conditions	1. A new group chat is created and stored on the Local Server. 2. Group Admin becomes the default admin of the group.
Dependencies	1. Local Server available to store group information. 2. Included by Manage Group Membership use case.

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	Group Admin selects “Create Group” option	**1.1**	System displays group creation form
2	-	**2.1**	System creates group record and assigns Group Admin as owner
3	-	**3.1**	System confirms group creation and returns to Manage Group Membership

Alternative Course of Action

S#	Actor Action	S#	System Response
1	Group Admin cancels group creation	**1.1**	System aborts group creation and discards input

Exception:

1. Server unavailable → group creation fails.
2. Group name already exists → error returned.

Use Case - 16 (Manage Group Membership)

Identifier	UC-16
Name	Manage Group Membership
Purpose	To allow a Group Admin to add or remove members in a group chat
Priority	High
Actors	Group Admin
Pre-conditions	1. Group Admin is logged in. 2. At least one group exists (may be created via Create Group Chat).
Post–conditions	1. Members added or removed from the selected group. 2. Updated member list stored on Local Server.
Dependencies	1. Local Server available. 2. Includes Create Group Chat use case for group creation if no group exists.

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	Group Admin selects an existing group	**1.1**	System displays group details and member list.
2	Group Admin chooses “Add Member” and inputs user details	**2.1**	System validates user and adds to group
3	Group Admin chooses “Remove Member”	**3.1**	System removes the selected member from the group

Alternative Course of Action

S#	Actor Action	S#	System Response
1	Group Admin attempts to manage membership but no group exists	**1.1**	System invokes Create Group Chat use case first

Exception:

1. Attempt to add non-existent user → error message.
2. Server unavailable → membership update fails.

Use Case - 17 (Manage User Permissions in Group)

Identifier	UC-17
Name	Manage User Permissions in Group
Purpose	To allow a Group Admin to assign or revoke roles/permissions for members inside a group.
Priority	Medium
Actors	Group Admin
Pre-conditions	1. Group Admin is logged in. 2. Group already exists and has members.
Post–conditions	1. Member permissions updated (e.g., make admin, revoke admin). 2. New permissions stored on Local Server.
Dependencies	1. Local Server available. 2. Group membership must already be established (Manage Group Membership).

Typical Course of Action (Primary Flow)

S#	Actor Action	S#	System Response
1	Group Admin opens group settings	**1.1**	System displays list of members and their current permissions
2	Group Admin selects a member and chooses “Make Admin” or “Revoke Admin”	**2.1**	System updates the member’s permission in the group record
3	-	**3.1**	System confirms permission change to Group Admin

Alternative Course of Action

S#	Actor Action	S#	System Response
1	Group Admin tries to manage permissions for a member who left the group	**1.1**	System displays “User not in group” error

Exception:

1. Server unavailable → permission change fails.
2. Group Admin tries to revoke their own admin rights → system prompts

confirmation or blocks action depending on policy.