**Multilingual Web Chat System**



**Submitted By: Hamna Imran**

**Supervised By: Ms. Memoona Afsheen**

**Department of Computer Sciences**

**Quaid-i-Azam University, Islamabad**

**(2021-2025)**

**Acknowledgment**

First of all, I would like to express my heartfelt gratitude to my supervisor, **Ms. Memoona Afsheen**, for her guidance, encouragement, and support throughout my Final Year Project. Her constructive feedback and expertise have been invaluable in helping me complete this work.

I am also thankful to my peers and colleagues who provided constructive feedback and supported me along the way.

Lastly, I appreciate everyone who contributed, directly or indirectly, to the completion of this project. Your support means a lot to me.

# 

**Abstract**

In today’s world where more than 7000 languages are spoken, seamless cross-language communication is still a challenge, especially in the corporate sector. Existing chat systems don’t support real-time communication across languages. This creates problems for tech businesses trying to connect globally.

Multilingual Web Chat System solves this by creating a user-friendly web-based multilingual chat application for tech support. It enables tech teams to communicate with their international clients seamlessly. The user will talk in their preferred language while the system translates messages and voice chats in real-time. We’re using the MERN stack and Google Translation API to make this possible.

The application has features like preferred language selection, multimedia sharing, chat history, and strong encryption to keep conversations private and secure. Our goal is to make global communication simple and accessible to unite people and ideas, despite language barriers.

# 

# 

# 

# 

Table of Contents

[Chapter 1 6](#_Toc190782359)

[Introduction 6](#_Toc190782360)

[1.2 Problem Definition 7](#_Toc190782361)

[1.3 Proposed Solution 7](#_Toc190782362)

[1.4 Purpose 7](#_Toc190782363)

[1.5 Objectives 7](#_Toc190782364)

[1.6 Scope 8](#_Toc190782365)

[What’s included: 8](#_Toc190782366)

[What’s not included: 8](#_Toc190782367)

[Limitations: 8](#_Toc190782368)

[1.7 Technologies and Tools 8](#_Toc190782369)

[1.9 Process Model 9](#_Toc190782370)

[1.10 Project Timeline 10](#_Toc190782371)

[Chapter 2 12](#_Toc190782372)

[Software Requirements Specification 12](#_Toc190782373)

[2.0. Introduction 13](#_Toc190782374)

[2.0.1 Purpose 13](#_Toc190782375)

[2.0.2 Definitions, Acronyms, and Abbreviations 13](#_Toc190782376)

[2.1 Overall Description 13](#_Toc190782377)

[2.1.1 Product Perspective 13](#_Toc190782378)

[2.1.2 Product Functions 13](#_Toc190782379)

[2.1.3 User Characteristics 14](#_Toc190782380)

[2.1.4 Constraints 14](#_Toc190782381)

[2.1.5 Assumptions and Dependencies 14](#_Toc190782382)

[2.2 Specific Requirements 15](#_Toc190782383)

[2.2.1 Functional Requirements (FR) 15](#_Toc190782384)

[2.2.2 Non-Functional Requirements (NFR) 15](#_Toc190782385)

[2.3 Major Inputs and Outputs 16](#_Toc190782386)

[Inputs: 16](#_Toc190782387)

[Outputs: 17](#_Toc190782388)

[2.4 Use Case Diagram 18](#_Toc190782389)

[2.5.1 Login 20](#_Toc190782390)

[2.5.2 Sign Up 21](#_Toc190782391)

[2.5.3 Google/Facebook Login (Social Login) 22](#_Toc190782392)

[2.5.4 Forgot Password 23](#_Toc190782393)

[2.5.5 Set Language 25](#_Toc190782394)

[2.5.6 Send Text Messages 26](#_Toc190782395)

[2.5.7 Receive Text Messages 26](#_Toc190782396)

[2.5.8 Automatic Translation of Text Messages 27](#_Toc190782397)

[2.5.9 Send Voice Messages 28](#_Toc190782398)

[2.5.10 Translate Voice Messages 29](#_Toc190782399)

[2.5.11 View Past Chat History 30](#_Toc190782400)

[2.5.12 Search Chat History 31](#_Toc190782401)

[2.5.13 Group Chat Creation 31](#_Toc190782402)

[2.5.14 Add/ Remove members 33](#_Toc190782403)

[2.5.15 Real-Time Notifications 34](#_Toc190782404)

[2.5.16 Profile Customization 35](#_Toc190782405)

[2.5.17 Logout 35](#_Toc190782406)

[2.5.18 View Profile 36](#_Toc190782407)

[2.6 System Sequence Diagrams 37](#_Toc190782408)

[2.6.1 Login 37](#_Toc190782409)

[2.6.2 Signup 38](#_Toc190782410)

[2.6.3 Logout 38](#_Toc190782411)

[2.6.4 Send Notifications 39](#_Toc190782412)

[2.6.5 View Profile 39](#_Toc190782413)

[2.6.6 Add/Remove Members 40](#_Toc190782414)

[2.6.7 Create Group 40](#_Toc190782415)

[2.6.8 Search by Keyword 41](#_Toc190782416)

[2.6.9 View History 41](#_Toc190782417)

[2.6.10 Recieve Message 42](#_Toc190782418)

[2.6.11 Send Message 42](#_Toc190782419)

[2.6.12 Set Language 43](#_Toc190782420)

[2.6.13 Forgot Password 43](#_Toc190782421)

[2.7 Domain Model 44](#_Toc190782422)

[Chapter 3 45](#_Toc190782423)

[Software Design Description 45](#_Toc190782424)

[Chapter 4 54](#_Toc190782425)

[Implementation and Testing 54](#_Toc190782426)

[4.1 Implementation: 55](#_Toc190782427)

[1. Frontend Development with React.js 55](#_Toc190782428)

[2. Backend Development with Node.js and Express.js 55](#_Toc190782429)

[3. Database Management with MongoDB 55](#_Toc190782430)

[4. Version Control with Git and GitHub 55](#_Toc190782431)

[5. API and Libraries Integration: 55](#_Toc190782432)

[4.2. Testing 56](#_Toc190782433)

[4.2.1 Introduction 56](#_Toc190782434)

[4.2.2 Test Strategy 56](#_Toc190782435)

[4.2.3 Features to be Tested 56](#_Toc190782436)

[4.3. Test Cases 57](#_Toc190782437)

[Chapter 5 64](#_Toc190782438)

[Conclusion and Future Work 64](#_Toc190782439)

[5.1 Conclusion 65](#_Toc190782440)

[5.2 Future Enhancements 65](#_Toc190782441)

[References 66](#_Toc190782442)

### 

# 

# 

# 

# 

# 

# Chapter 1

## Introduction

## 

## 

## 1.2 Problem Definition

In today’s globalized world, tech support teams face challenges communicating with global clients who speak different languages. Current solutions are either subscription-based or provide limited functionality. This makes it difficult to provide quick and effective support to international clients. This language barrier causes delays, misunderstandings, and frustration for both clients and support teams.

Current Solutions: Skype (limited languages translation), Slack, Copilot-live (subscription-based)

## 1.3 Proposed Solution

Multilingual Web Chat System is a MERN stack application that solves this problem by providing real-time text and voice chat translation. Users can communicate in their own language while the system handles translations in real-time.

## 1.4 Purpose

The purpose of Multilingual Web Chat System is to help tech support teams improve efficiency, enhance customer service, and drive business growth by breaking language barriers. It enables seamless communication between teams and clients, no matter their location or language.

## 1.5 Objectives

Here’s what we want to achieve with this project:

1. Provide real-time translation for text and voice chats.
2. Let users choose their preferred language for communication.
3. Keep chats private and protected with encryption.
4. Support features like group chats, multimedia sharing, and chat history.
5. Design a platform that’s simple and easy to use.
6. Build the system to handle more users and languages as needed.
7. Provide compatibility across different devices for accessibility (e.g. android, laptop, etc.)

## 1.6 Scope

Multilingual Web Chat System is a standalone web-based multilingual chat application.

### What’s included:

● Real-time translation for text and voice messages

● Support for multimedia (like images, emojis, and voice notes)

● Chat features for both individual and group conversations, with language settings

● Customizable user profile photo and status

● Status Indicators(online, offline, do not disturb)

● Secure communication

● A searchable chat history

● Notifications for new and missed messages

### What’s not included:

● Offline messaging or translation

● Complex language features like grammar correction

### Limitations:

● Only languages that Google Translation API supports

● Requires a stable internet connection to work properly

## 1.7 Technologies and Tools

We’re using the following technologies to develop Multilingual Web Chat System.

* Planning - MS word, Project Libre
* Analysis – Draw.io
* Documentation - MS Word
* Design – Figma, Canva
* Database - MongoDB Atlas
* Development - Visual Studio Code
* Google Translation API for translation

## 1.9 Process Model

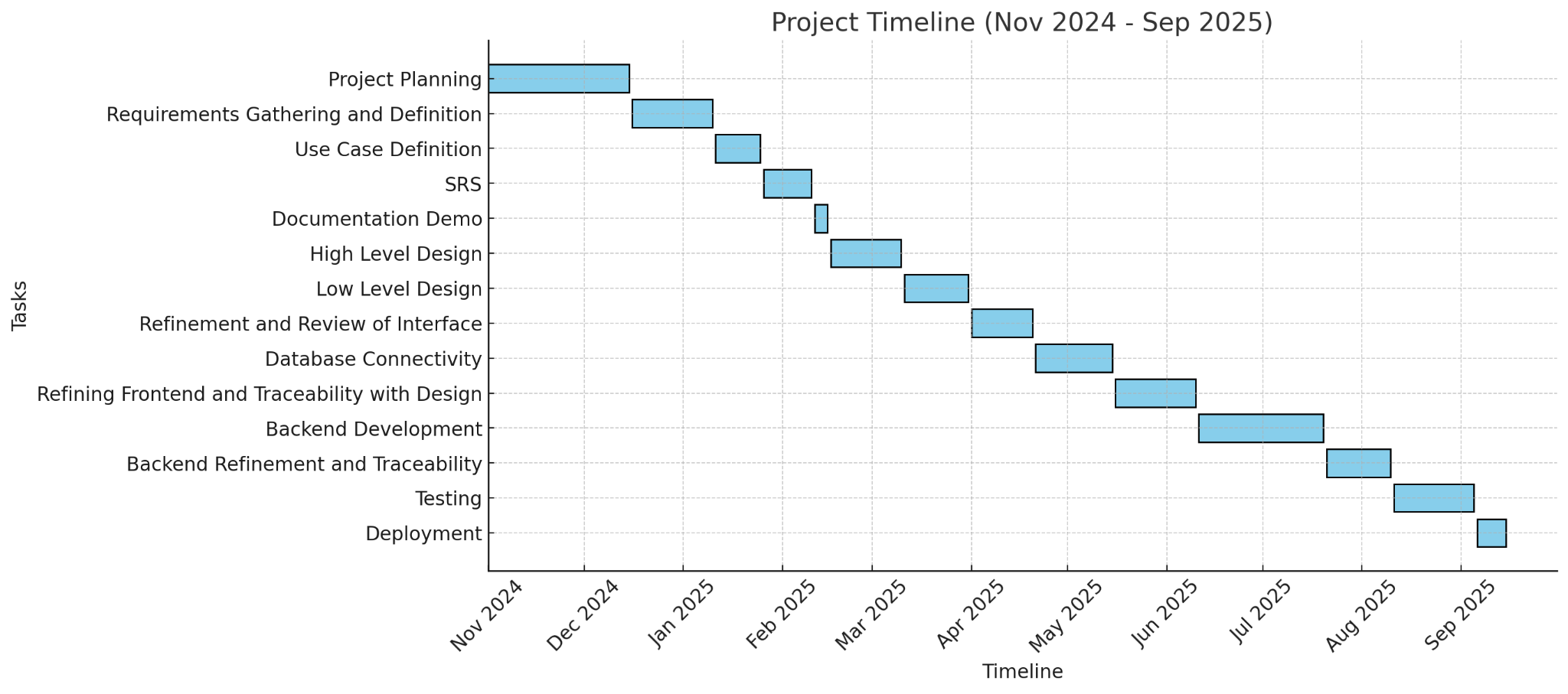
The **Agile Model** is the best choice for this project because:

Since I will have frequent meetings with your supervisor, the project needs a model that allows feedback at different stages. The Incremental Model makes sure that each phase is reviewed before moving forward.

If the supervisor suggests improvements, the model allows modifications in the increment without affecting the entire project.

Instead of waiting until the end, I can develop and test features(like messaging, translation, etc.) step by step. This reduces risk and makes it easier to identify issues early.

## 1.10 Project Timeline



**1.11 Risks and Contingencies**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk ID** | **Risk Description** | **Risk Category** | **Likelihood** | **Contingency Plan** |
| R1 | Real-time translation errors | Technical | High | Multiple translation APIs testing |
| R2 | Real-time voice-to-text transcription error | Technical | High | Multiple transcription APIs testing |
| R3 | Data breach or unauthorized access | Security | High | SSL, Encryption, authentication implementation |
| R4 | Server downtime affecting chat services | Operational | Medium | backup servers |
| R5 | latency in real-time translation and transcription | Technical | High | Optimize API calls, use WebSockets for faster processing, and implement local caching for common phrases |

# 

# Chapter 2

## Software Requirements Specification

### 2.0. Introduction

One of the most important documents in software development is the SRS document. It lays out how the system should work and acts as a roadmap for everyone involved. It helps turn an idea into clear, detailed instructions that developers can follow. It also serves as an agreement, outlining what the client needs and what the developers will deliver.

### 2.0.1 Purpose

This document defines the requirements for a **Multilingual Web** **Chat System** that enables real-time communication between users speaking different languages. The system will handle text and voice chat and automatically translate using the Google Translation API.

### 2.0.2 Definitions, Acronyms, and Abbreviations

* **API**: Application Programming Interface
* **FR:** Functional Requirement
* **NFR**: Non-Functional Requirement
* **SSL**: Secure Socket Layer
* **UI**: User Interface
* **UX**: User Experience
* **SRS**: Software Requirements Specification

## 2.1 Overall Description

### 2.1.1 Product Perspective

The system is a standalone web application that integrates with the Google Translation API for real-time language translation. It will be accessible via web browsers and will not require additional software installation.

### 

### 2.1.2 Product Functions

1. User Registration and Authentication
2. Language Selection and Translation
3. Text and Voice Messaging
4. Voice Message Transcription
5. Chat History and Search
6. Group Chat Functionality
7. Notifications
8. User Profiles and Status
9. End-to-End Encryption

### 2.1.3 User Characteristics

Multilingual Web Chat System is designed for all kinds of users belonging to different age groups and with varying exposure to technology. Naïve users may require more guidance to learn the system than others.

### 2.1.4 Constraints

* **Translation API limits** The free version of Google Translation API has usage restrictions.
* **Internet dependency** The system needs a stable internet connection for real-time translation.
* **Latency issues** Translations and transcriptions may have slight delays.
* **Voice accuracy** Speech-to-text transcription may not always be 100% accurate.

### 2.1.5 Assumptions and Dependencies

* Users have access to a stable internet connection.
* Google Translation API will be available and functional.

## 2.2 Specific Requirements

### 2.2.1 Functional Requirements (FR)

**FR0**: The system shall allow users to **register and log in securely**.

**FR1**: Users shall be able to **log in using Google and Facebook**.

**FR2**: The system shall allow users to **set their preferred language** for communication.

**FR3**: Users shall be able to **send and receive text messages**.

**FR4**: The system shall **automatically translate text messages** into the recipient's language.

**FR5**: Users shall be able to **send and receive voice messages**.

**FR6**: The system shall **translate voice messages.**

**FR7**: Users shall be able to **view past chat history**.

**FR8**: The system shall allow users to **search chat** using keywords.

**FR9**: The system shall support **group chats with real-time translation**.

**FR10**: Users shall receive **real-time notifications** for incoming messages.

**FR11**: The system shall provide **status indicators** (Online, Offline, Do Not Disturb).

**FR12**: Users shall be able to **customize their profiles** with pictures, statuses, and language preferences.

### 2.2.2 Non-Functional Requirements (NFR)

#### 1. Performance

* **NFR-1**: The system must support **up to 1000 concurrent users** without significant lag.
* **NFR-2**: Real-time translation must occur within **3 seconds** for text messages.
* **NFR-3**: The chat interface must load within **2 seconds** on a standard internet connection.

#### 2. Security

* **NFR-4**: User passwords must be stored using **hashing algorithms**.
* **NFR-5**: All communication must be encrypted using **SSL/TLS protocols**.

#### 3. Usability

* **NFR-7**: The system must have a **simple and intuitive UI/UX**.
* **NFR-8**: The system must be compatible with **Chrome, Firefox, and Edge** browsers.
* **NFR-9**: The system must provide **basic tooltips** for user guidance.

#### 4. Reliability

* **NFR-10**: The system must have an **uptime of 95%** during the testing phase.
* **NFR-11**: Chat history must be stored for at least **30 days**.

#### 6. Maintainability

* **NFR-13**: The codebase must be well-documented and follow **coding standards**.
* **NFR-14**: The system must log errors for debugging purposes.

#### 7. Compatibility

* **NFR-15**: The system must work on **desktop and mobile browsers**.

## 2.3 Major Inputs and Outputs

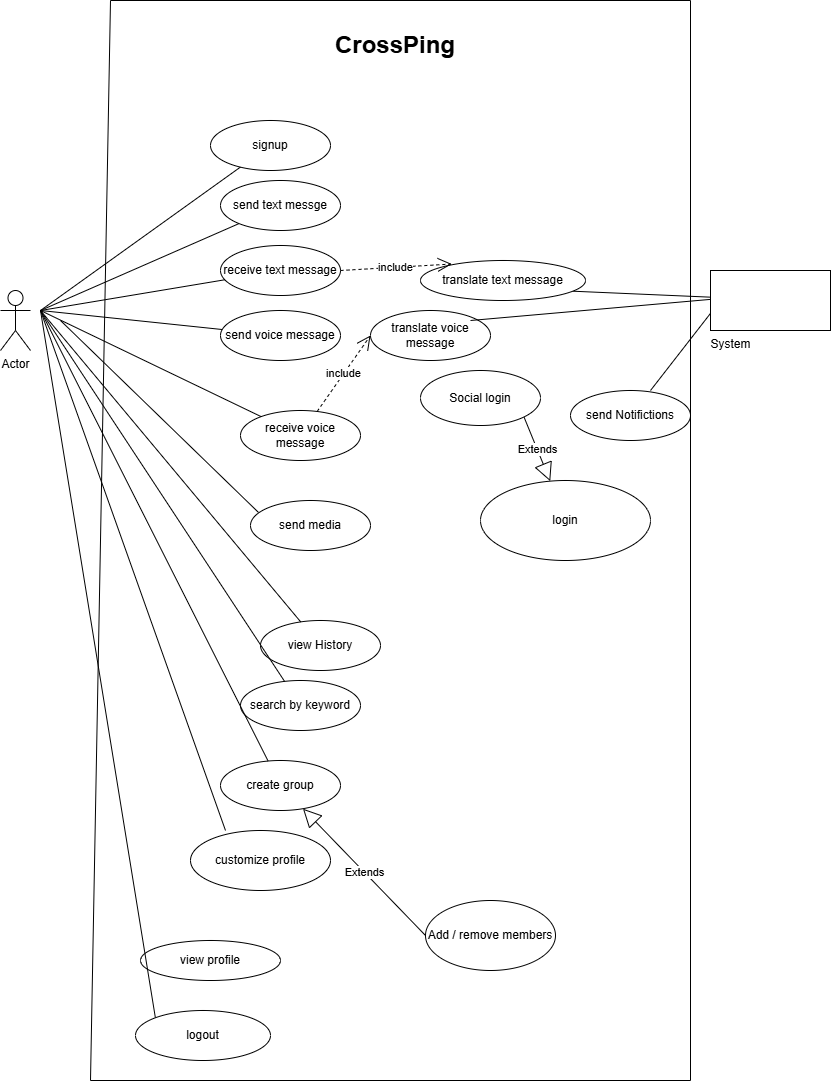
### Inputs:

* Login Credentials (Email/Password, Google/Facebook Login)
* Preferred Language Selection
* Text Messages (Sent by Users)
* Voice Messages (Sent by Users)
* Images & Emojis
* Profile Information (Name, Profile Picture, Status)
* Group Chat Creation Requests
* Search Keywords (For Chat History)
* Status Updates (Online, Offline, Do Not Disturb)

### Outputs:

* Translated Text Messages (Displayed in Recipient’s Preferred Language)
* Translated Voice Messages (Displayed as Text or Played in Translated Audio)
* Chat History Results (Based on User Search)
* User Profile Details
* Group Chat Messages (Translated in Real-time)
* Notification Alerts (For New Messages, Group Invites)
* Error Messages (Incorrect Login, Language Not Supported, Network Issues)

## 2.4 Use Case Diagram



Multilingual Web Chat

**2.4.1 Use Cases List**

**Authentication & User Management**

* Login
* Social Login (uses Login)
* Signup
* Forgot Password
* Logout

**Messaging**

* Send Text Message
* Receive Text Message
* Translate Text Message
* Send Voice Message
* Receive Voice Message
* Translate Voice Message

**Chat Management**

* View History
* Search by Keyword
* Create Group Chat
* Add/Remove Members

**Notifications & Profile Management**

* Send Notifications
* Customize Profile
* View Profile

**2.5 Use Cases’ Description**

### 2.5.1 Login

|  |  |
| --- | --- |
| ID | UC1 |
| Name | Login |
| Primary Actor | User |
| Pre-Condition | The user has a laptop, pc or mobile  The user has already signed up for an account  The user knows his password |
| Post Condition | The user has logged in to the web application successfully.  The system will redirect the user to connect Threads account page |
| Main Success Scenario | The user opens the app and clicks on the "Login" button  The system presents the login screen with email/username and password fields  The user enters their credentials and clicks "Login"  The system validates the credentials  The user successfully logs into the system and is redirected to the dashboard or home page |
| Alternative Flows | Invalid Input: If the user enters incorrect or incomplete information (e.g., invalid email, weak password), the system prompts them to correct it  Email Already Registered: If the email is already in use, the system asks the user to log in instead or use another email |
| Frequency | Occasionally |

### 2.5.2 Sign Up

|  |  |
| --- | --- |
| ID | UC2 |
| Name | Sign up |
| Primary Actor | User |
| Pre-Condition | The user has access to the internet and a device (PC, mobile, etc.) |
| Post Condition | User successfully registers an account and is redirected to the login screen or home page |
| Main Success Scenario | The user opens the app and clicks on the "Sign Up" button  The system presents the signup form (name, email, password, etc.)  The user enters their details and click "Submit"  The system validates the details (e.g., email format, password strength)  The user successfully registers and is redirected to the login page or home page |
| Alternative Flows | Invalid Credentials: If the user enters the wrong password or email, the system prompts them to re-enter the credentials  Account Locked: If there are multiple failed login attempts, the system locks the account for a specified period and notifies the user |
| Frequency | Rarely (first-time users only) |

### 2.5.3 Google/Facebook Login (Social Login)

|  |  |
| --- | --- |
| ID | UC2.1 |
| Name | Google/Facebook Login |
| Primary Actor | User |
| Pre-Condition | User has a Google or Facebook account |
| Post Condition | The user successfully logged in via Google or Facebook and is redirected to the login screen or home page |
| Main Success Scenario | The user opens the app and clicks "Login with Google" or "Login with Facebook"  The system redirects the user to the respective platform for authentication  User grants permissions  The system logs the user in and redirects them to the home page or dashboard |
| Alternative Flows | If the Facebook or Google accounts are not logged in, it is required to log in first |
| Frequency | Occasionally |

### 2.5.4 Forgot Password

|  |  |
| --- | --- |
| ID | UC2.2 |
| Name | Forgot Password |
| Primary Actor | User |
| Pre-Condition | User is on the login page and has forgotten their password |
| Post Condition | User successfully resets their password and is redirected to the login screen |
| Main Success Scenario | The user clicks on the "Forgot Password" link on the login page  The system prompts the user to enter their email address  The user enters their email and submits  The system sends a password reset link to the email address  The user clicks the link and is redirected to a page to create a new password  The user successfully sets a new password and is logged in automatically |
| Alternative Flows | *Email Not Found*: If the email is not registered, the system notifies the user and offers alternative options |
| Frequency | Occasionally |

### 2.5.5 Set Language

|  |  |
| --- | --- |
| ID | UC3 |
| Name | Set Language |
| Primary Actor | User |
| Pre-Condition | User is logged in |
| Post Condition | The user’s preferred language is saved and displayed throughout the application |
| Main Success Scenario | 1. The user navigates to the settings 2. User selects their desired language from the list of available languages 3. The system updates the UI to reflect the selected language |
| Alternative Flows | *Language Not Supported*: The system informs the user if the chosen language is unavailable |
| Frequency | Occasionally |

### 2.5.6 Send Text Messages

|  |  |
| --- | --- |
| ID | UC4 |
| Name | Send Text Messages |
| Primary Actor | User |
| Pre-Condition | The user is logged in, has an internet connection, and in an active chat conversation |
| Post Condition | The message is successfully sent |
| Main Success Scenario | The user types a message  The user clicks "Send"  The system transmits the message to the recipient |
| Alternative Flows | *Message Not Sent*: The system alerts user if there’s a network issue and prompts them to try again |
| Frequency | Frequently |

### 2.5.7 Receive Text Messages

|  |  |
| --- | --- |
| ID | UC4 |
| Name | Receive Text Messages |
| Primary Actor | User |
| Pre-Condition | User is logged in and has an active internet connection |
| Post Condition | User receives the message |
| Main Success Scenario | User waits for an incoming message  System notifies the user of a new text message  User opens the chat to view the received message |
| Alternative Flows | The user waits for an incoming message  System notifies the user of a new text message  User opens the chat to view the received message |
| Frequency | Frequently |

### 2.5.8 Automatic Translation of Text Messages

|  |  |
| --- | --- |
| ID | UC5 |
| Name | Automatic Translation of Text Messages |
| Primary Actor | User |
| Pre-Condition | Preferred language is chosen |
| Post Condition | Text message is automatically translated into the recipient’s language |
| Main Success Scenario | 1. User sends a message in their preferred language 2. The system detects the recipient’s language preference 3. System automatically translates the message into the recipient’s language 4. Recipient receives the translated message |
| Alternative Flows | Translation Error: System notifies the user if translation fails due to connectivity or backend issue |
| Frequency | Frequently |

### 2.5.9 Send Voice Messages

|  |  |
| --- | --- |
| ID | UC6 |
| Name | Send Voice Messages |
| Primary Actor | User |
| Pre-Condition | User is logged in and in an active chat conversation |
| Post Condition | Voice message is successfully sent |
| Main Success Scenario | User records a voice message  User clicks "Send"  System sends the voice message to the recipient |
| Alternative Flows | *Audio Fail*: System alerts user if audio is unavailable due to network issues |
| Frequency | Frequently |

### 2.5.10 Translate Voice Messages

|  |  |
| --- | --- |
| ID | UC7 |
| Name | Translate Voice Messages |
| Primary Actor | User |
| Pre-Condition | User has received a voice message |
| Post Condition | Voice message is successfully translated to preferred language |
| Main Success Scenario | User listens to a voice message  System automatically transcribes the voice message |
| Alternative Flows | System notifies the user if translation fails |
| Frequency | Frequently |

### 2.5.11 View Past Chat History

|  |  |
| --- | --- |
| ID | UC7 |
| Name | View Past Chat History |
| Primary Actor | User |
| Pre-Condition | User is logged in |
| Post Condition | User can view past chat conversation |
| Main Success Scenario | User navigates to the chat history page  System displays a list of previous conversations  User selects a conversation to view details |
| Alternative Flows | *No History Available*: System informs the user that no past conversations exist |
| Frequency | Ocassionally |

### 2.5.12 Search Chat History

|  |  |
| --- | --- |
| ID | UC8 |
| Name | Search Chat History |
| Primary Actor | User |
| Pre-Condition | User is logged in and has chat history, enters keyword to search. |
| Post Condition | User can search and filter past conversations based on keyword |
| Main Success Scenario | User enters a search term in the search bar  System filters and displays the matching conversations |
| Alternative Flows | *No Results Found*: System displays a message that no results match the search query |
| Frequency | Ocassionally |

### 2.5.13 Group Chat Creation

|  |  |
| --- | --- |
| ID | UC9 |
| Name | Group Creation |
| Primary Actor | User |
| Pre-Condition | User is logged in and have more than 0 contacts |
| Post Condition | A new group chat is created successfully, and the user is added to the group |
| Main Success Scenario | User clicks on the "Create Group" button from the dashboard or chat interface  The system prompts the user to provide a group name, select group language preferences, and add members (via search or contact list)  User enters the group name, selects the language for the group, and adds contacts to the group  User clicks "Create Group"  The system creates the group chat, adds the selected members, and applies language preferences  User is redirected to the newly created group chat |
| Alternative Flows | invalid Group Name: If the group name is already taken or invalid, the system prompts the user to choose another name  No Contacts Added: If the user does not select any contacts, the system will prompt them to add at least one contact to create the group  Group Creation Error: If there’s an issue creating the group (e.g., server failure), the system will show an error message |
| Frequency | Ocassionally |

### 2.5.14 Add/ Remove members

|  |  |
| --- | --- |
| ID | UC10 |
| Name | Add/ remove members |
| Primary Actor | User (Group Admin) |
| Pre-Condition | User is a member of an existing group chat |
| Post Condition | Members are successfully added or removed from the group |
| Main Success Scenario | User opens the existing group chat  User clicks on the "Group Settings" button  System presents the option to add or remove members  User selects the contacts to add or remove  The system updates the group and notifies members about the changes |
| Alternative Flows | Permission Error: If the user is not an admin or does not have permission to add/remove members, the system will show a message explaining the error |
| Frequency | Ocassionally |

### 2.5.15 Real-Time Notifications

|  |  |
| --- | --- |
| ID | UC11 |
| Name | Real-Time Notifications |
| Primary Actor | User |
| Pre-Condition | User is logged in and device is connected to internet. |
| Post Condition | User receives a notification for new messages |
| Main Success Scenario | User receives a message while offline or away from the app  System sends a push notification  User taps on the notification to view the message |
| Alternative Flows | *Notification Failed*: If notification fails, the user can manually check for new messages |
| Frequency | Frequently |

### 2.5.16 Profile Customization

|  |  |
| --- | --- |
| ID | UC11 |
| Name | Profile Customization |
| Primary Actor | User |
| Pre-Condition | User is logged in and device is connected to internet. |
| Post Condition | Profile picture or status is updated |
| Main Success Scenario | User navigates to their profile page  User selects the option to edit profile  User uploads a new picture or changes status  User saves the changes |
| Alternative Flows | *Profile Update Error*: System alerts user if changes couldn't be saved due to network issue. |
| Frequency | Ocassionally |

### 2.5.17 Logout

|  |  |
| --- | --- |
| ID | UC12 |
| Name | Logout |
| Primary Actor | User |
| Pre-Condition | User is logged in and device is connected to internet. |
| Post Condition | User is logged out of the application |
| Main Success Scenario | User clicks on the logout button  System logs the user out and redirects to the login screen |
| Alternative Flows | *Logout Error*: System alerts user if logout failed due to network or server issue |
| Frequency | Ocassionally |

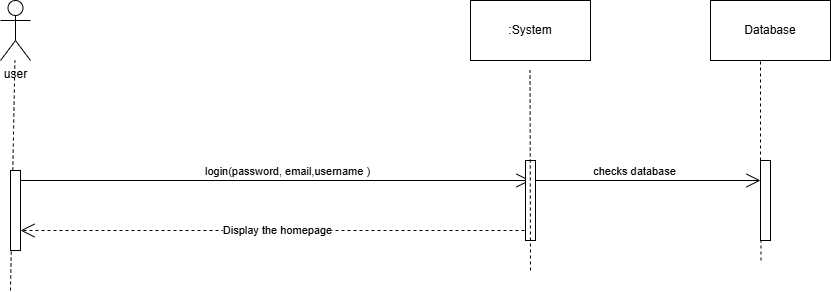
### 2.5.18 View Profile

|  |  |
| --- | --- |
| ID | UC13 |
| Name | View profile |
| Primary Actor | User |
| Pre-Condition | User is logged in and device is connected to internet. |
| Post Condition | The system displays the user’s profile information. |
| Main Success Scenario | The user opens the application.  The system displays the dashboard.  The user navigates to the profile section.  The system retrieves and displays the user’s profile details (name, email, profile picture, preferred language, and status). |
| Alternative Flows | Profile not displayed due to network issue |
| Frequency | Occasionally |

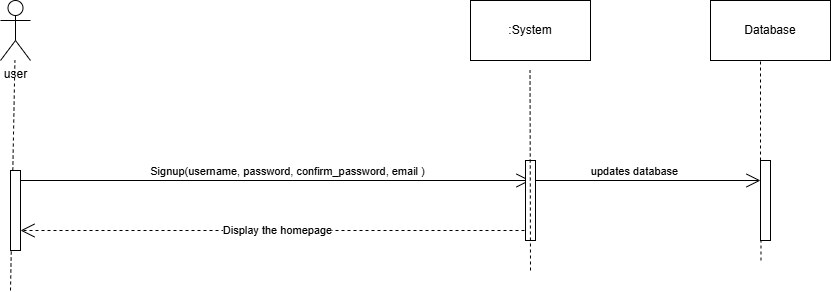
## 

## 2.6 System Sequence Diagrams

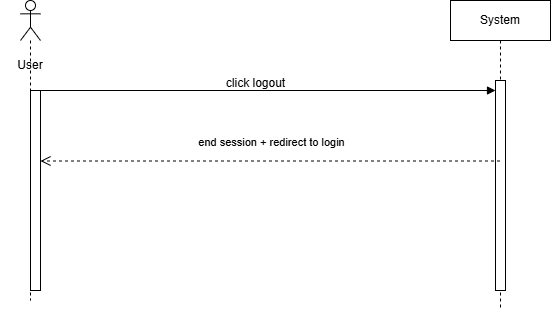
### 2.6.1 Login



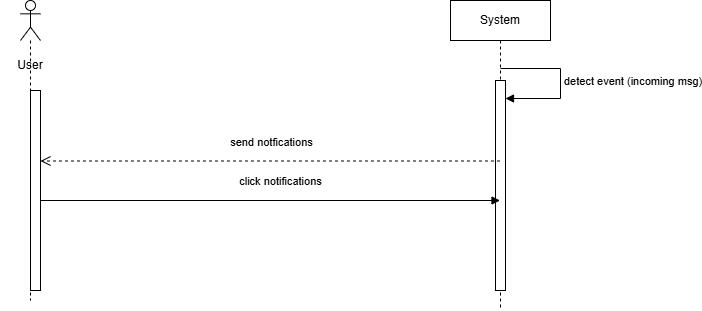
### 2.6.2 Signup



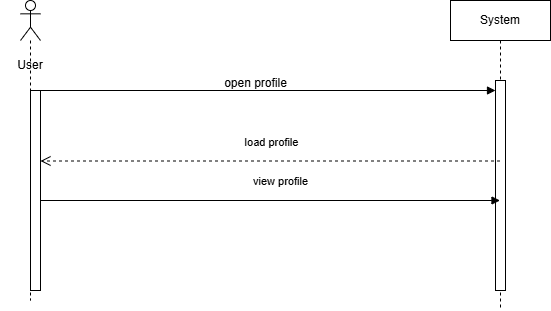
### 2.6.3 Logout



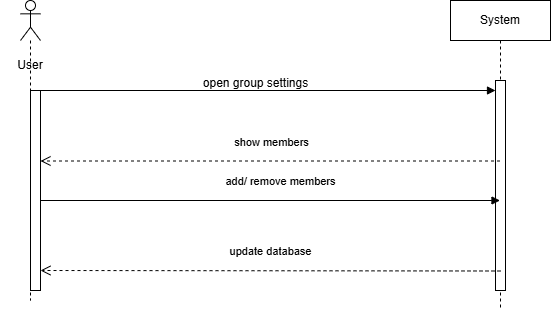
### 2.6.4 Send Notifications



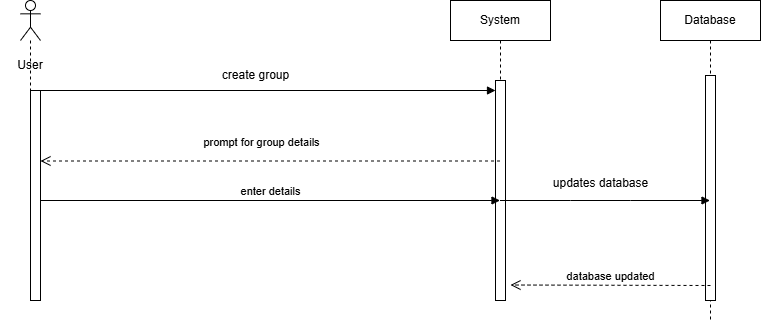
### 2.6.5 View Profile



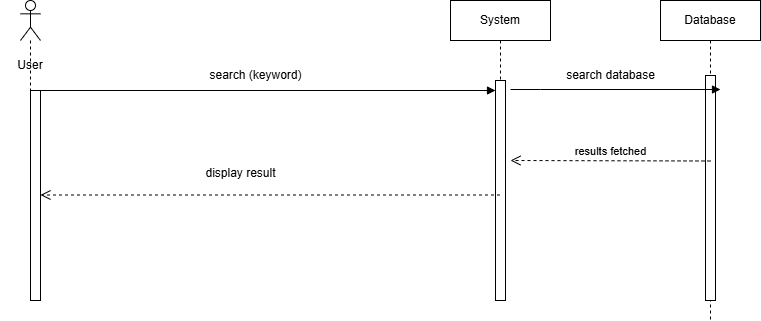
### 2.6.6 Add/Remove Members



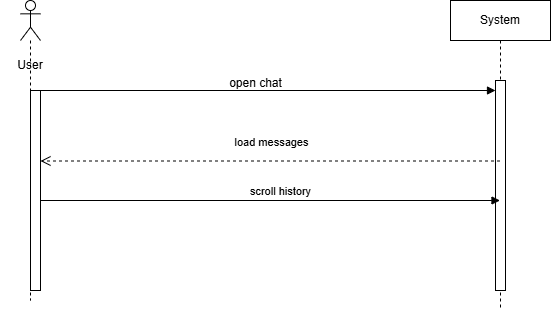
### 2.6.7 Create Group



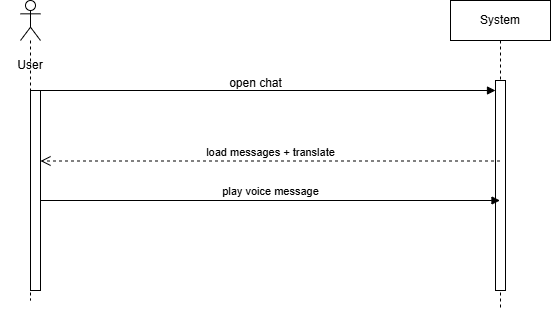
### 2.6.8 Search by Keyword



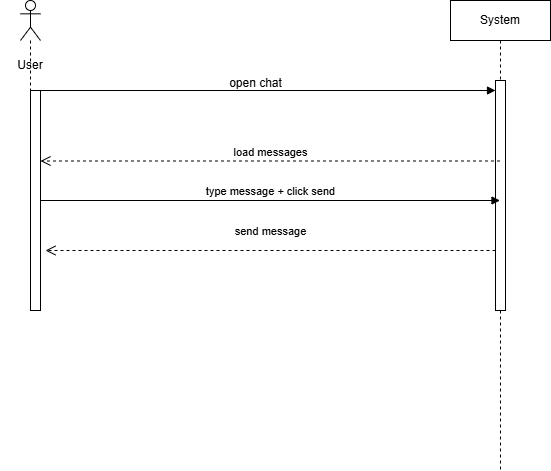
### 2.6.9 View History



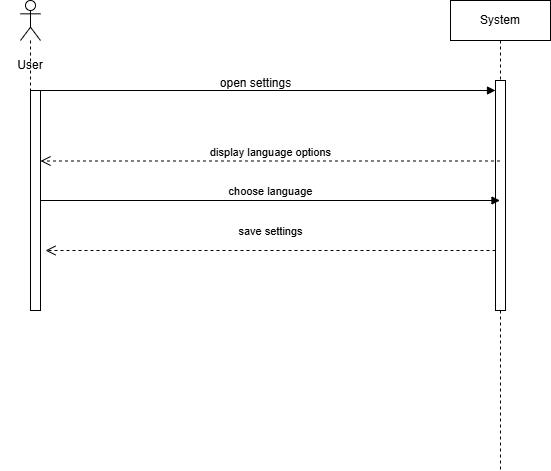
### 2.6.10 Recieve Message



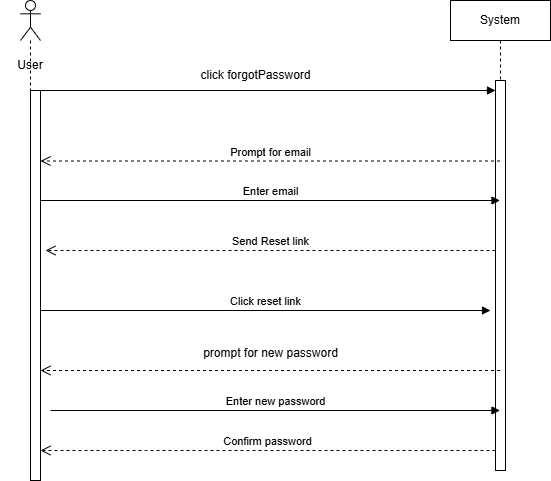
### 2.6.11 Send Message



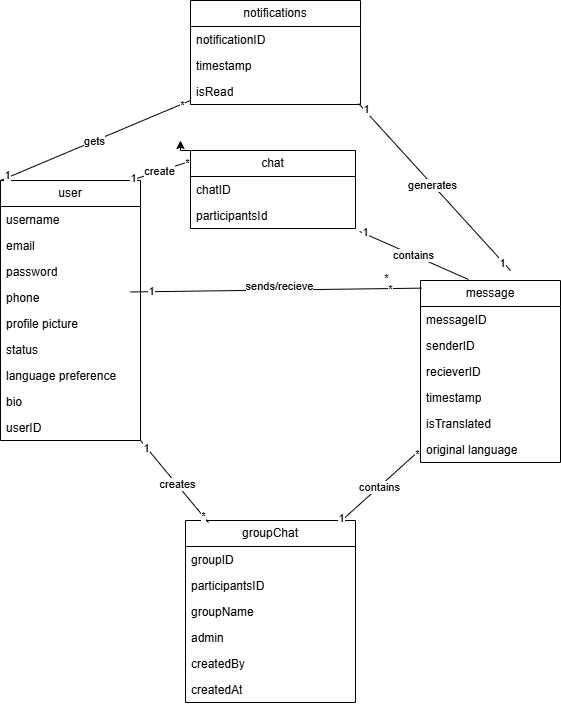
### 2.6.12 Set Language



### 2.6.13 Forgot Password



## 2.7 Domain Model



# 

# Chapter 3

## Software Design Description

**3.1.** **Introduction:**

The Software Design Description (SDD) serves as a means of communicating the design information of a software system to its stakeholders. It outlines the structure of the software and how it will fulfill the requirements. The SDD typically includes an interface design, class diagram, and a description of how the software will meet the requirements. It is an important document in the software development process and serves as a reference for developers throughout the project.

**3.2.** **Purpose:**

The Software Design Document aims to outline the design of the multilingual chat system and provide guidance for software development. This document will establish what needs to be developed and how it should be developed. The Design Document will provide detailed descriptions of the software and system to be built.

**3.3.** **Design Overview:**

The Software Design Document presents a comprehensive, detailed description of the system's low-level structure and design for each of its components.

**3.4**  **Chosen System Architecture:**

#### 3-tier architecture

Multilingual Web Chat System will be developed using 3-tier architecture. It has 3 different layers:

#### Presentation layer

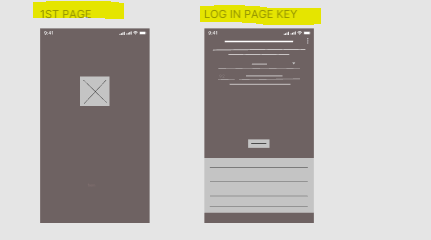
This layer is responsible for presenting user interfaces to different users.

#### Application layer

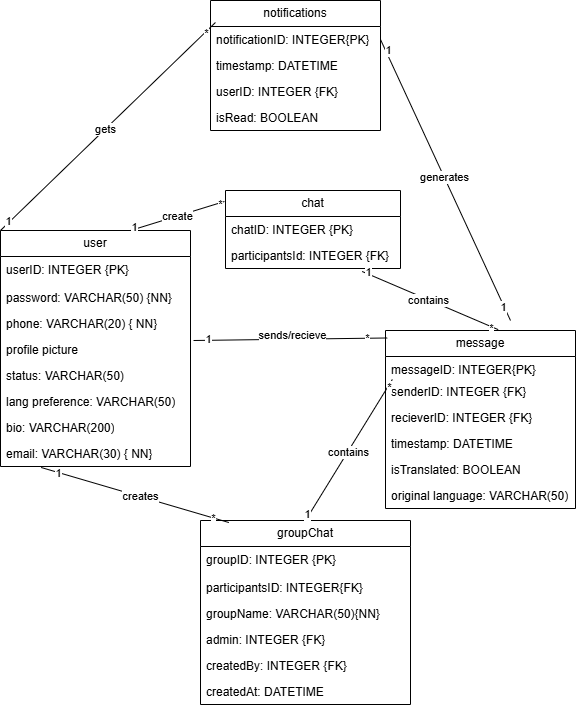
It contains business logic and this layer processes requests of users from presentation layer.

#### Data layer

This layer is used to store and manage data of the system.

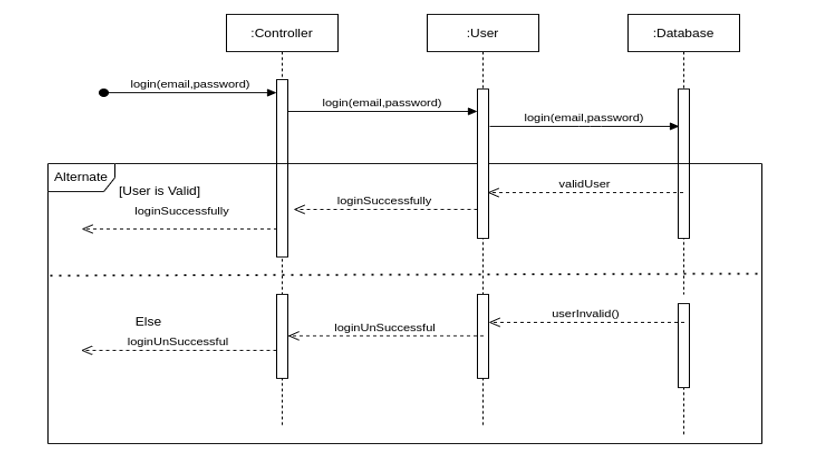
**.5. System Interface Wireframes**

**3.6. Database Design : ERD**

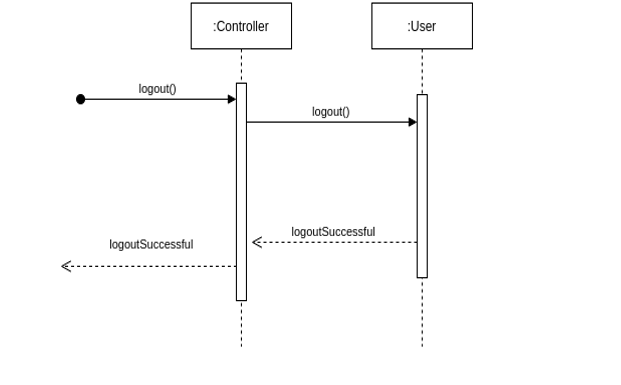
****

**3.7. Sequence Diagrams**

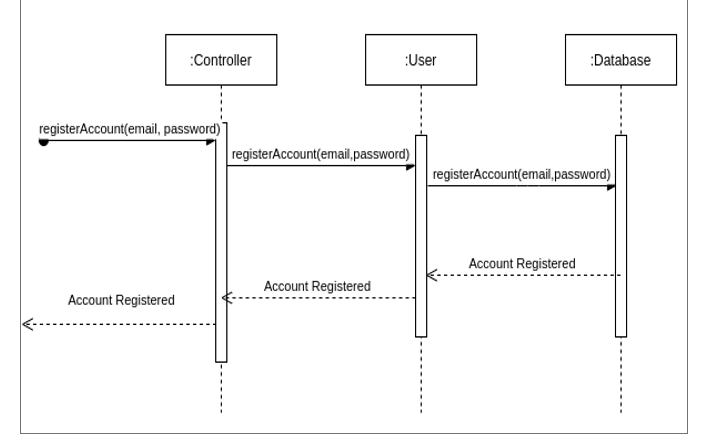
#### 3.7.1 Login

****

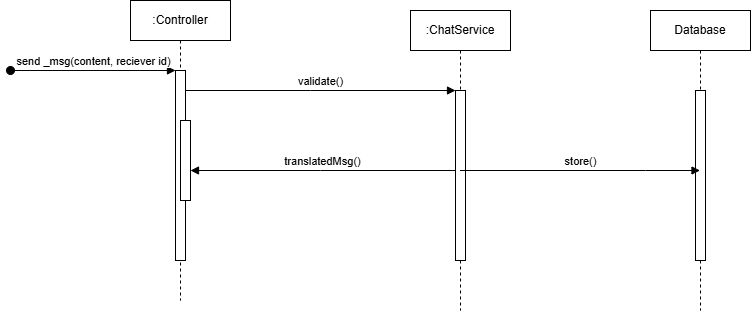
#### 3.7.2 Logout



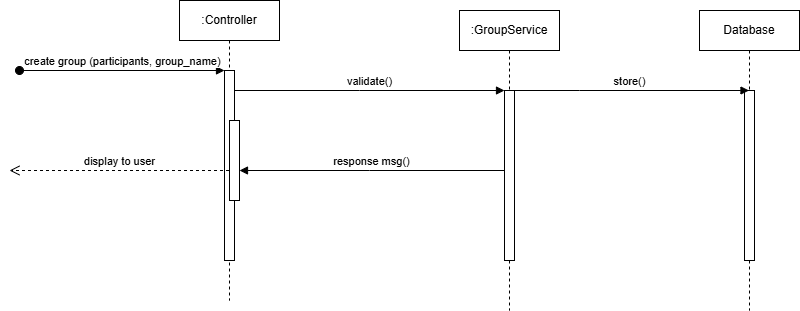
#### 3.7.3 signup

****

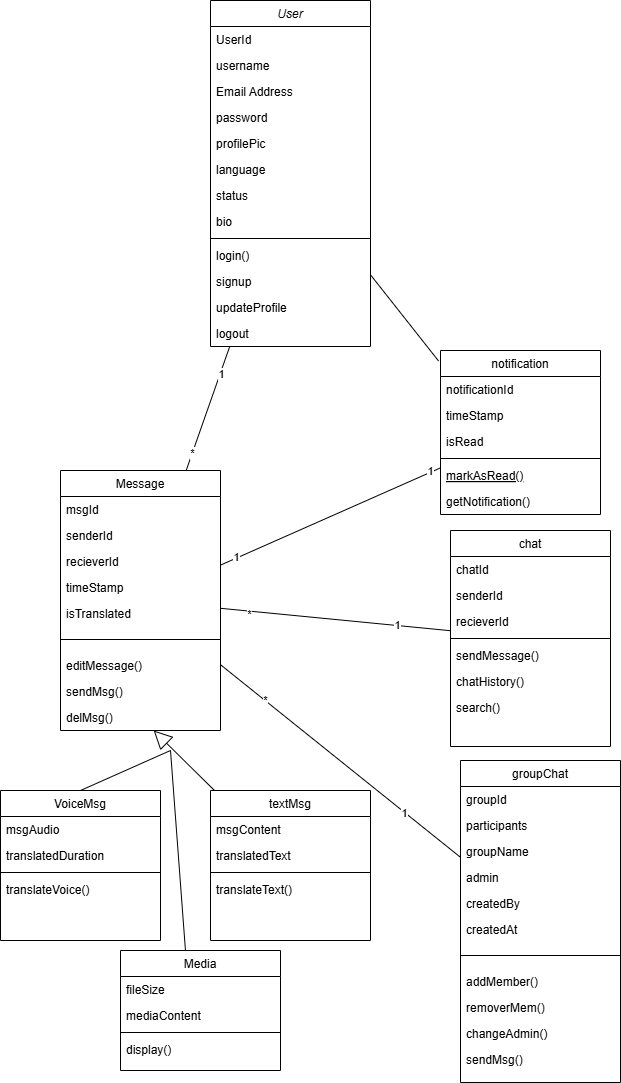
#### 3.7.4 Send Message



#### 3.7.5 Create Group



**3.8 Class Diagram**

****

# Chapter 4

# Implementation and Testing

## 

## 

## 4.1 Implementation:

### 1. Frontend Development with React.js

The frontend will be built using **React.js**, a popular JavaScript library known for creating dynamic and user-friendly interfaces. React’s component-based structure makes it easy to manage UI elements, ensuring a smooth and responsive user experience. It also has a huge ecosystem of tools and libraries that speed up development.

### 2. Backend Development with Node.js and Express.js

For the backend, **Node.js** will be used along with **Express.js**, a lightweight and flexible framework. Node.js is great for handling real-time data and high-performance applications, making it a perfect choice. Express.js simplifies backend development with easy-to-use routing, middleware, and authentication support.

### 3. Database Management with MongoDB

The application will store data in **MongoDB**, a flexible and scalable NoSQL database. Since MongoDB uses a document-based structure, it makes handling data much easier, especially for applications with dynamic content. Mongoose, an Object Data Modeling (ODM) library, will be used to manage database operations efficiently.

### 4. Version Control with Git and GitHub

To keep track of code changes and allow smooth collaboration, **Git** will be used for version control. The project will be hosted on **GitHub**, making it easy for the team to work together, review code, and manage different versions without losing track of progress.

### 5. API and Libraries Integration:

* **WebSockets (Socket.io):** Enables real-time communication for instant messaging.
* **Google Translation API:** Supports multilingual message translation.
* **Twilio API:** Allows voice messaging functionality.

## 4.2. Testing

### 4.2.1 Introduction

Software testing ensures that Multilingual Web Chat System is error-free, functions as expected, and delivers a smooth user experience. It plays a crucial role in verifying and validating the system’s reliability. Testing will be performed at every stage of development to maintain high quality.

### 4.2.2 Test Strategy

The testing approach focuses on identifying and resolving issues early to minimize risks. It includes:

* Unit Testing: Testing individual components like messaging and profile updates.
* Integration Testing: Ensuring smooth interaction between frontend, backend, and database.
* User Acceptance Testing (UAT): Validating that the system meets user needs before deployment.

### 4.2.3 Features to be Tested

The following features will be tested to ensure proper functionality:

1. Login
2. Social Login (use relationship)
3. Signup
4. Forgot Password
5. Send Text Message
6. Receive Text Message
7. Translate Text Message
8. Send Voice Message
9. Receive Voice Message
10. Translate Voice Message
11. View History
12. Search by Keyword
13. Create Group Chat
14. Add/Remove Members
15. Send Notifications
16. Customize Profile
17. View Profile
18. Logout

### 4.3. Test Cases

#### Test Case TC-1: Login

|  |  |
| --- | --- |
| Description | To verify that system can log in the user with valid credentials. |
| Setup | User is already registered. |
| Input | Username: aymenabbasi ; Password: xyZ12zYx |
| Instructions | Enter username.  Enter password.  Click on ‘login’ button. |
| Expected Result | System logs in. |
| Actual Result |  |

#### Test Case TC-2: Customize User Profile

|  |  |
| --- | --- |
| Description | To verify that users can update their profile information. |
| Setup | User is logged into the system. |
| Input | Updated profile details (e.g., name, profile picture, bio) |
| Instructions | Navigate to the profile page.  Edit the necessary fields.  Click on ‘Save Changes’ button. |
| Expected Result | Profile updates are saved successfully. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-3: Send Text Message

|  |  |
| --- | --- |
| Description | To verify that user can send a text message successfully. |
| Setup | User is logged into the system and has an active chat session. |
| Input | Text message input |
| Instructions | Open chat.  Type a message.  Click on ‘Send’ button. |
| Expected Result | Message is delivered successfully. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-4: Create Group

|  |  |
| --- | --- |
| Description | To verify that a user can create a new group chat. |
| Setup | User is logged into the system. |
| Input | Group name and selected members |
| Instructions | Navigate to the group chat section.  Click on ‘Create Group’.  Enter a group name and select members.  Click on ‘Create’ button. |
| Expected Result | Group is created successfully. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-5: Send Notifications

|  |  |
| --- | --- |
| Description | To verify that system sends notifications successfully. |
| Setup | User has notifications enabled. |
| Input | New message or system event |
| Instructions | Trigger a notification event.  Wait for the notification to appear. |
| Expected Result | Notification appears successfully. |
| Actual Result | . |
| Verdict |  |

#### Test Case TC-6: Receive Message

|  |  |
| --- | --- |
| Description | To verify that user can receive messages successfully. |
| Setup | User has an active chat session. |
| Input | Incoming message from another user |
| Instructions | Wait for a message to be received.  Check if the message appears in the chat. |
| Expected Result | Message is received successfully. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-7: Send Voice Message

|  |  |
| --- | --- |
| Description | To verify that user can send a voice message. |
| Setup | User is logged into the system and has microphone access. |
| Input | Voice recording |
| Instructions | Open chat window.  Press and hold the microphone button.  Record a voice message and release the button.  Click ‘Send’. |
| Expected Result | Voice message is sent successfully. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-8: View Profile

|  |  |
| --- | --- |
| Description | To verify that a user can view a profile. |
| Setup | User is logged into the system. |
| Input | Profile selection |
| Instructions | Navigate to a user’s profile.  Click on the profile to view details. |
| Expected Result | Profile details are displayed. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-9: Search by Keyword

|  |  |
| --- | --- |
| Description | To verify that users can search messages by keyword. |
| Setup | User is logged into the system. |
| Input | Search query |
| Instructions | Open the search function.  Enter a keyword.  Click on ‘Search’. |
| Expected Result | Matching messages are displayed. |
| Actual Result | . |
| Verdict |  |

#### Test Case TC-10: View History

|  |  |
| --- | --- |
| Description | To verify that users can view chat history. |
| Setup | User is logged into the system and has existing chats. |
| Input | Chat selection |
| Instructions | Open a chat window.  Scroll up to load previous messages. |
| Expected Result | Chat history is displayed. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-11: Set Language

|  |  |
| --- | --- |
| Description | To verify that user can change the system language. |
| Setup | User is logged into the system. |
| Input | Language selection |
| Instructions | Go to settings.  Select a new language.  Save changes. |
| Expected Result | System language is updated. |
| Actual Result |  |
| Verdict |  |

#### Test Case TC-12: Logout

|  |  |
| --- | --- |
| Description | To verify that users can log out successfully. |
| Setup | User is logged into the system. |
| Input | Logout button |
| Instructions | Click on the logout button. |
| Expected Result | User is logged out successfully. |
| Actual Result | . |
| Verdict |  |

# 

# 

# 

# Chapter 5

# Conclusion and Future Work

### 

## 5.1 Conclusion

The developed system successfully enables seamless user interactions, allowing for secure communication through messages, group chats, and notifications. By implementing a structured and efficient architecture, the system ensures data integrity, user authentication, and real-time messaging. The testing phase confirmed the system’s functionality, usability, and performance. Overall, the project achieves its objective of providing an interactive and user-friendly messaging platform.

## 5.2 Future Enhancements

While the system meets its current requirements, there is always room for improvement and expansion. Some potential future enhancements include:

1. **End-to-End Encryption** – Implementing advanced security measures to enhance user privacy and data protection.
2. **Voice and Video Calls** – Adding real-time communication features for a more interactive user experience.
3. **AI-Based Chatbot Integration** – Introducing an AI assistant for quick replies and automated responses.
4. **Multi-Language Support** – Expanding language capabilities to cater to a more diverse user base.

## 

## 

## References

1. Sommerville, I. (2015). *Software Engineering* (10th ed.). Pearson.
2. IEEE. (1998). *IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)*.
3. Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
4. Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer.
5. Acceptance Testing. (2011). Retrieved from:<http://softwaretestingfundamentals.com/acceptance-testing/>