

BRIDGE CHAT

UNIFIED MESSAGING ARCHITECTURE

THE TEAM

ANDREA A. VENTI FUENTES

BS in Computer Science

JUSTIN BONNER

BS in Computer Science, BS Data Science & AI, BS Math

SOPHIA MANTEGARI

BS in Computer Science, BS Data Science & AI

TABLE OF CONTENTS

1. What is Bridge Chat & Why it matters	3, 4
2. System architecture	5
3. Design Rational.....	6
4. Function Flow Diagram & Explanation	7, 8, 9
5. Class Design Concepts.....	10
6. Class Diagram	11
7. State Diagram.....	12
8. Nonfunctional Diagram.....	13
9. Why Libretranslate API.....	14
10. Demo.....	15
11. Summary.....	16

WHAT IS BIDGE CHAT?

UNIFIED MESSAGING PLATFORM

- Consolidates iMessage, WhatsApp, Discord, and GroupMe
- Built for cross-platform users who want one seamless inbox

KEY FEATURES

- Real-time multilingual translation
- End-to-end encryption
- Chat categorization (Work, Personal, Academic)
- Cloud-native architecture (Firebase + Flutter)

WHY BRIDGE CHAT?

THE PROBLEM

- Communication is fragmented across apps
- Language barriers hinder collaboration
- Users waste time switching between interfaces

OUR SOLUTION

- Unified interface
- Instant translations
- One secure place for all messaging

BRIDGE CHAT SYSTEM ARCHITECTURE

SYSTEM ACTORS

- User – initiates login, sends messages, receives translations
- Firebase Auth – handles secure authentication
- Firestore – stores users, messages, conversations
- Cloud Functions – runs backend logic when messages are sent
- LibreTranslate – performs translation using external API

SYSTEM OVERVIEW: DESIGN RATIONALE

01

ARCHITECTURAL STYLE

- Client–Server: Flutter frontend ↔ Firebase backend
- Event-Driven: Cloud Functions respond to database events

02

DESIGN PATTERNS

- Facade: Wraps LibreTranslate API behind translation handler
- Observer: Firestore syncs real-time updates to all clients
- MVVM/MVC: UI, services, and data models are clearly separated

03

FRAMEWORKS

- Flutter for UI
- Firebase (Auth, Firestore, Functions) for backend
- LibreTranslate for multilingual translation

BRIDGE CHAT FUNCTION FLOW

LOGIN

MESSAGE

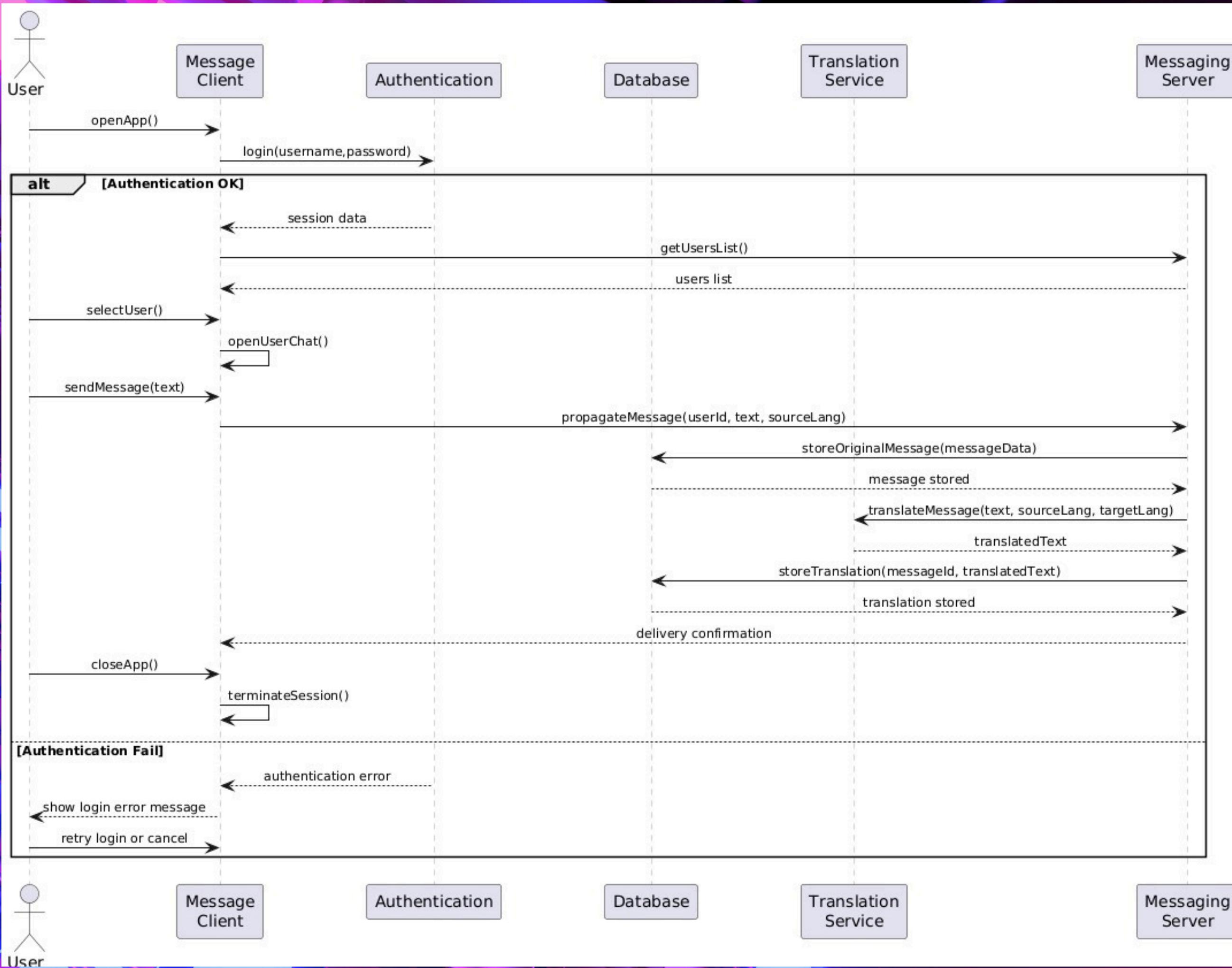
TRANSLATION

LOGOUT

- Users authenticate via Firebase
- Messages are sent and stored in Firestore
- Cloud Function triggers translation
- Translated message is stored
- Session closes cleanly on exit

BRIDGE CHAT: FUNCTION FLOW

8



BRIDGE CHAT FUNCTION FLOW

REACTIVE

- Credentials sent to Firebase Auth
- Successful login retrieves user list
- Failed login prompts retry

SESSION TERMINATION

- User exits → termination request
→ session ends cleanly

MESSAGING & TRANSLATION

- Message + metadata sent to server
- Message stored in Firestore
- Trigger → LibreTranslate → translated message returned
- Both versions saved and sent back to client

BRIDGE CHAT CLASS DESIGN CONCEPTS

USER:

**id, email, username,
preferred language**

MESSAGE:

**sender, text,
timestamp, translation**

CONVERSATION:
**participants, group
name, messages**

CHATSERVICE:
**sendMessage(),
getMessages()**

AUTHSERVICE:
**signUp(), signIn(),
signOut()**

CLASS DIAGRAMS

AuthService

- auth: FirebaseAuth
- firestone

- + signUp()
- + signIn()
- + signOut()

AuthScreen

- formkey
- email, password
- username
- preferredLanguage
- isSignUp

- + submit()
- + toggleForm()

MyApp

- isLoggedIn

- + routes configuration

SettingsScreen

- currentUser
- firestore
- preferredLanguage

- + updateSettings()
- + loadUserSettings()

RecentConversationScreen

- firestone

- + createOrGetConversation()
- + sendMessage()
- + getMessage()
- + deleteConversationForUser()

GroupSettingsScreen

- firestore
- currentUser
- participants

- + updateGroupName()
- + leaveGroup()
- + loadGroupData()

UtilityClasses

- supportedLanguages

HomeScreen

- (UI components)

NewConversationScreen

- searchController
- firestore
- chatService
- selectedUsers
- isGroupChat
- previousUsernames

- + searchUsers()
- + startConversationWithUser()
- + createGroupConversation()

ChatService

- firestone

- + createOrGetConversation()
- + sendMessage()
- + getMessage()
- + deleteConversationForUser()

ChatScreen

- messageController
- scrollController
- currentUser
- msgList
- otherUserLanguage
- myLanguage

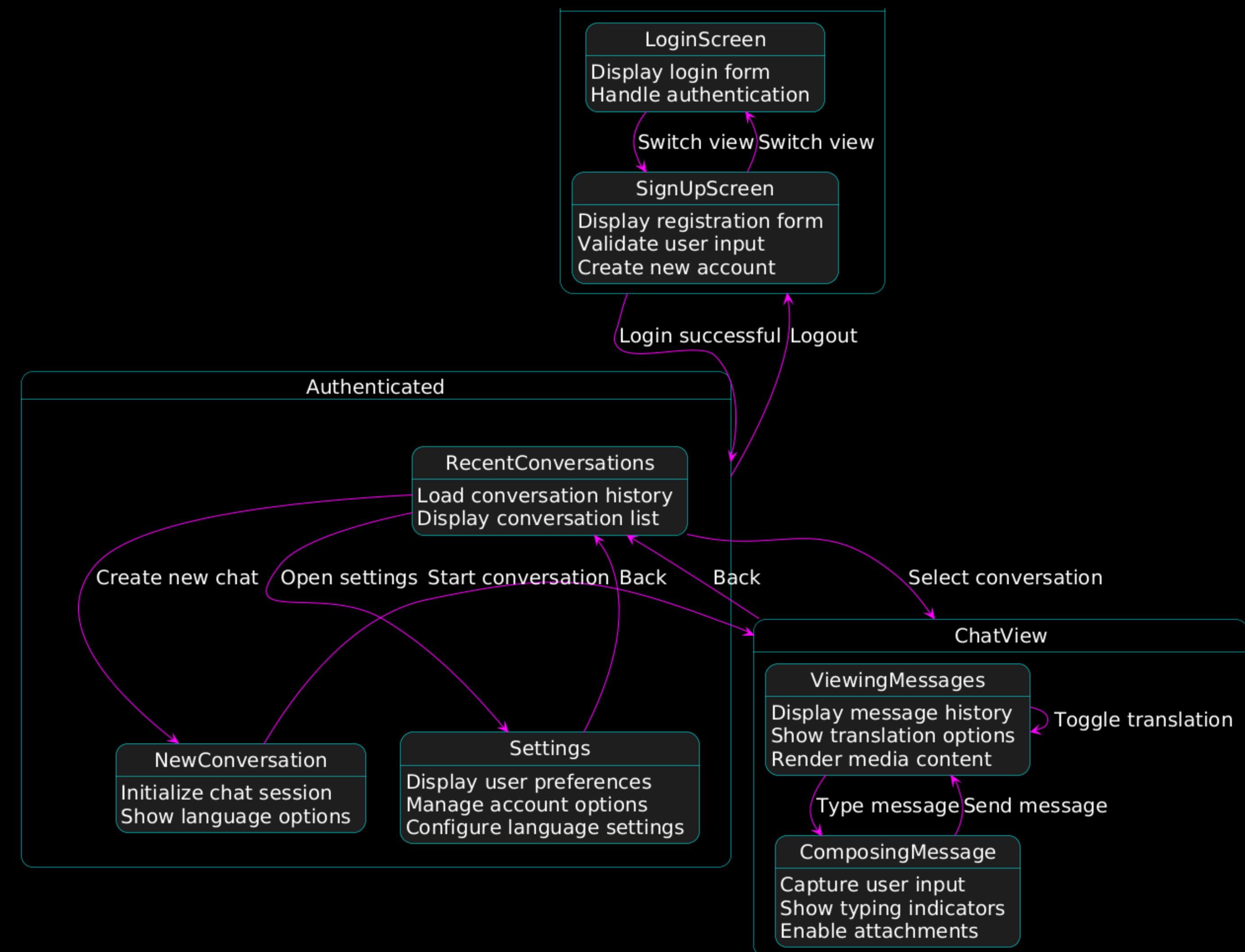
- + sendMessage()
- + translateText()
- + fetchParticipants()

MessageBubble

- senderId
- currentUserId
- originalText
- translatedText
- timestamp
- sourceLang
- targetLang

- + showOriginal()

BRIDGE CHAT: STATE DIAGRAM



BRIDGE CHAT

NON-FUNCTIONAL REQUIREMENTS

PERFORMANCE:
Firestore ensures
<200ms latency

SCALABILITY:
Firebase scales
automatically

USABILITY:
Responsive UI with
clean onboarding

SECURITY:
Firebase Auth + Firestore Rules
+ secret-managed API access

MAINTAINABILITY:
Modular services + CI with
GitHub Actions

WHY LIBRETRANSLATE?



Recent Conversations

No conversations yet.

Preferred Lan

Sign In

Welcome Back

Email: luna@gmail.com

Password:

Sign In

Don't have an account? [Sign Up](#)

SUMMARY

- One platform for all conversations
- Real-time, secure, multilingual messaging
- Built using scalable, open technologies
- Future-ready with modular components and AI integration

BRIDGE CHAT

THANK YOU