TED UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CMPE 492 - Senior Project Final Report

Submission Date

27.12.2024

Group Members:

Bartu Özen

Elif Nazlı Böke

Gülce Ayşe Döker

Toygar Yurt

Table of Contents

1 Introduction

1.1 Purpose of the system

The purpose of this system is to create a mobile messaging application that integrates advanced natural language processing features. This application aims to facilitate seamless communication across diverse languages by using a large language model (LLM) for real-time translation. Users will be able to send messages and share files while leveraging text and voice translation capabilities. The system is designed to be robust, user-friendly, and secure, catering to both individual and group communication needs.

1.2. Design goals

1.2.1 Usability

The system is designed with the end-user in mind, ensuring an intuitive and accessible interface for diverse user demographics. Features such as user authentication, real-time translation, and group messaging are implemented to enhance user satisfaction.

1.2.2 Scalability

To handle a growing user base, the system architecture supports horizontal and vertical scaling. This includes efficient database management and server-side optimizations to ensure consistent performance under increasing loads.

1.2.3 Reliability

Reliability is paramount for a messaging application. The system incorporates mechanisms for message delivery confirmation, data redundancy, and robust error-handling to ensure uninterrupted communication.

1.2.4 Performance

The application prioritizes low-latency performance, particularly for real-time features like translation and message delivery. Efficient algorithms and optimized resource utilization are employed to maintain high responsiveness.

1.2.5 Maintainability

The architecture adheres to modular design principles, making it easy to update and maintain. Comprehensive documentation and well-structured code ensure long-term maintainability.

1.2.6 Security

Security features include end-to-end encryption for messages, secure user authentication, and stringent data privacy measures. These safeguards protect user information and maintain trust in the application.

2 Final Architecture and Design

We have our application running on the client's device and it constantly communicates with the server. On the server side we used GeminiAPI.

Server

- Translation Service: Executes text translation using API's.
- Entity Layer: Defines User, Message and Conversation entities.
- Data Layer: Manages the database configuration and relationships.
- User Management: Handles user registration and profile update.
- Messaging Management: Handles the real time messaging.

Client

- View Models: Manages the UI and connects it to the backend.
- Screens: Displays the user interface.
- Request Manager: Makes HTTP requests and error handling.
- Persistence Manager: Stores data locally.

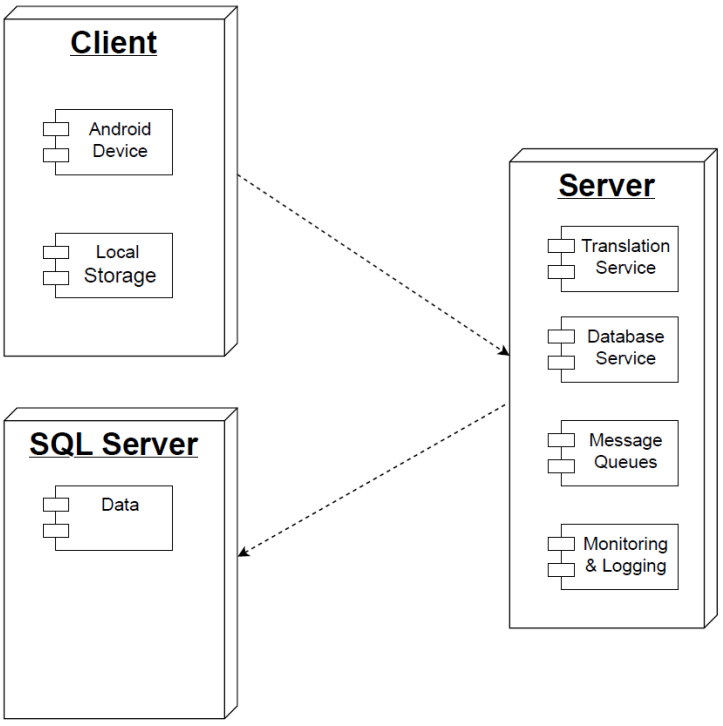

Figure 1: UML deployment diagram showing client-server communication

3 Test Cases

Test ID: 00100

Test Type/Category: Functional Unit Testing

Summary/Title/Objective: Login Page Functionality Testing

Procedure of Testing Steps:

- Navigate to the login page.
- Check the placement of all components, including text fields and buttons.
- Test the responsiveness of the "Login" button without entering credentials.
- Enter incorrect credentials and verify the error message.
- Enter valid credentials and ensure successful navigation to the home page.

Expected Results/Outcome: Each component functions as intended, and navigation occurs correctly based on input.

Priority/Severity: Major

Test ID: 00200

Test Type/Category: Functional Integration Testing

Summary/Title/Objective: User Registration and Validation

Procedure of Testing Steps:

- Open the registration page.
- Enter invalid data (e.g., incorrect phone number) and check for validation messages.
- Enter valid data and click the "Register" button.
- Verify confirmation functionality.
- Confirm that the user is successfully registered and navigated to the login page.

Expected Results/Outcome: Registration validations work correctly, and the user is successfully registered with valid inputs.

Priority/Severity: Major

Test ID: 00300
Test Type/Category: Functional Unit Testing
Summary/Title/Objective: Message Sending, Delivery, and Translation
Procedure of Testing Steps:

- Log in as a user.
- Navigate to the chat interface.
- Enter a message in the input field and click the "Send" button.
- Verify that the message appears in the sender's chat log with the correct timestamp.
- Check if the recipient receives the message in real-time.
- Verify that the message is translated into the recipient's preferred language and displayed correctly.
- Test with different language pairs (e.g., English to Spanish, French to German) to ensure accurate translation.

Expected Results/Outcome: Messages are sent, received, and translated accurately in real-time with correct timestamps.
Priority/Severity: Major

Test ID: 00400
Test Type/Category: Non-Functional Testing
Summary/Title/Objective: Scalability Testing
Procedure of Testing Steps:

- Simulate 5000 concurrent users sending messages.
- Monitor server performance and response times.
- Identify bottlenecks or failures under heavy load.

Expected Results/Outcome: The application handles 5000 users without significant performance degradation.
Priority/Severity: Critical

Test ID: 00500

Test Type/Category: Non-Functional Testing

Summary/Title/Objective: Security and Data Encryption

Procedure of Testing Steps:

- Attempt SQL injection attacks in input fields.
- Verify that messages are encrypted during transmission.
- Confirm that unauthorized users cannot access private conversations.
- Use a sandbox tool to check for potential malware in the app files.

Expected Results/Outcome: Data remains secure, and the application passes security assessments.

Priority/Severity: Critical

4 Consideration of Various Factors in Engineering Design

4.1 Public Health Considerations

The real-time messaging application promotes effective communication, enabling users to collaborate and maintain connections irrespective of language barriers. It enhances mental well-being by reducing isolation and fostering inclusivity. However, it must ensure that the interface does not cause cognitive overload, especially for users unfamiliar with advanced technology.

4.2 Public Safety Considerations

The application must prioritize user data privacy by implementing robust encryption mechanisms. Features such as real-time translation must prevent the leakage of sensitive information. Moreover, secure access control will mitigate risks associated with unauthorized usage, ensuring user safety.

4.3 Public Welfare Considerations

By breaking down language barriers, the application enhances global collaboration, fostering professional and personal connections. This improves job opportunities, academic discussions, and cross-cultural interactions, contributing positively to the users' social and economic welfare.

4.4 Global Considerations

The application supports multilingual functionalities, making it accessible across diverse regions and demographics. Careful localization ensures that cultural and linguistic nuances are respected, enhancing its global appeal. Future plans include expanding supported languages to encompass underserved linguistic communities.

4.5 Cultural Considerations

Cultural sensitivity is a cornerstone of the application's design. The translation algorithms are tuned to respect idiomatic expressions, avoiding literal translations that could lead to misunderstandings. The interface is designed to accommodate cultural norms in colors, text alignment, and symbols.

4.6 Social Considerations

The application fosters inclusivity by bridging communication gaps, allowing individuals from diverse backgrounds to collaborate effortlessly. This aligns with the goals of reducing social divides and creating opportunities for marginalized linguistic groups.

4.7 Environmental Considerations

The application's backend architecture leverages cloud-based services, minimizing the need for extensive physical infrastructure. Optimized code and resource-efficient servers contribute to reduced energy consumption, aligning with sustainability goals.

4.8 Economic Considerations

The application provides a cost-effective alternative to expensive translation services. Its scalability ensures affordability for individual users, businesses, and educational institutions. By facilitating global communication, it opens up new market opportunities, driving economic growth.

5 Tools and Technologies

- **Server**: Built using Python for backend operations and LLM integration.
- **Client**: Developed as an Android app in Kotlin.
- **Database**: SQLite is used for data management.
- **UML Diagrams**: Visualized architecture using Unified Modeling Language.
- **Standards**: Adhered to IEEE guidelines for software documentation and design.
- **User Interface:** Jetpack Compose is used for building the user interface.se

6 Glossary

API (Application Programming Interface): Defines how software components should interact

Channel: Communication mechanism between different application components

Database Context: The main class that coordinates Entity Framework functionality

Entity: A class that represents a database table

Gemini: An AI model that is generalized to be able to process various types of information, inclusive of text, code, audio, image, and video

HTTP: Hypertext Transfer Protocol for data communication

LLM (Large Language Models): Advanced AI models for natural language processing

SQL (Structured Query Language): A standard language for accessing and manipulating databases.

StateFlow: Data holder that emits updates to its collectors

ViewModel: Component that stores and manages UI-related data

7 References

- https://www.acm.org/code-of-ethics
- https://www.lucidchart.com/pages/how-to-draw-a-deployment-diagram-in-uml#:~:text=Turn%20on%20the%20UML%20shape,instance%2C%20or%20a%20basic%20object.
- https://www.visual-paradigm.com/tutorials/how-to-draw-deployment-diagram-in-uml/
- https://app.diagrams.net/
- https://developer.android.com/
- https://ai.google.dev/competition/projects/multimodal-gemini-15-flash-api?hl=tr
- https://www.sqlite.org/docs.html
- https://ktor.io/docs/welcome.html