# TED UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

# CMPE 491 - Senior Project Analysis Report

Submission Date

19.04.2024

**Group Members:**

Bartu Özen **22804498074**

Elif Nazlı Böke **11863076002**

Gülce Ayşe Döker **16483072168**

Toygar Yurt **4012670916**

# Table of Contents

# Analysis Report

## 1. Introduction

Real time translator will be an application which will comfort people in their social life. Main purpose of this application is to provide users an efficient and reliable communication experience with translation. Application will enable translation of text and spoken language by using speech recognition and natural language processing technologies.

## 2. Proposed system

### 2.1 Overview

The aim of the proposed system of our project is to enhance the communication and remove the language barrier via developing a mobile application. Each user will have this app in their phone. This app will capture the speeches, process it and translate it into the desired language. It will implement LLM since it can understand context, capture nuances, and generate more natural-sounding translations compared to traditional methods. Users will be able to make voice calls in which the conversation will be translated to the other user in real-time. Users will also be able to chat with each other; they will send text messages and the other user will get the message as translated on their side.

### 2.2 Functional Requirements

**User Authentication and Registration:**

- Users should be able to register for an account using their email address.
- Registered users should be able to log in securely to access the app's features.

**Profile Management:**

- Users should be able to create and manage their profiles, including adding profile pictures, personal information.
- Users should be able to edit their profiles and update information as needed.

**Friend Management:**

- Users should be able to search for other users and add them as friends.
- Users should be able to view their list of friends and manage friend requests (accepting or rejecting).
- Users should be able to remove friends from their friend list.

**Translation During Calls:**

- Users should be able to initiate calls with their friends from within the app.
- During a call, the app should translate the conversation in real-time based on the preferred languages of the users involved.

**Text Messaging with Translation:**

- Users should be able to send text messages to their friends within the app.
- Messages sent by users should be automatically translated into the recipient's preferred language.
- The app should support synchronous translation of text messages, ensuring that both parties see the translated message simultaneously.
- Users should have the option to switch between languages for text input and output.

**Language Support:**

- The app should support a wide range of languages for both spoken conversation and text translation.

**Notification System:**

- Users should receive notifications for incoming friend requests, messages, and call invitations.

**Accessibility and Usability:**

- User interface elements should be intuitive and easy to navigate for all users.

**Offline Functionality:**

- The app should provide some level of functionality even when offline, such as viewing previously sent messages and accessing cached translations.

### 2.3 Nonfunctional Requirements

- Performance and Responsiveness: The application should be fast and responsive, with minimal lag or delays.
- Scalability: The application should be scalable, with the ability to handle a growing number of users.
- Accuracy: The translation accuracy should meet or exceed industry standards for machine translation.
- Feedback Mechanism: The translator should provide feedback mechanisms to users, such as error messages or suggestions for improving input text.
- Continuous Improvement: User feedback should be taken for continuous improvement.
- Security and Privacy: Ensure that user data, including chat messages and call content, is encrypted during transmission and storage to protect against unauthorized access.

### 2.4 Pseudo requirements

- Develop the app in Kotlin language.
- Develop server-side infrastructure using Linux and support x86 or ARM architectures.
- Implement scalable and efficient database solutions such as PostgreSQL or MongoDB to handle large volumes of user data and message transactions.
- Implement Material Design principles for consistent and intuitive user interface across devices.
- Ensure compatibility with Android versions 5.0 (Lollipop) and above to reach a broad user base.
- Establish secure communication protocols between the Android app and the server infrastructure to protect user data and privacy.
- Implement rate-limiting and throttling mechanisms to prevent abuse and ensure fair resource allocation among users.
- Distribute the app through the Google Play Store.
- Adhere to copyright and intellectual property laws when handling user-generated content, ensuring proper attribution and permissions are obtained.

## 2.5 System models

### 2.5.1 Scenarios

User Registration:

- User navigates to the app and selects the registration option.
- User enters their details such as name, email, and password.
- User submits the registration form.
- The system validates the user's input and creates a new account.
- Confirmation email is sent to the user for account verification.

User Login:

- User opens the app and selects the login option.
- User enters their registered email and password.
- The system authenticates the user.
- Upon successful authentication, the user gains access to the app's features.

Adding Friends:

- User navigates to the friends section of the app.
- User selects the option to add a new friend.
- User enters the friend's username or email.
- The system searches for the friend's account and displays the result.
- User selects the friend they want to add and sends a friend request.
- The friend receives the request and can either accept or decline it.

Removing Friends:

- User goes to the friends section of the app.
- User selects the friend they want to remove.
- User confirms the removal action.
- The system removes the friend from the user's friend list.

Initiating a Call:

- User selects a friend from their friend list whom they want to call.
- User initiates a call request.
- The friend receives the call request and accepts it.
- Both users are connected via a synchronous translator call.

- As they speak, their conversation is translated in real-time to the selected language of each user.

Sending Text Messages:

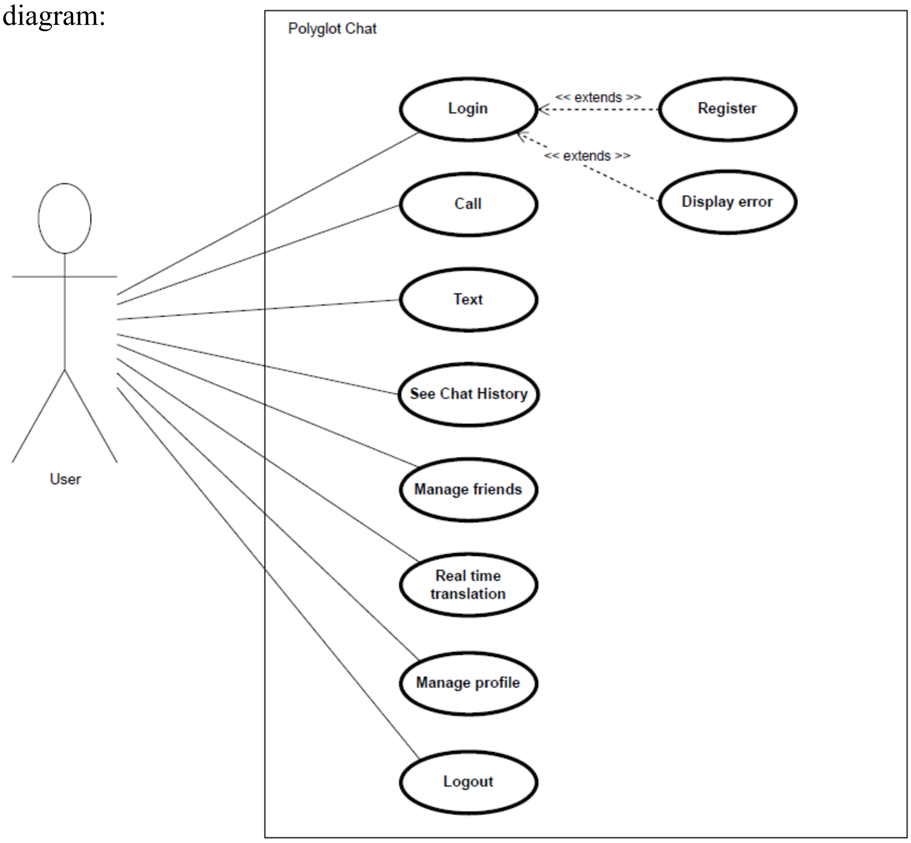
- User selects a friend to whom they want to send a message.
- User types the message in their preferred language.
- Upon sending, the system translates the message to the friend's preferred language in real-time.
- The friend receives the message in their preferred language.
- The friend can then respond, and the process repeats for each message exchange.

Logging Out:

- User decides to log out of the app.
- User selects the logout option from the menu.
- The system clears the user's session and logs them out.
- User is redirected to the login screen.

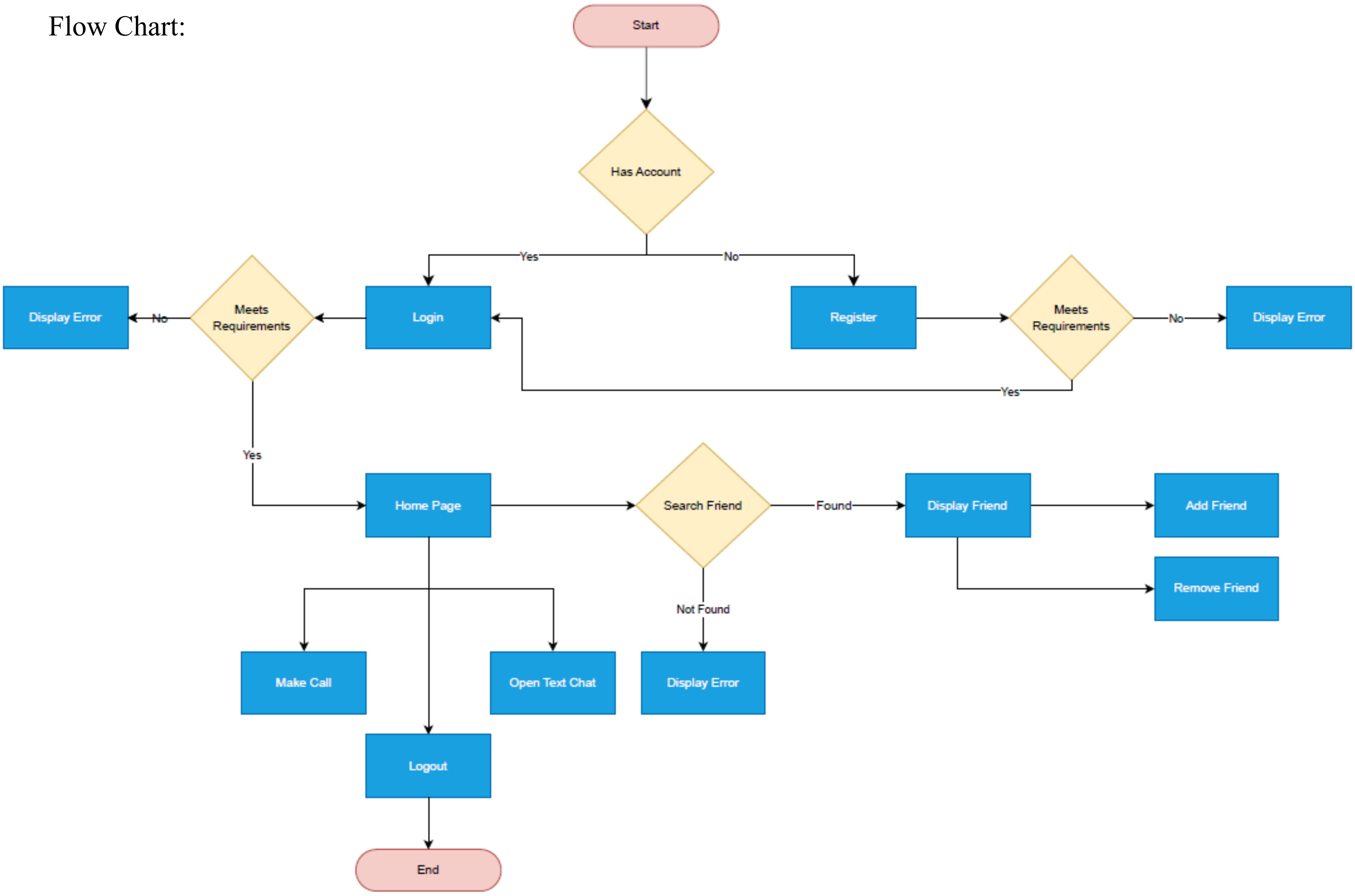
### 2.5.2 Use case model

Use case diagram:

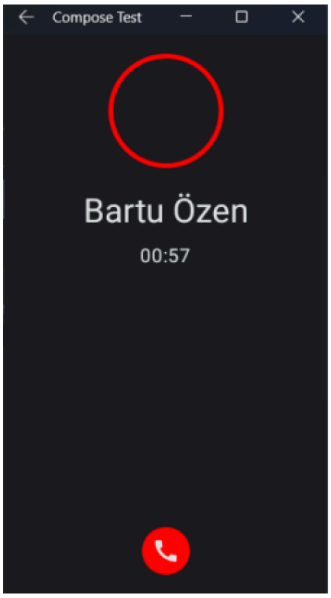


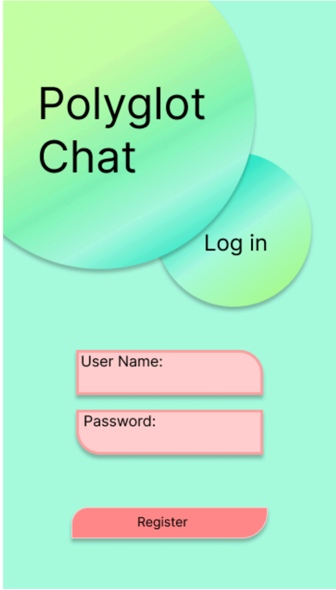
### 2.5.3 Dynamic model

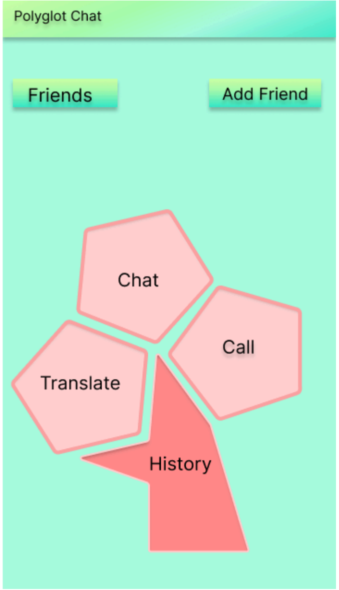
Flow Chart:



### 2.5.4 User interface - navigational paths and screen mock-ups







## 3. Glossary

- **LLM (Large Language Model)**: Refers to a language model capable of understanding context, capturing nuances, and generating natural-sounding translations, often used for enhancing translation accuracy.
- **Kotlin**: The programming language chosen for developing the mobile application.
- **Linux**: The operating system used for server-side infrastructure development.
- **PostgreSQL/MongoDB**: Scalable and efficient database solutions selected for handling large volumes of user data and message transactions.
- **Material Design**: Design principles developed by Google that emphasizes clean, minimalistic aesthetics, intuitive user experiences, and consistent visual elements across platforms and devices.
- **Google Play Store**: Platform for distributing the application to Android users.

## 4. References

- Interface Supportability Requirement Constraints. Retrieved From **interface Supportability requirement Constraints pseudo requirements | Course Hero**
- Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices. (20023, December 30). Retrieved From **https://www.altexsoft.com/blog/non-functional-requirements/**
- What is a flowchart?. Retrieved From **https://asq.org/quality-resources/flowchart**