



**Bilkent University**  
**Department of Computer Engineering**

**Senior Design Project**  
*T2314*  
*Linguist AI*

## **Analysis and Requirement Report**

22001943, Yağız Can Aslan, [can.aslan@ug.bilkent.edu.tr](mailto:can.aslan@ug.bilkent.edu.tr)  
22003017, Kardelen Ceren, [kardelen.ceren@ug.bilkent.edu.tr](mailto:kardelen.ceren@ug.bilkent.edu.tr)  
22002811, Selim Can Güler, [selim.guler@ug.bilkent.edu.tr](mailto:selim.guler@ug.bilkent.edu.tr)  
22002223, İlkim Elif Kervan, [elif.kervan@ug.bilkent.edu.tr](mailto:elif.kervan@ug.bilkent.edu.tr)  
22003850, Tolga Özgün, [tolga.ozgun@ug.bilkent.edu.tr](mailto:tolga.ozgun@ug.bilkent.edu.tr)

*Supervisor: Halil Altay Güvenir*  
*Course Instructors*  
*Atakan Erdem*  
*Mert Bıçakçı*

08.12.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

# Table of Contents

<b>1 Introduction.....</b>	<b>3</b>
<b>2 Current System .....</b>	<b>4</b>
<b>3 Proposed System.....</b>	<b>5</b>
<b>3.1 Overview .....</b>	<b>5</b>
<b>3.2 Functional Requirements .....</b>	<b>6</b>
3.2.1 User .....	6
3.2.2 Chatbot .....	6
3.2.3 Contextual Learning System .....	7
3.2.4 Admin.....	7
<b>3.3 Non-functional Requirements .....</b>	<b>8</b>
3.3.1 Usability .....	8
3.3.2 Reliability.....	8
3.3.3 Performance.....	9
3.3.4 Supportability .....	9
3.3.5 Scalability .....	10
3.3.6 Security & Privacy .....	10
<b>3.4 Pseudo Requirements .....</b>	<b>11</b>
<b>3.5 System Models .....</b>	<b>12</b>
<b>3.5.1 Scenarios .....</b>	<b>12</b>
<b>3.5.2 Use-Case Model.....</b>	<b>18</b>
3.5.2.1 Conversation UML Use Case Diagram .....	18
3.5.2.2 Unknown Word Bank UML Use Case Diagram .....	19
3.5.2.3 Gamification UML Use Case Diagram .....	20
<b>3.5.3 Object and Class Model .....</b>	<b>21</b>
<b>3.5.4 Dynamic Models .....</b>	<b>22</b>
3.5.4.1 Unknown Word Bank.....	22
3.5.4.1.1 Unknown Word Bank UML Activity Diagrams .....	22
3.5.4.1.2 Unknown Word Bank UML State Diagrams .....	23
3.5.4.2 Conversation.....	24
3.5.4.2.1 Conversation UML Activity Diagrams.....	24
3.5.4.2.2 Conversation UML State Diagrams .....	24
3.5.4.3 Gamification.....	25
3.5.4.3.1 Gamification UML Activity Diagrams.....	25
3.5.4.3.2 Gamification UML State Diagrams.....	26
3.5.5 User Interface.....	27
3.5.5.1 Implemented Screens .....	27
3.5.5.2 Screens on Figma .....	39
<b>4 Other Analysis Elements .....</b>	<b>40</b>
<b>4.1 Consideration of Various Factors in Engineering Design .....</b>	<b>40</b>
4.1.1 Constraints .....	40
4.1.2 Standards.....	42
<b>4.2 Risks and Alternatives .....</b>	<b>43</b>
<b>4.3 Project Plan.....</b>	<b>45</b>
<b>4.3.1 Gantt Chart.....</b>	<b>55</b>
<b>4.4 Ensuring Proper Teamwork .....</b>	<b>56</b>
<b>4.5 Ethics and Professional Responsibilities.....</b>	<b>63</b>
4.5.1 Data Privacy.....	63
4.5.2. Responsible Chatbot Responses .....	63
4.5.3. Ethical Topic Filtering .....	63
4.5.4. Professional Responsibilities .....	63
4.5.5. Making Sure the Promised Methods Work .....	64
<b>4.6 Planning for New Knowledge and Learning Strategies .....</b>	<b>64</b>
4.6.1 Technical Areas.....	64
4.6.2 Non-technical areas.....	65
<b>5 Glossary .....</b>	<b>65</b>
<b>6 References .....</b>	<b>66</b>

# Analysis and Requirement Report

*T2314: Linguist AI*

## 1 Introduction

English is one of the most spoken languages in the world, with millions of native and non-native speakers [1]. It has become the default language of international events, businesses, technology, etc. Therefore, many people aim to learn English. For instance, in the Turkish education system, students are taught English as a second language for ten years starting from elementary school. That said, most students cannot speak English properly due to the ineffective traditional teaching methods, including grammar and vocabulary memorization using textbooks and flashcards. Students cannot incorporate the language into their everyday lives using the aforementioned methods and, therefore, forget the memorized rules of the language. Also, there is a lack of resources when it comes to writing and speaking materials, as the main focus of language teaching is based on grammar and vocabulary. However, language is not something to be memorized, it is a culture that should be experienced.

In addition to the traditional language learning techniques, there are applications that students can use to learn English, like Cambly and Duolingo. Duolingo is a mobile application that offers lessons for various skill levels [2]. These lessons aim to teach grammar and vocabulary using different activities such as filling in the blanks, translating sentences, speaking, and listening. On the other hand, Cambly is an online platform designed to connect students with native speakers [3]. While using Cambly, students have a chance to interact with a native speaker during the classes. However, the lessons in Duolingo are not customized to individuals and might get boring after a while. Also, they are designed for beginner and intermediate learners. Therefore, someone who wants to speak more advanced or learn academic vocabulary cannot use Duolingo. As for Cambly, it allows students to experience the language culture. However, it might not be affordable for all students. Also, students with self-esteem problems and shyness would be uncomfortable using this app.

LinguistAI is an online mobile application where users can learn vocabulary and practice their language skills using a chatbot. Our application detects the skill level of the user during the chat sessions and adapts to the user's level. Also, the application determines if the user is correctly using a vocabulary to make sure the user is actively learning. Moreover, it personalizes the conversations based on a user model that includes the user's interests, demography and more. Users can have a conversation with the bot about a topic or scenario they like such as buying a coffee or plane ticket. Our application will include gamification elements such as achievements, leaderboards, and daily streaks. The main focus of Linguist AI is to teach words to users by repeatedly using unknown words for a user in different contexts and measuring the level of knowledge of the user for each word. All in all, Linguist AI will enable users to learn English tailored to their level by eliminating artificial learning items such as flashcards and textbooks and offering a sincere conversational environment.

The rest of this report will describe the current system of Linguist AI, the proposed system's functional and non-functional requirements, and system models. Then it will explain engineering principles taken into account, possible risks and alternative plans for the risks, and the project plan.

## **2 Current System**

Currently, learners practice their English vocabulary, speaking, and writing skills either in classes or using websites and mobile apps. Classes are held either in traditional schools, including middle schools, high schools, and universities, or in language learning schools, such as Wall Street English, Berlitz, and The British Council, either face-to-face or online. While they offer professional assistance, their lessons are costly, their classes are only available a limited amount of times per week, and they do not offer personalized teaching as standardized curriculums have a one-size-fits-all approach. Private tutors such as those on websites like Cambly and Italki, on the other hand, can adapt to a student's needs, but they also suffer from being expensive and not always being available. Finally, language exchanges, where partners teach each other their native languages, while free, may have timezone conflicts, are inadequate for the less demanded languages, and may prove frustrating for the partner who is more proficient in the foreign language than the other.

Self-learning websites and mobile apps, such as Duolingo, Babbel, Busuu, and Mondly, tend to offer bite-sized lessons on vocabulary or grammar, sometimes with example dialogues. Companies including Rosetta Stone, Memrise, or opensource Anki, feature flashcards to learn vocabulary, supported by audio. Most of these apps stand out with their gamification features to motivate learners to study. While these apps are helpful when it comes to vocabulary drilling, they lack features for learners to practice speaking or writing on diverse topics.

Certain apps, like Memrise and Duolingo, published features where the user can speak with a chatbot based on an LLM (Large Language Model) as recently as December 2022, and March 2023 respectively, with Mondly announcing a chatbot with a human-like appearance in October 2023. Using AI for language learning is still in its infancy, and therefore these apps currently only provide chatting with the AI for pre-determined topics, such as ordering food where the chatbot pretends to be a waiter. However, certain competitors, like Talkpal AI or character.ai offer free reign with their chatbots, proving that such an idea is possible to implement. Moreover, studies conducted using assistant chatbots akin to Siri show that speaking with them with the intent of learning a language does improve foreign language skills [4, 5, 6], while motivating and increasing the confidence of learners as they do not fear making mistakes as they would with a human partner [7, 8].

### 3 Proposed System

In the following subsections, our proposed system with additional features to the current system is explained in terms of its functional and non-functional requirements. System diagrams such as use case, activity, and state for core functionalities are listed as well.

#### 3.1 Overview

LinguistAI will be a mobile application aimed to help English as a foreign language learners practice their vocabulary, writing, and speaking skills through chatting via text or speech with an LLM-based chatbot. Target users will be expected to have at least A2 level proficiency in English to ensure the chats can be meaningful.

The chatbot will be able to detect the user's skill level by inspecting their vocabulary and grammar in order to adapt their conversation accordingly. When users click on a word, they will be able to see its definition, along with example sentences. If desired, they will be able to save the words they do not know in a list, or they will add words to user-created lists manually. The chatbot will then integrate the words from these lists into future conversations, placing them in diverse contexts. This approach allows users to familiarize themselves with unknown vocabulary by encountering it in various real-life situations. In addition, users will be able to speak to the chatbot and listen to its responses in order to improve their speaking and listening skills.

Furthermore, vocabulary will be tested at the start and at the end of the conversation with the chatbot via multiple-choice questions to gauge how much the user learned through interacting with the AI. The questions, along with how well the user uses various vocabulary in their own messages will be used to determine whether the words are learned.

Finally, we will add gamification to our application, following the tested and true methods of our competitors. We will incorporate engaging and compelling gamification concepts, including login/chat streaks, experience points, chatbot customization, leaderboards, a friendship system, achievements, and daily quests. These elements aim to inspire users and enhance their motivation for studying. Similarly, while chatting, the bot will create a user profile, including the user's demographic data, their likes and dislikes, and suggest new topics that might interest the user. The standout feature of LinguistAI is its ability to engage with users in a conversational format, allowing it to seamlessly track and maintain the flow of conversation for continued responses.

## **3.2 Functional Requirements**

### **3.2.1 User**

- The user can register/login by using their email.
- The user can reset their password using their email.
- The user can have a conversation with the chatbot by texting with the chatbot.
- The user can also choose to have a conversation with the chatbot by speaking.
- The user can listen to the pronunciation of the chatbot's texts.
- During their conversation with the chatbot, the user will be able to select a word they do not know and add it to their unknown word storage.
- The user can see the meaning of the words they select, along with an example sentence including the word.
- The user can see the list of unknown words they saved so far, with their meanings and example sentences available on click.
- The user will be presented with weekly tests to determine how well they know unknown vocabulary.
- The user can access their profile to see and edit their personal information.
- The user can see other users' profiles with their basic information and follow them.
- The user can view the leaderboard to compare themselves with their friends in terms of the number of words they learned in a particular time span.
- The user can be notified on a daily basis to remind them of their daily practice.
- The user can be notified of the progress of the other users they follow to encourage them to use the application.
- The user can see their daily streak on a pop-up when they initiate a chat with the bot for the first time in the day.
- The user can win awards and badges based on their learning statistics that are based on time spent on the app, number of words learned, language level-ups, etc.
- The user will be presented with predefined role-play scenarios as prompts.
- The users will be able to opt-out of model personalization using the settings.

### **3.2.2 Chatbot**

- The chatbot will initiate the conversation when the user opens up the chat screen.
- The chatbot can adapt its texts to be similar to the user's proficiency level.
- The chatbot will use words from the unknown word list more frequently.
- The chatbot will be given a summary of chat history and will adapt to it for further responses.
- The chatbot will be given a user model including information about demography, interests, occupation, etc.

### **3.2.3 Contextual Learning System**

- The user's proficiency level can be measured by the system based on the recorded unknown and known words for the particular user.

- When the chatbot receives a sentence which includes a word that is unknown to the user, the system will evaluate the well-usedness of that particular word to update the confidence of the user knowing the word.
- When the chatbot uses a word that is unknown to the user, the system will evaluate the answer coming from the user to see whether the answer makes sense to update the confidence level of the user knowing the unknown word.
- The system will analyze each message sent by the user and save the user's level of familiarity of each word which will be used by the machine learning model.

### **3.2.4 Admin**

- The admin will be able to ban the users reported by the bot. The bot will filter user messages. If users send malicious texts that include sexist, racist, discriminating, or hateful phrases, the bot filter will detect these and report them to admins. Upon careful review the admins can decide to ban the reported users.
- Admins will be able to generate statistical reports and view them in their dashboard. These reports can be the number of active users, most learned words, and minutes spent having a conversation with the bot.

### **3.3 Non-functional Requirements**

Requirements and constraints on external interfaces, performance, dependability, maintainability, reusability, security, etc. : Requirements and constraints on external interfaces, performance, dependability, maintainability, reusability, security, etc.

This section mentions the non-functional requirements under Usability, Reliability, Performance, Supportability, Scalability and Security.

#### **3.3.1 Usability**

Linguist AI will be a mobile application. Since Linguist AI will be a mobile application, the users will be able to use our application anywhere which can include while commuting on the bus, when waiting in line, while waiting for food, and many more examples. As such, it is vital that the application is accessible and easy to use. The users must be able to utilize our application without struggle. Therefore, the application must have an intuitive and simple user interface, ensure accessibility for all types of users (different types of color blind etc.), have a responsive design, maintain optimal performance, include methods for gathering user feedback and error reporting, be compatible across a variety of platforms and maintain a consistent design. The application also should have an easy-to-follow and informative user onboarding procedure upon first use, to maintain user retention and make the application easier to understand and use. In order to adhere to the aforementioned usability guidelines for our project, the users must be able to reach all parts of the application with at most 5 clicks. For ease of use, the navigation bar should be visible on all screens, and the back buttons must always be in the same place (for instance, top left). We have defined the following constraints to measure usability:

- Each main subsystem, such as chat and leaderboard, should be accessible within 2 clicks for every logged-in user.
- Each login step should take 4 steps at maximum with valid credentials.
- All back buttons should be placed on the top left of the screen, with 16 eDPI for left and top margins.
- Each main subsystem should have a visible bottom navigation.
- All network requests should be asynchronous so that users can interact with the front end while the backend response is being generated.
- All in-progress network requests should be clearly indicated to the user with a loading visual placed as the closest element to the target component.

#### **3.3.2 Reliability**

In order for our application to achieve its goals, the application must provide a consistent experience. All systems of our application (backend servers, databases, LLM integration, mobile application etc.) must cooperate smoothly without major errors. To provide a smooth experience to our users, all of our systems should have a 95% uptime. The user must be able to continuously use the application without encountering a major error in any step. Moreover, the responses of our conversational AI models must not be strange or unsettling, the users must know that whenever they use our application, they will be subject to the same experience.



### 3.3.3 Performance

As stated in section 2.2.1, our users can use our application almost anywhere. As such, it is important for our application to be responsive. For this reason, the performance of the application will be measured through various metrics such as time to load (TTL), mean time to respond (MTTR) and time to first byte (TTFB). An upper bound is declared for all of these metrics in various contexts. The constraints considered for such metrics are as follows:

- The time to the first byte of the chat system should be strictly under 10 seconds. This also applies to a potential version of the chat which uses text streaming via sockets.
- Mean response time of the authentication service should be less than 5 seconds.
- Average time to load a page on the mobile application should be less than 3 seconds.
- Adding new words to the word list should take less than 2 seconds.
- Loading user profiles should take less than 4 seconds.
- Mean response time of the main subsystems besides chat and authentication should be less than 4 seconds.

If these conditions are not met, the users will have to wait a considerable amount of time to utilize our features which will decrease our application's usage and language teaching effectiveness significantly. With these metrics in mind, our expectation is having a UI that is performant and responsive, so the application does not feel clunky, slow or old, and the connections between the subsystems of our project such as backend servers, databases, LLM integration and mobile application happening relatively fast to enable the aforementioned smooth and responsive experiences.

### 3.3.4 Supportability

In order to reach a broader user base, our mobile application must be available for a variety of platforms, including Android and iOS. As such, we must maintain the application for different operating systems and mobile devices. Our application must be able to be maintained and expanded upon easily. This can be addressed by following good software development practices and adhering to tested and established design paradigms such as Object Oriented Programming. In order to adhere to these values, we established the following constraints:

- The minimum Android SDK version should be 24.0 which encompasses more than 98% of the active Android user base, and is the recommended minimum by Google.
- The minimum iOS version should be 15 which is the version recommended by Apple. 94% of users use it.

### **3.3.5 Scalability**

Since our application is a mobile application, it will be easy for new users to download and start using our application. Therefore, our systems (backend servers, databases, LLM integration, mobile application etc.) must be prepared and ready to handle a surge in users and server loads. Our system must be able to handle at least 1000 concurrent users. We must also develop our systems in such a manner that migration, adding new features or distributing our application to new vendors (for mobile applications, like App Store, Google Play Store etc.) will not cause significant problems.

### **3.3.6 Security & Privacy**

Linguist AI will store user data to provide a specialized learning experience. The stored data may include names, interests, passwords, emails, conversation history and many more types of information. Henceforth, it is crucial that our systems are as secure as possible to protect sensitive data and that we handle such data with extreme care. To achieve this, we can utilize complex encryption algorithms and various other tried and tested security protocols. Moreover, we will follow GDPR (General Data Protection Regulation) and KVKK (Kişisel Verilerin Korunması Kanunu) guidelines closely and will seek user consent whenever necessary, to strike a balance between personalization and privacy. We are implementing the following procedures for our security purposes:

- All passwords must be hashed with SHA-256 and salted.
- All sessions must be given a session token by JWT and all requests should have a valid access token.
- All users should be provided with a refresh token to regenerate the access token when expired.
- Access token should be valid for 15 minutes.
- Refresh token should be valid for 24 hours.

### 3.4 Pseudo Requirements

1. Github and Jira will be used to maintain the development process of our project.
2. React Native will be used as a framework to develop our application for the most common mobile operating systems, such as iOS and Android, with a single codebase.
3. Expo will be used to maintain the project architecture for the client side.
4. React Native Platform will be used to create responsive layouts for different device sizes and to create layouts that are adaptive to landscape and portrait modes.
5. React Navigation will be used to create Stack-based routing and bottom tab routing and navigation for the entire application.
6. Axios will be used to send requests over the Internet and intercept them to prevent unwanted behavior when needed.
7. Expo ImagePicker will be used when users want to add a profile picture to their profiles.
8. Expo Notification will be used to push notifications to the customer.
9. Backend will be developed using Spring Boot with Java.
10. Slf4j will be used for logging server activity.
11. Relational MySQL database will be the main point of storing data.
12. Amazon AWS S3 will be used as a secondary database for non-relational data such as files.
13. Java Persistence API (JPA) and Hibernate ORM will be used to ease database CRUD operations.
14. The backend will be deployed to a Virtual Desktop Server serving Ubuntu hosted at GoDaddy.
15. Pre-trained Large Language Models (LLMs) such as Llama2, GPT-4, Gemini will be fine-tuned with publicly available conversational datasets to fit the context of the application.
16. New machine learning models will be trained with publicly available datasets for tasks such as text filtering, text analysis and text summarization.
17. Large Language Models (LLMs) will be deployed on a remote Ubuntu server, running with respective quantization methods to support CPU processes.
18. LLM services will be served leveraging microservice architecture, and a Flask application will be connected to an API Gateway.
19. The API Gateway will also be hosted in the same Ubuntu machine and will be run on Spring Boot architecture, handling the redirection of API calls and authentication.

## 3.5 System Models

### 3.5.1 Scenarios

#### Scenario 1: Have a conversation

**Actors:** User

**Entry Conditions:** User taps the new conversation button

**Exit Conditions:** User closes the app

or

User taps the back button

or

User taps on any menu button

#### **Flow of Events:**

1. User solves a multiple-choice test based on the activated word bank list(s) and some other control/known words.
2. User's daily chat streak is updated if it's the first conversation of the day.
3. User's text is sent to the filter.
4. Assuming user's text passed the filter, system updates user's profile based on text.
5. System checks for daily quests and achievements based on user's text. If any, shows them on screen.
6. If the user correctly used an unknown word, increase that specific word's knowledge confidence.
7. System generates and shows a response based on user's text, user's profile, user's activated word bank list(s).
8. User sends a text.
9. Steps 2-7 are repeated until the user taps on the end conversation button.
10. User solves a second multiple-choice test that contains the same words as the first test but with different sentences. The word knowledge confidence levels are adjusted accordingly
11. The conversation is marked as completed.

### **Scenario 2: Continue a previous conversation**

**Actors:** User

**Entry Conditions:** User taps an unfinished conversation and sends a text

**Exit Conditions:** User closes the app  
or  
User taps the back button  
or  
User taps on any menu button

**Flow of Events:**

1. Steps 2-10 from Scenario 1 are to be followed.

### **Scenario 3: Chatbot starts a previous conversation**

**Actors:** User

**Entry Conditions:** User's last conversation was 18 hours ago

**Exit Conditions:** User closes the app  
or  
User taps the back button  
or  
User taps on any menu button

**Flow of Events:**

1. System generates a text based on user's profile to initiate the conversation.
2. A notification is sent to the user.
3. User taps on the notification.
4. App opens on the chat screen.
5. User sends a text.
6. Steps 1-10 are followed from Scenario 1.

#### **Scenario 4: User text does not pass the filter**

**Actors:** User

**Entry Conditions:** User sends a text on the chat screen that could not pass the filter on step 3 of scenario 1.

**Exit Conditions:** User closes the app  
or  
User taps the back button  
or  
User taps on any menu button

**Flow of Events:**

1. A warning message is sent.
2. User's text is saved for future reference by the admins.

#### **Scenario 5: Add word on chat to a list**

**Actors:** User

**Entry Conditions:** User is on the chat screen and taps on a word

**Exit Conditions:** User closes the app  
or  
User taps the back button  
or  
User taps on any menu button

**Flow of Events:**

1. A pop-up with word's definition, example sentences, and the last chosen list is shown.
2. Optionally, user chooses a word list from the dropdown list.
3. User taps on the add button.
4. Word is added to the given list.

### **Scenario 6: Create new word list**

**Actors:** User

**Entry Conditions:** User is on the word bank screen

**Exit Conditions:** User closes the app

or

User taps the back button

or

User taps on any menu button

#### **Flow of Events:**

1. User writes a name on the text field.
2. User taps on the add a new word list button.
3. Screen changes to words view, showing the empty new list

### **Scenario 7: Activate word list**

**Actors:** User

**Entry Conditions:** User is on the word bank screen

**Exit Conditions:** User closes the app

or

User taps the back button

or

User taps on any menu button

#### **Flow of Events:**

1. User chooses one or multiple word lists.
2. User taps on the activate button.
3. All the selected lists are activated.

### **Scenario 8: Deactivate word list**

**Actors:** User

**Entry Conditions:** User is on the word bank screen

**Exit Conditions:** User closes the app

or

User taps the back button

or

User taps on any menu button

**Flow of Events:**

1. User chooses one or multiple word lists.
2. User taps on the activate button.
3. All the selected lists are activated.

### **Scenario 9: Add word manually**

**Actors:** User

**Entry Conditions:** User is on the words in word list screen

**Exit Conditions:** User closes the app

or

User taps the back button

or

User taps on any menu button

**Flow of Events:**

1. User enters a word on the text field.
2. User taps add new word button.
3. If the word is a valid English word, it is added to the word list shown on the screen.



### **Scenario 10: Add friend**

**Actors:** Users

**Entry Conditions:** User is on the leaderboard screen

**Exit Conditions:** User closes the app

or

User taps the back button

or

User taps on any menu button

#### **Flow of Events:**

1. User chooses the friends leaderboard screen.
2. User enters a username at the text field.
3. Users with similar usernames are listed.
4. User taps on the add a friend button next to a username.
5. A notification is sent to the second user.
6. Second user taps on the notification or opens the app and taps the leaderboard screen.
7. Second user is shown information about the friend request on a pop-up.
8. If the second user accepts, the users are set as friends.

### 3.5.2 Use-Case Model

#### 3.5.2.1 Conversation UML Use Case Diagram

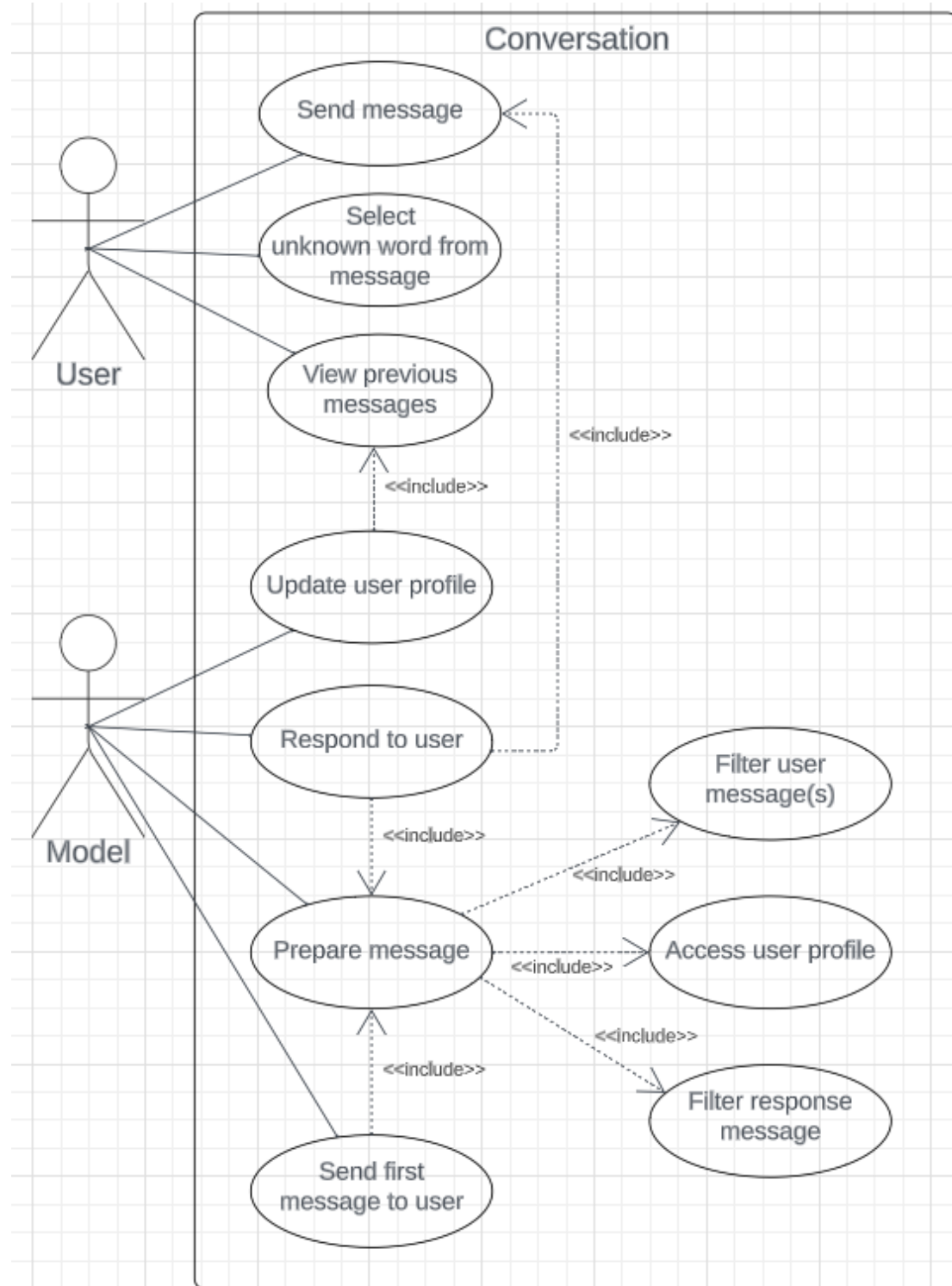


Fig. 1. Conversation UML Use Case Diagram

### 3.5.2.2 Unknown Word Bank UML Use Case Diagram

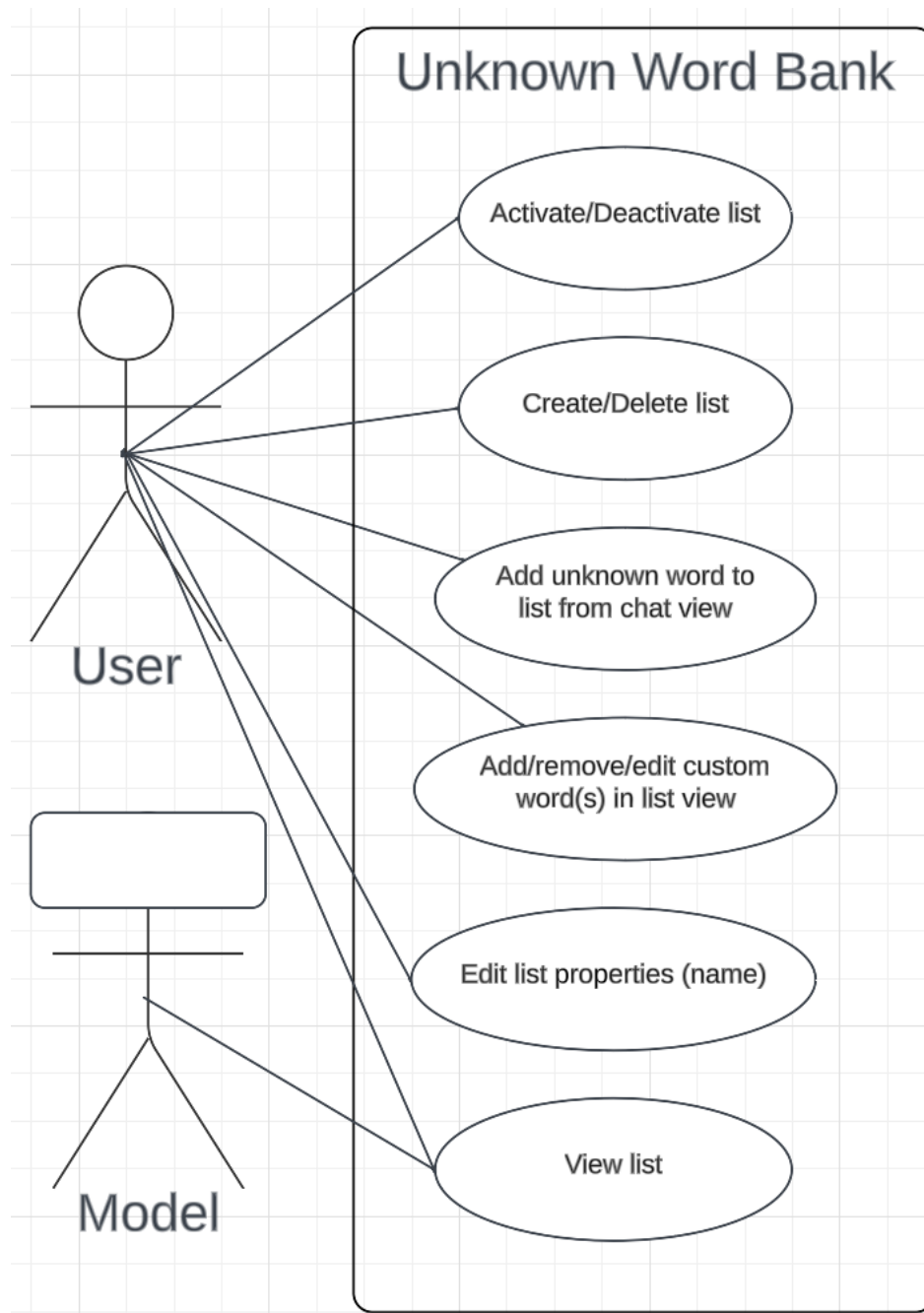


Fig. 2. Unknown Word Bank UML Use Case Diagram

### 3.5.2.3 Gamification UML Use Case Diagram

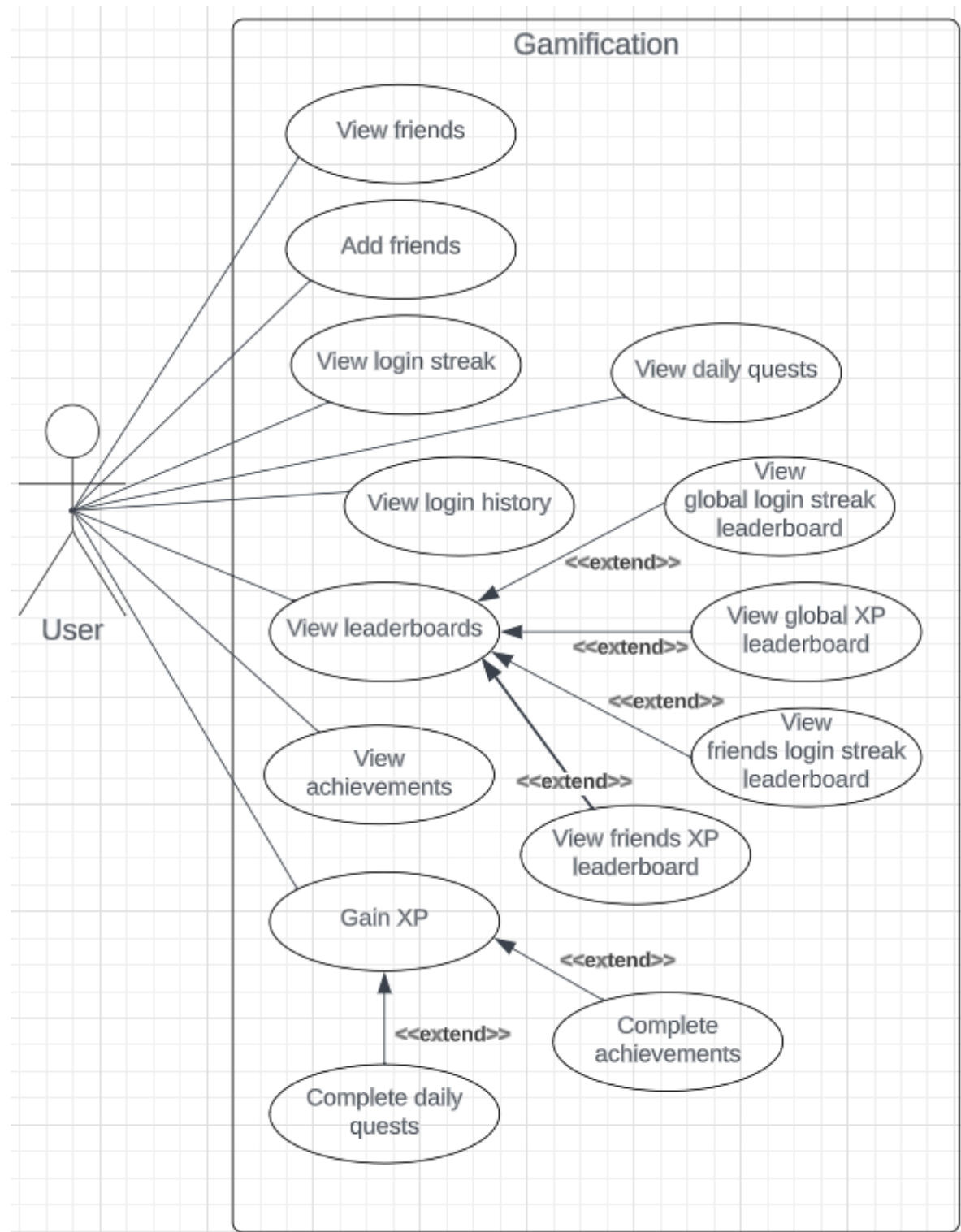


Fig. 3. Gamification UML Use Case Diagram

### 3.5.3 Object and Class Model

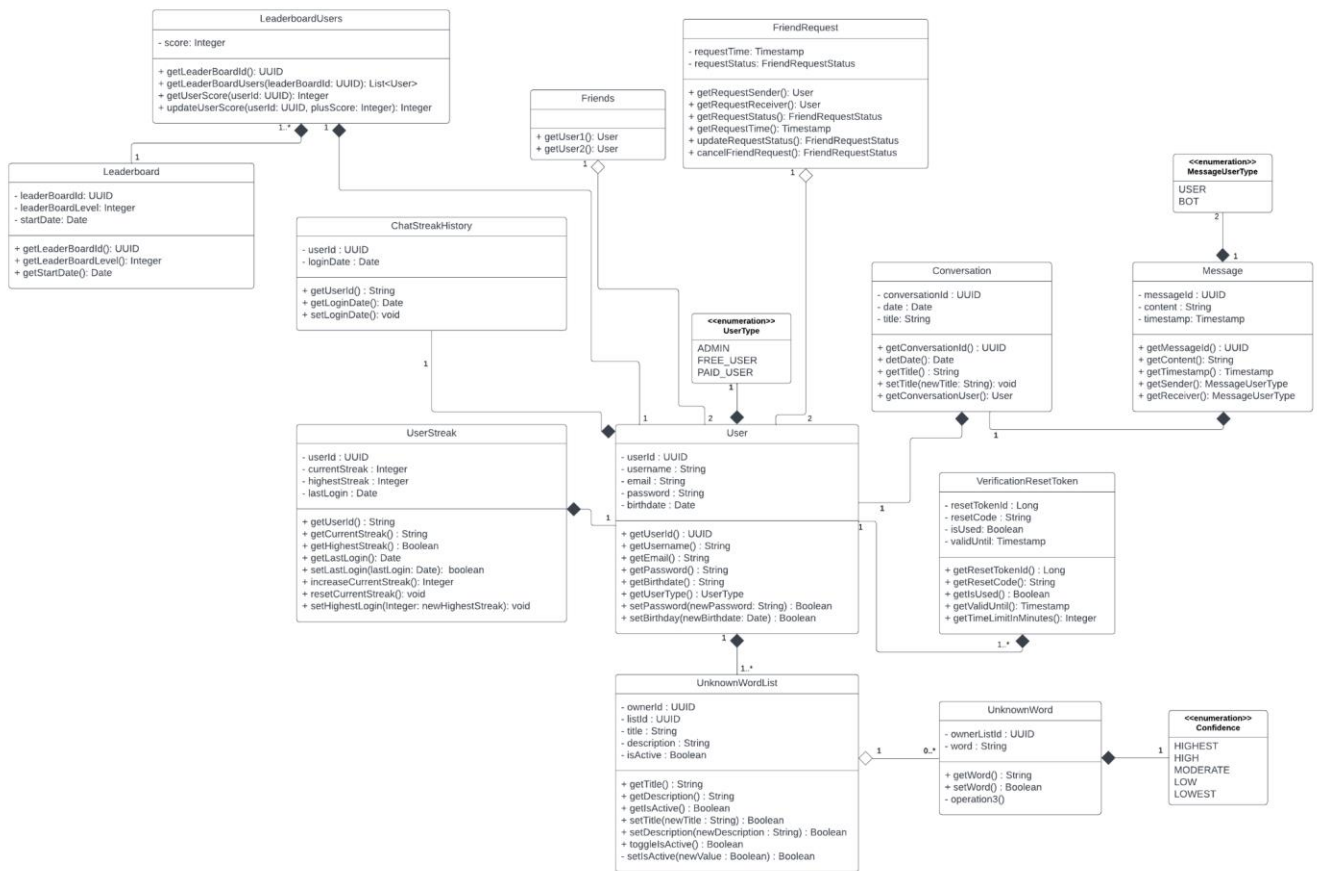


Fig. 4. UML Class Diagram

You can see the high-quality version of this diagram here:

[http://elif.kervan.ug.bilkent.edu.tr/linguistai/class\\_diagram.png](http://elif.kervan.ug.bilkent.edu.tr/linguistai/class_diagram.png)

### 3.5.4 Dynamic Models

#### 3.5.4.1 Unknown Word Bank

This section includes the Activity and State UML diagrams for all “Unknown Word Bank” related features and logic.

##### 3.5.4.1.1 Unknown Word Bank UML Activity Diagrams

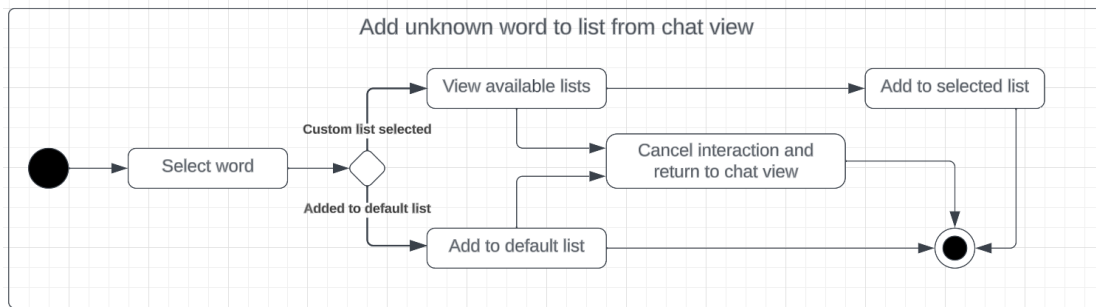


Fig. 5. Add unknown word to list from chat view UML Activity Diagram

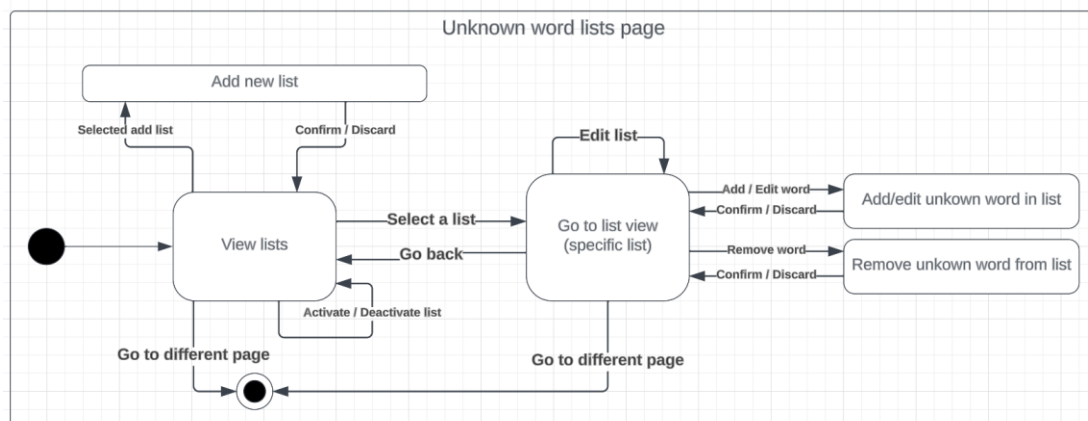


Fig. 6. Unknown word lists page UML Activity Diagram

### 3.5.4.1.2 Unknown Word Bank UML State Diagrams

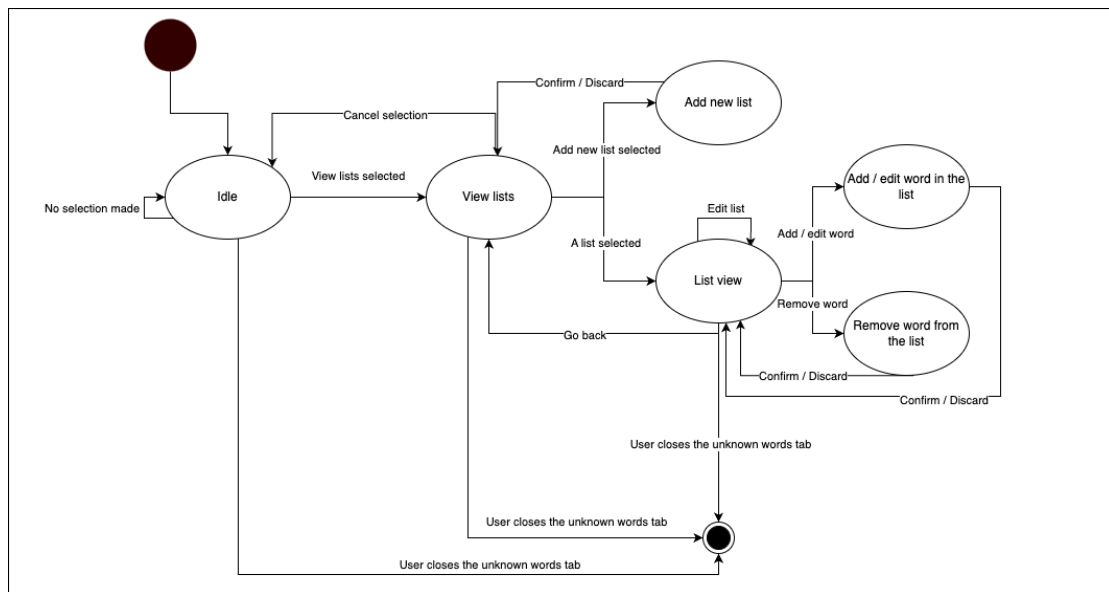


Fig. 7. Add unknown word to list from chat view UML State Diagram

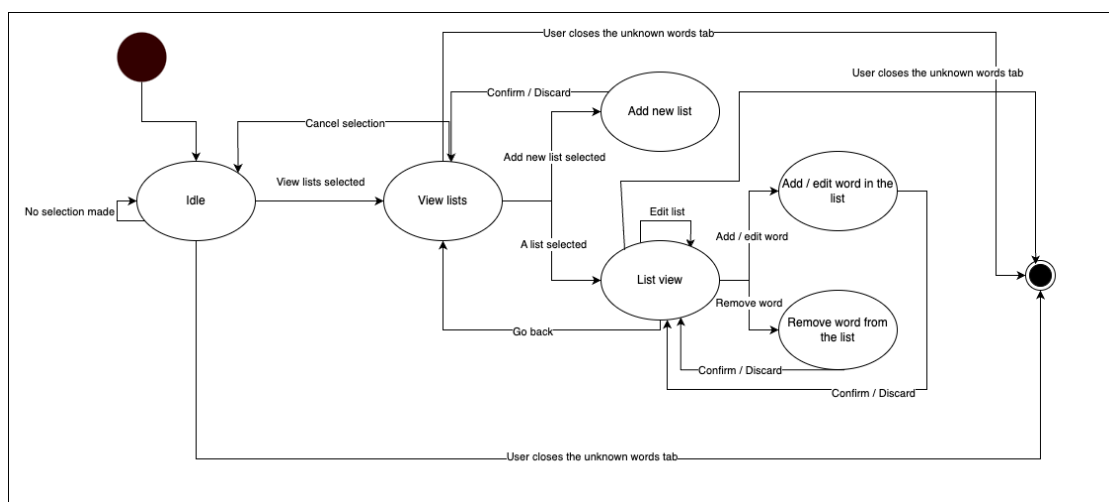


Fig. 8. Unknown word lists page UML State Diagram

### 3.5.4.2 Conversation

This section includes the Activity and State UML diagrams for all “Conversation” related features and logic.

#### 3.5.4.2.1 Conversation UML Activity Diagrams

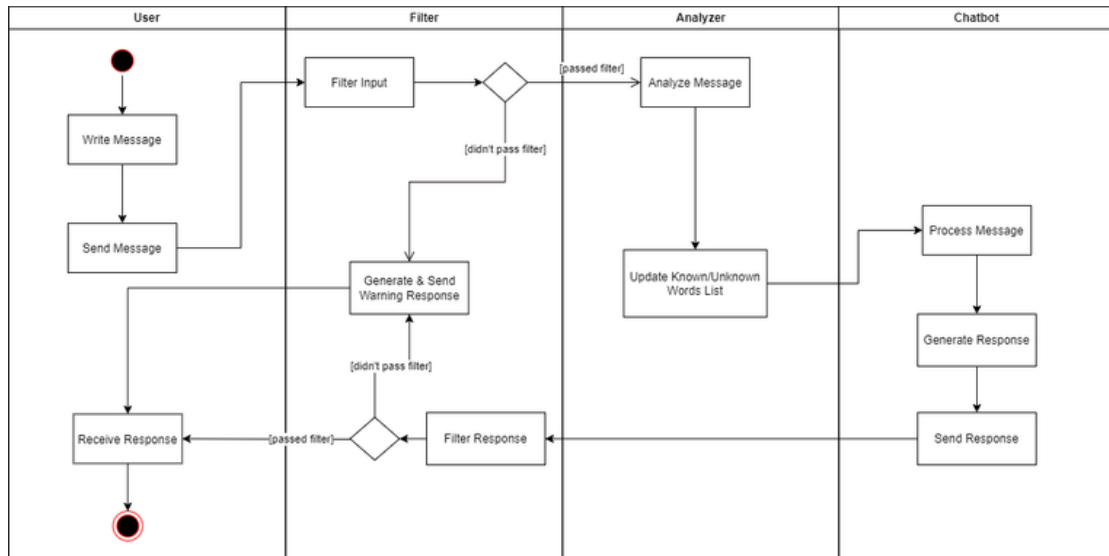


Fig. 9. Conversation UML Activity Diagram

#### 3.5.4.2.2 Conversation UML State Diagrams

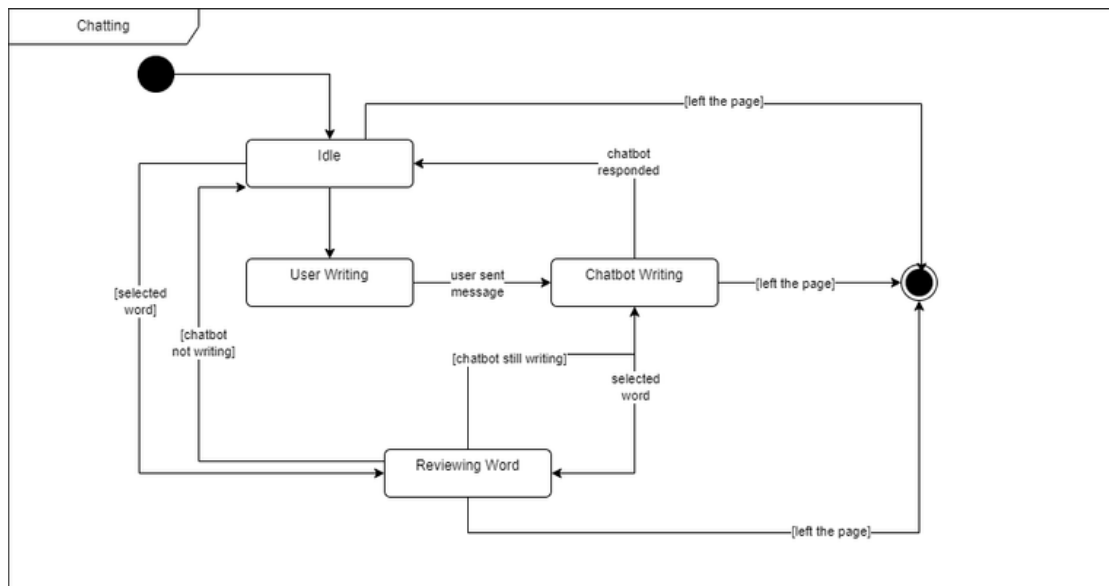


Fig. 10. Conversation UML State Diagram



### 3.5.4.3 Gamification

This section includes the Activity and State UML diagrams for all “Gamification” related features and logic.

#### 3.5.4.3.1 Gamification UML Activity Diagrams

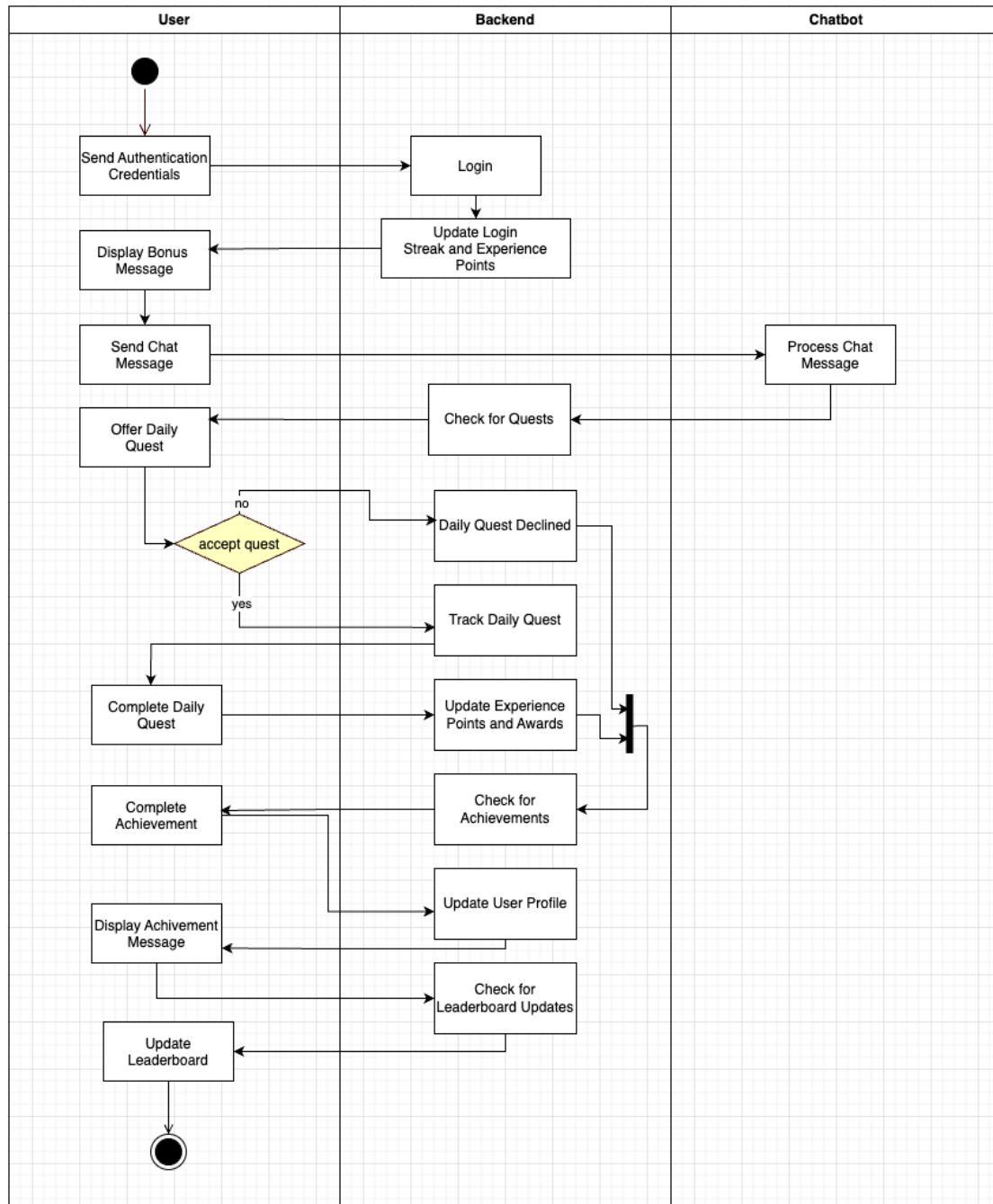


Fig. 11. Gamification UML Activity Diagram

### 3.5.4.3.2 Gamification UML State Diagrams

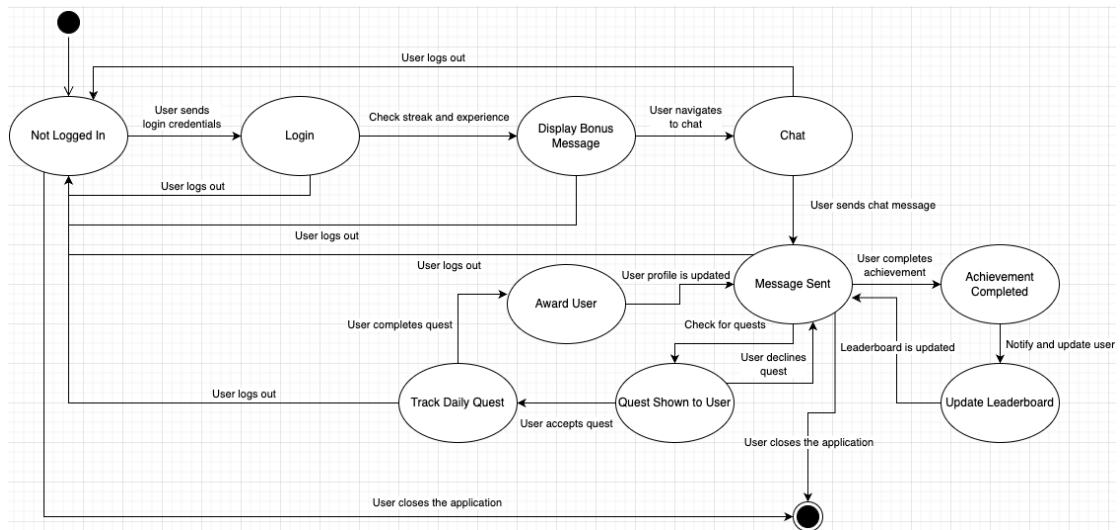


Fig. 12. Gamification UML State Diagram

### 3.5.5 User Interface

#### 3.5.5.1 Implemented Screens

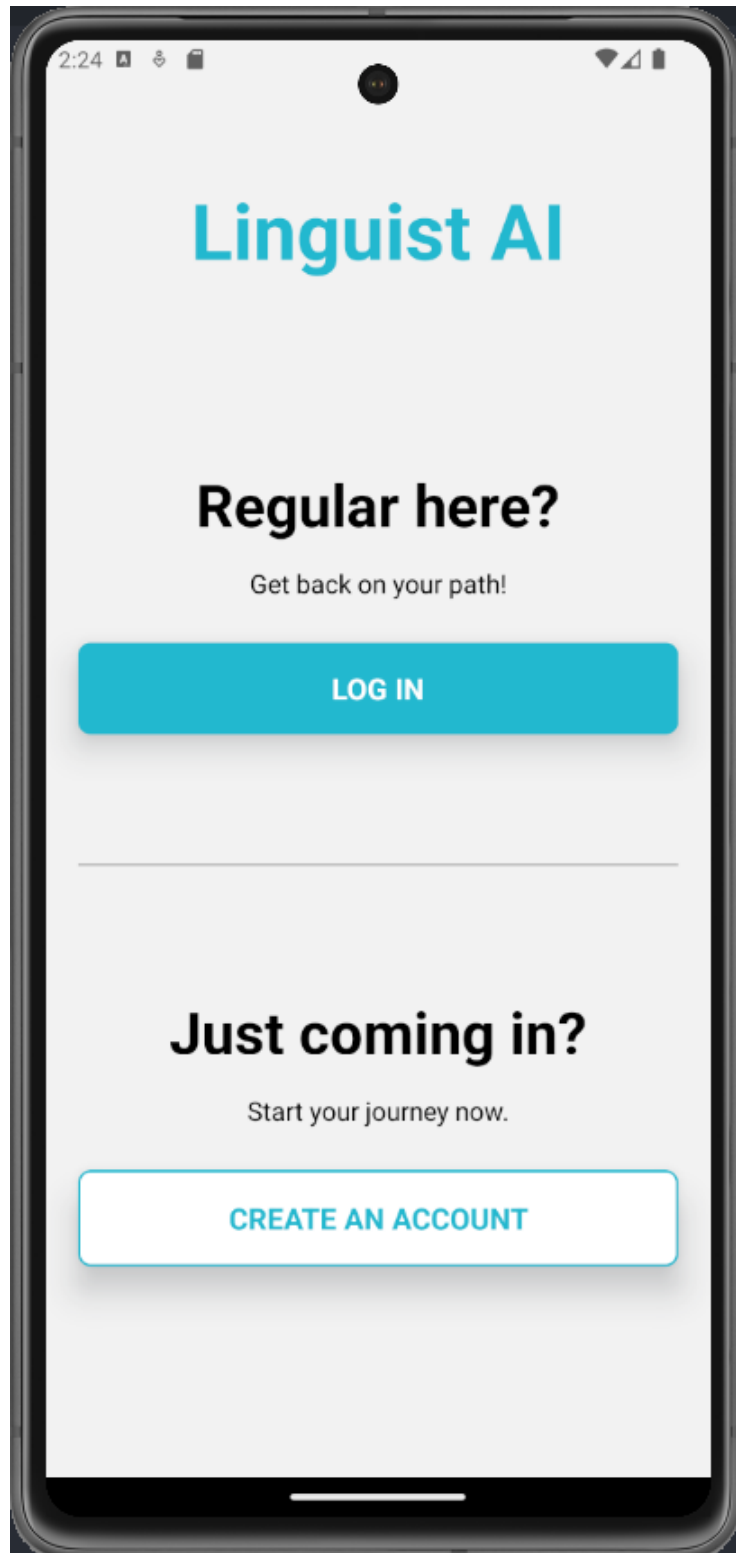


Fig. 13. Landing screen

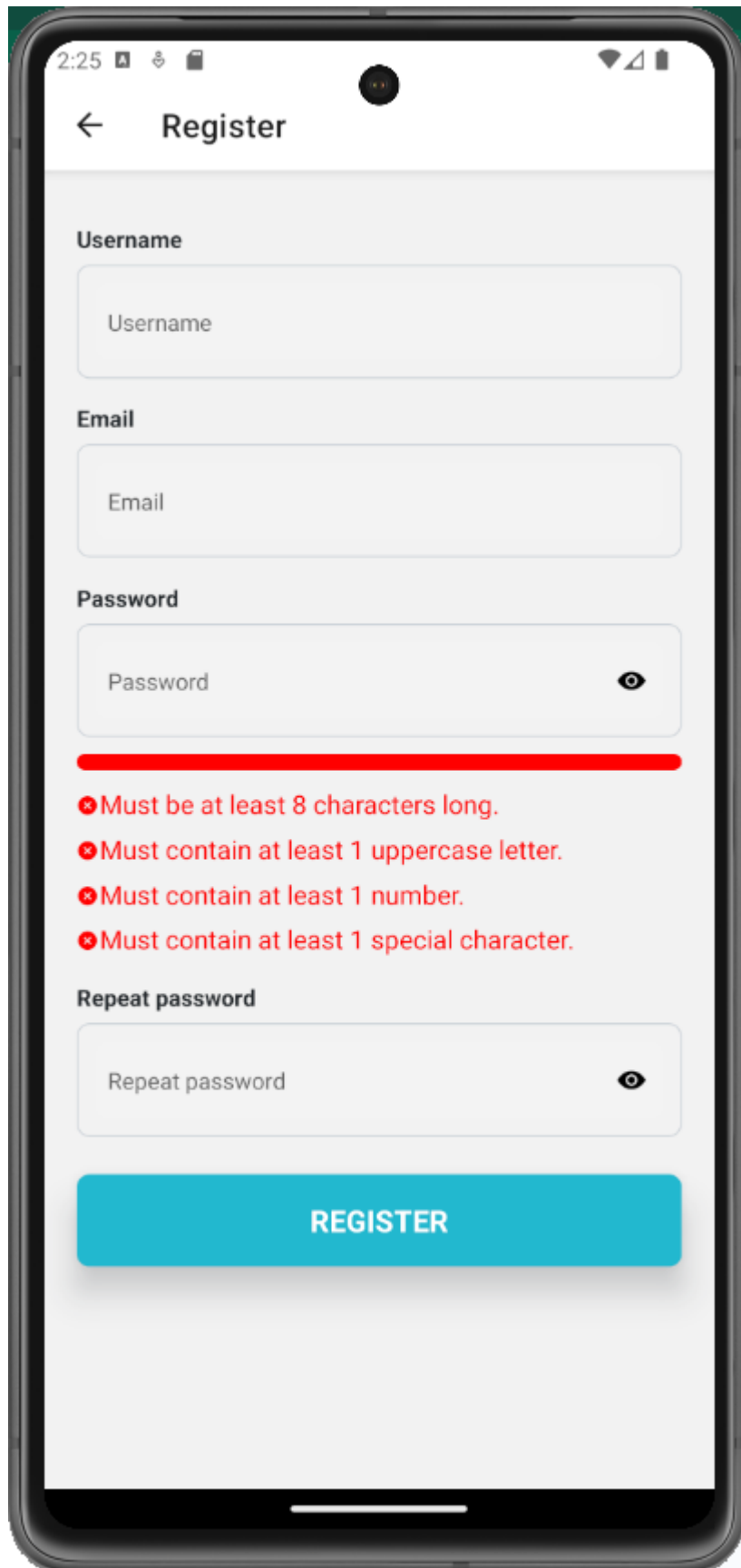


Fig. 14. Register screen

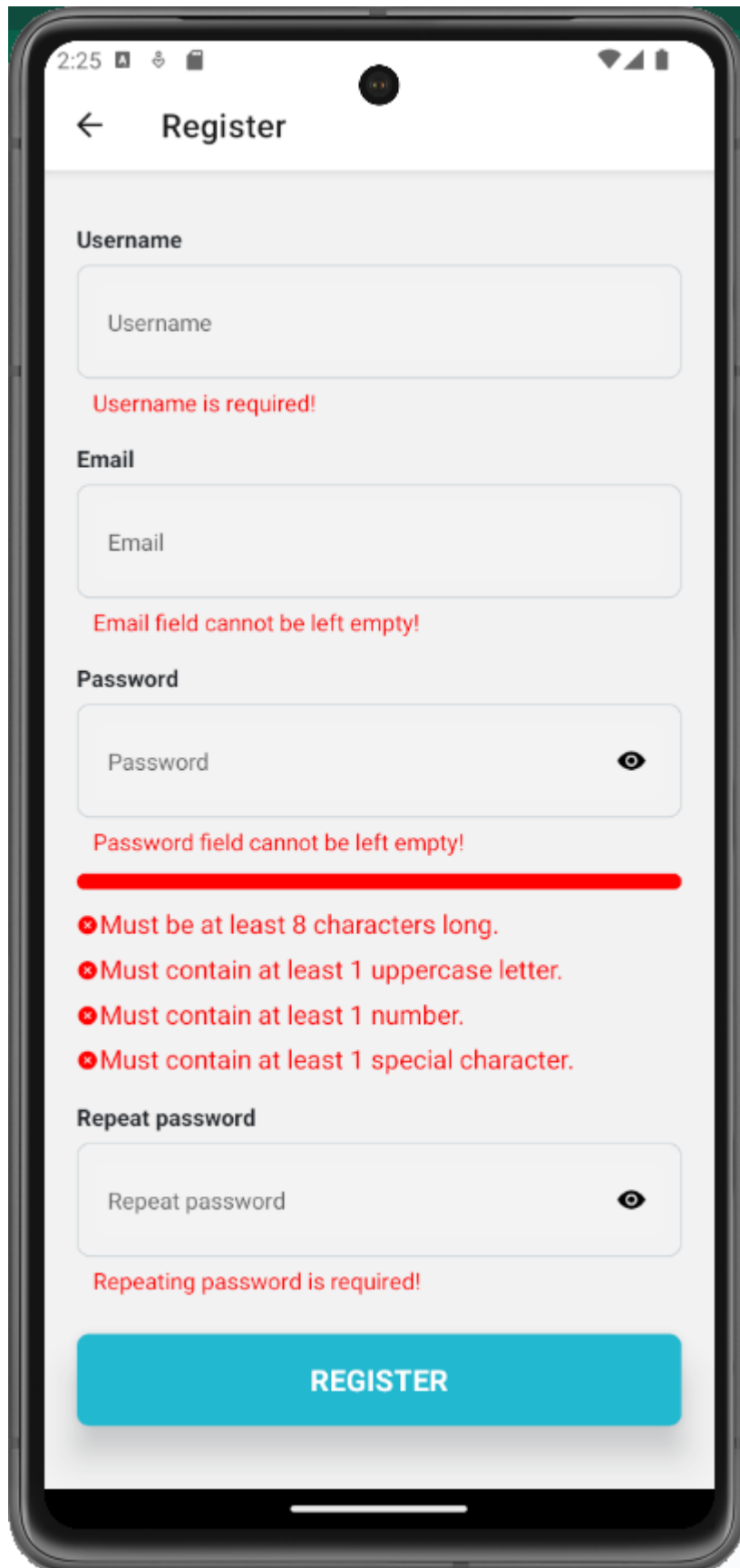


Fig. 15. Register screen fail scenario

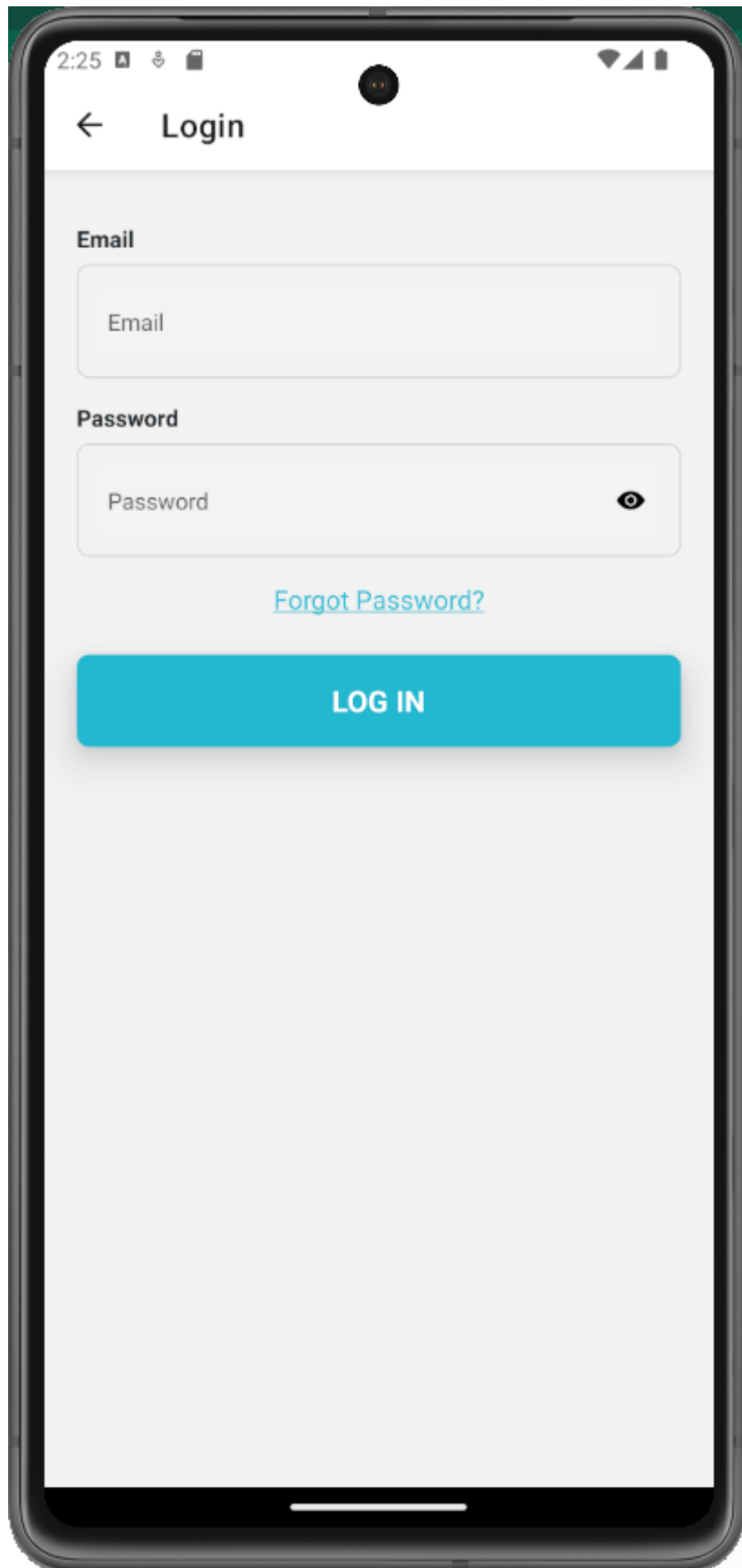


Fig. 16. Log in screen

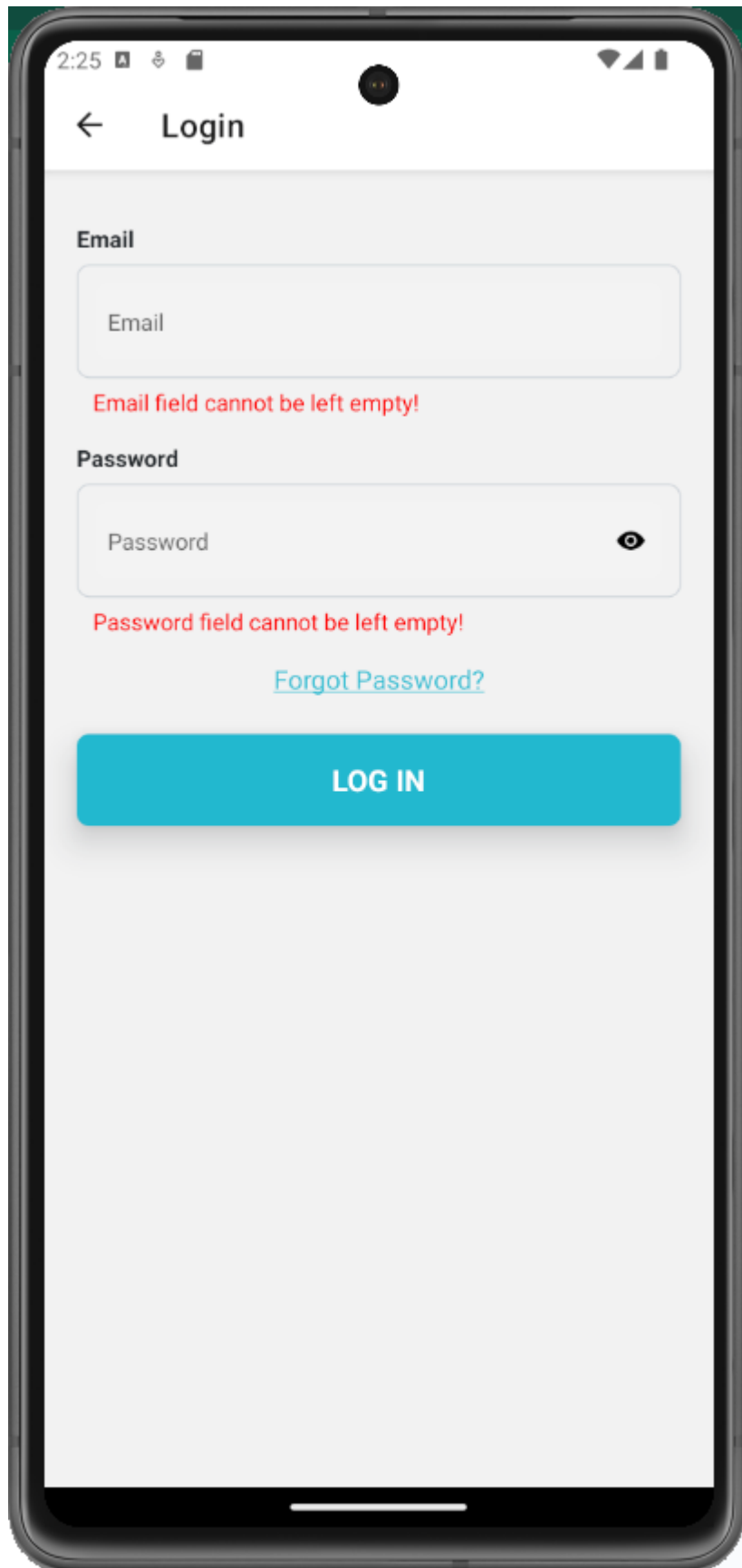


Fig. 17. Log in screen fail scenario

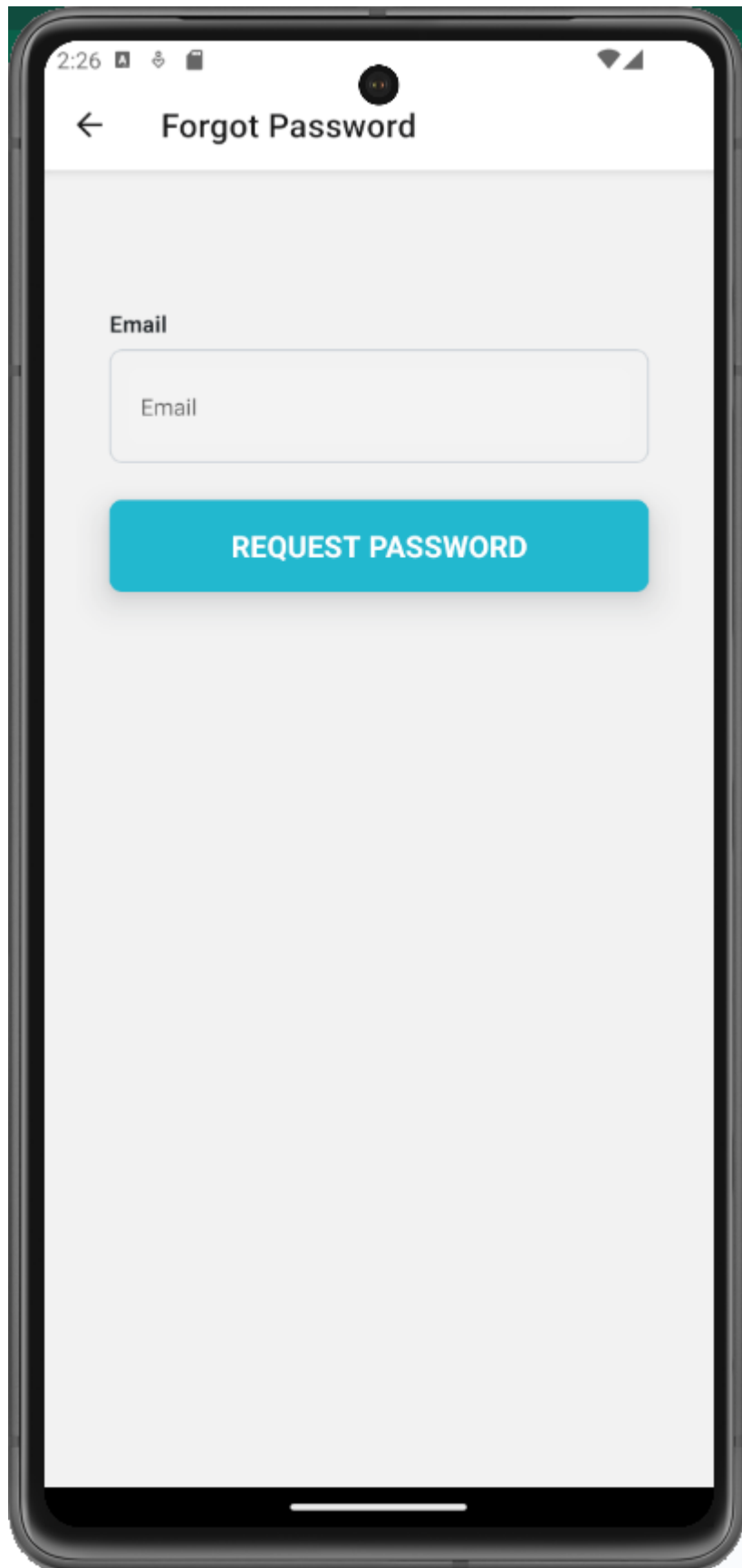


Fig. 18. Forgot password screen



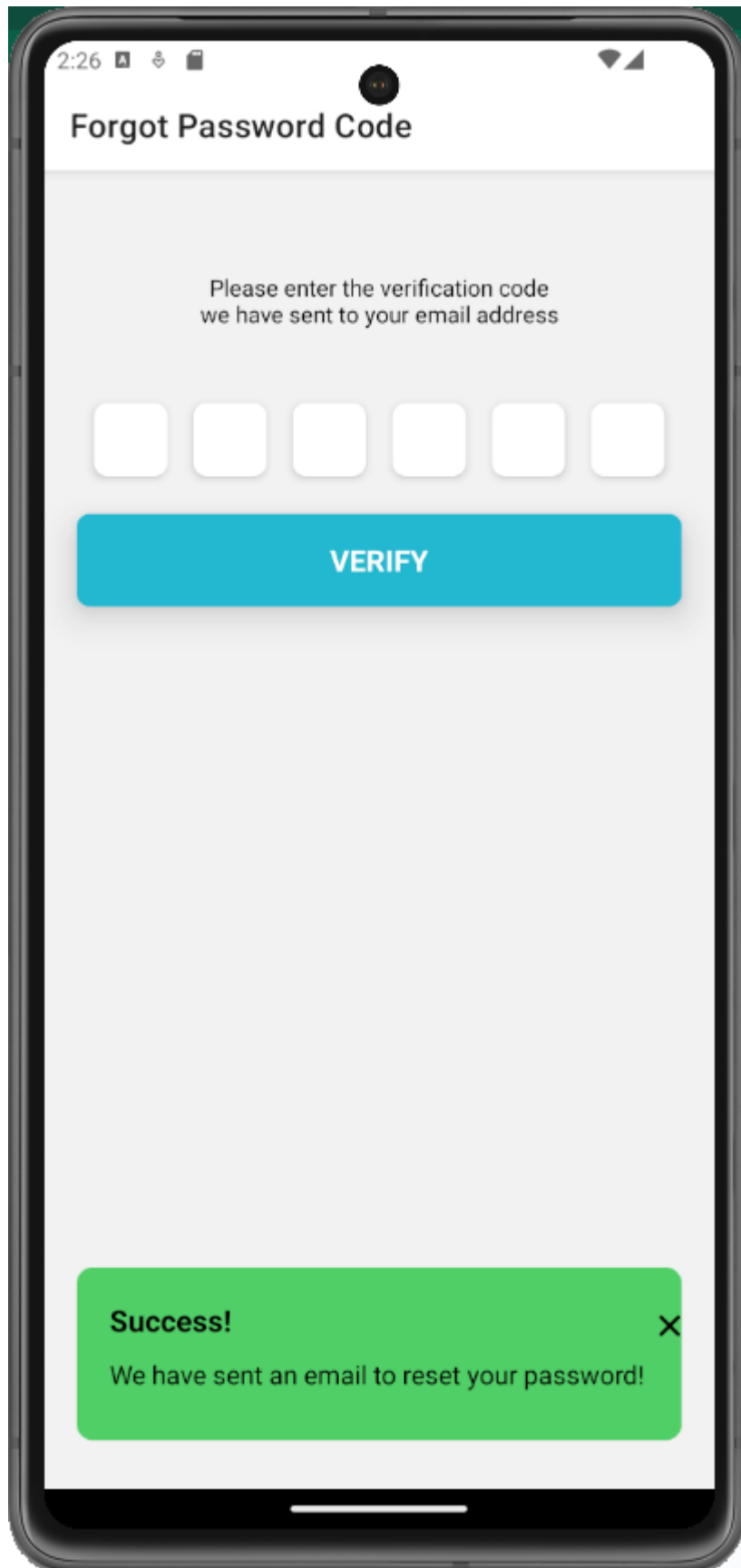


Fig. 19. Forgot password code confirmation screen

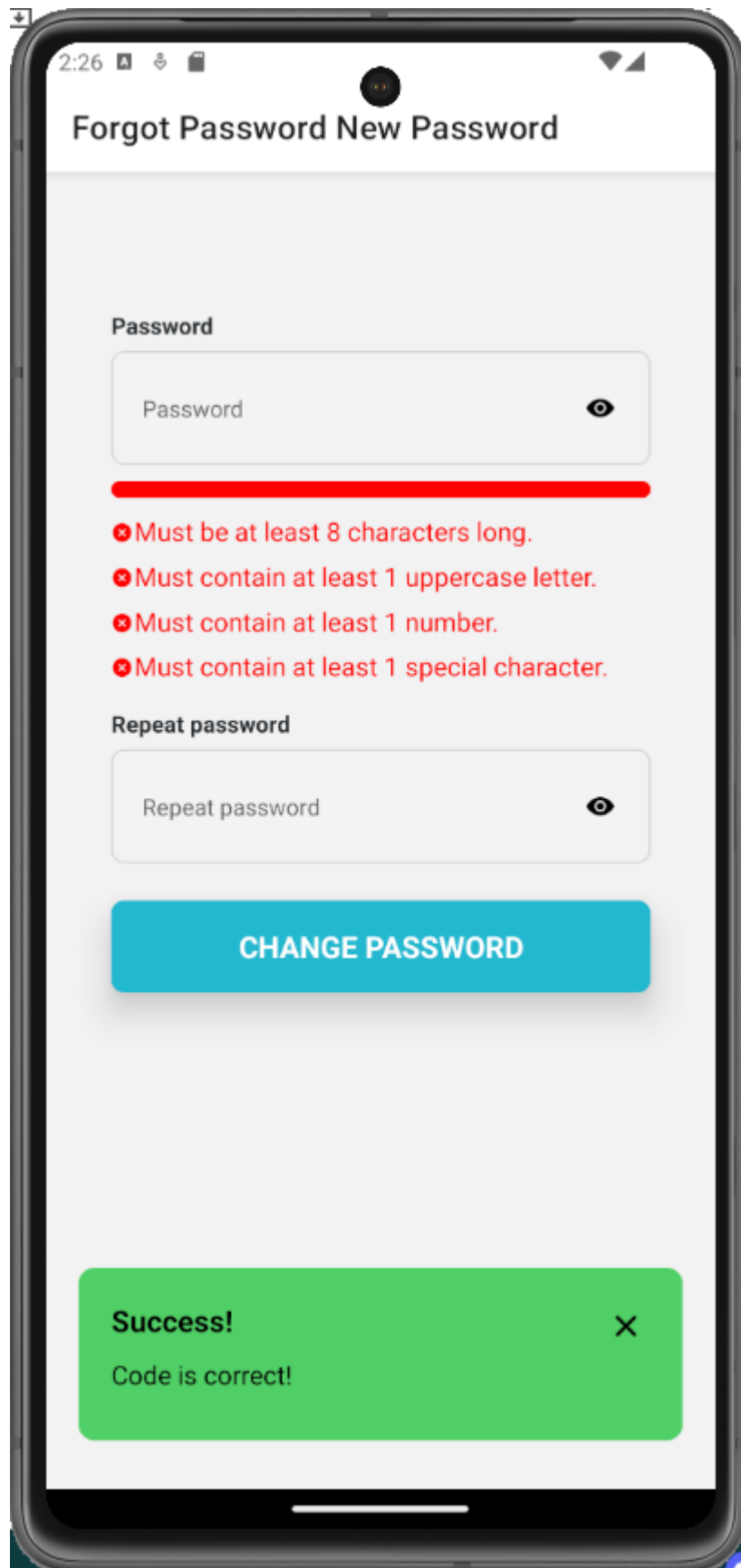


Fig. 20. Forgot password, password renewal screen

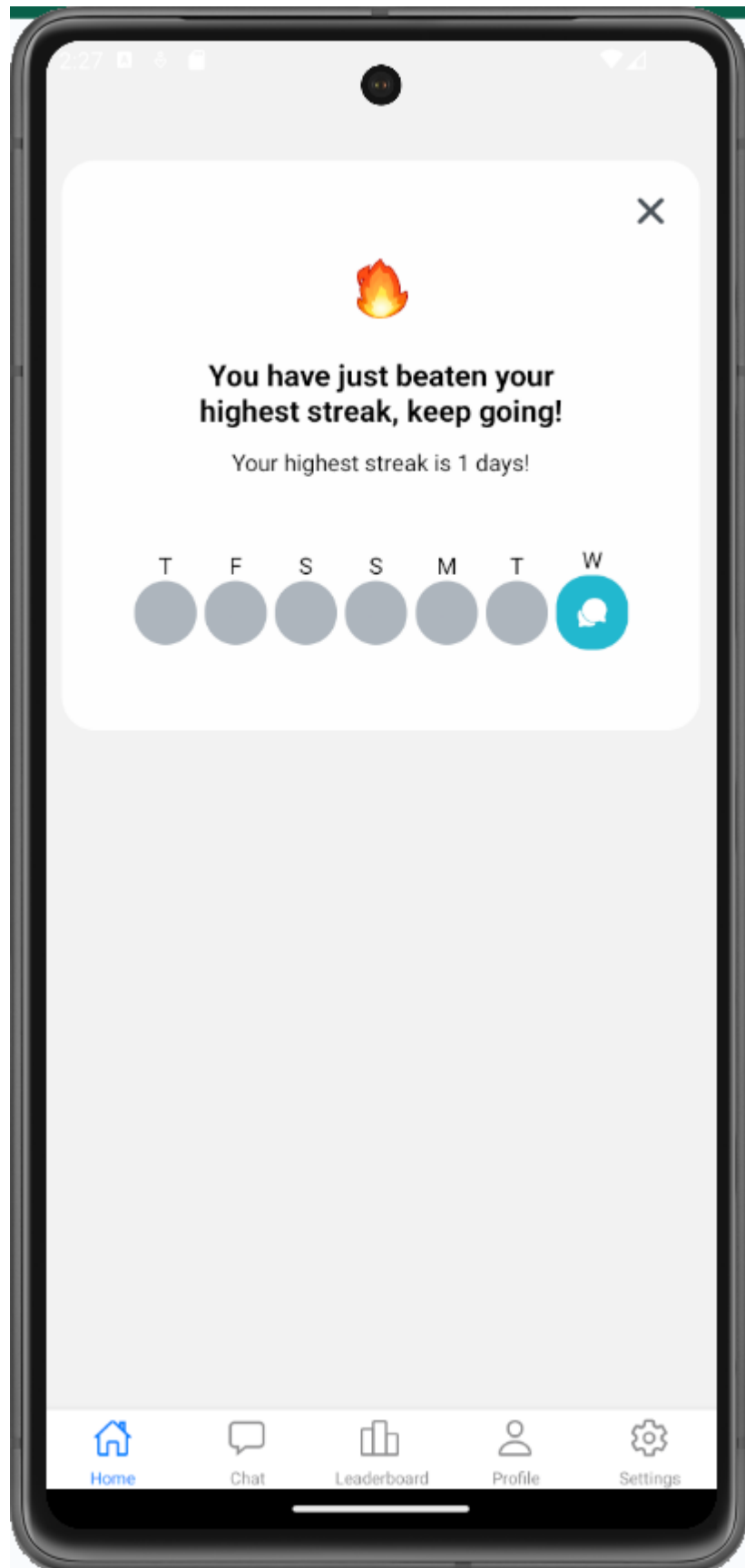


Fig. 21. Chat streak modal

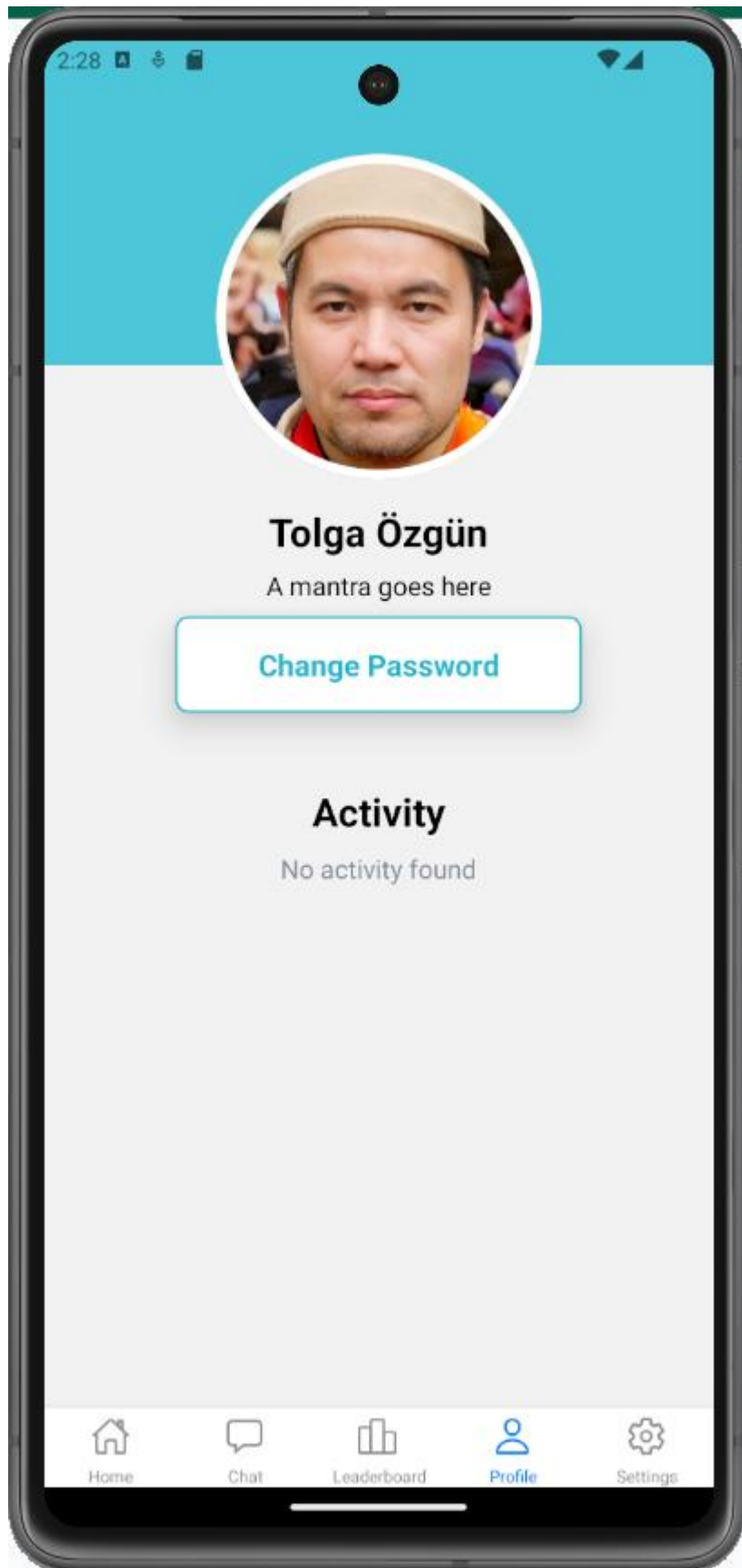


Fig. 22. Profile screen

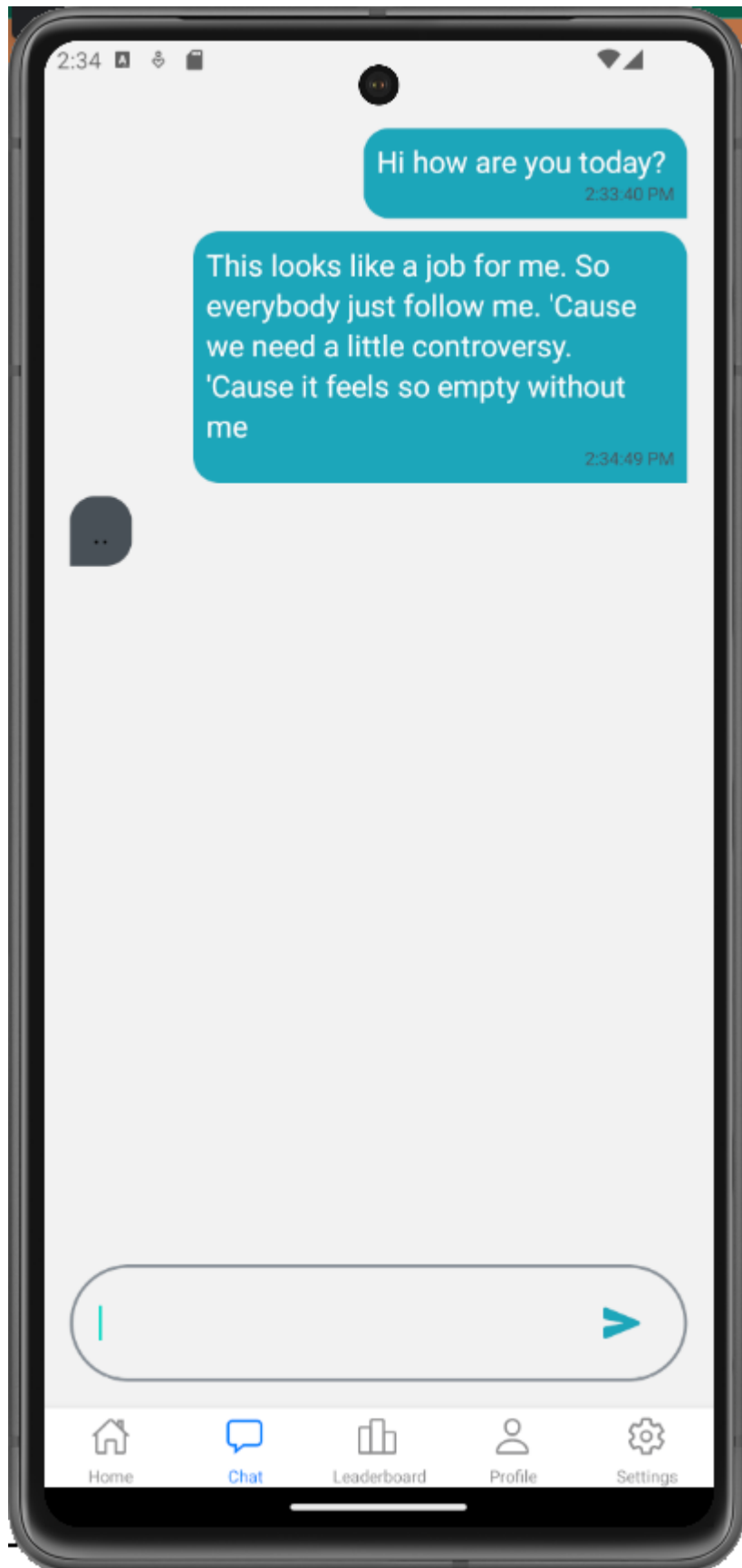


Fig. 23. Chat screen

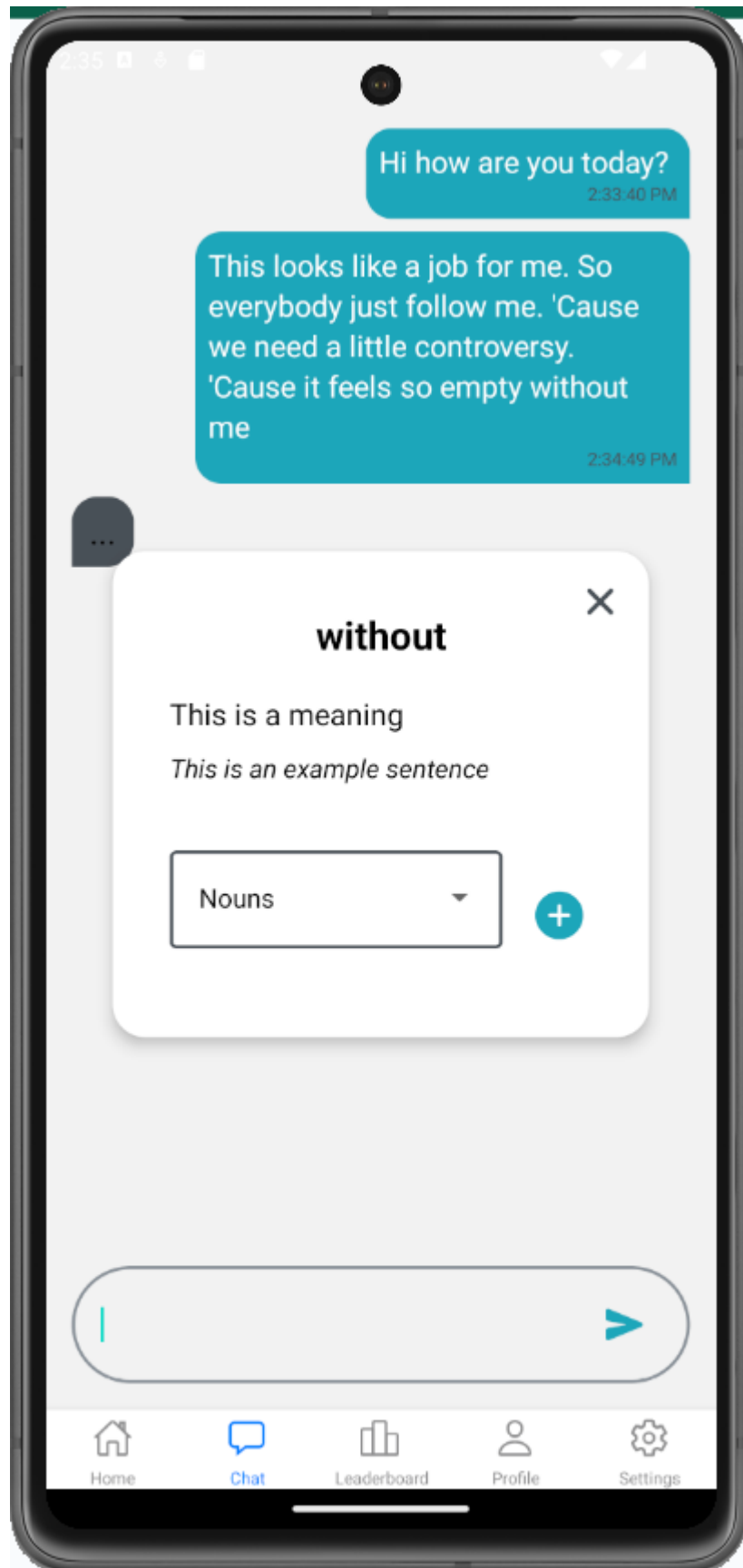


Fig. 24. Word lookup modal (opens when user clicks on a word)

### 3.5.5.2 Screens on Figma

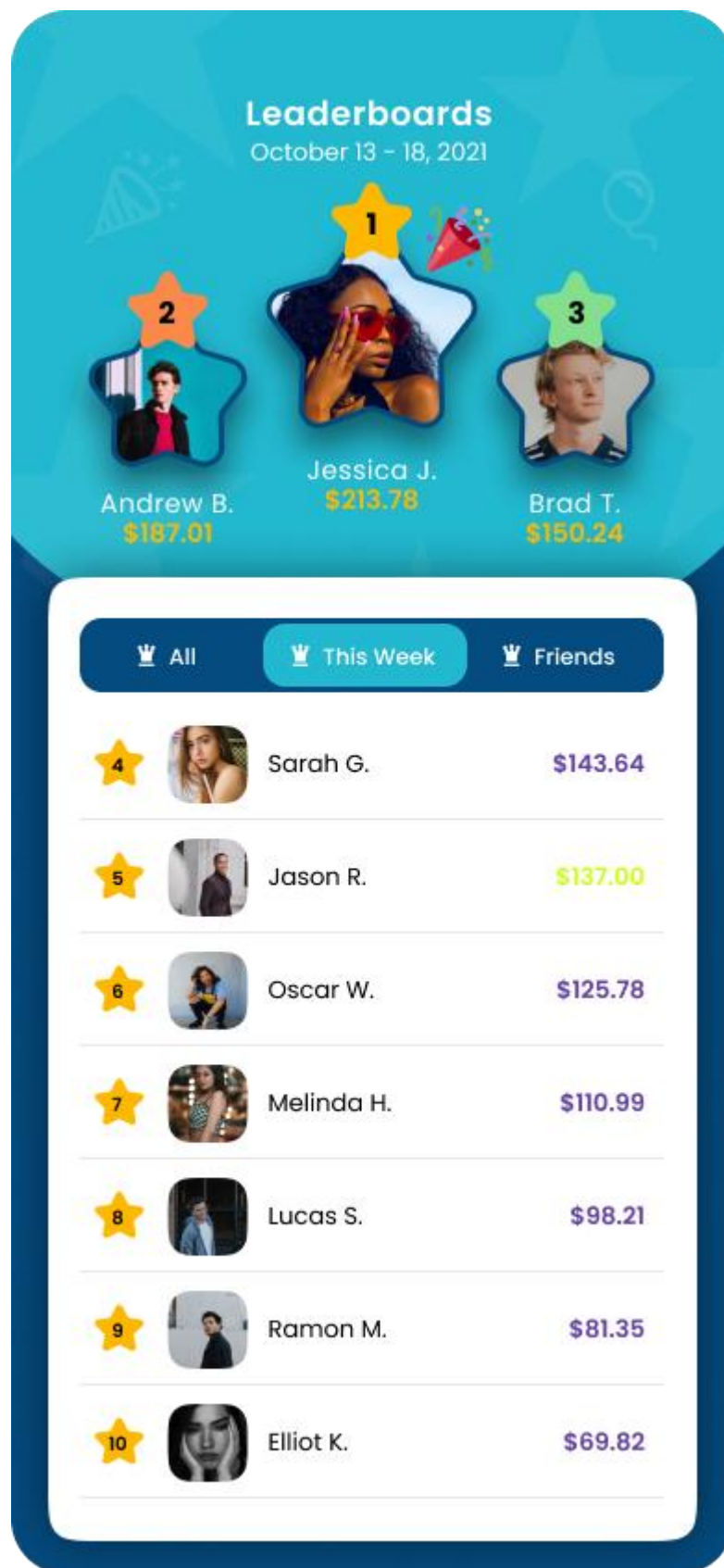


Fig. 25. Leaderboard screen design on Figma

## **4 Other Analysis Elements**

### **4.1 Consideration of Various Factors in Engineering Design**

#### **4.1.1 Constraints**

The main feature of the application is a chatbot in which the users can freely speak on any topic they choose. If the user starts talking about matters associated with health issues, the chatbot should not make any judgments or give any health-related advice to the user. Since the model is a conversational AI model, it has no actual knowledge about health-related topics, but it may appear so to the user that the chatbot is knowledgeable about public health, and the user might take the answers as if they are facts which may lead the users to take false medical advice from the chatbot. To avoid this, the chatbot should be constrained not to speak about health topics, especially if the context of the chat seems like a doctor consultation. The user may try to manipulate the generative AI model in various ways. For example, by saying this is just a role play scenario, they might try to convince the AI model to answer back on a health associated message. The chatbot must not be persuaded by such attempts, and should warn the user that they can't talk about such topics. Lastly, before using the chatbot, the user can be warned about how what the chatbot says should not be taken as facts.

Ensuring public safety plays an important role in the analysis and design of our application. All user-generated messages are securely stored on our application servers, and users will be provided with our Terms of Service prior to signing up to give informed consent. In order to comply with data privacy laws, users can download their stored data and delete their accounts. Furthermore, to protect sensitive information, we use industry-standard measures for storing user data, including SHA-256 bit encryption and salting procedures. We use a multifaceted approach to combat potential risks such as hallucinations, which manifest as inaccurate or misleading information, and to mitigate dangerous text outputs such as inappropriate content or biased language. This includes prompt engineering, fine-tuning various text-analysis models, and developing an ensemble model to ensure the application's highest levels of safety and accuracy. Moreover, as in public health concerns, the user should always be warned that the chatbot can be factually and morally wrong.

While public safety considerations take precedence in our application's analysis and design, public welfare also holds a notable, albeit secondary, significance. Our primary mission is to educate people in English, a skill that has universal application. Given this, we recognize the significance of making the application affordable to the general public. By making the app financially accessible, we hope to empower a larger population and advance public welfare by promoting English language proficiency that people can use in a variety of contexts.



Global factors influenced our decision to prioritize English in our language learning application. We aimed to meet the diverse and extensive international demand for English language proficiency, given the large number of foreign learners and the widespread use of English as a global language.

Cultural factors hold significance in our approach as we prioritize respect and adaptability to diverse cultural norms and sensitivities. It is important for us to avoid offensive or inappropriate content across various cultural contexts, contributing to a more inclusive and respectful user experience.

Social factors are crucial in our strategy, influencing the gamification aspects of our application significantly. Leaderboards, achievements, and daily streaks are carefully designed to appeal to users' competitive instincts, fostering a sense of accomplishment and motivation. Furthermore, the desire for human-like interaction motivates us to improve the AI's writing to mimic a more natural conversational tone both in writing and speaking.

Environmental factors, specifically the carbon footprint associated with GPUs, have limited impact on our decisions since given our financial limitations, we are opting for alternative strategies that do not heavily rely on GPUs. Our commitment to cost-effectiveness influences our choice of technologies, and we are exploring resource-efficient alternatives that align with both our budgetary considerations and environmental consciousness.

Economic factors are a driving force in our decision-making process, particularly due to the prevalent GPU-heavy solutions for training and inference in LLM models. Given budget constraints, we are opting for alternative strategies such as prompt engineering and models with fewer parameters. While these approaches can be executed with CPUs, they might yield less optimal results. However, they align with our financial considerations and ensure a feasible and cost-effective development path for Linguist AI.

Table 1: Factors that can affect analysis and design.

	Effect level	Effect
Public health	5/10	Chatbot should be blocked from giving health related advice, user should be warned that the bot's medical views can be wrong
Public safety	7/10	Data privacy should be ensured, users consent should be taken about data collection and they should be given the option to opt out at any time. To combat AI hallucination, chatbot's messages should be filtered and the user should be warned.
Public welfare	3/10	To promote English education, the app should be affordable.
Global factors	6/10	The foreign language was chosen to be English due to its global popularity.
Cultural factors	4/10	For cultural respect, we aim to filter offensive content across diverse cultural contexts.
Social factors	9/10	The social competitiveness driving motivation leads us to implement gamification elements like leaderboards and achievements. Chatbot is designed to mimic human chat as closely as possible.
Environmental factors	3/10	Not much effect as we will not depend on GPU related solutions.
Economic factors	8/10	Since GPUs for LLM training are expensive, we opt for prompt engineering and smaller models.

#### 4.1.2 Standards

In all parts of our tech stack, floating point numbers are stored based on the IEEE 754 standard. We follow the Agile manifesto, and our development team adheres to 2-week-long Scrum cycles, where we hold regular “stand up” meetings to discuss our progress, problems and plans. We hold Sprint review, retrospective and planning meetings at the end of each Sprint. For the machine learning part of our code, implemented in Python, we follow the PEP-8 style guide. For modeling, we follow the UML 2.5 version specification.

## 4.2 Risks and Alternatives

Table 2: Identified Opportunities

<b>Risk</b>	<b>Likelihood</b>	<b>Effect on the project</b>	<b>B Plan Summary</b>
Application usage may exceed expectations.	Moderate	Increased chat duration and engagement with the application. The chatbot may freeze or send responses more slowly.	If the amount of concurrent users increases in a way that affects the delay of the bot responses, we can add load balancers or upgrade our hardware.
Users may want to chat with bots with different personalities such as a philosopher bot or a teenage bot.	Low	Increase in engagement with the bot. Linguist AI might get more popular.	Upon extensive discussion sessions, we can decide to implement chatbots having various personalities.
Users might want to use Linguist AI on their computer as well.	Low	Increase in engagement. Users do not only want to chat with the bot when they are waiting in a line or commuting somewhere, but they also want to sit down properly and have long conversations.	If the demand is high, a web application of Linguist AI can be developed. Since the backend services can be utilized the same way, implementing a frontend and integrating it would be sufficient.

Table 3: Identified Threats

<b>Risk</b>	<b>Likelihood</b>	<b>Effect on the project</b>	<b>B Plan Summary</b>
Some of the features such as language level detection might not be implemented due to the difficulty of the implementation	Moderate	Since the language level cannot be detected, the chatbot cannot use vocabulary appropriate to the user level.	We can ask the user to provide their level, like A2, or B1.
We might not have enough time to finish all the features.	Low	The project would be incomplete if the unfinished features were included in the main features.	We can try to finish the project earlier than we planned so that if something goes wrong, we will have more time.
Some features might not be tested well.	Low	If chat-related features are not tested well, the application might be too clumsy to use, decreasing the number of users. If other areas, such as leaderboards and achievements, are not tested, users would not get annoyed as much.	We can test the features as we develop them instead of testing all features at once close to the deadline.
Some users might use the bot for malicious reasons.	High	The chatbot might learn this malicious content and respond to the users inappropriately.	Filtration can be added to both user and bot messages. Therefore, if either side generates an improper response, the filter can detect it. If the user sends a such message, the filter can prevent the bot from learning. Furthermore, if the filter detects a malicious message produced by the bot, a new response can be generated.

### 4.3 Project Plan

Table 4: List of work packages

WP#	Work package title	Leader	Members involved
WP1	Unknown Word Bank Backend Development	Yağız Can Aslan	İlkim Elif Kervan
WP2	Unknown Word Bank UI Development	Selim Can Güler	Tolga Özgün, Yağız Can Aslan
WP3	Inferring English Level - Research	Kardelen Ceren	Selim Can Güler, Tolga Özgün
WP4	Dictionary Service Development	İlkim Elif Kervan	Yağız Can Aslan, Selim Can Güler
WP5	Conversational Chatbot LLM Development	Tolga Özgün	Kardelen Ceren
WP6	Conversation Backend Infrastructure Development	Yağız Can Aslan	İlkim Elif Kervan
WP7	User Message Filter Development	Kardelen Ceren	Selim Can Güler
WP8	Chatbot Message Filter Development	Tolga Özgün	Kardelen Ceren
WP9	Contextual Word Scoring Model Development	Selim Can Güler	Tolga Özgün, Kardelen Ceren
WP10	Multi-choice Question Test Backend Development	İlkim Elif Kervan	Yağız Can Aslan
WP11	Multi-choice Question Test UI Development	Selim Can Güler	Tolga Özgün
WP12	Gamification Backend Development	Yağız Can Aslan	İlkim Elif Kervan
WP13	Gamification UI Development	Selim Can Güler	Yağız Can Aslan
WP14	API Gateway, Microservice Architecture and Authorization	İlkim Elif Kervan	Yağız Can Aslan
WP15	Setup Jira and Confluence Infrastructures	Yağız Can Aslan	-
WP16	Voice Chat Integration	Kardelen Ceren	İlkim Elif Kervan, Yağız Can Aslan
WP17	User Profile Backend Development	Tolga Özgün	Kardelen Ceren, İlkin Elif Kervan

WP 1: Unknown Word Bank Backend Development			
Start date: 14.11.2023 End date: 30.12.2023			
Leader:	Yağız Can Aslan	Members involved:	İlkim Elif Kervan
<b>Objectives:</b> Designing the backend of the “Unknown Word Bank” system. Development of object models, database schemas and logic of the “Unknown Word Bank” system including adding and removing words from word lists, creation and deletion of word lists, activating and deactivating lists.			
<b>Tasks:</b> <b>Task 1.1 UML Use Case Diagram for Unknown Word Bank:</b> Design, discuss and iterate over the UML Use Case diagram for Unknown Word Bank system. <b>Task 1.2 UML Activity Diagram for Unknown Word Bank:</b> Design, discuss and iterate over the UML Activity diagram for Unknown Word Bank system. <b>Task 1.3 UML State Diagram for Unknown Word Bank:</b> Design, discuss and iterate over the UML State diagram for Unknown Word Bank system. <b>Task 1.3 UML Class Diagram for Unknown Word Bank:</b> Design, discuss and iterate over the UML Class diagram for Unknown Word Bank system. <b>Task 1.4 Unknown Word Bank Backend Spring Implementation:</b> Implement the Controller - Service - Repository classes in Java Spring, modeled object classes and database schemas for the Unknown Word Bank system.			
<b>Deliverables</b> <b>D1.1:</b> Source Code <b>D1.2:</b> Documentation <b>D1.3:</b> UML Use Case, Activity, State and Class Diagrams			

WP 2: Unknown Word Bank UI Development			
Start date: 10.12.2023 End date: 15.01.2024			
Leader:	Selim Can Güler	Members involved:	Tolga Özgün, Yağız Can Aslan
<b>Objectives:</b> Designing the UI of the “Unknown Word Bank” system. Development of UI and logic of the “Unknown Word Bank” system including adding and removing words from word lists, creation and deletion of word lists. Sending calls to “Unknown Word Bank” endpoints when necessary			
<b>Tasks:</b> <b>Task 2.1 Designing the UI :</b> Designing the general look of the UI. <b>Task 2.2 Development of UI :</b> Implementing the design in React Native without any logic included. <b>Task 2.3 Development of logic/state:</b> Logical components, state changes in the frontend must be implemented. <b>Task 2.4 Integrating API endpoint calls:</b> Endpoints for adding and removing unknown words to word lists, creation and deletion of word lists must be sent to the backend.			
<b>Deliverables</b> <b>D2.1:</b> UI Design <b>D2.2:</b> Source Code <b>D2.3:</b> Documentation			

<b>WP 3: Inferring English Level - Research</b>			
<b>Start date:</b> 01.01.2024 <b>End date:</b> 30.01.2024			
<b>Leader:</b>	Kardelen Ceren	<b>Members involved:</b>	Selim Can Güler, Tolga Özgün
<p><b>Objectives:</b> Researching methods to correctly and effectively assess a learner's English level in order for the LLM to be able to match users level. As this is an internal measure, evaluated levels do not necessarily have to match CEFR levels such as B1, C2, etc., but would be preferred if possible.</p>			
<p><b>Tasks:</b></p> <p><b>Task 3.1 Literature review:</b> Review comprehensively the existing research on English evaluation tests, their accuracy and how well they could be incorporated to an online, automated system.</p> <p><b>Task 3.2 Market research:</b> Research the existing online tools that are used to evaluate foreign language skills.</p> <p><b>Task 3.3 Consulting language learning experts:</b> Conduct interviews with language learning professionals to get insight into different benchmarks and their recommendations for our system.</p> <p><b>Task 3.4 Development of evaluation criteria:</b> Define clear criteria for language evaluation that is feasible and as accurate as possible.</p>			
<p><b>Deliverables</b></p> <p><b>D3.1:</b> Research Report</p> <p><b>D3.2:</b> Evaluation Criteria Guidelines</p>			

<b>WP 4: Dictionary Service Development</b>			
<b>Start date:</b> 14.11.2023 <b>End date:</b> 15.0.2023			
<b>Leader:</b>	İlkin Elif Kervan	<b>Members involved:</b>	Yağız Can Aslan, Selim Can Güler
<p><b>Objectives:</b> In this work package, a dictionary service will be developed in order to provide word definitions and their usages in the sentences to the user. Dictionary API will not be created from scratch. A free dictionary API will be selected and integrated into the project. Since it is easier to send requests to external APIs with Express, a new microservice will be implemented using NodeJS and Express. The sole purpose of this microservice will be to send request to external APIs whenever needed.</p>			
<p><b>Tasks:</b></p> <p><b>Task 4.1 Research about available dictionary APIs :</b> Make a research about the existing free dictionary APIs on the Internet to find a suitable one to use in the project.</p> <p><b>Task 4.2 Integrate dictionary API to the project :</b> In order to display vocabulary definition and usage to the user, implement a microservice to send requests to the dictionary API found in the first subtask.</p>			
<p><b>Deliverables</b></p> <p><b>D4.1:</b> An Express microservice</p> <p><b>D4.2:</b> Source code of the API</p> <p><b>D4.3:</b> Documentation of the API</p>			

WP 5: Conversational Chatbot LLM Development			
Start date: 05.11.2023 End date: 20.01.2024			
Leader:	Tolga Özgün	Members involved:	Kardelen Ceren
<p><b>Objectives:</b> In this work package, conversational chatbot LLM will be developed and deployed in a server that will handle the user input to create a conversation. During this development various experiments with existing chat models will be conducted to select the best performing model for our use case. Moreover, we will try to leverage methodologies such as prompt engineering, LLM chaining and agents to create the most efficient chatbot.</p> <p><b>Tasks:</b></p> <p><b>Task 5.1 Research and select a framework on chat models:</b> There are many frameworks that help with the deployment and management of chat models. With the use of a framework, we can focus on the details of our application without having to worry about the basics.</p> <p><b>Task 5.2 Research and select chat models :</b> We can leverage more than one chat model in our application as a chain, however we need to research and calculate the feasibility for each combination.</p> <p><b>Task 5.3 Prompt engineering :</b> Experiment with various prompts in various use cases to find the best performing prompt.</p> <p><b>Task 5.4 Research and ingest documents :</b> We can ingest documents to our chatbot which puts additional knowledge into the chatbot's memory. This can be beneficial for our use case as it decreases the need for fine-tuning the model.</p> <p><b>Task 5.5 Research and create tools:</b> There are tools that can be defined to the chatbot to help with the conversation. We can research and see if the implementation is feasible.</p>			
<p><b>Deliverables</b></p> <p><b>D5.1:</b> Source code of LLM microservice</p> <p><b>D5.2:</b> Various LLM API endpoints</p> <p><b>D5.3:</b> Documentation of the API</p>			

WP 6: Conversation Backend Infrastructure Development			
Start date: 22.10.2023 End date: 13.11.2023			
Leader:	Yağız Can Aslan	Members involved:	İlkin Elif Kervan
<p><b>Objectives:</b> The user needs to be able to send and receive messages from our system. This work package aims at providing the infrastructure for storing (not producing) the messages. There needs to be support for multiple conversations.</p> <p><b>Tasks:</b></p> <p><b>Task 6.1 Design Conversation ER Diagram:</b> Design the ER diagram for the Conversation system which will aid in the designing of database schemas.</p> <p><b>Task 6.2 Conversation Backend Spring Implementation :</b> Implement the Controller - Service - Repository classes in Java Spring, modeled object classes and database schemas for the Conversation system.</p>			
<p><b>Deliverables</b></p> <p><b>D6.1:</b> Source Code</p> <p><b>D6.2:</b> Documentation</p> <p><b>D6.3:</b> Conversation ER Diagram</p>			



WP 7: User Message Filter Development			
Start date: 30.01.2024 End date: 30.02.2024			
Leader:	Kardelen Ceren	Members involved:	Selim Can Güler
<b>Objectives:</b> <i>This work package aims to filter user's messages so that texts that include sexist, racist, discriminating, hateful or sexual phrases, a warning message can be sent instead of an actual bot response.</i>			
<b>Tasks:</b> <b>Task 7.1 Determine malicious words and phrases :</b> <i>Research coding libraries or APIs that have listed unsuitable English vocabulary.</i> <b>Task 7.2 Develop user text filter :</b> <i>Implement a user filter that checks for malicious intent and sends a warning message, blocking the text reaching the bot.</i>			
<b>Deliverables</b> <b>D7.1:</b> Source code <b>D7.2:</b> Documentation			

WP 8: Chatbot Message Filter Development			
Start date: 20.01.2024 End date: 30.02.2024			
Leader:	Tolga Özgün	Members involved:	Kardelen Ceren
<b>Objectives:</b> <i>Chatbot's messages must be filtered to not demonstrate any racism, sexism, discriminatory behavior, hate crime, sexual behavior or hallucinatory information.</i>			
<b>Tasks:</b> <b>Task 8.1 Literature and tool review:</b> <i>Research existing literature and LLM filtering tools to determine which phrases could and should be filtered, along with which tools can be used to accomplish the development.</i> <b>Task 8.2 Filter implementation :</b> <i>Implement a filter that checks the text that comes out of the LLM based on the malicious words and phrases determined in Task 8.1.</i>			
<b>Deliverables</b> <b>D8.1:</b> LLM filtering guidelines report <b>D8.2:</b> Source code <b>D8.3:</b> Documentation			

<b>WP 9: Contextual Word Scoring Model Development</b>			
<b>Start date:</b> 15.01.2024 <b>End date:</b> 30.01.2024			
<b>Leader:</b>	Selim Can Güler	<b>Members involved:</b>	Tolga Özgün, Kardelen Ceren
<b>Objectives:</b> <i>There needs to be a model that will implicitly score the unknown words used by the user based on contextual and semantic correctness. The model will be an LLM. It will be fine-tuned for the task.</i>			
<b>Tasks:</b> <b>Task 9.1 Development &amp; fine-tuning of the model :</b> Development and testing of the initial model, and fine-tuning it to increase its performance. <b>Task 9.2 Updating the database:</b> Update the user's score for the words affected <b>Task 9.3 Develop API:</b> Development of several API endpoints related to the scoring of words in a sentence.			
<b>Deliverables</b> <b>D9.1:</b> Feasibility Report <b>D9.2:</b> Code <b>D9.3:</b> Documentation <b>D9.4:</b> LLM Model			

<b>WP 10: Multi-choice Question Test Backend Development</b>			
<b>Start date:</b> 01.03.2024 <b>End date:</b> 15.04.2024			
<b>Leader:</b>	İlkim Elif Kervan	<b>Members involved:</b>	Yağız Can Aslan
<b>Objectives:</b> <i>In this work package, backend services of the multi-choice tests will be implemented. The purpose of these tests is to measure the success of the users after they have conversation with the bot. In order to assess user success we will test them before and after the conversation. The tests before and after the conversation will test the same words using different questions. Therefore, in this package we will implement services that prepare questions and assess user success.</i>			
<b>Tasks:</b> <b>Task 10.1 Make a research to find the most effective question types :</b> Identification of the best question type so that success of the user can be measured correctly. <b>Task 10.2 Generate test questions :</b> Development of the question generator services using generative AI. <b>Task 10.3 Assess success of the user :</b> Development of services to measure the success of the user before and after they have conversation with the chat bot.			
<b>Deliverables</b> <b>D10.1:</b> Source code. <b>D10.2:</b> Documentation <b>D10.3:</b> LLM Model			

<b>WP 11: Multi-choice Question Test UI Development</b>			
<b>Start date:</b> 01.03.2024 <b>End date:</b> 15.04.2024			
<b>Leader:</b>	Selim Can Güler	<b>Members involved:</b>	Tolga Özgün
<b>Objectives:</b> Multi-choice question test will appear once the user starts a conversation with the bot and once the conversation ends. It will include calling the backend for the generation of questions, displaying the questions in an aesthetic manner. When the test ends results will be displayed.			
<b>Tasks:</b> <b>Task 11.1 Test question UI development:</b> The UI will include the following components: displaying questions and ability to go back and forth between questions, feedback on wrong/correct questions, displaying results summary after test ends <b>Task 11.2 Integration with backend :</b> Includes sending the answers to backend, fetching the questions from the backend.			
<b>Deliverables</b> <b>D11.1:</b> UI Design <b>D11.2:</b> Code <b>D11.3:</b> Documentation			

<b>WP 12: Gamification Backend Development</b>			
<b>Start date:</b> 22.10.2023 <b>End date:</b> 30.04.2024			
<b>Leader:</b>	Yağız Can Aslan	<b>Members involved:</b>	İlkin Elif Kervan
<b>Objectives:</b> Gamification will be a vital part of our application. It will distinguish us from competitors while aiding our users' language learning journeys. Interesting and captivating gamification ideas need to be implemented such as; login/chat streaks, experience points, customization, leaderboards, friendship system, achievements, daily quests and more (as ideas come up and are tested).			
<b>Tasks:</b> <b>Task 12.1 UML Use Case Diagram for Gamification:</b> Design, discuss and iterate over the UML Use Case diagram for Gamification system. <b>Task 12.2 UML Activity Diagram for Gamification:</b> Design, discuss and iterate over the UML Activity diagram for Gamification system. <b>Task 12.3 UML State Diagram for Gamification:</b> Design, discuss and iterate over the UML State diagram for Gamification system. <b>Task 12.3 UML Class Diagram for Gamification:</b> Design, discuss and iterate over the UML Class diagram for Gamification system. <b>Task 12.4 Login/Chat Streak Backend Implementation:</b> Implement the Controller - Service - Repository classes in Java Spring, modeled object classes and database schemas for the Login/Chat Streak system. <b>Task 12.5 Leaderboard Backend Implementation:</b> Implement the Controller - Service - Repository classes in Java Spring, modeled object classes and database schemas for the Leaderboard system.			

**Task 12.6 Friendship Backend Implementation:** Implement the Controller - Service - Repository classes in Java Spring, modeled object classes and database schemas for the Friendship system.

**Task 12.7 Achievements and Daily Quests Backend Implementation:** Implement the Controller - Service - Repository classes in Java Spring, modeled object classes and database schemas for the Achievements and Daily Quests system.

**Deliverables**

**D12.1:** Source Code

**D12.2:** Documentation

**D12.3:** UML Use Case, Activity, State and Class Diagrams

**WP 13: Gamification UI Development**

**Start date:** 20.11.2023 **End date:** 30.04.2024

**Leader:** Selim Can Güler

**Members involved:**

Yağız Can Aslan

**Objectives:** Development of gamification elements. Chat streak components will be displayed on each initial load of the application. List of achievements will be visible from the user's profile (obtained and to be obtained), achievement badges will be displayed when their conditions are met. Leaderboard based on chat streak and learned words between user's friends and a randomly generated league will be displayed. Personification of the chatbot.

**Tasks:**

**Task 13.1 Personification of the chatbot :** Giving a custom persona to the chatbot including its name, profile image, general behaviors.

**Task 13.2 Listing accomplished achievements :** Listing the predefined obtained and to be obtained achievements on the user's profile, on a separate tab.

**Task 13.3 Listing badges when conditions are met :** There should be a notification when one of the achievements are accomplished. A badge will be given for each achievement.

**Task 13.4 Leaderboard for chat streak and learned words :** A leaderboard where the user can see their ranking among their friends or in the league they are assigned to based on both their chat streak and the number of learned words.

**Task 13.5 Chat Streak:** A chat streak component displaying the user's current and highest streak with an appealing visual will be developed. It will be displayed on each login. It must also be accessible from the home page.

**Deliverables**

**D13.1:** UI Design

**D13.2:** Code

**D13.3:** Documentation

<b>WP 14: API Gateway, Microservice Architecture, Authentication and Authorization</b>			
<b>Start date:</b> 22.10.2023 <b>End date:</b> 15.12.2023			
<b>Leader:</b>	İlkin Elif Kervan	<b>Members involved:</b>	Yağız Can Aslan
<b>Objectives:</b> Implementing API Gateway and microservice architecture. Configuring API Gateway to direct the incoming requests to related services. Developing security for gateway and microservices. Engineering login and register services and integrating it with the database for authentication and authorization.			
<b>Tasks:</b> <b>Task 14.1 Create API Gateway :</b> Implementing API Gateway with Spring Boot and configuring it to redirect requests to related microservices. <b>Task 14.2 Implement security :</b> Developing login and register services using JWT. Implementing authorization and authentication layers in the API gateway and microservices to secure the endpoints from malicious requests.			
<b>Deliverables</b> <b>D14.1:</b> Spring Boot API Gateway <b>D14.2:</b> Authorization and authentication layers <b>D14.3:</b> Source code <b>D14.4:</b> Code documentation			

<b>WP 15: Setup Jira and Confluence Infrastructures</b>			
<b>Start date:</b> 15.10.2023 <b>End date:</b> 22.10.2023			
<b>Leader:</b>	Yağız Can Aslan	<b>Members involved:</b>	-
<b>Objectives:</b> Setup our issue tracking, version control and bug reporting infrastructures. Setup Jira workflow diagrams and issue types. Publish crucial Confluence workflow guidelines and category-based folders.			
<b>Tasks:</b> <b>Task 15.1 Setup Jira and Issue Types:</b> Setup our issue tracking, version control and bug reporting infrastructures. Setup Jira workflow diagrams and issue types. <b>Task 15.2 Setup Crucial Confluence Documents:</b> Publish crucial Confluence workflow guidelines and category-based folders.			
<b>Deliverables</b> <b>D15.1:</b> Jira URL <b>D15.2:</b> Jira Issue Types <b>D15.3:</b> Confluence URL <b>D15.4:</b> Multiple Confluence Documents & Guidelines			

<b>WP 16: Voice Chat Integration</b>			
<b>Start date:</b> 20.02.2024 <b>End date:</b> 30.03.2024			
<b>Leader:</b>	Kardelen Ceren	<b>Members involved:</b>	İlkim Elif Kervan, Yağız Can Aslan
<b>Objectives:</b> <i>Implementing speech-to-text for the user and text-to-speech for the chatbot. It is important to choose a voice that is as human-sounding as possible.</i>			
<b>Tasks:</b> <b>Task 16.1 Research speech-to-text and text-to-speech libraries :</b> <i>Research libraries and APIs that enable voice chat integration.</i> <b>Task 16.2 Implement speech-to-text :</b> <i>Integrate the speech-to-text libraries to transcribe user's speech to be fed into the main model that generates the response.</i> <b>Task 16.3 Implement text-to-speech :</b> <i>Integrate the text-to-speech libraries to voice chatbot's responses if desired. A human-like voice should be selected.</i>			
<b>Deliverables</b> <b>D16.1:</b> Source code <b>D16.2:</b> Documentation			

<b>WP 17: User Profile Backend Development</b>			
<b>Start date:</b> 20.02.2024 <b>End date:</b> 30.03.2024			
<b>Leader:</b>	Tolga Özgün	<b>Members involved:</b>	Kardelen Ceren, İlkin Elif Kervan
<b>Objectives:</b> <i>Users' profile, i.e. their likes and dislikes and demographic information, will be kept in the database and frequently updated based on their messages. This profile will serve to generate messages from the chatbot that will interest the user to engage in long conversations.</i>			
<b>Tasks:</b> <b>Task 17.1 User Profile Generator LLM Development :</b> <i>Implement an LLM using prompt engineering that outputs the user profile when an old profile and user's latest messages are given.</i> <b>Task 17.2 User Profile Backend Development:</b> <i>Implement the Controller - Service - Repository classes in Java Spring, modeled object classes and database schemas for the user profile system by modifying the existing user model.</i> <b>Task 17.3 Research and Improve:</b> <i>Research to find better methodologies which may include leveraging agents, tools and chaining.</i>			
<b>Deliverables</b> <b>D17.1:</b> Source code <b>D17.2:</b> Documentation <b>D17.3:</b> LLM			

### 4.3.1 Gantt Chart

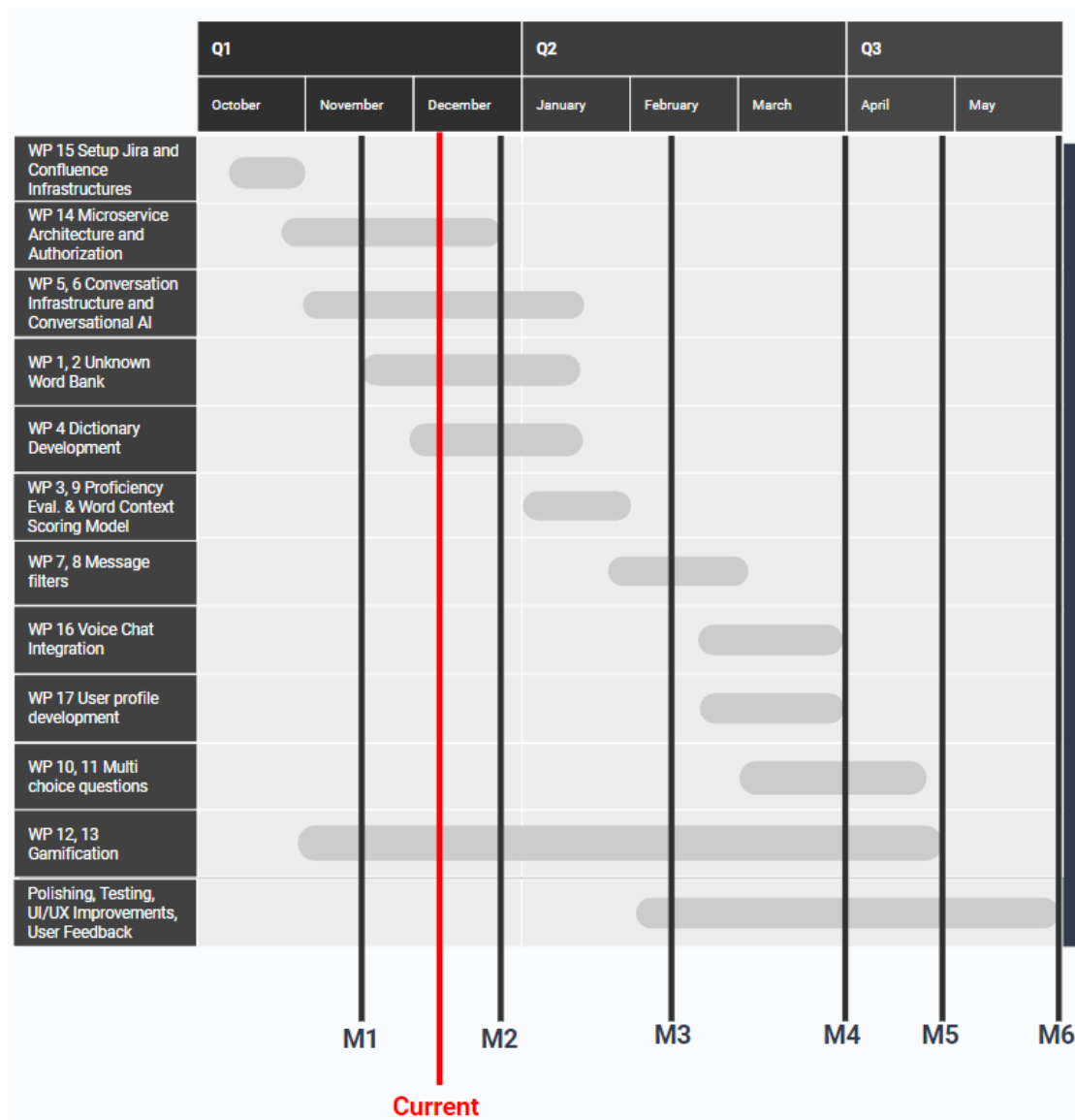


Fig. 26. Linguist AI Gantt Chart

#### Explanation of Milestones;

- **M1:** First Minimum Viable Product (MVP) Ready
- **M2:** Fall Semester Demo Version Ready
- **M3:** Ready for General User Beta Testing
- **M4:** All Main Features are Implemented
- **M5:** All Gamification Features are Ready for Testing
- **M6:** CS Fair Demo Version Ready

## 4.4 Ensuring Proper Teamwork

Proper teamwork is a crucial asset for any software development team. Our team consists of 5 senior Computer Engineering students who work on different and busy schedules. Hence, it is crucial for us to properly coordinate our teamwork and establish the necessary guidelines and framework to adequately distribute the workload and project responsibilities, while also keeping in touch.

To achieve the aforementioned goals, we employ various industry-proven and widely integrated methodologies, tools, and software in our development process. Some notable examples include but are not limited to;

- Jira, Confluence, Google Drive, Git, GitHub, and GitHub Issues for issue tracking, version control, repository hosting, common documentation, bug reporting and other technical and development/planning oriented processes,
- Discord, WhatsApp and Zoom for constant communication and online meeting platforms and,
- Weekly in-person progress meetings with our supervisor.

We also follow a tailored Scrum workflow configured to work with our schedules. We decided to follow the Scrum workflow as it was deemed to be more efficient to be a self-organized team. Our development team adheres to 2-week-long Scrum cycles, where we hold regular “stand up” meetings to discuss our progress, problems and plans. We hold Sprint review, retrospective and planning meetings at the end of each Sprint. Moreover, in order to streamline our development process, we have set up tailored pull request and commit conventions. For more information, see the figures below for the Confluence pages of the conventions, and some sample screenshots from Jira and GitHub.

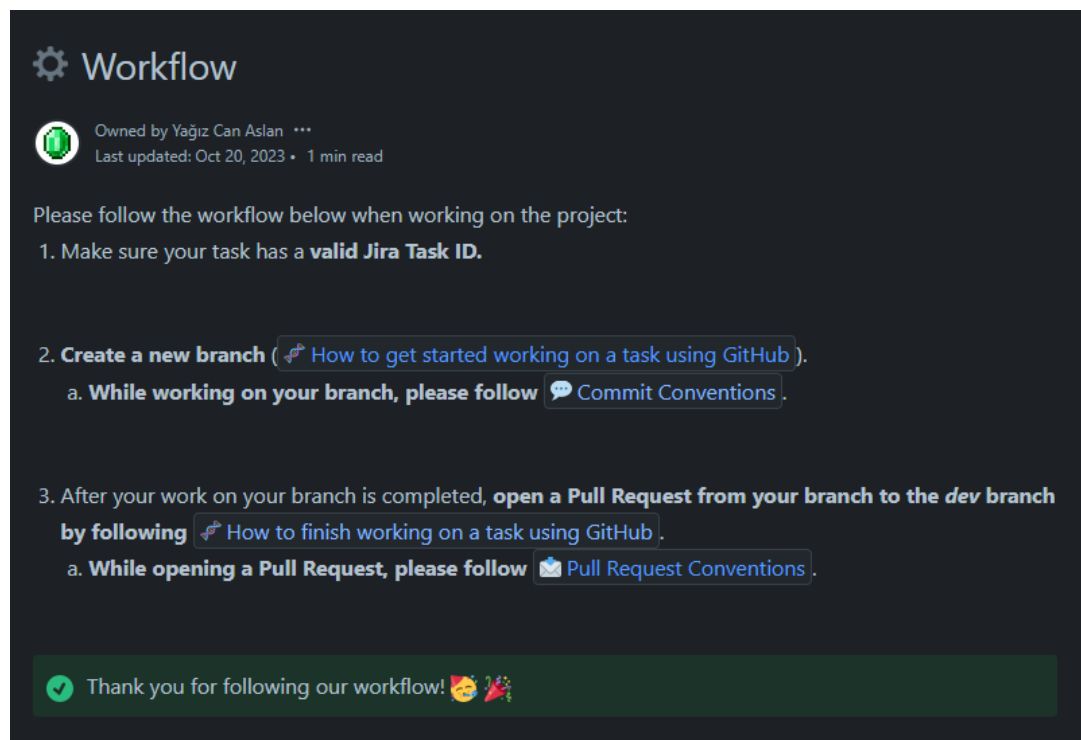


Fig. 27. Linguist AI Workflow

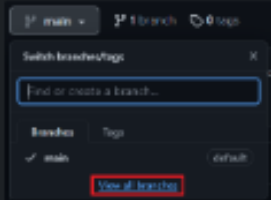
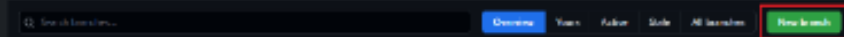
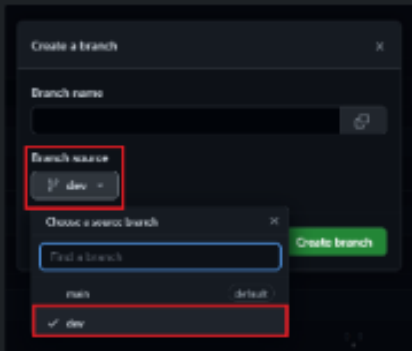
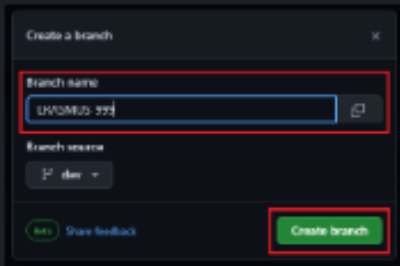


## How to get started working on a task using GitHub

Owned by Yagiz Can Aslan · Last updated: Oct 20, 2023 · 2 min read

This short document demonstrates how to get started working on a task using new branches in GitHub.


### Instructions

- Go to our project's GitHub repository, locate and click "View all branches".
 
- A page showing all active branches will open. Click "New branch".
 
- Select the branch source as "dev".
 
- Name your branch with the name of your Jira Task ID (example: **LINGUIST-999**, NOT **ERASMUS-999**) and click "Create branch".
 
- Congratulations! You can now get started working on your task from the new branch 🎉

**📌** This document only covers how to get started working on a task using GitHub. The document for how to proceed upon task completion is here: [How to finish working on a task using GitHub](#).

Fig. 28. Linguist AI Branch Creation Instructions

## Commit Conventions

 Owned by Yağız Can Aslan \*\*\*  
Last updated: Oct 20, 2023 • 1 min read


Please follow the conventions below when committing your changes to our GitHub repository.

Use the commit naming "style":

- 🚀 **"feature(LINGUIST-9999): Jira task title/task description"**: when implementing *\*new\* features*.
- 🛠️ **"chore(LINGUIST-9999): Jira task title/task description"**: when making *small changes* to existing code segments/comments.
- 🔧 **"fix(LINGUIST-9999): Jira task title/task description"**: when fixing *bugs/errors/typos*.
- 🧪 **"test(LINGUIST-9999): Jira task title/task description"**: when *implementing/changing test cases*.
- 🧑‍🔧 **"refactor(LINGUIST-9999): Jira task title/task description"**: when *changing/improving/removing existing feature implementations*.
- 💎 **"misc(LINGUIST-9999): Jira task title/task description"**: any other type of commit that is *not* included in this list.

Fig. 29. Linguist AI Commit Conventions

## How to finish working on a task using GitHub

 Owned by Yağız Can Aslan \*\*\*  
Last updated: Oct 20, 2023 • 1 min read

When you finish working on a task and commit your changes to your respective branch, you can open a **"Pull Request (PR)"**, where you can request your branch "LINGUIST-9999" to be merged into the branch "dev", and request code review(s).

Once the reviewer(s) approve your PR, your branch can be successfully merged into the "dev" branch (if no merge conflict(s) exist).



 Please make sure your Pull Request follows [Pull Request Conventions](#) before posting.

Fig. 30. Linguist AI Task Finishing Instructions

## Pull Request Conventions

Owned by Yağız Can Aslan •••  
Last updated: Oct 20, 2023 • 1 min read

Please make sure your Pull Request (PR) includes the following sections (which will be provided by a file called "pull\_request\_template.md"):

- **Description:** A short description of the PR, and changes made.
- **Checklists:** This is just a header, no explanation is needed here.
  - **Development and Testing**
    - Related unit test(s) have been implemented and/or updated: Yes/No
    - Application changes have been tested: Yes/No
    - Related documents have been updated (if any): Yes/No
  - **Security**
    - Security impact and vulnerabilities of the change(s) have been considered: Yes/No
  - **Time and Space Complexity**
    - Time complexities of algorithms are not above  $O(N^3)^*$  and are considered carefully: Yes/No
    - Space complexities of algorithms are considered carefully: Yes/No
  - **Code Review**
    - PR has a valid Jira Task ID: Yes/No
    - PR title follows  **Commit Conventions**, and is succeeded by a short explanation: Yes/No

**feat(ERASMUS-87): Re-organize the project structure for types and tests.**


 **Merged** can-aslan merged 5 commits into `dev` from `ERASMUS-87` 17 days ago

Figure 1: Example PR Title


 Make sure task id starts with "LINGUIST", not "ERASMUS".

Fig. 31. Linguist AI Pull Request Conventions

## How to create and merge branches

1. Create your branch from the `dev` branch, as stated in our [getting started on a task Confluence guide](#).
2. When your work is done, merge back to the `dev` branch, as stated in our [finishing work on a task Confluence guide](#).
3. After a few merges to the `dev` branch, open a PR to the `test` branch. The `test` branch is utilized for collectively testing the interactions of a variety of features before merging to the `main` branch.
4. When tests are successful, open a PR to the `main` branch from the `test` branch.

**dev** branch: *the branch which we work from.*

**test** branch: *the branch where we merge our changes and new features to test.*

**main** branch: *the branch which has the tested and working features.*

Fig. 32. Linguist AI Branch Instructions

🔴 LINGUIST-118 Bottom Navigation Icons Are Not Clickable	MISCELLANEOUS BUGS	TO DO ▾	- ▾ ⚙️
🔴 LINGUIST-121 Determine LLM evaluation metrics and dataset	CONVERSATIONAL AI	TO DO ▾	- = KC
🔴 LINGUIST-122 Research LLM deployment library	CONVERSATIONAL AI	TO DO ▾	- = KC
🔴 LINGUIST-123 API Gateway Security Research	MICROSERVICE ARCHIT...	IN PROGRESS ▾	- = 🧑
🔵 LINGUIST-125 Configure API Gateway	MICROSERVICE ARCHIT...	IN PROGRESS ▾	- = 🧑
🔴 LINGUIST-124 Fork Dictionary API and Examine How It Works	WORD DICTIONARY	TESTING ▾	- = 🧑
🔵 LINGUIST-128 Unknown Word Bank Backend Development	UNKNOWN WORD BANK	ON HOLD ▾	- ⬆️ 🧑
📄 LINGUIST-134 Requirements Report - Introduction	REQUIREMENTS REPORT	TO DO ▾	🕒 08 DEC - = 🧑
📄 LINGUIST-135 Requirements Report - Current System	REQUIREMENTS REPORT	TO DO ▾	🕒 08 DEC - = 🧑
📄 LINGUIST-136 Requirements Report - Proposed System	REQUIREMENTS REPORT	TO DO ▾	🕒 08 DEC - = 🧑
📄 LINGUIST-137 Requirements Report - Other Analysis Elements	REQUIREMENTS REPORT	TO DO ▾	🕒 08 DEC - = 🧑
🔴 LINGUIST-141 Register Does Not Validate Email and Password Strength	MISCELLANEOUS BUGS	TO DO ▾	- = 🧑
🔵 LINGUIST-142 Design Conversation UML Diagrams	REQUIREMENTS REPORT	IN PROGRESS ▾	- ⬆️ 🧑
🔵 LINGUIST-165 Design Gamification (Streak, Leaderboard, Friends) UML Diagrams	REQUIREMENTS REPORT	TO DO ▾	- ⬆️ 🧑
🔴 LINGUIST-174 Examine Different Unknown Word Mapping Solutions	UNKNOWN WORD BANK	TO DO ▾	- ⬆️ 🧑
🔴 LINGUIST-175 Remove Unnecessary userId Field in UserStreak	GAMIFICATION V1	TO DO ▾	- ▾ 🧑
✅ LINGUIST-176 Organize Jira Tasks and Report Title Formatting for Requirements Report	REQUIREMENTS REPORT	DONE ▾	- = 🧑

Fig. 33. Sample Jira Sprint Backlog

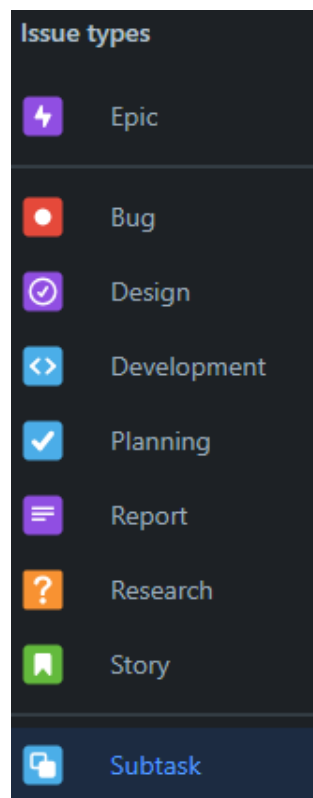



Fig. 34. Jira Issue Types

## Add dynamic language support (i.e. i18n) #16

Open

can-aslan opened this issue on Oct 24 · 7 comments



can-aslan

commented on Oct 24 · edited

Member


...


Instead of having string literals and/or other hard-typed information shown to users, we can find and use a more structured way that;

- allows for *easier multiple language support*,
- and for *all messages to be stored together*.

<https://www.i18next.com/>


+ Maintainability, + Reliability, + Supportability, + Accessibility

 1



can-aslan

added the enhancement label on Oct 24




tolgaozgun

commented on Oct 24

Member

...

Applies to both mobile and spring. [@can-aslan](#) would you take the task for technic implementation of localization for React Native part?




tolgaozgun

commented on Oct 24

Member


...

[@KardelenCeren](#) would you be interested in localization in the Spring part?



tolgaozgun

added mobile backend labels on Oct 24



can-aslan


commented on Oct 24

Member

Author

...

[@tolgaozgun](#) I have experience with i18n in Spring and React. I can setup both.



can-aslan

commented on Oct 24 · edited by jira bot

Member

Author

...

Will be handled with [LINGUIST-62](#) and [LINGUIST-63](#).

Assignees

No one assigned

Labels

backend

enhancement

mobile

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

Notifications

Customize

Unsubscribe

You're receiving notifications because you were mentioned.

3 participants






Fig. 35. Sample GitHub Issue Conversation

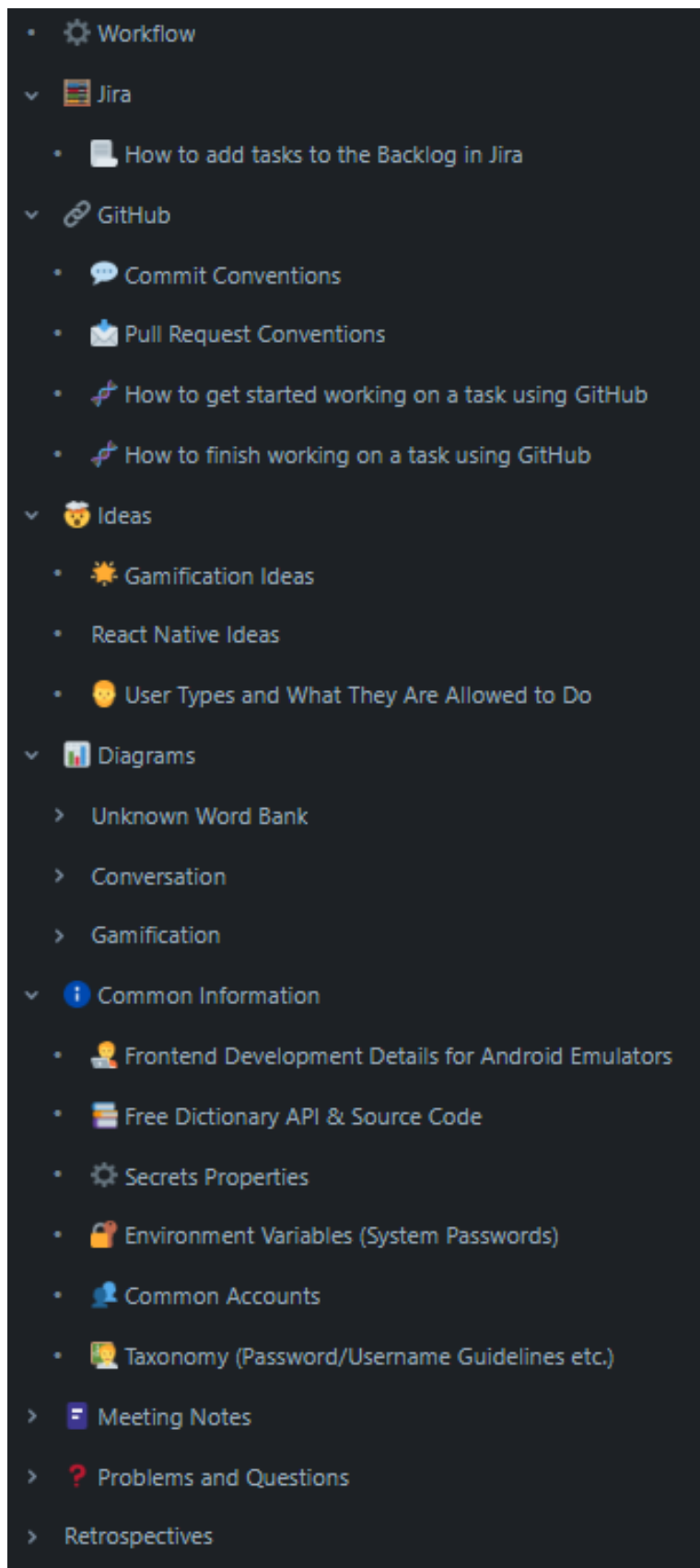


Fig. 36. Confluence Pages Summary Overview

## **4.5 Ethics and Professional Responsibilities**

### **4.5.1 Data Privacy**

In order to teach the user's in the best possible way, Linguist AI uses personalization. Personalization involves storing certain data of users to be utilized later. We understand that as the developers, it is our responsibility to be transparent about clearly telling users what data is being collected, how it is being used, and with whom it is being shared. Therefore, we are aware that a detailed privacy policy within the app is essential.

Being transparent is the first part of the process. It is important that the user explicitly consent to data collection and use for the sake of personalization. If the user does not consent, the data should not be used or stored in any way for any purpose. Since one of our core features includes personalized conversations with the chatbot, we should explicitly state this and warn the user that the performance of the chatbot might not be as satisfying as possible in case no personal data is used.

We are also aware that we must implement security measures to prevent unauthorized access and use to protect user data, on top of transparency and consent.

### **4.5.2. Responsible Chatbot Responses**

We are aware that the chatbot might produce inaccurate or controversial responses due to the nature of LLMs. We are going to implement mechanisms such as prompt engineering to prevent chatbot from producing such outcomes. We will also add a filter layer to the response of the LLM, so that we can prevent any content that is produced despite prompt engineering. The filtering layer will filter out potentially offensive, discriminatory or harmful content.

Moreover, we will explicitly communicate with the user's that they are talking with a machine and not a human. Thereby, the user won't be misled by chatbot's capabilities.

### **4.5.3. Ethical Topic Filtering**

We are going to determine sensitive topics such as hate speech, political extremism, discriminatory behavior, sexism, etc. and tell the chatbot to generally avoid speaking in conversations. We will also add an additional filter that will prevent the model from discussing sensitive topics by using keyword identification or sentiment analysis.

### **4.5.4. Professional Responsibilities**

As the developers, we know that being transparent with the users with any ethical concerns, or responding quickly to incidents reported by them is crucial. In that regard, we believe that continuous improvement will result in a better system overall. We will also comply with data privacy regulations such as GDPR. We believe sticking to such principles is really important in the development process.

#### 4.5.5. Making Sure the Promised Methods Work

Our main promise with Linguist AI is that we are teaching the users words they do not know by either implicitly inferring the words they don't know or explicitly having user input about words they do not know. As professionals, we should make sure we offer what we promise. In that regard, we aim to implement mechanisms to measure the "learning level" for each word by using methods such as multiple-choice question tests and using LLMs to measure users' understanding of the words they don't know. In that way, we can calculate statistics about the level of users' improvement.

### 4.6 Planning for New Knowledge and Learning Strategies

After designing the core purpose of our project and defining its boundaries, we came up with technical and non-technical areas that we need to have proper knowledge on to continue building our project.

#### 4.6.1 Technical Areas

- Developing a mobile application
  - None of the team members had previous experience in building a mobile application. After defining the purpose and the boundaries of our project, we decided on the main parts each of the members will work on. After that, those who will work directly in building the mobile application planned learning about mobile application development. We chose to use React Native to build the mobile application. For that purpose, we planned to follow online learning methods, and during this semester we took a Udemy course about using React Native for mobile application development. Our approach for the future is more on the side of learning by doing the project and teaching us concepts that we do not know.
- Designing a scalable architecture for our system
  - To make sure that we will have a scalable architecture, we decided that we needed to know more about the current architectural approaches and their details. For that purpose, we assigned team members to learn more about microservice architecture, network communication methods used in microservice architecture such as gRPC, HTTP, and messaging queues, and which is the most feasible one in terms of our structure. We plan to learn more about this issue in the future by reading blog posts about software architectures from companies like Microsoft and reading more about them. We also decided that our architecture will change based on our needs, so we will learn more by doing it as well.
- Choosing the most suitable database(s) for our architecture
  - Since our application included a chatting system with an LLM we planned to conduct research on how similar systems utilize different database systems. We did an initial research on how Discord uses NoSQL databases by reading their analysis of different approaches. We plan to iteratively learn more about how large-scale systems such as Discord work by using online learning methods.



- Measuring the success of how well the application teaches words
  - We are planning to find a method to measure our success in teaching words by discussing it with our supervisor. Currently we have two methods for measuring our success: using an LLM to grade the use of certain words by the user, making multiple-choice question tests at the end of conversation sessions. We plan to learn more about how well these approaches work by getting feedback from actual users.
- Measuring the English level of a user
  - While defining the scope of our application we had a meeting with Tijen Akşit, FAE Program Director. We received feedback about the usability of our application and how we can classify the level of English of the users. We plan to have another meeting with them to learn more about which information we can use to develop a system to measure the English level of a user.

#### 4.6.2 Non-technical areas

- Learning more about language teaching in casual settings
  - We had a meeting with Özlem Doğan, who is an instructor of the Spanish courses in Bilkent. We discussed the process of language learning and how we can design our application in a way that the process is natural to the users so they don't feel like they are memorizing words. Also, we talked about the factors that can attract learners to keep using our app.
- Creating a unique visual language for the best user experience
  - We had an initial meeting with Jülide Akşiyote Görür, from the COMD department, about building a visual language for our application. We plan to meet with them after building a sample visual language of our own to learn more about how we can do better in that regard.

## 5 Glossary

- Large Language Model (LLM): Complex artificial intelligence systems that can understand and generate human-like text. These models are trained on massive amounts of data to understand language patterns, semantics, and context. As a result, LLMs produce coherent and contextually relevant text, making them useful for tasks such as chatbot conversations, content production, and language translation. Most notable LLMs include OpenAI's ChatGPT, Meta's Llama, Google's BERT and PaLM.
- Gamification: The incorporation of game-like elements, such as points and rewards, into non-game contexts in order to increase user engagement and motivation. It aims to create an interactive and enjoyable experience by applying game design principles, encouraging users to actively participate and achieve goals within the app.

## 6 References

- [1] E. H. Dyvik, "The most spoken languages worldwide 2023," Statista. [Online]. Available: <https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/> [Accessed: Nov 14, 2023].
- [2] "Learn a language for free," Duolingo. [Online]. Available: <https://www.duolingo.com/> [Accessed: Oct 27, 2023].
- [3] "Online English classes with native speakers," Cambly. [Online]. Available: <https://www.cambly.com/english?lang=en> [Accessed: Oct 27, 2023].
- [4] S. Bibauw, T. François, and P. Desmet, "Discussing with a computer to practice a foreign language: Research synthesis and conceptual framework of dialogue-based call," *Computer Assisted Language Learning*, vol. 32, no. 8, pp. 827–877, 2019. [Online]. Available: <https://doi.org/10.1080/09588221.2018.1535508>. [Accessed: Nov 16, 2023].
- [5] N.-Y. Kim, "Chatbots and Korean EFL Students' English Vocabulary Learning," *Journal of Digital Convergence*, vol. 16, no. 2, pp. 1–7, 2018. [Online]. Available: <https://doi.org/10.14400/JDC.2018.16.2.001>. [Accessed: Nov 16, 2023].
- [6] M.-H. Hsu, P.-S. Chen, and C.-S. Yu, "Proposing a task-oriented chatbot system for EFL learners speaking practice," *Interactive Learning Environments*, vol. 31, no. 7, pp. 4297–4308, 2021. [Online]. Available: <https://doi.org/10.1080/10494820.2021.1960864>. [Accessed: Nov 16, 2023].
- [7] L. K. Fryer et al., "Bots for language learning now: Current and future directions," *Language Learning & Technology*, vol. 24, no.2, pp. 8-22, Jun. 2020. [Online]. Available: <https://scholarspace.manoa.hawaii.edu/server/api/core/bitstreams/950396a1-e7a1-4eac-bd27-c3d194ce77e2/content>. [Accessed: Nov 16, 2023].
- [8] S. Lee et al., "On the effectiveness of robot-assisted language learning," *ReCALL*, vol. 23, no. 1, pp. 25–58, 2011. [Online]. Available: <https://doi.org/10.1017/s0958344010000273>. [Accessed: Nov 16, 2023].