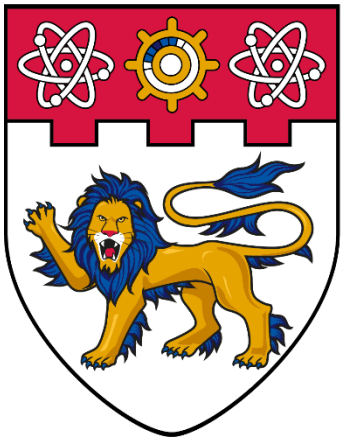


NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

SCSE23-1152

ChatAlone: An Offline Anonymous Messaging Application for Enhanced Privacy and Open Communication

Submitted By: Lin Run Yu

Matriculation Number: U2020490F

Supervisor: Dr. Owen Noel Newton Fernando

Submitted in partial fulfilment of requirements for degree in Computer Science by Nanyang Technological University

Contents

Abstract

Chat applications such as WhatsApp, WeChat, and Telegram have become ubiquitous in our daily interactions in our digital era. However, these applications compromise user anonymity by requiring personal information and a constant internet connection. While there are many anonymous chat applications available, they lack certain features such as multi-language interface and playing games.

This project aims to create ChatAlone, a Bluetooth-based chat application that allows users to communicate anonymously without needing Wi-Fi or cellular data. By utilizing Bluetooth's shorter range, ChatAlone enhances privacy by reducing eavesdropping chances. It is developed using Flutter and can send text messages, send images and play games. However, ChatAlone is likely to be misused due to anonymity. To combat this, an offensive content filter is implemented to prevent offensive content from being sent. ChatAlone supports English, Simplified Chinese, Malay, and Tamil and is available on Android and iOS devices.

Acknowledgments

Before continuing, I would like to express my gratitude to following people:

- Dr. Owel Noel Newton Fernando, who is my supervisor for this final-year project, for giving me inspirational ideas to work on and guiding me throughout.
- Prof Cham Tat Jen for reviewing and assessing my work.
- How Yi De, Carissa Lee Xue Wei and Jordan Yuen Jia Jun for their past year contributions to ChatAlone.

It was an amazing opportunity to work on this project to develop and practice what I have learned to develop this application.

List of Figures

List of Tables

1. Introduction

1.1. Background

Chat applications have become ubiquitous in our daily interactions in our digital era. Some popular chat applications today include but are not limited to WhatsApp, WeChat, Facebook Messenger, and Telegram [1]. These applications help people communicate with one another without being physically present. With one click, messages are sent instantaneously from one computing device to another with minimal effort. However, while chat applications make it easier for us to communicate, it compromises our anonymity. This is because most chat applications mandate users to reveal their personal information, such as phone numbers, before allowing users to use their service. They also require constant internet connection through Wi-Fi or cellular. Anonymity is essential as people are more expressive when they are anonymous. One study by Ji-Yong Yang and Changbeom Choi discovered that student participation increased after introducing a new teaching method using an anonymous chat service [2]. In addition, messages are vulnerable to eavesdropping attacks whilst transmitting through Wi-Fi or cellular [3]. One way to mitigate such attacks is by using Bluetooth. Bluetooth's maximum range is significantly shorter than Wi-Fi or cellular [4]. Therefore, hackers must be closer to their target devices to carry out their attacks.

1.2. Objectives

This project aims to create ChatAlone, a chat application that allows users to communicate with other users anonymously without Wi-Fi or cellular connection. Users need not reveal their personal information, such as phone numbers, before using ChatAlone. Communication is done through a peer-to-peer connection using Bluetooth. Development is done using Flutter.

1.3. Scope

ChatAlone is initially made available only in Singapore. This allows for a more controlled environment to monitor ChatAlone's features. ChatAlone is also available only on Android and iOS mobile operating systems. Limiting ChatAlone to these two operating systems ensures ease of development and maintenance. Furthermore, as of October 2024, Android and iOS account for 67.6% and 31% of Singapore's mobile operating system market share, respectively [5]. Therefore, ChatAlone is still accessible to many users in Singapore despite being limited to only two mobile operating systems.

2. Literature Review

2.1. Existing Solutions

These are some anonymous offline chat applications that have been developed:

- **Briar**: Briar is a free and open-source messaging app that allows users to communicate using Bluetooth, Wi-Fi or Tor. It features peer-to-peer encrypted messaging and stores data locally on user's device [6].
- **Bridgefy**: Bridgefy uses Bluetooth to send messages from user's phone to their friends over a distance of 100 meters. It is typically used during natural disasters, large events, and at school [7].
- **FireChat**: FireChat was an offline message that was widely used during major protests in Hong Kong, Taiwan, Iran and Iraq [8]. However, it has been discontinued.

Despite their rich features, Briar, Bridgefy and FireChat lack certain features that helps enhance user experience, as evident in following literatures.

2.2. Understanding Cultural Norms to Design Inclusive User Experiences

In a globally interconnected world, software applications have become universally accessible and widely adopted. However, many prominent platforms still maintain a predominantly West-centric approach. This presents challenges for users from non-Western cultural backgrounds, who may not be well-verse in western culture [9].

Briar, Bridgefy and FireChat only supports English as their interface language. In a multilingual country like Singapore, relying solely on English excludes a significant portion of Singaporeans who are more comfortable with other languages. This language barrier can hinder accessibility and adoption of such apps in multilingual societies.

Therefore, ChatAlone supports all of Singapore's official languages—English, Malay, Simplified Chinese, and Tamil—as its interface languages. This ensures that it is more accessible to users across various linguistic backgrounds in Singapore.

2.3. Role of Multiplayer Games in Enhancing Engagement

Research has shown that multiplayer games significantly contribute to users' social and psychological well-being. This is done primarily through enhancing social interaction, cognitive stimulation, and emotional connection. This is because multiplayer games offer unique environments for players to develop social ties and collaborative skills. In addition, multiplayer games provide mental stimulation, help to reduce stress, and enhance mood by fostering a sense of community and social support among players. Furthermore, games provide an environment for skill development and cognitive engagement, where users practice problem-solving and adaptability, contributing to a positive user experience [10].

Bridgefy, Briar, and FireChat lack interactive features such as games. This limits their potential for engagement and makes conversations feel static and less engaging.

Therefore, ChatAlone has games to allow users to play games such as Tic Tac Toe, Connect 4, and Othello while engaging in conversations to make chatting more enjoyable and offer users a chance to relax.

2.4. Influences of Anonymity and Social Modeling

Research has shown that anonymity leads to significantly increase aggression in online environments, where participants engaged more in aggressive behaviors when they felt unidentifiable. This is a phenomenon known as cyber-disinhibition, where users will express hostility that they would typically be suppressed during face-to-face interactions due to social norms and accountability [11].

There are no restrictions on Briar, Bridgefy and FireChat on what content cannot be sent. As such, users are more likely to send offensive content such as profanity-filled messages or pornographic images to others.

Therefore, an offensive content filter is implemented on ChatAlone to prevent users from sending offensive content. This ensures that more fruitful conversations can be had through ChatAlone.

3. ChatAlone Requirements

3.1. Functional Requirements

1. Settings
    1.1. Application must prompt new user to setup.
        1.1.1. Application must prompt user to input their username.
            1.1.1.1. Username must be between 1 to 25 characters.
            1.1.1.2. Username must not contain special characters (@, |, $...).
            1.1.1.3. Username must not contain profanity.
        1.1.2. Application must allow user to select their preferred language.
        1.1.3. Application must allow user to select theme.
        1.1.4. Application must check username for special characters or profanity.
            1.1.4.1. Application must provide error messages if invalid input is provided.
        1.1.5. After successful setup, user must be redirected to 'Home' page.
    1.2. Application must allow users to update settings.
        1.2.1. Application must provide 'Settings' page where users can update their username, language, and theme.
        1.2.2. Application must validate new usernames to avoid special characters or profanity.
            1.2.2.1. Application must provide error messages if invalid input is provided.
        1.2.3. Settings must take effect after user confirms changes.

2. Chat Session
    2.1. Application must allow user to join person chat.
        2.1.1. Application must display list of nearby users with Bluetooth enabled.
        2.1.2. Application must allow user to select person to chat with and send request.
        2.1.3. Application must notify user if their request to chat is accepted or denied.
    2.2. Application must allow users to create group chat.
        2.2.1. Application must prompt user to input their group name.
            2.2.1.1. Group name must be between 1 to 25 characters.
            2.2.1.2. Group name must not contain special characters (@, |, $...).
            2.2.1.3. Group name must not contain profanity.
        2.2.2. Application must notify other users of group chat existence.
    2.3. Application must allow users to join group chat.
        2.3.1. Application must display nearby group chats that user can join.
        2.3.2. Application must notify user if their request to join group chat is denied or accepted.

3. Message
   3.1. Application must allow user to send message.
      3.1.1. Application must allow users to send text messages in person or group chat.
      3.1.2. Application must validate messages for profanity before sending them.
      3.1.3. Application must display message in conversation area after sending them.
   3.2. Application must allow user to copy message.
      3.2.1. Application must store copied messages onto clipboard.
   3.3. Application must allow users to edit previously sent messages.
      3.3.1. Application must display original message above input field during editing.
      3.3.2. Application must validate edited message for profanity.
   3.4. Application must allow users to delete previously sent messages.
      3.4.1. Application must ask for confirmation before permanently deleting message.
   3.5. Application must allow users to reply to specific received messages.
      3.5.1. Replied message must be displayed above input field before sending.
      3.5.2. Replied message must be linked to new message after sending.
   3.6. Application must allow users to react to received messages using emojis.
      3.6.1. Reaction must be displayed below message it applies to.
   3.7. Application must allow users to send messages that self-destruct after set time.
      3.7.1. Application must allow users to set self-destruct timer before sending message.
      3.7.2. Message must be automatically deleted after set time has passed.

4. Image
   4.1. Application must allow users to select and send image in person or group chat.
   4.2. Application must provide confirmation before sending image.
   4.3. Application must ensure image size is not more than 20KB.
      4.3.1. Application must notify user if image is more than 20KB.
   4.4. Application must validate image for profanity or nudity.
      4.4.1. Application must notify user if image contains profanity or nudity.

5. Game
   5.1. Application must allow users to initiate a game (Tic-Tac-Toe, Connect 4, or Othello) in person chat.
      5.1.1. Application must notify other user of game request
      5.1.2. Application must allow other user to accept or deny.
   5.2. Application must allow users to end ongoing game.
      5.2.1. Application must return both users to person chat session after game ends.

6. End Chat Session
    6.1. Application must allow users to end person chat session.
        6.1.1. Application must send termination command to other user.
        6.1.2. Application must return both users to ‘Home’ page after session ends.
    6.2. Application must allow users to leave group chat session.
        6.2.1. Application must notify other users when someone leaves group chat.
    6.3. Application must allow group chat creator to end group chat session.
        6.3.1. Application must notify all group participants that session is over.
        6.3.2. Application must return all users to ‘Home’ page after session ends.

7. View Old Chats
    7.1. Application must allow users to view old person or group chat sessions.
    7.2. Application must display list of old chats for selection.
    7.3. Application must notify user if no old chats are available.

8. Search
    8.1. Application must allow users to search for specific messages in both active and old chats.
    8.2. Application must highlight search results within chat.
    8.3. Application must provide navigation options (e.g., up/down arrows) for users to browse search results.

3.2. Non-Functional Requirements

1. Reliability
   1.1. Application must maintain stable Bluetooth connections for as long as possible.
   1.2. All messages sent between users must be delivered within 5 seconds after being sent.

2. Usability
   2.1. User interface must be intuitive and easy to navigate.
   2.2. Application must allow users to select any of English, Simplified Chinese, Malay, and Tamil as its interface language.

3. Security
   3.1. Users' information must not be required to use application.
   3.2. Application must store all data locally.

4. Availability
   4.1. Users must be able to download application from Apple App Store in Singapore.
   4.2. Users must be able to download application from Google Play Store in Singapore.

5. Scalability
   5.1. Application must support group chats of up to 5 devices without significant performance degradation.
   5.2. Application must remain responsive and stable in long-running chat sessions for at least 15 minutes.

3.3. Use Case Diagram

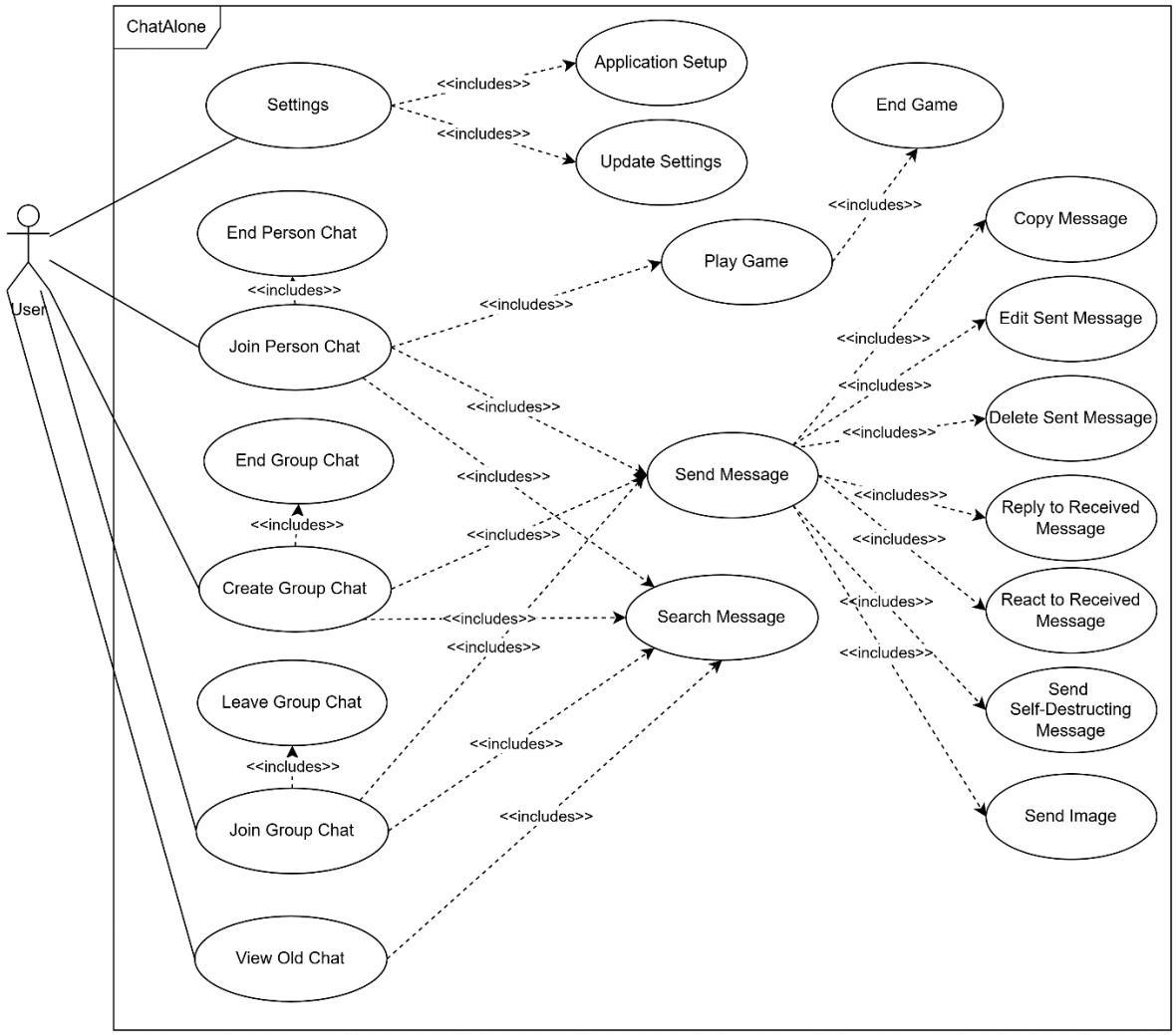


Figure 1: Use Case Diagram

3.4. Use Case Descriptions

Use Case ID:	1		
Use Case Name:	Application Setup		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	New User		
Description:	User installs and configures ChatAlone for first time		
Preconditions:	User has downloaded ChatAlone		
Postconditions:	User has successfully set up ChatAlone		
Priority:	High		
Use Frequency:	Once per device		
Event Flow:	1. User opens application 2. User input their desired username 3. User selects their desired language 4. User selects their theme 5. User clicks 'Confirm' button 6. Application brings user to 'Home' page		
Alternative Flows:	AF1: No username 1. User clicks 'Confirm' button without entering username 2. Application prompts user to enter username AF2: Invalid username 1. User enters special characters or profanities 2. User clicks 'Confirm' button 3. Application prompts user to not enter special characters or profanities		
Exceptions:	None		
Includes:	None		
Assumptions:	User is using latest version of Android or iOS		
Notes & Issues:	Ensure compatibility with various Android or iOS versions		

Table 1: 'Application Setup' Use Case

Use Case ID:	2		
Use Case Name:	Update Settings		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User updates settings like username, language or theme		
Preconditions:	User is at 'Home' page		
Postconditions:	Settings are updated to reflect user's preferences		
Priority:	Medium		
Use Frequency:	Occasionally		
Event Flow:	1. User press 'Settings' button on 'Home' page 2. Application brings user to 'Settings' page 3. User selects setting to be changed (username, language, theme). 4. User press 'Confirm' button		

	5. Application brings user to ‘Home’ page
Alternative Flows:	AF1: No username 1. User clicks ‘Confirm’ button without entering username 2. Application prompts user to enter username AF2: Invalid username 1. User enters special characters or profanities 2. User clicks ‘Confirm’ button 3. Application prompts user to not enter special characters or profanities
Exceptions:	None
Includes:	None
Assumptions:	User understands implications of their settings choices
Notes & Issues:	Test for responsiveness to setting changes

Table 2: 'Update Settings' Use Case

Use Case ID:	3		
Use Case Name:	Join Person Chat		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User joins person chat with other user		
Preconditions:	User has Bluetooth enabled and is at ‘Home’ page		
Postconditions:	User is actively participating in person chat		
Priority:	High		
Use Frequency:	Frequently		
Event Flow:	1. User press ‘New Chat’ button 2. Application opens ‘New Chat’ dialog 3. User press ‘Person Chat’ button 4. Application brings user to ‘Nearby Person’ page 5. Application displays nearby persons 6. User selects person to request chat 7. Other user accepts request 8. User press ‘Message’ button to start chatting		
Alternative Flows:	AF1: Request is denied by other user 1. Other user denied request 2. Application notifies user that request has been denied		
Exceptions:	Bluetooth connectivity issues		
Includes:	Send Message, Play Game, Search Message, End Person Chat		
Assumptions:	Users are within Bluetooth range		
Notes & Issues:	None		

Table 3: 'Join Person Chat' Use Case

Use Case ID:	4		
Use Case Name:	Create Group Chat		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024

Actors:	User
Description:	User creates group chat
Preconditions:	User has Bluetooth enabled and is at 'Home' page
Postconditions:	User is actively participating in group chat
Priority:	High
Use Frequency:	Regularly
Event Flow:	1. User press 'New Chat' button 2. Application opens 'New Chat' dialog 3. User press 'Create Group Chat' button 4. Application prompts user to create group name 5. User enters group name and clicks 'Confirm' button 6. Application brings user to 'Nearby Person' page 7. Application displays nearby persons 8. User selects person to join group chat 9. Other user accepts request 10. User press 'Message' button to start chatting
Alternative Flows:	AF1: No group name 1. User clicks 'Confirm' button without entering group name 2. Application prompts user to enter group name AF2: Invalid group name 1. User enters special characters or profanities 2. User clicks 'Confirm' button 3. Application prompts user to not enter special characters or profanities AF3: Request is denied by other user 1. Other user denied request 2. Application notifies user that request has been denied
Exceptions:	Bluetooth connectivity issues
Includes:	Send Message, Search Message, End Group Chat
Assumptions:	Users are within Bluetooth range
Notes & Issues:	Monitor for scalability issues with large groups.

Table 4: 'Create Group Chat' Use Case

Use Case ID:	5		
Use Case Name:	Join Group Chat		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User joins group chat		
Preconditions:	User has Bluetooth enabled and is at 'Home' page		
Postconditions:	User is actively participating in group chat		
Priority:	High		
Use Frequency:	Regularly		
Event Flow:	1. User press 'New Chat' button 2. Application opens 'New Chat' dialog 3. User press 'Create Group Chat' button 4. Application brings user to 'Nearby Group' page 5. Application displays nearby groups		

	6. User selects group to join 7. Other user accepts request 8. User press ‘Message’ button to start chatting
Alternative Flows:	AF1: Request is denied by other user 1. Other user denied request 2. Application notifies user that request has been denied
Exceptions:	Bluetooth connectivity issues
Includes:	Send Message, Search Message, Leave Group Chat
Assumptions:	Users are within Bluetooth range
Notes & Issues:	Monitor for scalability issues with large groups.

Table 5: 'Join Group Chat' Use Case

Use Case ID:	6		
Use Case Name:	Send Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User sends text messages within person or group chat		
Preconditions:	User is in active person or group chat		
Postconditions:	Message is visible to all participants in chat		
Priority:	High		
Use Frequency:	Very High		
Event Flow:	1. User enters message in chat input field 2. User presses 'Send' button 3. Application displays message in chat window		
Alternative Flows:	AF1: Message contains profanity 1. User enters message containing profanity 2. Application notifies user to not send those messages		
Exceptions:	Delivery issues due to Bluetooth instability		
Includes:	Copy Message, Edit Sent Message, Delete Sent Message, Reply to Received Message, React to Received Message, Send Self-Destructing Message, Send Image		
Assumptions:	User has active Bluetooth connection		
Notes & Issues:	Ensure message delivery confirmation feature is robust		

Table 6: 'Send Message' Use Case

Use Case ID:	7		
Use Case Name:	Copy Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User copies message from chat to clipboard		
Preconditions:	Message is present in chat window		
Postconditions:	Message is stored in clipboard.		
Priority:	Low		
Use Frequency:	Occasionally		

Event Flow:	1. User long presses on message 2. Application opens context menu 3. User selects 'Copy' from context menu 4. Application copies message to clipboard
Alternative Flows:	None
Exceptions:	Clipboard access permissions denied
Includes:	None
Assumptions:	None
Notes & Issues:	None

Table 7: 'Copy Message' Use Case

Use Case ID:	8		
Use Case Name:	Edit Sent Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User edits message they have previously sent		
Preconditions:	User has already sent message		
Postconditions:	Message is updated		
Priority:	Medium		
Use Frequency:	Rarely		
Event Flow:	1. User long presses on message they sent 2. Application opens context menu 3. User selects 'Edit' from context menu 4. Application displays original message above chat input field 5. User edits message 6. User presses ‘Send’ button 7. Application updates message		
Alternative Flows:	AF1: User decides not to edit message afterward 1. Application displays original message above chat input field 2. User press ‘X’ button AF2: Message contains profanity 1. User enters message containing profanity 2. Application notifies user to not send those messages		
Exceptions:	Unable to edit on other user’s phone due to Bluetooth instability		
Includes:	None		
Assumptions:	None		
Notes & Issues:	None		

Table 8: 'Edit Sent Message' Use Case

Use Case ID:	9		
Use Case Name:	Delete Sent Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User deletes message they have previously sent		

Preconditions:	User has already sent message
Postconditions:	Message is deleted
Priority:	Medium
Use Frequency:	Rarely
Event Flow:	1. User long presses on message they sent 2. Application opens context menu 3. User selects 'Delete' from context menu 4. Application ask user whether to delete 5. User press ‘Confirm’ button 6. Application deletes message
Alternative Flows:	AF1: User decides not to edit message afterward 1. Application ask user whether to delete 2. User press ‘Cancel’ button
Exceptions:	Unable to delete on other user’s phone due to Bluetooth instability
Includes:	None
Assumptions:	None
Notes & Issues:	None

Table 9: 'Delete Sent Message' Use Case

Use Case ID:	10		
Use Case Name:	Reply to Received Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User replies to message they received		
Preconditions:	User received message from other user		
Postconditions:	Reply is linked to received message		
Priority:	High		
Use Frequency:	Frequently		
Event Flow:	1. User long presses on message they received 2. Application opens context menu 3. User selects 'Reply' from context menu 4. Application displays original message above chat input field 5. User enters their message 6. User presses ‘Send’ button 7. Application sends message with original message displayed on top		
Alternative Flows:	AF1: User decides not to reply message afterward 1. Application displays original message above chat input field 2. User press ‘X’ button AF2: Message contains profanity 1. User enters message containing profanity 2. Application notifies user to not send those messages		
Exceptions:	Delivery issues due to Bluetooth instability		
Includes:	None		
Assumptions:	None		
Notes & Issues:	None		

Table 10: 'Reply to Received Message' Use Case

Use Case ID:	11		
Use Case Name:	React to Received Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User reacts to message using emojis		
Preconditions:	User received message from other user		
Postconditions:	Reaction is displayed below message.		
Priority:	Low		
Use Frequency:	Frequently		
Event Flow:	1. User long presses on message they received 2. Application opens context menu 3. User selects 'React' from context menu 4. Application displays emojis 5. User selects emoji 6. Application displays emoji below message		
Alternative Flows:	AF1: User decides not to react message afterward 1. Application displays emojis 2. User press ‘Cancel’ button		
Exceptions:	Delivery issues due to Bluetooth instability		
Includes:	None		
Assumptions:	None		
Notes & Issues:	None		

Table 11: 'React to Received Message' Use Case

Use Case ID:	12		
Use Case Name:	Send Self-Destructing Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User sends message that automatically deletes after set time		
Preconditions:	User is in active person or group chat		
Postconditions:	Message deletes itself after set time.		
Priority:	Medium		
Use Frequency:	Rarely		
Event Flow:	1. User enters message in chat input field 2. User long presses 'Send' button 3. Application display dialog for user to set time 4. User sets time 5. User press ‘Send’ button 6. Application displays message in chat window for set time 7. Application deletes message after set time		
Alternative Flows:	AF1: User decides not to send self-destructing message afterward 1. Application display dialog for user to set time 2. User press ‘Cancel’ button AF2: Message contains profanity 1. User enters message containing profanity		

	2. Application notifies user to not send those messages
Exceptions:	Delivery issues due to Bluetooth instability
Includes:	None
Assumptions:	User has active Bluetooth connection
Notes & Issues:	Ensure message delivery confirmation feature is robust

Table 12: 'Send Self-Destructing Message' Use Case

Use Case ID:	13		
Use Case Name:	Send Image		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User sends image to other users in person or group chat		
Preconditions:	User is in active person or group chat		
Postconditions:	Image is visible to all participants in chat		
Priority:	High		
Use Frequency:	Regularly		
Event Flow:	1. User press 'Send Image' button 2. Application displays user's images 3. User chooses image 4. Application ask user whether to send 5. User press 'Send' button 6. Application displays image in chat window		
Alternative Flows:	AF1: User decides not to send image afterward 1. Application ask user whether to send 2. User press 'Cancel' button AF2: Image is 20KB or more 1. User chooses image that is 20KB or more 2. Application notifies user that those images cannot be sent AF3: Image contains nudity or profanity 1. User sends image containing nudity or profanity 2. Application notifies user to not send those images		
Exceptions:	Technical issues prevent image upload		
Includes:	None		
Assumptions:	None		
Notes & Issues:	Validate supported image formats and size limitations		

Table 13: 'Send Image' Use Case

Use Case ID:	14		
Use Case Name:	Play Game		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User initiate Tic-Tac-Toe, Connect 4, or Othello		
Preconditions:	User is in active person chat		
Postconditions:	User plays Tic-Tac-Toe, Connect 4, or Othello		

Priority:	Low
Use Frequency:	Occasionally
Event Flow:	1. User press 'Play Game' button 2. Application displays game menu 3. User chooses game 4. Application sends game request to other user 5. Other user accepts request 6. Application brings user to a game page
Alternative Flows:	AF1: Request is denied by other user 1. Other user denied request 2. Application notifies user that request has been denied
Exceptions:	Unable to play game due to Bluetooth connectivity issues
Includes:	End Game
Assumptions:	None
Notes & Issues:	Monitor game performance and ensure fair play mechanics

Table 14: 'Play Game' Use Case

Use Case ID:	15		
Use Case Name:	End Game		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User decides to end ongoing game		
Preconditions:	User is engaged in game		
Postconditions:	Game session is terminated		
Priority:	High		
Use Frequency:	Rarely		
Event Flow:	1. User press 'End Game' button 2. Application ask user whether to end game 3. User press ‘Confirm’ button 4. Application sends terminate game command to other user 5. Application brings both user and other user to ‘Person Chat Session’ page		
Alternative Flows:	AF1: User decides not to end game afterward 1. Application ask user whether to end game 2. User press ‘Cancel’ button		
Exceptions:	Unable to send terminate game command to other user due to Bluetooth connectivity issues		
Includes:	None		
Assumptions:	None		
Notes & Issues:	Ensure smooth transition back to ‘Person Chat Session’ page after game ends		

Table 15: 'End Game' Use Case

Use Case ID:	16
Use Case Name:	End Person Chat

Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User decides to end person chat session		
Preconditions:	User is in active person chat		
Postconditions:	Person chat session is closed		
Priority:	High		
Use Frequency:	Frequently		
Event Flow:	1. User press 'Exit' button 2. Application ask user whether to end person chat session 3. User press 'Confirm' button 4. Application sends terminate person chat command to other user 5. Application brings both user and other user to 'Home' page		
Alternative Flows:	AF1: User decides not to end person chat session afterward 1. Application ask user whether to end person chat session 2. User press 'Cancel' button		
Exceptions:	Unable to send terminate person chat session command to other user due to Bluetooth connectivity issues		
Includes:	None		
Assumptions:	None		
Notes & Issues:	Ensure smooth transition back to 'Home' page after game ends		

Table 16: 'End Person Chat' Use Case

Use Case ID:	17		
Use Case Name:	Leave Group Chat		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User leaves group chat they are part of		
Preconditions:	User is member of group chat		
Postconditions:	User is no longer part of group chat		
Priority:	High		
Use Frequency:	Frequently		
Event Flow:	1. User press 'Exit' button 2. Application ask user whether to leave group chat session 3. User press 'Confirm' button 4. Application notifies other users 5. Application brings user to 'Home' page		
Alternative Flows:	AF1: User decides not to end person chat session afterward 1. Application ask user whether to leave group chat session 2. User press 'Cancel' button		
Exceptions:	Unable to notify other users due to Bluetooth connectivity issues		
Includes:	None		
Assumptions:	None		
Notes & Issues:	Ensure smooth transition back to 'Home' page after game ends		

Table 17: 'Leave Group Chat' Use Case

Use Case ID:	18		
Use Case Name:	End Group Chat		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User decides to end group chat session		
Preconditions:	User is group chat creator		
Postconditions:	Group chat session is closed		
Priority:	High		
Use Frequency:	Frequently		
Event Flow:	1. User press 'Exit' button 2. Application ask user whether to end group chat session 3. User press 'Confirm' button 4. Application sends terminate group chat command to other users 5. Application brings both user and other users to 'Home' page		
Alternative Flows:	AF1: User decides not to end person chat session afterward 1. Application ask user whether to end group chat session 2. User press 'Cancel' button		
Exceptions:	Unable to send terminate group chat session command to other users due to Bluetooth connectivity issues		
Includes:	None		
Assumptions:	None		
Notes & Issues:	Ensure smooth transition back to 'Home' page after game ends		

Table 18: 'End Group Chat' Use Case

Use Case ID:	19		
Use Case Name:	View Old Chats		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User views old person chat or group chat sessions		
Preconditions:	User is at 'Home' page		
Postconditions:	User reads old chat		
Priority:	Low		
Use Frequency:	Occasionally		
Event Flow:	1. User press 'Old Chat' button 2. Application opens 'Old Chat' dialog 3. User press 'Person Chat' or 'Group Chat' button 4. Application brings user to 'Person Chat Session or 'Group Chat Session' page 5. Application displays old person chat or group chat 6. User selects old chat 7. Application brings user to 'Old Chat Session' page		
Alternative Flows:	AF1: No old chats 1. Application brings user to 'Person Chat Session or 'Group Chat Session' page		

	2. Application notifies user that there are no old chats
Exceptions:	Errors in loading old chats due to corrupted data
Includes:	Search Message
Assumptions:	None
Notes & Issues:	Consider storage limitations for storing old chats

Table 19: 'View Old Chats' Use Case

Use Case ID:	20		
Use Case Name:	Search Message		
Created By:	Run Yu	Last Updated By:	Run Yu
Date Created:	9/9/2024	Date Last Updated:	10/10/2024
Actors:	User		
Description:	User searches for specific messages within chat session		
Preconditions:	User is in person or group chat that is active or old		
Postconditions:	User finds specific messages based on their query		
Priority:	Medium		
Use Frequency:	Frequently		
Event Flow:	1. User press on 'Search' button 2. Application displays search bar 3. User enters their query 4. Application highlights all messages containing query 5. User clicks up or down arrows to navigate 6. Application scrolls up or down to display highlighted queries		
Alternative Flows:	AF1: No results 1. User enters search term that yields no results 2. Application notifies user there are no results		
Exceptions:	Search function is unavailable due to technical issues		
Includes:	None		
Assumptions:	None		
Notes & Issues:	Search needs to be fast and accurate		

Table 20: 'Search Message' Use Case

4. ChatAlone Design

4.1. System Architecture

ChatAlone is designed with 3-tier layered architecture. It is divided into presentation layer, logic layer and data layer. Each layer has its own focused responsibilities, thus reducing dependencies between different parts [12]. Presentation layer is responsible for ChatAlone's user interface where users interact with ChatAlone. Logic layer is responsible for ChatAlone's core functionality and rules. Data layer is responsible for data storage and retrieval. This architecture provides clear separation of concerns, making ChatAlone more modular and easier to update or expand.

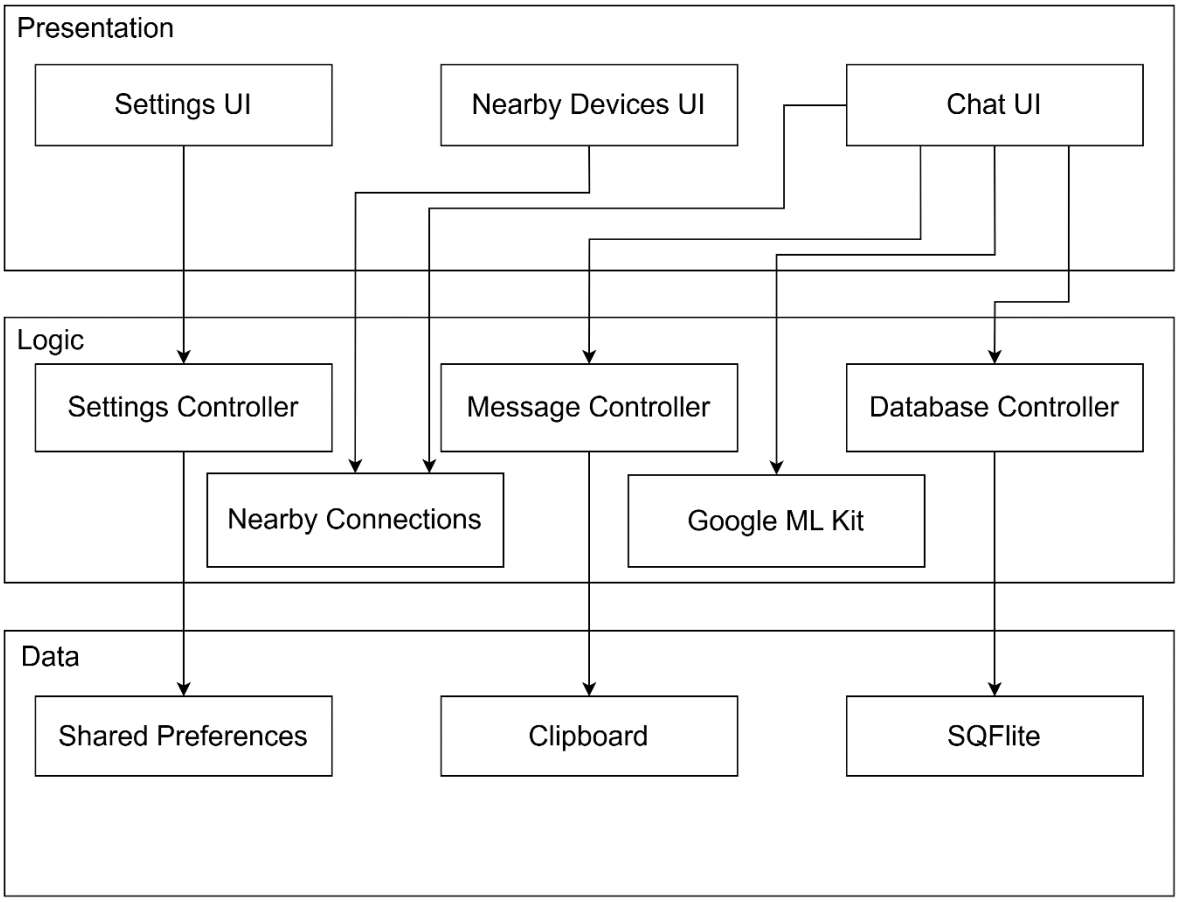


Figure 2: System Architecture Diagram

4.2. Database Design

ChatAlone utilizes combination of Shared Preferences, Clipboard and SQFlite to store data. Shared Preferences is used to store lightweight data such as username, language and theme configurations. Clipboard is used to allow users to copy messages, making it easy to share text content across various applications. SQFlite, Flutter's plugin for SQLite, is used to store person chat and group chat conversation histories. Tables below shows ChatAlone's SQFlite's database schema diagram. This schema abides by Third Normal Form (3NF), where there are no transitive dependencies. This means that there is no non-key attribute that is dependent on another non-key attribute [13].

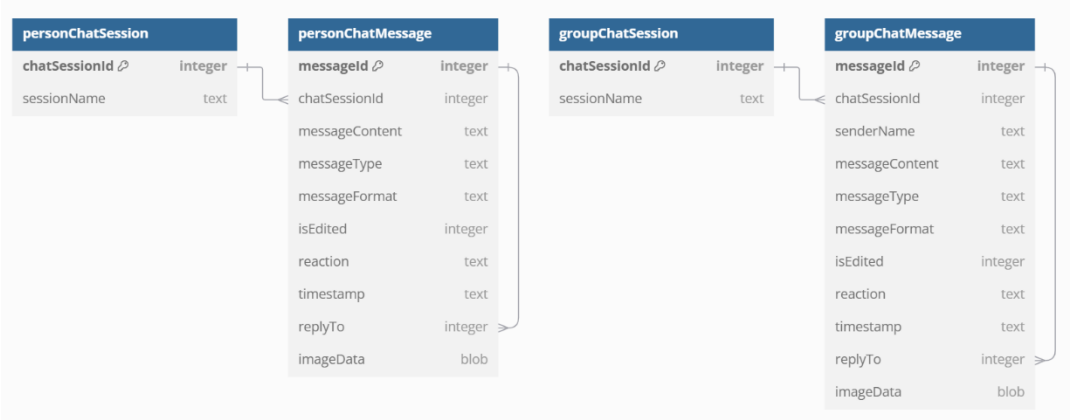


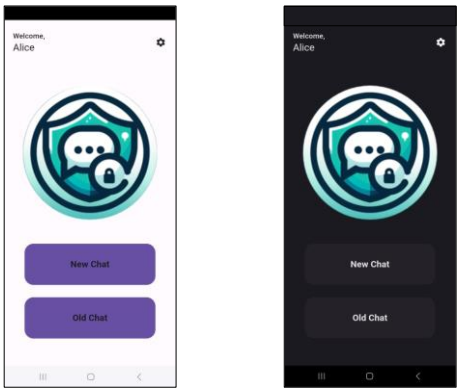
Figure 3: SQFlite Database Schema Diagram

Attribute	Data Type	Description
chatSessionId	Integer	ID for person or group chat session.
sessionName	Text	Name of person or group chat session. Session name for person chat is receiver's name while session name for group chat is determined by group chat creator.
messageId	Integer	ID for messages.
senderName	Text	Name of sender, only applicable to group chat sessions.
messageContent	Text	Content of messages.
messageType	Text	Type of messages, whether is it sender message, received message, or system message.
messageFormat	Text	Formatting of messages, whether it is bold, underline, italic, etc.
isEdited	Integer	Whether message has been modified after sent.
reaction	Text	Reaction to messages using emojis (👍 , ❤️ , 🤣).
timestamp	Text	Time message is sent.
replyTo	Integer	Reference to messageId message is replying to.
imageData	Binary Large Object	Image data that is base64 encoded.

Table 21: SQFlite Database Attributes' Definitions

4.3. User Interfaces

Light or Dark Mode: ChatAlone supports both light mode and dark mode. Users can manually decide which mode they want, or they can follow their device settings. This ensures that it is visually appealing and comfortable to use in various lighting conditions.



Light Mode Dark Mode

Figure 4: Light or Dark Mode

Multi-language Support: ChatAlone supports English, Malay, Simplified Chinese and Tamil as its interface language. Multi-language support is implemented by using Flutter's localization system [14]. ARB files are used to store key-value pairs for each supported language. Users choose their preferred language in ChatAlone's settings, and their choice is saved using Shared Preferences. ChatAlone's UI is updated in real-time when specific language is selected.

English Malay Simplified Chinese Tamil

Figure 5: Various Languages' Interfaces

Settings: ChatAlone's 'Settings' page is designed to offer users to configure and personalize their application. It allows users to set their username, language and theme preferences. It will be displayed when users first setup. Users can also access it from 'Home' page after setting up.

Figure 6: 'Settings' Page

Home: Whenever users launch ChatAlone after setting up, they will be directed to its 'Home' page. They are greeted by welcoming message at top. There is also 'Settings' button at top right corner to access 'Settings' page. Below, there are 2 main functional buttons, 'New Chat' and 'Old Chat'. Pressing 'New Chat' allows users to start person chat, create group chat or join group chat. Pressing 'Old Chat' button allows users to view their old person or group chat session.

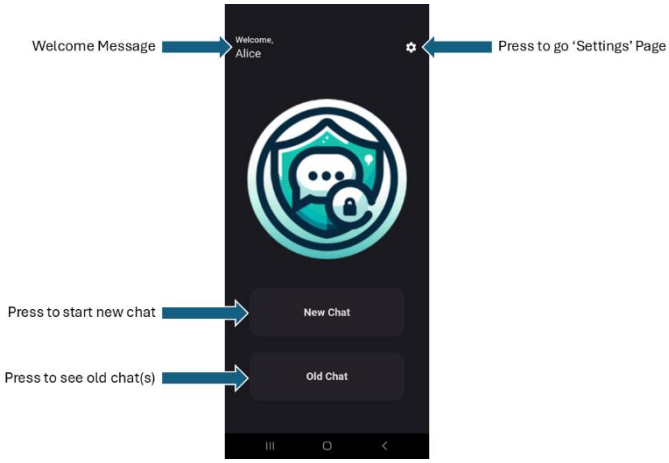


Figure 7: 'Home' Page

Person Chat Nearby Person: When users want to start person chat, they are first directed to 'Nearby Person' page. If there are other users who are nearby using ChatAlone, their name will be displayed. User can then press 'Connect' button to connect with them via Bluetooth. Upon connection, users can press 'Message' button to start person chat session.

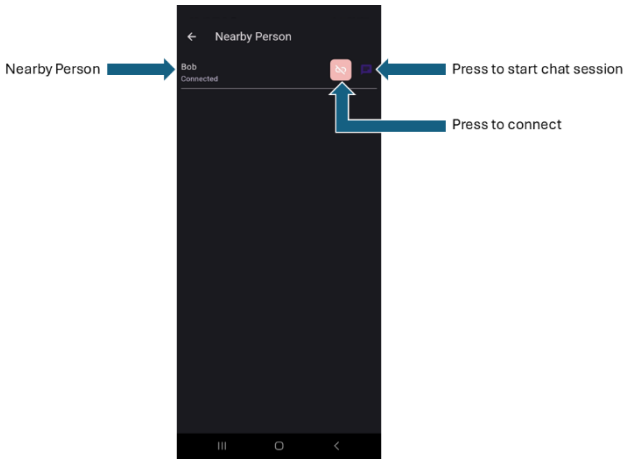


Figure 8: 'Person Chat Nearby Person' Page

Person Chat Session: 'Person Chat Session' page is where users get to chat with one another. Person's name is prominently shown at top left corner. Users can type their desired message in message input box and send it by clicking 'Send' button at bottom right corner. If users want to send self-destructing messages, they can do so by long-pressing 'Send' button. Message sent by user is at right side while messages received is at left side. If users wish to search message, they can do so by pressing 'Search' button at top right corner. Users can also press 'Menu' button at bottom right to send images or play games. After users are done with chatting, they can exit by pressing 'Exit' button at top right corner.

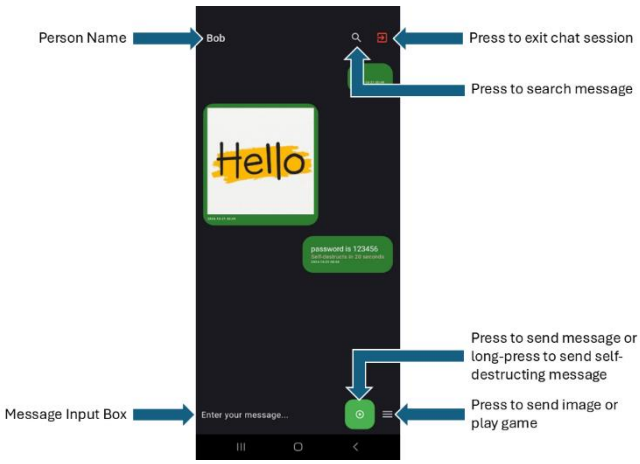


Figure 9: 'Person Chat Session' Page

Tic-Tac-Toe: One game offered by ChatAlone is Tic-Tac-Toe. For Tic-Tac-Toe, users take turns placing Xs or Os on 3x3 grid. Their aim is to be first to align three symbols in a row, column, or diagonal. User who initiates Tic-Tac-Toe will be 'X' and will always go first. When 'Your turn' is displayed, user can place their symbol on their desired tile. After placing, page will display 'Opponent's turn' for opponent to play, and user will not be able to place anything until their opponent make their move. Users can end game anytime by pressing 'End Game' button, in which both users will be redirected back to their chat session.

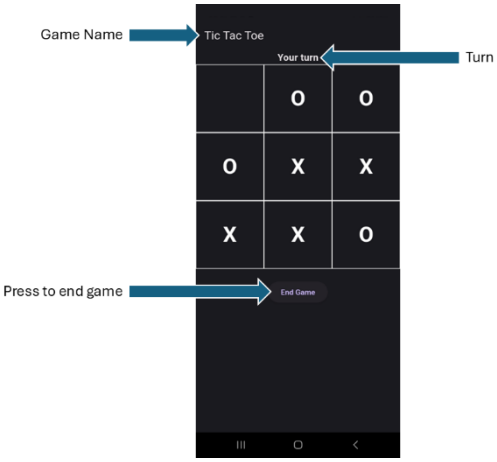


Figure 10: 'Tic-Tac-Toe' Page

Connect 4: Another game offered by ChatAlone is Connect 4. For Connect 4, users alternate dropping colored discs into vertical grid. Their aim is to connect four of their discs in row, column, or diagonal. User who initiates Connect 4 will have red disc and will always go first. When 'Your turn' is displayed, user can drop their disc on their desired column. After dropping, page will display 'Opponent's turn' for opponent to play, and user will not be able to place anything until their opponent make their move. Users can end game anytime by pressing 'End Game' button, in which both users will be redirected back to their chat session.

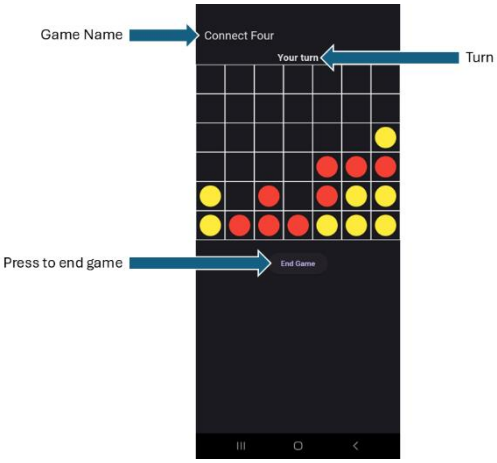


Figure 11: 'Connect 4' Page

Othello: Last game offered by ChatAlone is Othello. For Othello, users take turns placing discs on 8x8 board. Their aim is to flip as many of their opponent's discs as possible. User who initiates Othello will have black disc and will always go first. When 'Your turn' is displayed, user can place their disc on their desired tile. After placing, page will display 'Opponent's turn' for opponent to play, and user will not be able to place anything until their opponent make their move. Users can end game anytime by pressing 'End Game' button, in which both users will be redirected back to their chat session.

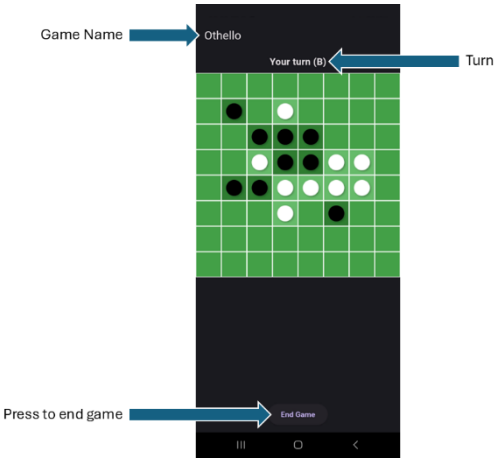


Figure 12: 'Othello' Page

Create Group Chat Nearby Person: When users want to create group chat, they are first required to come up with a group name. They are then directed to 'Nearby Person' page. If there are other users who are nearby using ChatAlone who wants to join group, their name will be displayed. User can then press 'Connect' button to connect with them via Bluetooth. Upon connection, users can press 'Start Chat' button to start group chat session.

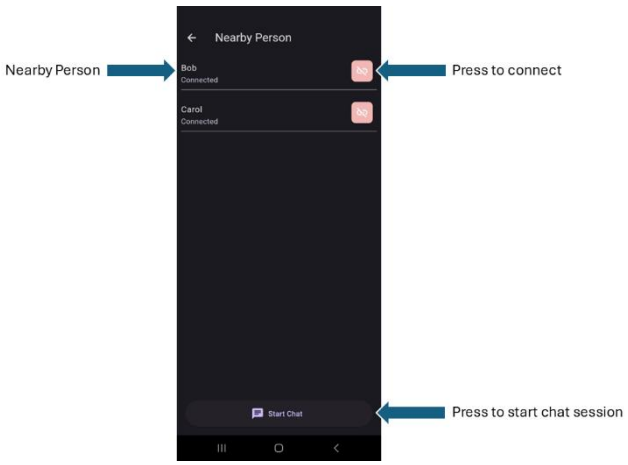


Figure 13: 'Create Group Chat Nearby Person' Page

Join Group Chat Nearby Group: When users want to join group chat, they are first directed to 'Nearby Group' page. If there are other groups who that are created nearby, that group name will be displayed. User can then press 'Connect' button to connect with that group via Bluetooth. Upon connection, users can press 'Message' button to start group chat session.

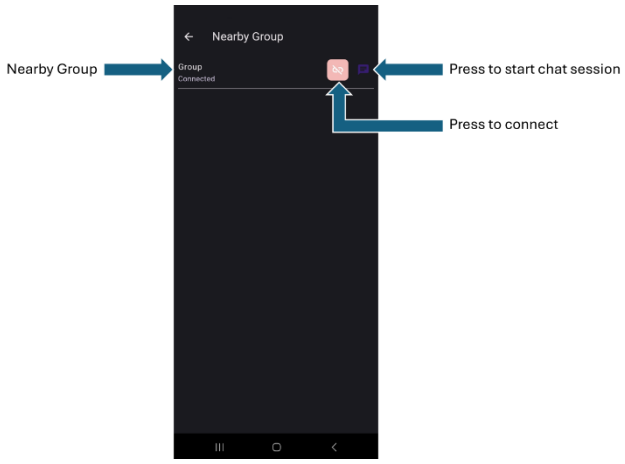


Figure 14: 'Join Group Chat Nearby Group' Page

Group Chat Session: 'Group Chat Session' page is where group members get to chat with one another. Group's name is prominently shown at top left corner. Users can type their desired message in message input box and send it by clicking 'Send' button at bottom right corner. If users want to send self-destructing messages, they can do so by long-pressing 'Send' button. Message sent by user is at right side while messages received is at left side. If users wish to search message, they can do so by pressing 'Search' button at top right corner. Users can also press 'Image' button at bottom right to send images. For group members, exiting group chat will trigger notification to inform remaining members that they have left. However, if group creator decides to end chat, group chat session will be terminated for all members.

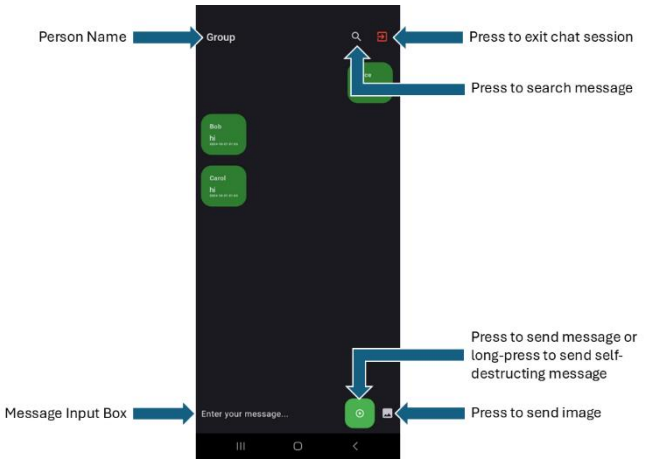


Figure 15: 'Group Chat Session' Page

Old Chat: After users end person or group chat session, their messages are stored locally on their phones. Users can access it by clicking on 'Old Chat' button on their 'Home' page. Old chat sessions are displayed in descending order, with newest chat session at top and oldest at bottom. Users can also delete old chat sessions by pressing 'Trash' button.

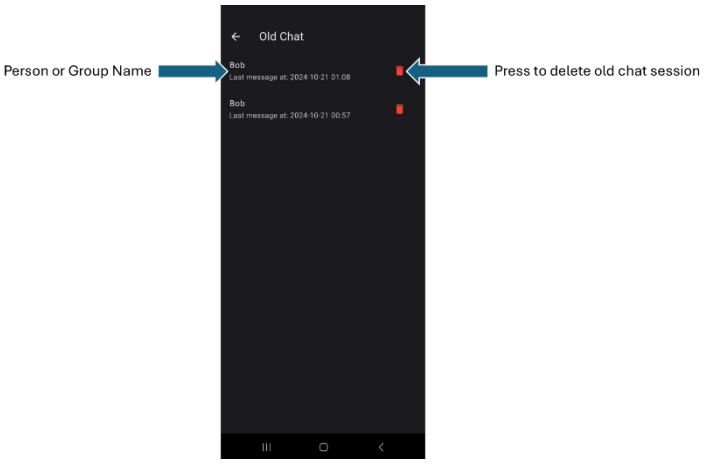


Figure 16: 'Old Chat' Page

Old Chat Session: Once user selects chat from 'Old Chat' page, messages from old chat session will be displayed for user to see. User can also press 'Search' button at top right corner to search for old messages.

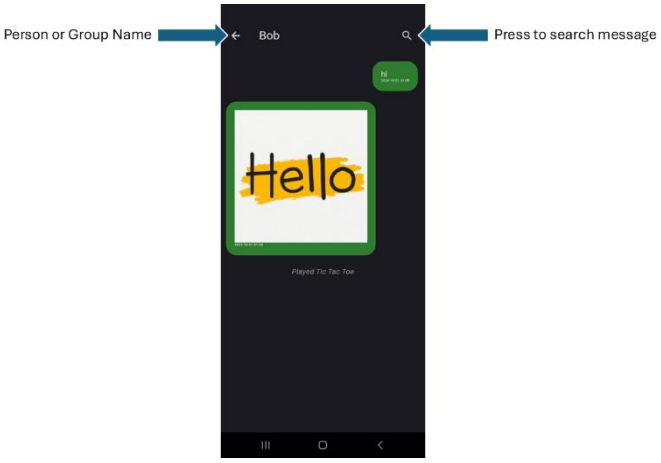


Figure 17: 'Old Chat Session' Page

5. ChatAlone Implementation

5.1. Features Implemented

Feature	Description
Person Chat	ChatAlone enables two users to communicate anonymously via Bluetooth without needing an internet connection. This allows users to send and receive messages in real-time, with messages delivered instantly over a secure Bluetooth connection.
Group Chat	In addition to person chat, ChatAlone provides a group chat feature that allows multiple users to communicate simultaneously over a Bluetooth network. Users can create a group chat, and nearby devices can join. This feature is particularly useful in settings such as conferences or social gatherings.
Message Formatting	ChatAlone allows users to format their messages using various text styles. Users can change formats by using a toolbar located below an app bar. It supports bold, italic, underline, strikethrough, and highlighted text. Additionally, URLs are automatically detected and rendered as links, allowing users to easily navigate to websites directly just by clicking on them.
Copy Messages	ChatAlone allows users to copy messages to their clipboard. This is done by long-pressing on a message to open a context menu with options to copy. Through copying a message, users can easily share its content in other apps or save it for later use.
React to Received Messages	ChatAlone supports reactions using emojis, allowing users to express their emotions or responses quickly without typing out a full message. By long-pressing on a message, users can choose from a set of predefined emojis (such as thumbs up, heart, and laughter) to react to a message. These reactions are then displayed below its message.
Reply to Received Messages	ChatAlone allows users to reply to received messages. Users select a message they want to reply to by long pressing that message. This feature is particularly useful in fast-paced conversations, where it may be necessary to reference an earlier message to provide context.
Edit Sent Messages	ChatAlone allows users to update their messages after they have been sent. Editing messages allow users to modify their content, with an edited message labeled as “edited”. This feature is particularly useful when sender makes minor grammatical or spelling errors in their messages.
Delete Sent Messages	ChatAlone allows users to delete their messages after they have been sent. Deleting messages removes them from both senders’ and recipients’ devices. This feature is particularly useful when sender sends messages that they regret after it is sent.
Self-Destructing Messages	To enhance privacy, ChatAlone allows users to send self-destructing messages. This is done by long pressing ‘Send’ button. Users can choose to make it self-destruct after a set period (e.g., 10 seconds). Once time is up, that self-destructing message is automatically deleted. This feature is ideal for sharing sensitive information that should not be stored permanently.

Send Images	ChatAlone allows users to send images with one another for both person chat and group chat. This makes conversations more expressive and interactive. It uses Bluetooth technology to transmit images directly between devices, ensuring that even in offline environments, users can still engage in rich multimedia exchanges.
Games	To make ChatAlone more engaging, users can enjoy following turn-based games such as Tic-Tac-Toe, Connect 4, and Othello without needing to be physically present or connected to internet: • Tic-Tac-Toe: Players take turns placing Xs or Os on a 3x3 grid, aiming to be first to align three symbols in row, column, or diagonal. • Connect 4: Players alternate dropping colored discs into vertical grid, aiming to connect four of their discs in row, column, or diagonal. • Othello: Players take turns placing discs on 8x8 board, with goal of flipping as many of their opponent's pieces as possible.
Offensive Content Filter	To maintain safe and respectful communication environment, ChatAlone has offensive content filter that detects content with profanity or nudity. Besides detecting profanity in text messages, ChatAlone can also scan images to see if it has profanity or nudity in it. If users attempt to send those content, it will not be sent. This discourages negative and disrespectful communication.
View Old Chats	After users are done with chatting with one another, ChatAlone saves all messages in that chat session locally on their phones. Users can revisit their own conversations anytime they want. Users can also delete their own chat session if they choose to do so.
Search Messages	ChatAlone offers powerful message search feature that allows users to efficiently locate specific messages within their conversation history in both active and old chats. When users enter search term, ChatAlone instantly scans and highlights all messages containing that term. To further streamline navigation, ChatAlone provides directional arrows that enable users to quickly jump between different instances of that search term within their conversation history.

Table 22: Features Implemented

5.2. Packages Used

Package	Version	Description
flutter_localizations	SDK: flutter	Enables localization and internationalization for Flutter applications [15].
flutter_test	SDK: flutter	Provides testing tools and libraries for Flutter applications [16].
device_info_plus	10.1.2	Retrieves device information, such as model and operating system details [17].
flutter_nearby_connections	1.1.1	Enables nearby device connections for Bluetooth functionality [18].
flutter_nude_detector	0.0.4+1	Detects nudity in images, useful for content moderation [19].
flutter_styled_toast	2.0.0	Provides customizable toast messages for temporary pop-up notifications [20].
gallery_saver	2.3.2	Saves images and videos to device's gallery or media library [21].
google_mlkit_text_recognition	0.10.0	Uses Google’s ML Kit for recognizing text within images [22].
image_picker	0.8.7	Allows users to pick images from gallery or capture new ones with camera [23].
intl	0.18.1	Provides internationalization support, including date formatting and number localization [24].
intl_utils	2.0.0	Helps manage localization files, especially for generating translations [25].
path	1.8.0	Utilities for handling and manipulating file paths [26].
permission_handler	11.3.1	Manages permissions for accessing device features, such as camera or Bluetooth [27].
profanity_filter	2.0.0	Detects and filters profanity in text input [28].
shared_preferences	2.0.13	Provides persistent storage for simple data in key-value pairs [29].
sqflite	2.0.0+4	SQLite database for local data storage [30].
url_launcher	6.3.0	Opens URLs in device's default browser or other supported applications [31].
uuid	4.5.0	Generates unique identifiers (UUIDs) for various purposes within applications [32].

Table 23: Packages Used

5.3. Bluetooth Connection

Bluetooth connection is done using Flutter Nearby Connections. When user clicks ‘Connect’ button in find ‘Nearby Person’ or ‘Nearby Group’ page, Flutter Nearby Connections activates advertise and listening functions. When device is advertising, it transmits packet across multiple channels, and devices listening on those channels will receive packet, leading to device discovery. Upon successful connection, its status will be updated and displayed on screen. Users can then chat with each other in ‘Person Chat Session’ or ‘Group Chat Session’ page.

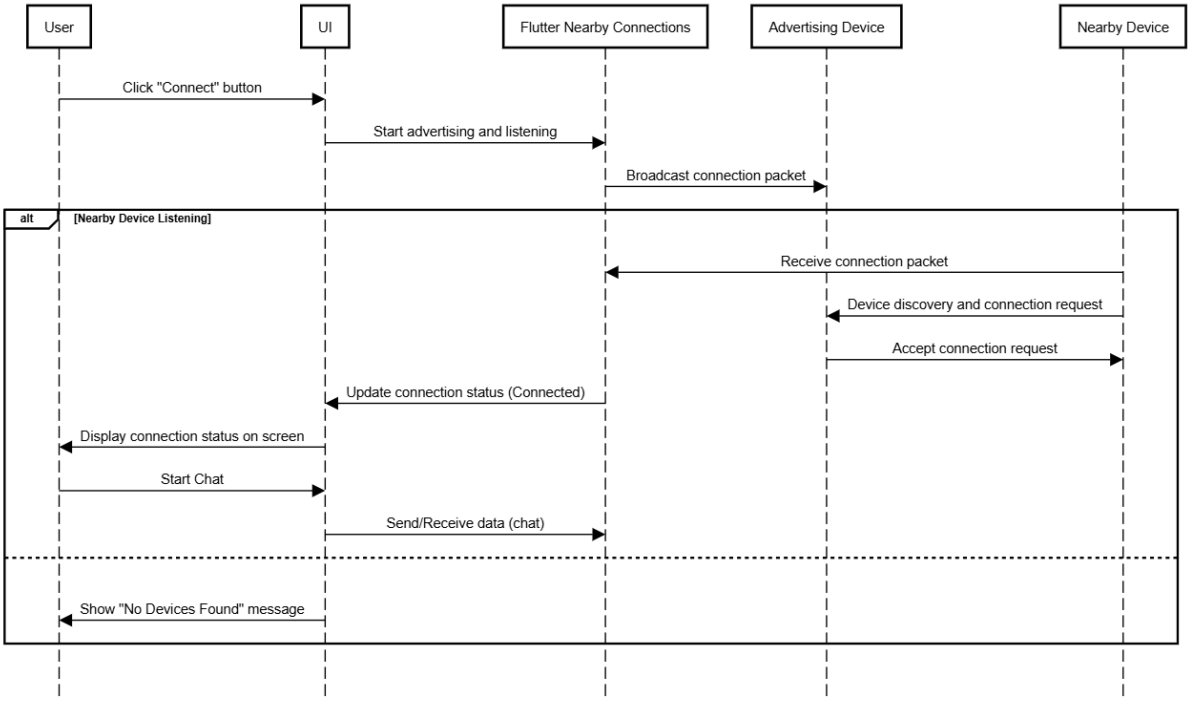


Figure 18: ‘Bluetooth Connection’ Sequence Diagram

```
void init() async {
  nearbyService = NearbyService();
  await nearbyService.init(
    serviceType: 'mpconn',
    deviceName: 'group:${widget.groupName}',
    strategy: Strategy.P2P_CLUSTER,
    callback: (isRunning) async {
      if (isRunning) {
        await nearbyService.stopAdvertisingPeer();
        await nearbyService.stopBrowsingForPeers();
        nearbyService.startAdvertisingPeer();
        nearbyService.startBrowsingForPeers();
      }
    },
  );
  subscription =
      nearbyService.stateChangedSubscription(callback: (devicesList) {
    for (var element in devicesList) {
      if (Platform.isAndroid) {
        if (element.state == SessionState.connected) {
          nearbyService.stopBrowsingForPeers();
        }
      }
    }
    setState(() {
      devices.clear();
      devices.addAll(devicesList
          .where((d) => d.deviceName.contains('joining:'))
          .map((d) {
        d.deviceName = d.deviceName.replaceAll('joining:', '');
        return d;
      }).toList());
      connectedDevices.clear();
      connectedDevices.addAll(devicesList
          .where((d) => d.state == SessionState.connected)
          .toList());
    });
  });
}
```

Figure 19: Code for Finding Nearby Devices

5.4. Send Message

To send message, user types their desired text into input field on chat screen and presses ‘Send’ button. ChatAlone first checks message for any profanities using its offensive content filter. If message contains profanity, ChatAlone displays warning alert to user and message is not sent. Else, message is transmitted to connected device over Bluetooth. Upon successful transmission, message is displayed in conversation area for both sender and recipient.

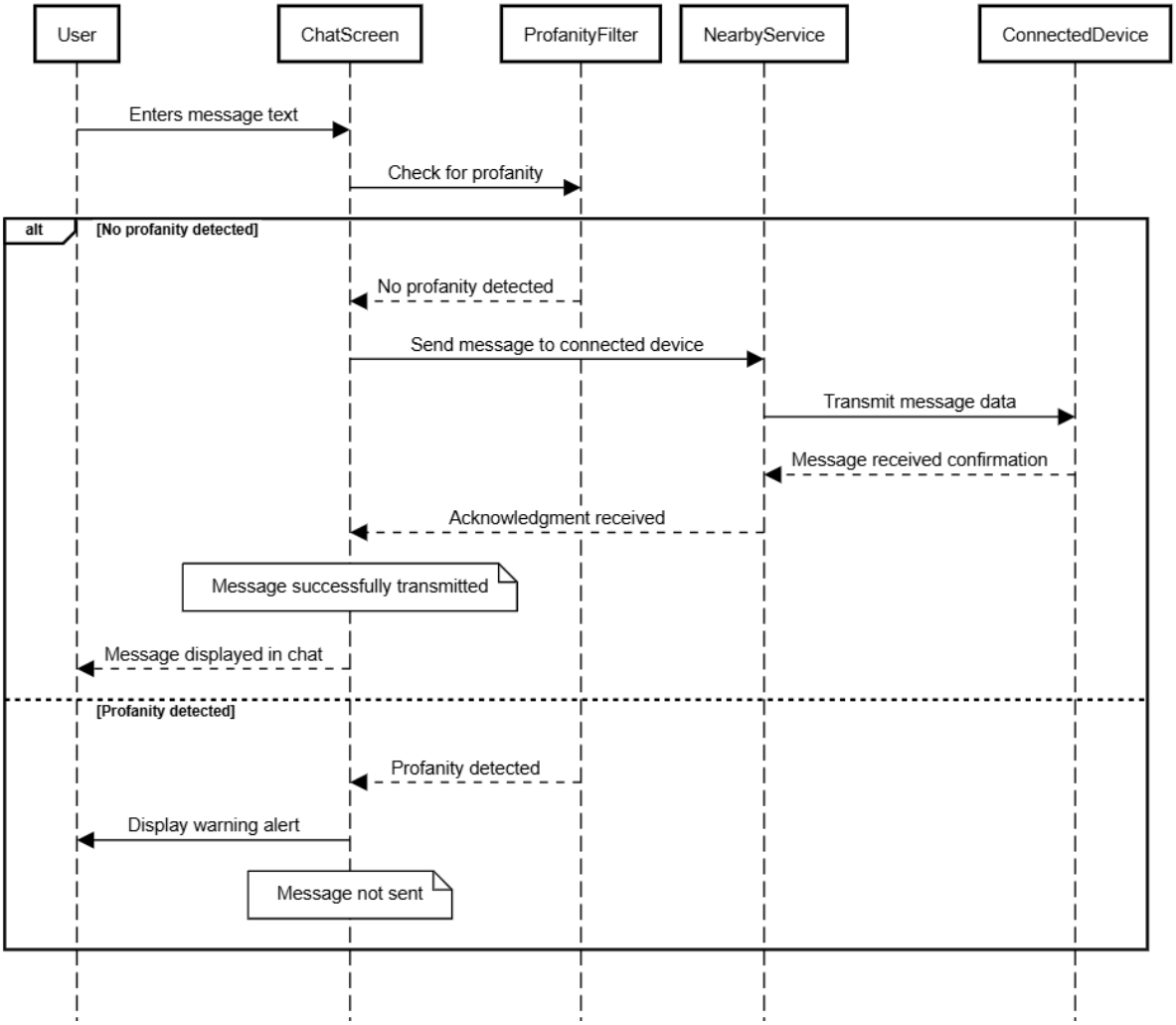


Figure 20: 'Send Message' Sequence Diagram

```
void sendMessage(String content,
    {Uint8List? imageData, bool autoDelete = false, int? timeInSeconds}) {
  String message = imageData != null ? "image" : "message";
  String finalMessage = "$message|$content${replyToMessage != null
          ? "|reply|${replyToMessage!.messageContent}"
          : ""}${autoDelete ? "|auto|$timeInSeconds" : ""}";
  if (imageData != null) {
    String base64Image = base64Encode(imageData);
    finalMessage += "|$base64Image";
  }
  var obj = ChatMessage(
    chatId: chatId,
    personName: widget.connectedDevice.deviceName,
    messageContent: content,
    messageType: "sender",
    messageFormat: message,
    autoDelete: autoDelete,
    timeRemaining: autoDelete ? timeInSeconds ?? 10 : null,
    timestamp: DateTime.now().toUtc().add(const Duration(hours: 8)),
    replyTo: replyToMessage,
    imageData: imageData,
  ); // ChatMessage
  addMessageToList(obj);
  widget.nearbyService
      .sendMessage(widget.connectedDevice.deviceId, finalMessage);
  myController.clear();
  setState(() {
    replyToMessage = null;
  });
}
```

Figure 21: Code for Sending Message

5.5. Send Image

To send image, ChatAlone uses Flutter's Image Picker plugin. Due to Nearby Connections' file size limitation, images must be under 20KB. Selected image undergoes Offensive Content Filter to detect for profanity and nudity. Profanity is checked using Google's ML Kit Text Recognition, and nudity is checked using Flutter Nude Detector. It utilizes Google's ML Kit Image Labeling with TensorFlow Lite model to detect nudity. If either is found, ChatAlone displays warning alert to user and image is not sent. Otherwise, image is base64 encoded as string and sent to other users. Other users will then decode that string and be displayed to them.

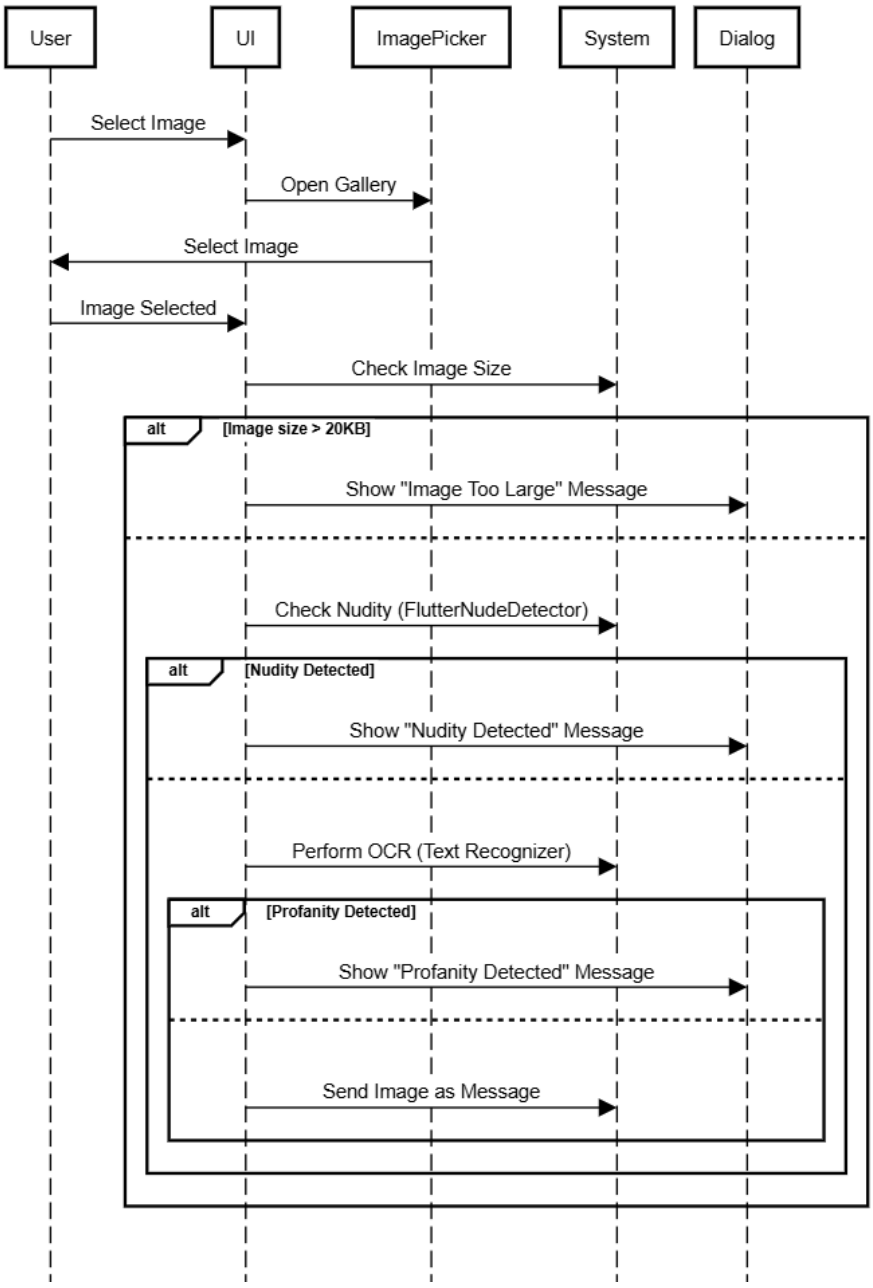


Figure 22: 'Send Image' Sequence Diagram

```
void pickImage() async {
    final ImagePicker picker = ImagePicker();
    final XFile? image = await picker.pickImage(source: ImageSource.gallery);
    if (image != null) {
      Uint8List imageData = await image.readAsBytes();
      int imageSize = imageData.lengthInBytes;
      if (imageSize >= 20 * 1024) {
        showDialog(···
      } else {
        final containsNudity =
            await FlutterNudeDetector.detect(path: image.path);
        if (containsNudity) {
          showDialog(···
        } else {
          final inputImage = InputImage.fromFilePath(image.path);
          final textRecognizer = TextRecognizer();
          try {
            final RecognizedText recognizedText =
                await textRecognizer.processImage(inputImage);
            final String ocrText = recognizedText.text;
            if (ocrText.isNotEmpty && filter.hasProfanity(ocrText)) {
              showDialog(···
            } else {
              sendMessage("Image", imageData: imageData);
            }
          } catch (e) {
            ScaffoldMessenger.of(context).showSnackBar(···
          } finally {
            textRecognizer.close();
          }
        }
      }
    } else {
      print("No image was selected.");
    }
}
```

Figure 23: Code for Sending Image

5.6. Play Game

To play game, user first opens game menu in person chat to display available games like Tic Tac Toe, Connect Four, and Othello. After selecting, ChatAlone sends game initiation request to other connected device through Bluetooth. Other connected device can either accept or decline request. If game request is accepted, ChatAlone opens game for both users to start playing in real time. If game request is declined, ChatAlone notifies user that game has been declined.

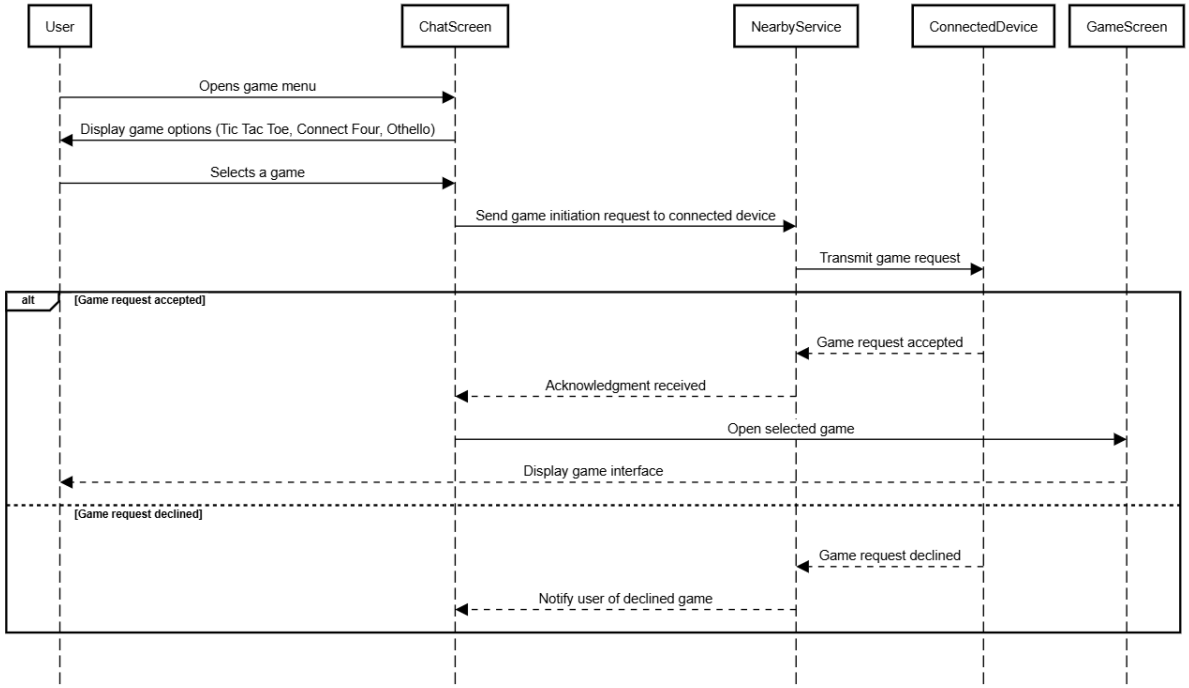


Figure 24: 'Play Game' Sequence Diagram

```
void startTicTacToe() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text(S.of(context).startTicTacToe),
        content: const Text("Are you sure you want to start Tic Tac Toe?"),
        actions: <Widget>[
          TextButton(
            child: Text(S.of(context).cancel),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ), // TextButton
          TextButton(
            child: Text(S.of(context).start),
            onPressed: () {
              sendSystemMessage(S.of(context).playedTicTacToe);
              Navigator.of(context).pop();
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(
                  builder: (context) => TicTacToePage(
                    nearbyService: widget.nearbyService,
                    connectedDevice: widget.connectedDevice,
                    currentDevicePlayer: "X",
                    myData: widget.myData,
                    chatState: messages.isNotEmpty ? messages : null,
                  ), // TicTacToePage
                ), // MaterialPageRoute
              );
              widget.nearbyService.sendMessage(
                  widget.connectedDevice.deviceId, "tictactoe|start|0");
            },
          ), // TextButton
        ], // <Widget>[]
      ); // AlertDialog
    },
  );
}
```

Figure 25: Code for Starting Tic-Tac-Toe

6. ChatAlone Testing

6.1. Username Validation

Requirements: Between 1 to 25 characters, no special characters, no profanity

Preconditions: Application must be at 'Settings' page

Test Cases:

ID	Description	Steps
UV1	Minimum length (1 character), no special characters, no profanity	1. Input 'a' into username input field 2. Press 'Done'
UV2	Maximum length (25 characters), no special characters, no profanity	1. Input 'abcdefghijklmnopqrstuvwxy' into username input field 2. Press 'Done'
UV3	Slightly below minimum length (0 character)	1. Press 'Done' without inputting anything
UV4	Slightly above maximum length (26 characters)	1. Input 'abcdefghijklmnopqrstuvwxyz' into username input field 2. Press 'Done'
UV5	Contains special character	1. Input '@' into username input field 2. Press 'Done'
UV6	Contains profanity	1. Input 'shit' into username input field 2. Press 'Done'

Table 24: 'Username Validation' Test Cases

Results:

ID	Expected Outcome	Result
UV1	Application transits to 'Home' page	Pass
UV2	Application transits to 'Home' page	Pass
UV3	Application does not transit to 'Home' page and shows error message: "Please enter username"	Pass
UV4	Application does not transit to 'Home' page and shows error message: "Username should be 25 characters or less"	Pass
UV5	Application does not transit to 'Home page' and shows error message: "Username should not contain special characters"	Pass
UV6	Application does not transit to 'Home page' and shows error message: "Username should not contain profanity"	Pass

Table 25: 'Username Validation' Test Results

6.2. Group Name Validation

Requirements: Between 1 to 25 characters, no special characters, no profanity

Preconditions: Application must display ‘Create Group Chat’ dialog

Test Cases:

ID	Description	Steps
GN1	Minimum length (1 character), no special characters, no profanity	1. Input ‘a’ into group name input field 2. Press ‘Done’
GN2	Maximum length (25 characters), no special characters, no profanity	1. Input ‘abcdefghijklmnopqrstuvwxy’ into group name input field 2. Press ‘Done’
GN3	Slightly below minimum length (0 character)	1. Press ‘Done’ without inputting anything
GN4	Slightly above maximum length (26 characters)	1. Input ‘abcdefghijklmnopqrstuvwxyz’ into group name input field 2. Press ‘Done’
GN5	Contains special character	1. Input ‘@’ into username input field 2. Press ‘Done’
GN6	Contains profanity	1. Input ‘shit’ into username input field 2. Press ‘Done’

Table 26: 'Group Name Validation' Test Cases

Results:

ID	Expected Outcome	Result
GN1	Application transits to ‘Nearby Person’ page	Pass
GN2	Application transits to ‘Nearby Person’ page	Pass
GN3	Application does not transit to ‘Nearby Person’ page and shows error message: “Please enter username”	Pass
GN4	Application does not transit to ‘Nearby Person’ page and shows error message: “Group name should be 25 characters or less”	Pass
GN5	Application does not transit to ‘Nearby Person’ page and shows error message: “Group name should not contain special characters”	Pass
GN6	Application does not transit to ‘Nearby Person’ page and shows error message: “Group name should not contain profanity”	Pass

Table 27: 'Group Name Validation' Test Results

6.3. Bluetooth Connection

Requirements: Devices must be able to transmit messages via Bluetooth

Preconditions: 2 Android phones with Bluetooth turned on, application must be in nearby person page

Test Cases:

ID	Description	Steps
BC1	Device discovery	1. Bring 2 devices near each other 2. Observe if device A's username appears on device B, vice versa
BC2	Successful pairing	1. Press 'Message' button to go into 'Person Chat Session' page on both devices 2. Input 'a' into message input field on device A 3. Press 'Send'
BC3	Connection stability	1. Leave devices alone for 15 minutes 2. After 15 minutes, input 'a' into message input field on device A 3. Press 'Send'

Table 28: 'Bluetooth Connection' Test Cases

Results:

ID	Expected Outcome	Result
BC1	Device A's username appears on device B, vice versa	Pass
BC2	Device B receives 'a'	Pass
BC3	Device B receives 'a'	Pass

Table 29: 'Bluetooth Connection' Test Results

6.4. Message Validation

Requirements: Minimum 1 character, no profanity

Preconditions: Bluetooth connection works between 2 devices, application must be in 'Person Chat Session' page

Test Cases:

ID	Description	Steps
M1	Minimum length (1 character), no special characters, no profanity	1. Input 'a' into username input field 2. Press 'Send'
M2	Slightly below minimum length (0 character)	1. Press 'Send' without inputting anything
M3	Contains profanity	1. Input 'shit' into message input field 2. Press 'Send'

Table 30: 'Message Validation' Test Cases

Results:

ID	Expected Outcome	Result
M1	Application sends message and other device receive message	Pass
M2	Application does not send message and shows error message: "Please enter your message"	Pass
M3	Application does not send message and shows error message: "Please do not send message with profanity"	Pass

Table 31: 'Message Validation' Test Results

6.5. Image Validation

Requirements: No profanity, no nudity

Preconditions: Bluetooth connection works between 2 devices, application must display image gallery

Test Cases:

ID	Description	Steps
I1	Image without profanity or nudity, below 20KB in size	1. Select an image without profanity or nudity, below 20KB in size 2. Press 'Send'
I2	Image without profanity or nudity, 20KB in size	1. Select an image without profanity or nudity, 20KB in size 2. Press 'Send'
I3	Image contains profanity	1. Select an image that contains profanity and is below 20KB in size 2. Press 'Send'
I4	Image contains nudity	1. Select an image that contains nudity and is below 20KB in size 2. Press 'Send'

Table 32: 'Image Validation' Test Cases

Results:

ID	Expected Outcome	Result
I1	Application sends image	Pass
I2	Application does not send message and shows error message: "Image should be less than 20KB"	Pass
I3	Application does not send message and shows error message: "Please do not send image with profanity"	Pass
I4	Application does not send message and shows error message: "Please do not send image with nudity"	Pass

Table 33: 'Image Validation' Test Results

7. Conclusion

7.1. Summary

ChatAlone is an anonymous and feature-rich messaging application designed to facilitate seamless peer-to-peer communication over Bluetooth. With a wide array of functionalities, including message formatting, editing, reactions, replies, sending images, sending self-destructing messages, games, and a built-in offensive content filter, ChatAlone is well-suited for various use cases, from casual conversations to more formal communications. Its multi-language interface, combined with robust Bluetooth connectivity and cross-platform support, makes it an ideal choice for anyone looking to use an offline chat application for privacy reasons.

7.2. Limitations & Future Works

ChatAlone has some limitations that can be resolved in future works:

- **Detect non-nude pornographic content**: ChatAlone currently cannot detect non-nude pornographic content. Advanced machine learning can be used to detect non-nude pornographic content, ensuring that inappropriate materials are flagged and prevented from being shared, even if they do not contain explicit nudity.
- **Transfer files larger than 20KB**: ChatAlone currently cannot transfer files larger than 20KB. Image transfer feature can be upgraded into file transfer feature that can share richer content like larger images, videos and documents without size limitations.
- **More Games**: ChatAlone currently has limited games for users to play. To make ChatAlone more engaging and versatile, more games such as chess can be introduced within person chat to expand on current game options (Tic-Tac-Toe, Connect 4, Othello).

8. References

[1] Statista, "Most popular global mobile messenger apps," Statista, 2023. [Online]. Available: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps. [Accessed: 10-Oct-2024].

[2] J.-Y. Yang and C. Choi, "Enhancing interactiveness of problem-based programming course using anonymous chatting service," in Proc. 16th International Conference on Information Technology Based Higher Education and Training (ITHET), Ohrid, Macedonia, 2017. [Online]. Available: https://doi.org/10.1109/ITHET.2017.8067828. [Accessed: 10-Oct-2024].

[3] K. Ramezanpour, J. Jagannath, and A. Jagannath, "Security and privacy vulnerabilities of 5G/6G and WiFi 6: Survey and research directions from a coexistence perspective," Computer Networks, vol. 221, 109515, 2023. [Online]. Available: https://doi.org/10.1016/j.comnet.2022.109515. [Accessed: 10-Oct-2024].

[4] Sandeep, "Bluetooth vs Wi-Fi vs Cellular – Difference and Comparison," January 9, 2023. [Online]. Available: https://differbtw.com/difference-between-bluetooth-wi-fi-and-cellular. [Accessed: 10-Oct-2024].

[5] Statista, "Mobile operating system market share in Singapore 2023," Statista, 2023. [Online]. Available: https://www.statista.com/statistics/1349426/singapore-mobile-os-market-share. [Accessed: 10-Oct-2024].

[6] Briar Project, "Briar: Secure messaging, anywhere," Briar Project, 2023. [Online]. Available: https://briarproject.org. [Accessed: 10-Oct-2024].

[7] Bridgefy, "Bridgefy: Offline messaging app," Bridgefy, 2023. [Online]. Available: https://bridgefy.me. [Accessed: 10-Oct-2024].

[8] L. Hern, "FireChat: the messaging app powering the Hong Kong protests," The Guardian, Sep. 29, 2014. [Online]. Available: https://www.theguardian.com/world/2014/sep/29/firechat-messaging-app-powering-hong-kong-protests. [Accessed: 10-Oct-2024].

[9] J. Chang, "Understanding Cultural Norms to Design Inclusive User Experiences," UXmatters, Sep. 2024. [Online]. Available: https://www.uxmatters.com/mt/archives/2024/09/understanding-cultural-norms-to-design-inclusive-user-experiences.php. [Accessed: 10-Oct-2024].

[10] Y. Zhang, S. Zhao, W. Zhao, and W. Lu, "Age differences in perceptions of online community participation among non-users: An extension of the Technology Acceptance Model," Frontiers in Psychology, vol. 12, art. 698799, 2021. [Online]. Available: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2021.698799/full. [Accessed: 10-Oct-2024].

[11] A. G. Zimmerman and G. J. Ybarra, "Online aggression: influences of anonymity and social modeling," Psychology of Popular Media Culture, vol. 5, no. 2, pp. 181–193, 2016. [Online]. Available: https://doi.org/10.1037/ppm0000038. [Accessed: 10-Oct-2024].

[12] "Three-Tier Architecture," IBM, 2024. [Online]. Available: https://www.ibm.com/topics/three-tier-architecture. [Accessed: 10-Oct-2024].

[13] GeeksforGeeks, "Third Normal Form (3NF)," GeeksforGeeks, 2023. [Online]. Available: https://www.geeksforgeeks.org/third-normal-form-3nf/. [Accessed: 10-Oct-2024].

[14] Flutter, "Internationalization in Flutter," Flutter Documentation, 2023. [Online]. Available: https://docs.flutter.dev/ui/accessibility-and-internationalization/internationalization. [Accessed: 10-Oct-2024].

[15] "Flutter Localizations," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/flutter_localizations. [Accessed: 10-Oct-2024].

[16] "Flutter Test," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/flutter_test. [Accessed: 10-Oct-2024].

[17] "Device Info Plus," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/device_info_plus. [Accessed: 10-Oct-2024].

[18] "Flutter Nearby Connections," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/flutter_nearby_connections. [Accessed: 10-Oct-2024].

[19] "Flutter Nude Detector," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/flutter_nude_detector. [Accessed: 10-Oct-2024].

[20] "Flutter Styled Toast," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/flutter_styled_toast. [Accessed: 10-Oct-2024].

[21] "Gallery Saver," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/gallery_saver. [Accessed: 10-Oct-2024].

[22] "Google ML Kit Text Recognition," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/google_mlkit_text_recognition. [Accessed: 10-Oct-2024].

[23] "Image Picker," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/image_picker. [Accessed: 10-Oct-2024].

[24] "Intl," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/intl. [Accessed: 10-Oct-2024].

[25] "Intl Utils," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/intl_utils. [Accessed: 10-Oct-2024].

[26] "Path," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/path. [Accessed: 10-Oct-2024].

[27] "Permission Handler," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/permission_handler. [Accessed: 10-Oct-2024].

[28] "Profanity Filter," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/profanity_filter. [Accessed: 10-Oct-2024].

[29] "Shared Preferences," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/shared_preferences. [Accessed: 10-Oct-2024].

[30] "Sqflite," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/sqflite. [Accessed: 10-Oct-2024].

[31] "URL Launcher," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/url_launcher. [Accessed: 10-Oct-2024].

[32] "UUID," Pub.dev, 2023. [Online]. Available: https://pub.dev/packages/uuid. [Accessed: 10-Oct-2024].