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

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

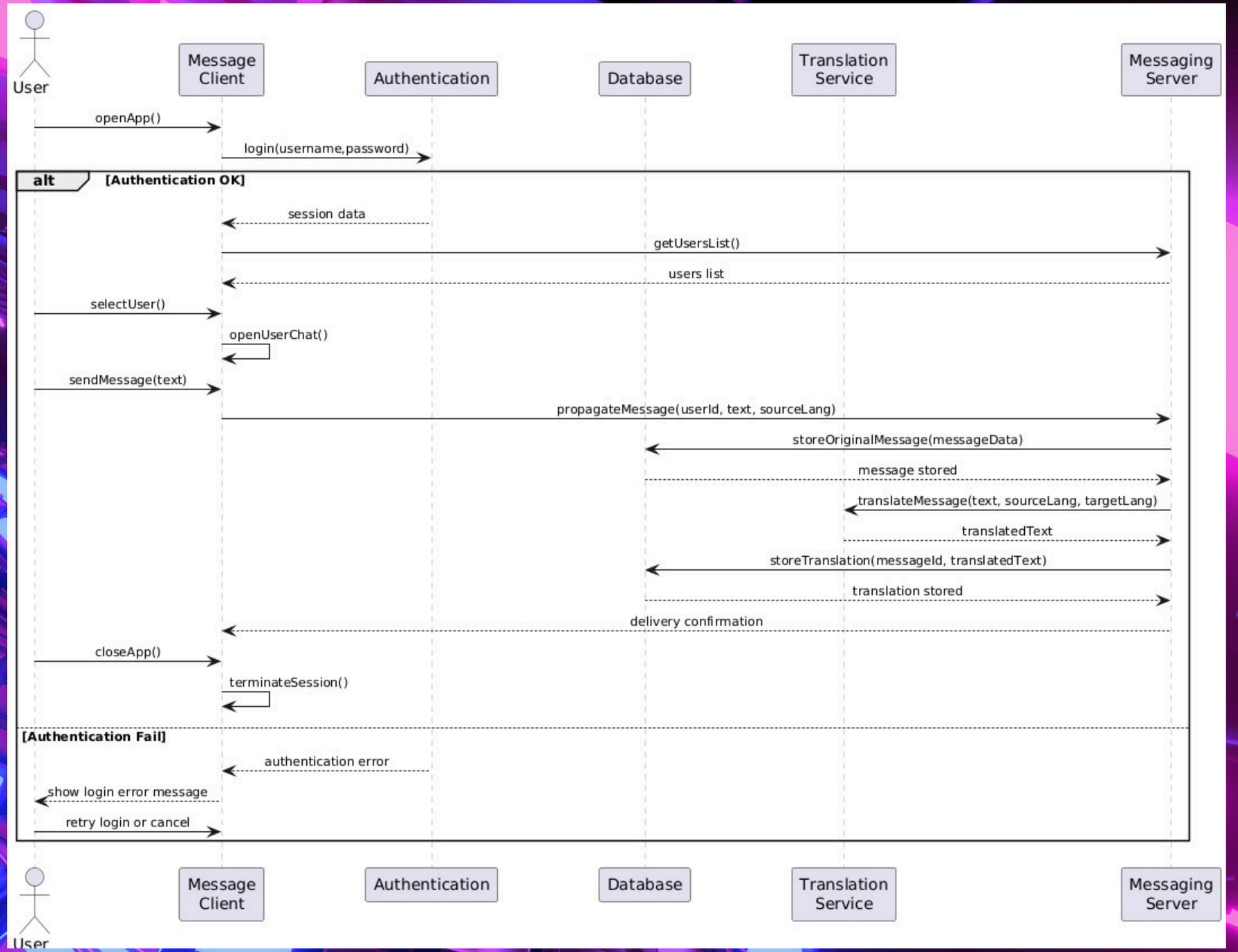
- Flutter for UI
- Firebase (Auth, Firestore, Functions) for backend
- LibreTranslate for multilingual translation

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



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

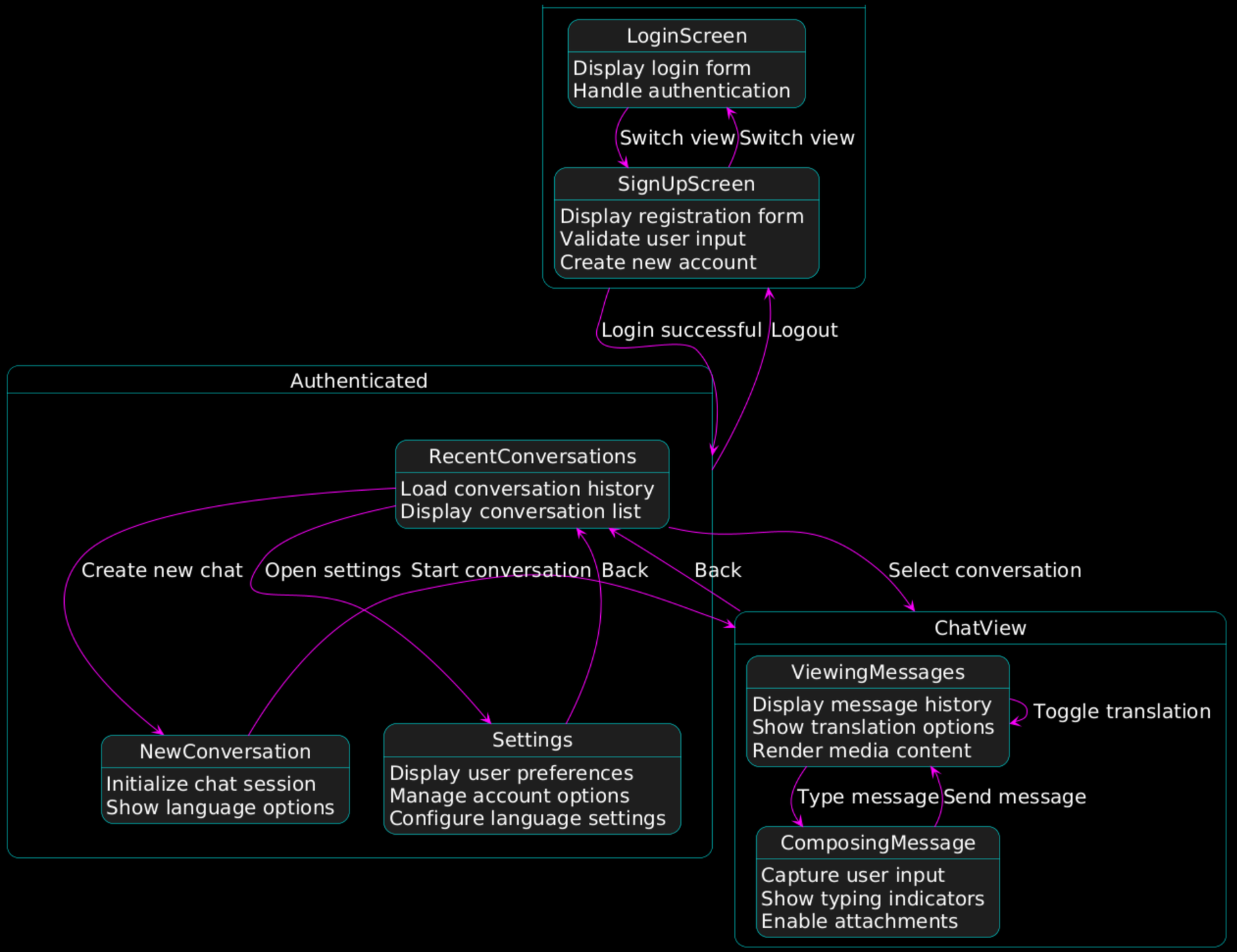
CONVERSATION:
participants, group name, messages

CHATSERVICE:
sendMessage(), getMessages()

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

CLASS DIAGRAMS

BRIDGE CHAT: STATE DIAGRAM



BRIDGE CHAT
NON-FUNCTIONAL REQUIREMENTS

PERFORMANCE:
Firestore ensures <200ms latency

SCALABILITY:
Firebase scales automatically

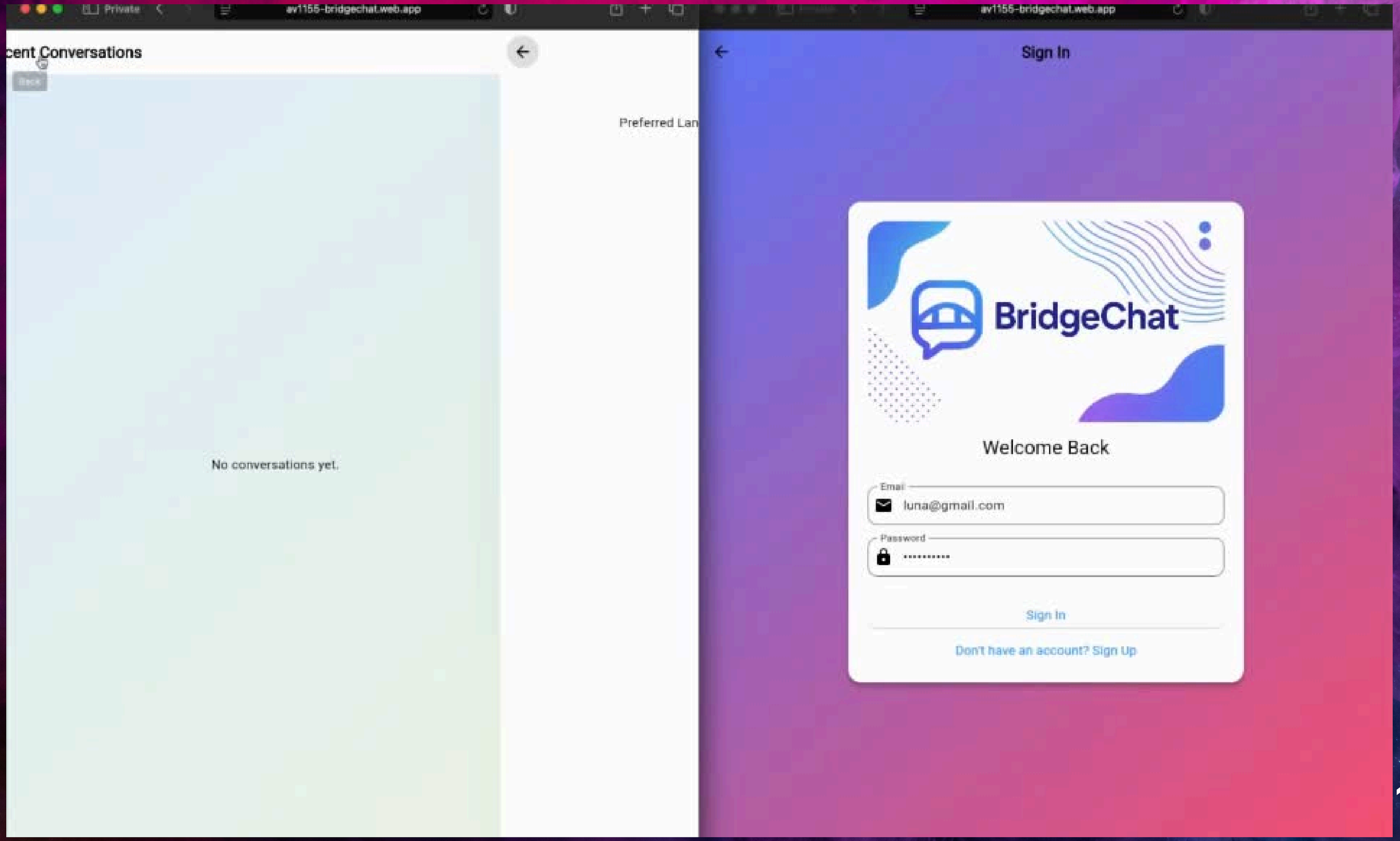
USABILITY:
Responsive UI with clean onboarding

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

MAINTAINABILITY:
Modular services + CI with GitHub Actions

WHY LIBRETRANSLATE?

VS


Private
av1155-bridgechat.web.app
cent Conversations
Back
Preferred Lan
No conversations yet.
av1155-bridgechat.web.app
Sign In
BridgeChat
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