

Title of the project: AI-Ym Mobile Application

Team Members: Ayaulym Sayakova, Anel Anuash, Aliya Shalabayeva, Nargiza

Iminova, Dana Zhunussova

GROUP 12

Project Advisor/Co-Advisors: Associate Professor Anara Sandygulova

Co-supervisor: Marat Isteleyev

1. Executive Summary

AI-YM is a project that helps make Kazakh Sign Language learning and communication more accessible. Many learners of Kazakh Sign Language lack a quick and convenient way to learn sign language. It was observed that there is a lack of convenient tools that bring together learning materials, a dictionary, practice exercises, and translation support in one place. The project's central goal is to combine text-to-sign translation (TTS), sign-to-text recognition (STT), a sign language dictionary, and learning courses with gamified quizzes in one mobile application.

The project followed an iterative development approach during the 2 semesters. The team analyzed the backend and sign language resources, designed the application structure and user interface, and implemented the main application modules. After that, the team added to the system a language practice module and improved integration with backend services. Also, the team refined the translation interface, redesigned the animation approach for text-to-sign output, and created an avatar generation pipeline.

The mobile client was built using Flutter. It is being connected to backend services through APIs for such purposes as user authentication, course content, dictionary data, quizzes, and translation-related processing. The app is using Riverpod to manage state and Dio to handle API requests. It also communicates with backend services developed in Node.js and Express, with PostgreSQL & MySQL for data storage. For sign recognition and avatar generation, external tools were also integrated. As a result, the project delivers a working prototype of a mobile application that provides an opportunity for convenient learning, translating, and practicing sign language in a single app.

2. Introduction

The project focuses on a problem that deaf, hard-of-hearing people, as well as other people who want to use a sign language face on a regular basis. In Kazakhstan, tools for Kazakh Sign Language are scattered across different platforms, videos, dictionaries, and other learning materials. As a result, people do not have a convenient tool that combines vocabulary learning, structured lessons, practice, and translation support for the sign language in one single place.

The project aims to reduce this gap by providing an application where users can learn signs, search for vocabulary, practice through quizzes, and use translation-related tools in one centralized application. The system gives a learning environment that can help users access sign-language materials more easily and interact with Kazakh Sign Language through a user-friendly mobile interface.

The target users are:

- Deaf and hard-of-hearing people who need convenient tools for communication and learning.
- Hearing people who want to use Kazakh Sign Language for communication with Deaf and hard-of-hearing people.
- Educators who want to have get an access to a convenient dictionary and learning resources.

The system consists of such entities:

- A mobile application that provides the user interface on Android and iOS.
- Backend services to manage user authentication & data, learning courses, dictionary content, quizzes, and translation-related requests.
- A translation module that supports text-to-sign and sign-to-text translation.
- A dictionary and learning module that provides vocabulary, visual materials, lessons, and progress tracking.
- A practice module that uses quizzes, XP, and streaks to support repeated learning.

By bringing together learning content, dictionary, STT & TTS translation & animation, the project makes a step towards a more inclusive environment for Kazakh Sign Language users and learners.

3. Identification of constraints

During the development of the AI-YM application various constraints were considered. The implementation of the AI-YM application was influenced by several economic, social, ethical, and technical factors. Consequently, main design decisions and implementation choices were affected.

Starting with the **economic and resources constraints**: as a student project, our application was developed with limited financial resources. Since we could not afford premium cloud hostings and high quality avatar animation creation tools, our main goal was to create an accessible yet working application. As a result, NodeJs based microservice and Media-Pipe/Blender based sign recognition and avatar animation choices were utilized. However, future scaling of the application for larger user groups will require investments in data storage and maintenance.

Moving forward to the **social constraints**, our main audience include both hard-of-hearing individuals and hearing users. Thus, designing an interface that is intuitive for all users is a key. Additionally, various user needs and familiarity with technology also affect the adoption of the app. To address this constraint, our AI-YM mobile application includes features that cover the needs of both deaf and hearing individuals.

Ethical constraints include the privacy of user's content and copyright issues of media assets. To implement a fully functional application that bridges the communication gap between deaf and hearing individuals real-time videos and pre-recorded video lessons were used. Those videos are part of the "Translation" and "Learning" pages and considered as sensitive and personal data. Our system must ensure data privacy and avoid any misuse of the collected information. Next ethical constraint covers the copyright issue, in particular, all videos and datasets used for "Dictionary" and "Learning" modules are owned by the Human-Robot-Interaction laboratory within Nazarbayev University. This content ownership was highly considered while designing the AI-YM application.

Our AI-YM project operates within Kazakhstan, and is constrained with **political/legal factors**. Since the project's main goal was to support local sign language, Kazakh-Russian Sign Language (K-RSL) datasets were prioritized. The application was developed in compliance with legal regulations. Storage and user data such as account details, in-app recorded-videos, progress records strictly followed by the Law of the Republic of Kazakhstan on "Personal Data and Its Protection". All the provided educational materials within the application are subject to educational standards. To ensure that these materials meet the required norms, all dictionary entries and course lectures were validated by Kazakh Sign Language experts.

Health and Safety constraints were also identified. A major health and safety constraint is directly related to the real-time translation accuracy, especially during the emergency situations. If the deaf or hearing user of the application will rely on our application in urgent situations and translation is wrong, then it could lead to negative consequences. For this reason, we implemented a Welcome screen as a disclaimer that our application serves as an educational platform.

Last and one of the most important constraints include the **sustainability constraints**. The AI-YM project promotes social inclusion and education which can significantly contribute to society. Its main goal is to provide an equally accessible application to people with and without disabilities. This mission will definitely remain relevant to reduce communication barriers over time. Continuous maintenance of the application will preserve a linguistic heritage and improve inclusive interactions. To keep the technical sustainability, Flutter SDK that provides codebase for both Android and iOS devices was applied. This choice will allow future students at Nazarbayev University to consistently maintain the app with future updates.

Overall, the above-mentioned economic, social, ethical, political, health and sustainability constraints influenced design and functionality decisions.

4. Background

4.1 Background

The World Health Organization estimates that over 430 million individuals globally have disabling hearing loss - almost 5 per cent of the world population (World Health Organization, 2025). To the majority of these people, sign language is the main form of communication: a visual-gestural language, rich in signs, based on handshapes, movements, facial expressions, and posture instead of sound (Bragg et al., 2019; Imashev et al., 2024). Sign languages are not universal and each nation forms its own linguistic system. The major visual-gestural language in Kazakhstan is the Kazakh-Russian Sign Language (K-RSL). It has been estimated that there are over 30,800 hearing-impaired people in Kazakhstan (Zholshiyeva et al., 2025), and although there are sign language interpreters who act as communication bridges, their population is not enough to cater to the daily needs.

The current smartphones and AI-driven applications are an interesting chance to fill this gap due to their ubiquity and multimedia processing in real-time (David et al., 2024a). Nevertheless, the K-RSL digital space is not yet well-developed - the majority of the current sign language technologies are focused on popular languages like American Sign Language, whereas K-RSL is not given much attention in the academic literature and commercial software.

The initial AI-YM system, created at Nazarbayev University during the previous academic year, was an online K-RSL recognition, translation, and learning system. Although it proved machine learning to be feasible in processing Kazakh sign language, its web-based nature restricted its use in real-life scenarios. The present project builds upon the AI-YM ecosystem by creating a cross-platform mobile application with Flutter, connecting it with the existing backend services, and creating a new gamified Practice module and two separate avatar development projects as detailed in the sections below.

4.2 Related Work and Literature Review.

4.2.1 Sign language recognition and translation systems.

Sign language processing encompasses isolated sign recognition, continuous recognition, translation, and sign production. The interdisciplinary review by Bragg et al. (2019) highlights the importance of convergence of artificial intelligence, linguistics, HCI, and Deaf community engagement to make effective systems. One common observation is that technically competent systems do not always work in practice because of the lack of consideration of linguistic subtleties and cultural context.

MediaPipe is a cross-platform framework by Google, and the heart of many modern recognition systems, offering hardware-independent body pose, hand, and facial landmark extraction of video (Lugaresi et al., 2019). MediaPipe Holistic is used in the AI-YM platform to identify full-body pose landmarks (33 points) and hand landmarks (21 points per hand). These

keypoints are then normalized, joint angles processed and inputted as speed-invariant features in the sign recognition model. In the case of K-RSL, in particular, the FluentSigners-50 dataset, which includes over 43,250 annotated samples of everyday K-RSL use by the local Deaf community members (Mukushev et al., 2022) , has been a key to training and benchmarking recognition models on the platform.

4.2.2 Sign Language Mobile Applications.

David et al. (2024b) conducted a systematic review of 45 articles on sign language mobile applications that were divided into education-focused, translation/communication-focused, and recognition-focused articles. Gamification has been shown to be effective in enhancing engagement and retention of learning through education-based applications like E-Tutor, FING, and ASL Bloom (ASL Bloom, n.d.). ASL Bloom specifically inspired the strategy of the AI-YM team to track progress and component-based UI design. Applications that concentrate on translation use CNNs, 3D avatars (Sign4Me, DeafWorld), and linguistic representation systems like HamNoSys and SigML. Other systems rely on additional hardware like smart gloves or smartwatches, which pose cost and portability challenges. The AI-YM mobile app does not have such dependencies, as it only uses the smartphone camera.

A similar precedent can be found in the Shuwa/SignTown project (The Chinese University of Hong Kong, 2024), created at the Chinese University of Hong Kong, which proposes an AI-based sign language learning system, where real-time gesture recognition and feedback are provided using regular webcams, which proves the feasibility of hardware-independent and accessible solutions. This was the direct informant to the AI-YM team to focus on the standard smartphone cameras as the only capture device.

4.2.3 Text-to-Sign Translation Avatar (MediaPipe + Pygame) in Real-Time.

One of the most technically challenging parts of a sign language platform is sign language production, the process of creating sign language output based on text using an animated avatar. Initial methods used stick-figure drawings or video clips that were pre-recorded, both of which lacked expressiveness and vocabulary. The initial AI-YM web platform employed a stick-figure Text-to-Sign module, which, though practical, was not as visually natural as needed to facilitate effective communication.

This is substituted by the present project of a real-time avatar system that is based on MediaPipe Holistic and Pygame (Lugaresi et al., 2019). As the user types, the backend looks up the sign keypoint information and the application displays an animated, human-like avatar by moving a skeleton using the extracted full-body and hand landmarks. One of the main issues in this work was reverse-engineering the legacy microservice pipeline (which processes input by using a chain of text input, AI model lookups, .pose file assembly, and avatar rendering) since it

used an undocumented binary .pose format, and there was no direct mapping to the landmark schema of MediaPipe. The team unraveled the binary format and created a new animation script that converts old joint representations to standard MediaPipe landmarks, allowing it to be smoothly integrated with the existing backend without affecting other services.

This method creates a more expressive and linguistically correct avatar than the old stick-figure system, and is lightweight enough to play back in real-time on a mobile at the desired 2030 FPS. Imashev et al. (2024) emphasized the role of Deaf community in the assessment of avatars quality in K-RSL; this fact influenced the choice of the team to make Deaf and hard-of-hearing users the main participants in usability testing of the new avatar.

4.2.4 Avatar Video Regeneration Pipeline (SMPL-X +HaMeR + Blender)

In addition to the real-time translation avatar, a more long-term project is being undertaken to recreate the current database of K-RSL interpreter videos as avatar animations. The concept is to substitute the raw video of human interpreters with a uniform avatar image of all dictionary material. The pipeline is operated in three phases. To estimate body pose, first, each interpreter video is estimated with SMPL-X (Pavlakos et al., 2019), a parametric human body model that represents the torso, limbs, and facial expression as a small set of shape and pose parameters. But since sign language is hands-first, SMPL-X does not offer enough hand detail to be used in this case. To overcome this, the pipeline uses HaMeR (Pavlakos et al., 2024), a transformer-based Hand Mesh Recovery model that recovers detailed 3D hand meshes using monocular video with much higher accuracy than general-purpose body models. The estimated hand parameters of HaMeR are then combined into the SMPL-X representation, resulting in a single full-body pose sequence in which the hands are controlled by the more accurate estimates of HaMeR instead of the defaults of SMPL-X. These combined pose sequences are then imported into Blender in the last step and animated using a rigged humanoid avatar mesh and rendered as video to be used in place of the raw interpreter footage throughout the Dictionary module. This pipeline is still being actively developed and has not yet been integrated into the platform. Its finalization will be a milestone in the future and initial outputs will be discussed with the K-RSL experts and members of the Deaf community prior to implementation.

4.2.5 Mobile Application Frameworks and Microservice Architecture.

Flutter was chosen as the mobile development platform because it can be compiled to both Android and iOS to reach as many types of devices as possible. The backend is microservices-based: Sign-to-Text (Flask/Python), Text-to-Sign (Flask/Python), Dictionary (Node.js/MySQL), and Learn and Practice modules (Node.js/PostgreSQL) are independent services, which are connected to a central NGINX API gateway. This pattern facilitates independent scalability and resilience, and runs on ps.kz infrastructure to serve K-RSL services in a locally-operated environment.

4.2.6 Gamification in Learning Language.

The design of the Practice module is based on gamified language learning apps like Duolingo, whose experience point system, streak tracking, and multiple-choice test have been demonstrated to enhance user engagement and retention of learning. The AI-YM Practice module implements these principles on K-RSL vocabulary: users earn points on correct answers, streaks, and browse the content by the same thematic categories as the Dictionary module - a self-reinforcing learning experience across Dictionary, Learn and Practice in a single mobile experience.

4.3 Overview and Rationale of Methodology.

The analysis of the existing literature shows that there is a certain trend: although considerable technical advancements have been achieved in sign language recognition and translation, the current solutions to the underrepresented languages like K-RSL are either not integrated, hardware-specific, or not mobile enough. The AI-YM mobile application is a solution to this by providing a mobile-first design that combines translation, dictionary, learning, and gamified practice. Two complementary avatar development projects, a real-time MediaPipe/Pygame translation avatar and an offline SMPL-X/Blender video regeneration pipeline, directly overcome the long-standing shortcoming of low-quality sign animations in K-RSL systems. Collectively, the selected methodology is supported by the literature, the experience of the first-generation AI-YM web platform, and the real communicative and educative requirements of the Deaf and hard-of-hearing population in Kazakhstan.

5. Project Approach

5.1 System Overview and Architecture

The proposed solution is a cross-platform mobile application designed to provide accessible, real-time communication and learning tools for Kazakh Sign Language (KSL). The system follows a client-server architecture with modular microservices, ensuring scalability, maintainability, and efficient feature integration.

On a high level, the system is comprised of:

- **Frontend (Mobile Client):** It is created in Flutter, which allows a single codebase on both Android and iOS platforms.
- **Backend Layer:** A set of independent services responsible for translation, authentication, dictionary, learning, and practice modules

- **API Gateway:** This is a key routing element, which facilitates all incoming traffic to the mobile application in the form of HTTP requests, and routes each request to the relevant service.
- **Databases and AI Models:** Units of storage and processing of user data, dictionary entries, quiz progress, and sign language recognition.

The mobile client can only communicate via the API gateway as indicated in the architecture diagram (Figure 1) hence separation of concerns is achieved and each of the backend modules can be scaled independently.

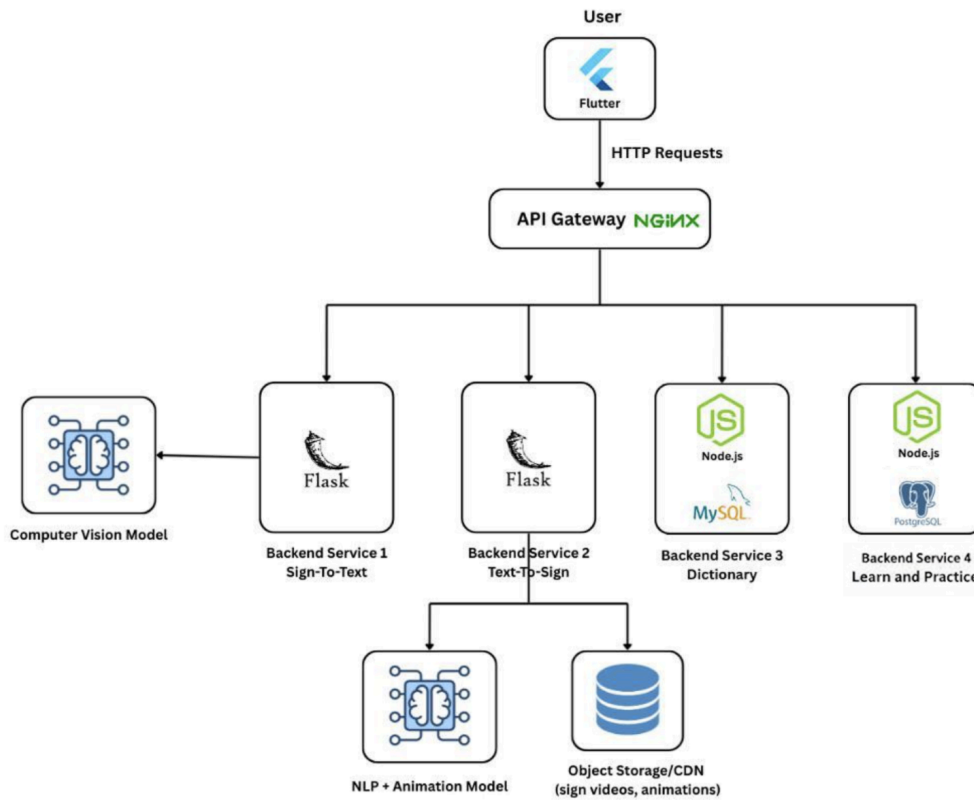


Figure 1: Architecture Diagram

This architecture enables efficient handling of multiple features simultaneously, easy integration of AI-based services (sign recognition and animation), independent development and deployment of system components.

5.2 Core Features and Functional Modules

The system is organized into several key functional modules:

1. Translation Module

- Sign-to-Text: Captures the real-time camera data and transmits them to a computer vision model to identify gestures.
- Text-to-Sign: Converts user input text into animated sign language representations

An important deliverable of this project is the custom animation pipeline that produces user-friendly sign representations through the use of motion processing of keypoints rather than non-animated or ambiguous images.

2. Dictionary Module

- Provides searchable access to sign language vocabulary
- Displays word definitions along with visual demonstrations
- Organized to support both learning and translation workflows

3. Practice Module (Gamified Learning)

Created as an independent module to fill the gap in interactive learning.

Includes:

- Multiple-choice quizzes
- XP-based scoring system
- Streak tracking to encourage daily engagement
- Topic-based organization aligned with dictionary content

The workflow of this module (Figure 2) shows how quiz requests are processed, validated, and scored through a dedicated service and database .

4. Learn Module

- Delivers educational material in a structured way in the form of video lectures.
- Tracks user progress and completion
- Supports certification upon course completion

5. User Management Module

- Secure authentication and authorization with email verification
- Profile management
- Progress tracking across all modules

The system ensures secure handling of user credentials using token-based authentication, encrypted storage mechanisms and email verification for safety measures.

5.3 Workflow

The system is request-response based where the users interact with the mobile application and everything is done in the back-end services. After authentication, users gain access to the

application's functionality and can initiate different operations such as translation, learning, or practice.

The API gateway initiates a request to the mobile client to the backend with each user action. The gateway routes the request to the appropriate service, where it is processed. The result is then returned to the client and displayed to the user.

To illustrate this process in detail, the workflow of the Practice module is described below (Figure 2).

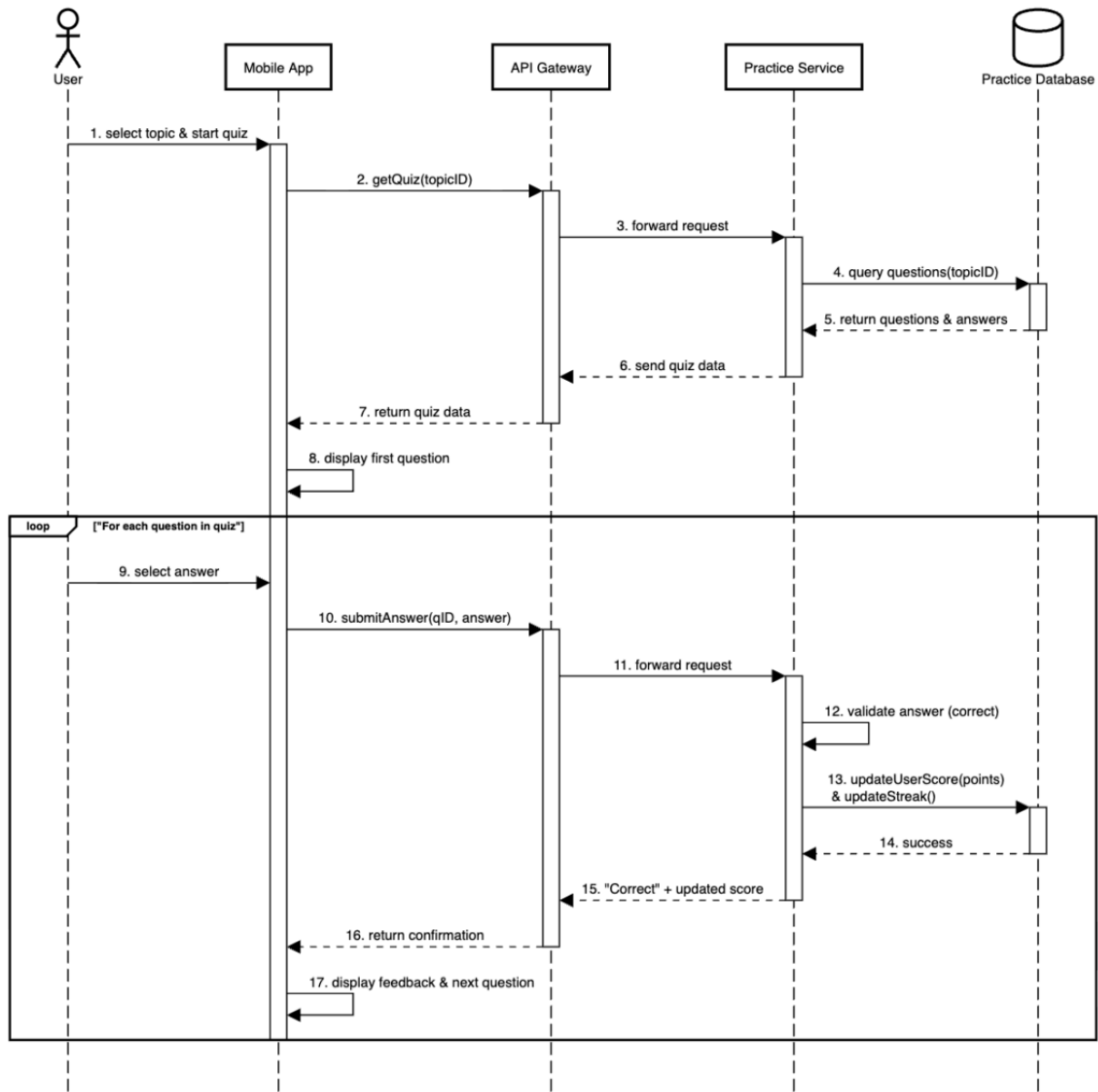


Figure 2: Sequence Diagram: Workflow of the Practice module

5.4 Use Case Diagram and Roles

The use case diagram represents the interaction between different actors and the system, showing how users engage with the application to perform various tasks.

It illustrates how user actions (such as initiating translation, searching for information, or engaging with learning activities) are mapped to system functionalities. The diagram also shows relationships between use cases, including included processes such as authentication and internal operations required to support user requests.

Additionally, the diagram captures the involvement of system-level actors (such as administrators and testers) who interact with the system for management, monitoring, and evaluation purposes.

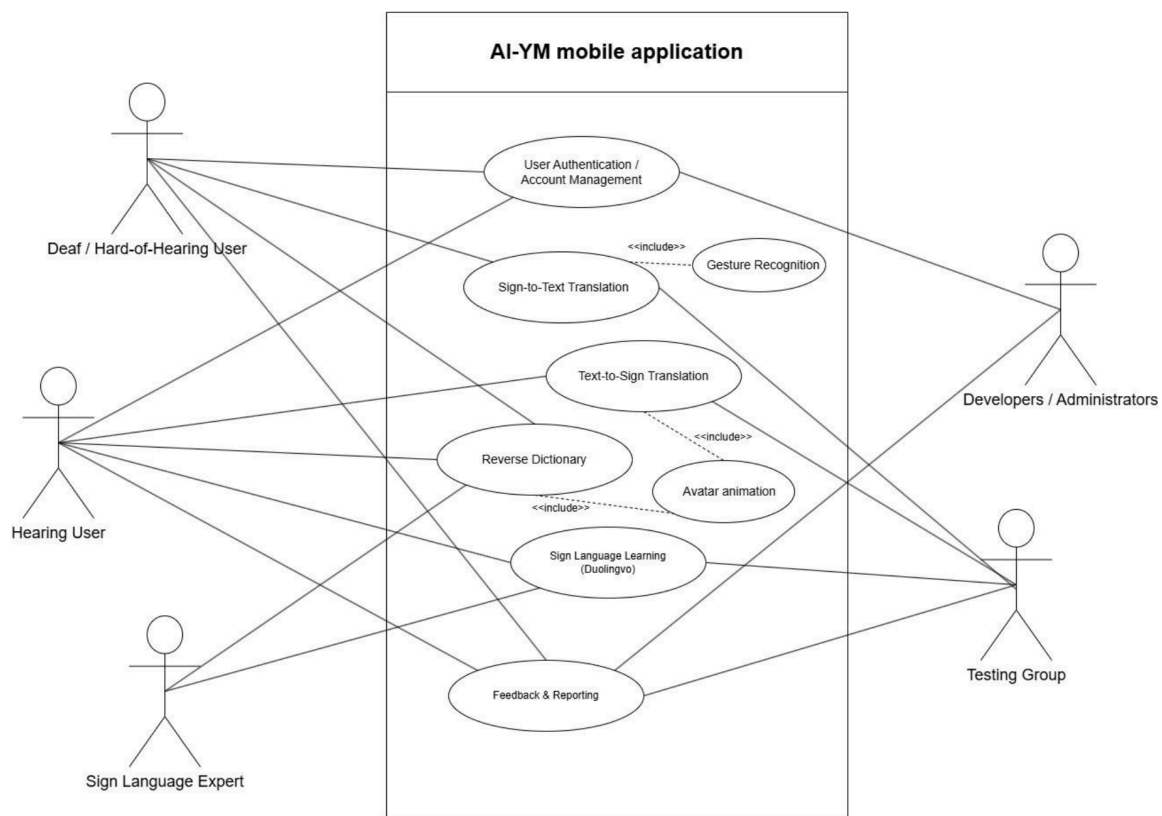


Figure 3: Use Case Diagram

5.5 Tools and Processing Logic

The system is implemented using a combination of mobile, backend, and processing technologies.

The mobile application is developed using **Flutter**, enabling cross-platform deployment on Android and iOS. Backend services handle user authentication, data processing, and communication between modules, with structured databases used for storing user data, dictionary entries, and quiz progress.

In the case of the text-to-sign translation module, a custom animation pipeline is written in **Python**. The system uses an external tool (`text_to_gloss_to_pose`) to convert input text into pose data, which is then parsed using the **pose_format** library. Landmark coordinates are processed using **NumPy**, and each frame is rendered using **Pillow (PIL)** to generate an animated GIF. Other utilities like `math`, `subprocess`, and `os` are applied to do geometry computations, execution and management of processes and files.

The practice module and authorization system are implemented on a unified backend server using **Node.js** with the **Express** framework. Quiz information and user data, score, experience points, and streaks are stored in a **PostgreSQL** database. **Sequelize** is used to manage database interactions, schema and migrations as well as structured data modeling.

Another avatar generation pipeline was created to enhance the quality and scalability of sign language material. This part relies on **MediaPipe** to extract pose and hand landmarks in the videos of recorded sign language translations and **Blender** to create a 3D animation avatar relying on these motions. This will allow transforming low quality or uneven video recordings into standard and easy to use visual representations, which can be utilized to improve the dictionary and learning modules in subsequent versions of the systems.

5.6 Third-party Components

The system incorporates a number of external elements to aid functionality that the system does not directly implement.

For sign-to-text translation, the system integrates an existing legacy RSL Gesture Recognition Service, implemented as a Flask-based microservice. This service processes video input from the client, applies MediaPipe for hand landmark detection, and uses a pre-trained classification model to recognize gestures and map them to corresponding dictionary words.

For text-to-sign translation, the system utilizes the open-source **spoken-to-signed-translation** project, namely, the `text_to_gloss_to_pose` component that transforms the textual input into structured pose data (sign-language-processing, 2024). Our custom animation pipeline then further processes this output to create user friendly visual representation.

The dictionary module is based on a structured dataset provided by the **HRI Lab**, which contains sign language vocabulary and associated resources. This dataset is integrated into the

system through backend services and serves as the basis for search, translation, and learning functionalities. The learn module also uses video lecture content provided by the **HRI Lab**.

External libraries and frameworks such as **MediaPipe** (for landmark extraction) and **Blender** (for avatar generation) are used to support advanced processing and visualization tasks.

5.7 Team Development Process

To ensure effective development of the computing-based solution, the team followed a parallel development approach, where different modules of the system were implemented simultaneously.

The team was constantly in touch with one another, using frequent meetings and online cooperation tools. High-level planning and documentation were performed using **Notion**, task tracking and sprint management were performed using **Jira**, and day-to-day communication was provided and quick issues were resolved using **Discord**. **Git**-based workflows were used to manage version control, enabling the team members to work on different features and merge them effectively.

6. Project Execution

6.1 What was implemented

During the first semester, the team focused on design, system understanding, and initial development. A significant part of the work involved analyzing the existing backend system and planning how to integrate a new mobile application without breaking existing functionality. This required understanding the data flow, APIs, and constraints of the current system before introducing new components.

At the design stage, the full structure of the mobile application was defined, including the main modules: Translate (sign-to-text and text-to-sign), Dictionary, Learn, and Practice. UI/UX prototypes were created in Figma to guide development and ensure a clear and accessible user experience .

Initial development of the Flutter mobile application was started, including:

- implementation of authentication (registration and login)
- integration with backend services for user management
- development of core pages such as Home, Profile, Translation, and Dictionary
- partial implementation of the Learn module

In parallel, work on the animation system was initiated. The legacy animation pipeline was analyzed, and a new approach based on Mediapipe and custom rendering was explored. An animation script draft was created and tested separately, and proved to have greater visual clarity than the initial stick-figure representation.

In the second semester, development was based on functionality completion, system components enhancement, and unifying all modules to form a single application. Flutter app was also enhanced by adding modules and enhancing the integration of the back end. Moreover, the new Practice module was introduced, namely, quiz functionality, scoring logic, and user progress tracking.

The animation system was significantly redesigned. Using the concerns of the first semester, the old pipeline was substituted by a new one that takes pose data, computes bounding boxes, and displays a structured animation along with body and hand motions. This led to greater stability in the positioning and better readability of gestures.

Also, a pipeline of generating an avatar was built based on MediaPipe and Blender that enabled converting the videos of sign language into standardized 3D avatars. This was an element that would enable further development of the dictionary with better visual material.

The team also worked on integrating all components, including mobile application, backend services, translation modules, animation pipeline. This integration ensured consistent communication between modules and unified system behavior.

Finally, the system was tested to ensure stability and usability, and deployment processes were initiated, including preparing the application for distribution through platforms such as the App Store and Google Play.

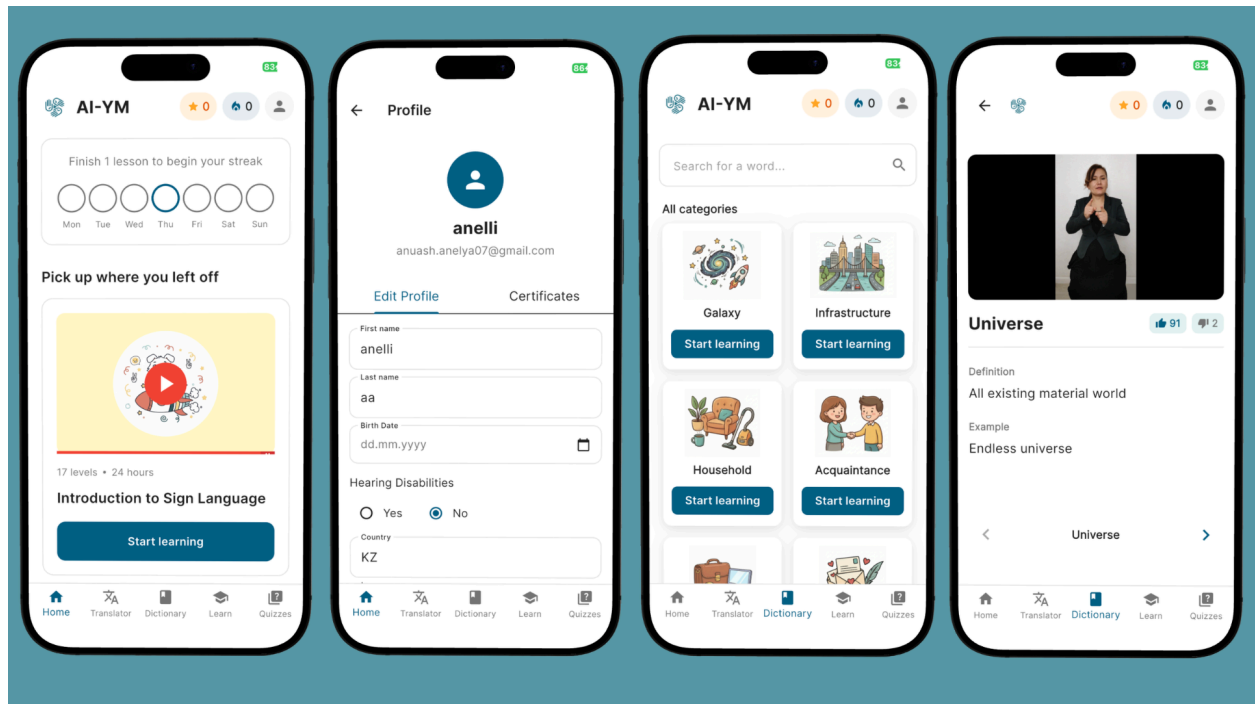


Figure 3: AI-YM mobile application screenshots

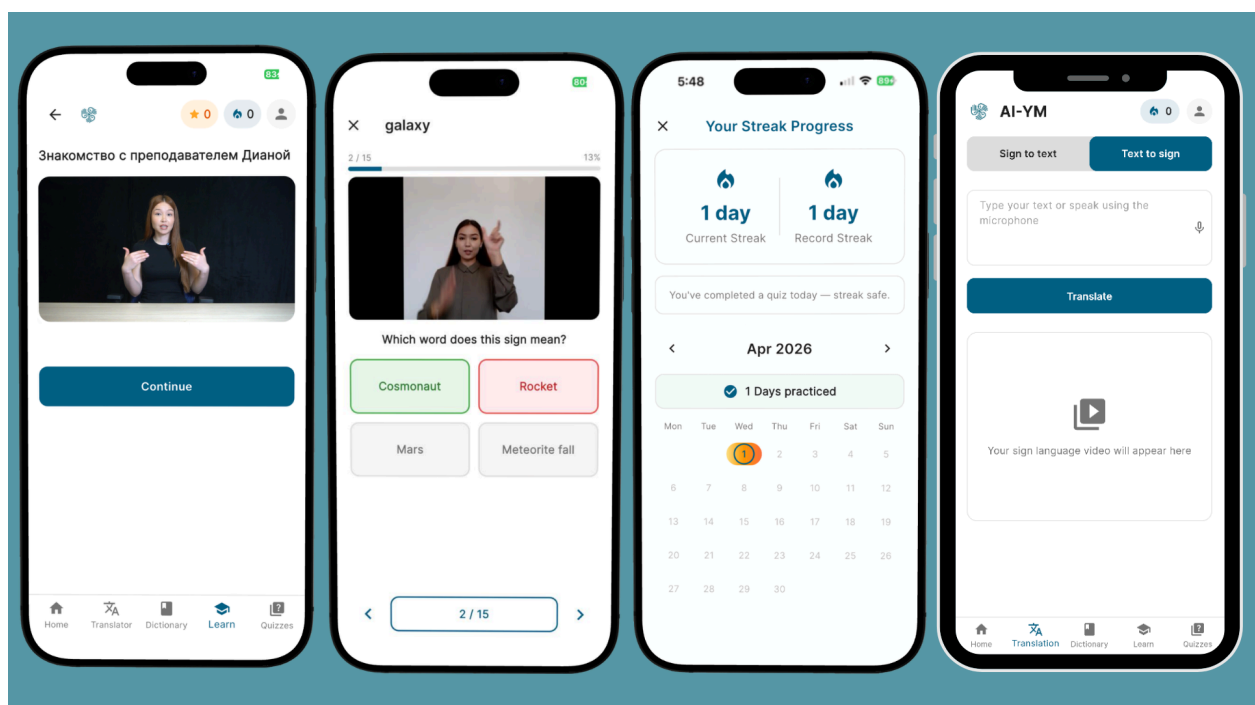


Figure 4: AI-YM mobile application screenshots

6.2 Problems encountered

1. Incompatibility between new animation approach and existing system

In the first semester, a new animation methodology was created based on MediaPipe-extracted keypoints and a script-written render. This implementation generated animations from JSON landmark data and produced visually clearer results compared to the legacy stick-figure system.

This method, however, could not be incorporated into the current system. The main text-to-sign pipeline was built on the use of .pose files produced by the translation model, and the novel implementation was built on MediaPipe landmarks. These two representations were essentially different, and thus there was no straightforward method to connect the new animation script to the current backend without re-creating the entire pipeline.

To resolve this, the approach was revised in the second semester. The team did not replace the .pose-based system, instead, adapting the new rendering logic to accept and work with .pose files. The last implementation reads data with the pose_format library, finds body and hand landmarks, and renders a structured avatar on top of the data.

With this solution, the system was able to maintain compatibility with the already existing translation pipeline and greatly enhance the visual quality of the animation and it met the integration and usability objectives.

2. The SMPL-X model lacks enough hand detail and the difficulty of integrating it with HaMeR.

In the process of creating the offline avatar video regeneration pipeline, it was apparent early that SMPL-X was not sufficient to capture hand gestures with the detail that sign language demands. Although SMPL-X offers a good model of overall body pose and facial expression, the model of hands is relatively rough - the articulation of fingers in particular is not as fine as required to differentiate between signs that only differ in subtle hand shapes or finger positions, which is typical of K-RSL.

To resolve this, the team added HaMeR to SMPL-X. HaMeR is a transformer-based hand mesh recovery model that recovers detailed 3D hand meshes using monocular video and performs much better on fine-grained hand pose than general body models. The proposed pipeline was to estimate body pose using SMPL-X, estimate hand pose using HaMeR, and then transfer HaMeR hand parameters to the SMPL-X representation and feed the result to Blender to render an avatar.

In reality, though, the key technical challenge was to merge the two models. The coordinate systems, joint conventions and mesh topologies of SMPL-X and HaMeR are different. To overlay the HaMeR hand output onto the SMPL-X body in such a manner that geometrically consistent, visually consistent results were obtained, especially at the wrist joint

where the two models meet, the parameter spaces had to be carefully aligned and transformed. It was challenging to have the fingers to render properly in Blender without any visible seams, unnatural bending, or loss of articulation detail at the wrist, and the team is still in the process of solving the remaining problems. This is the main open issue in the avatar regeneration pipeline, and one of the reasons why this component is still in active development.

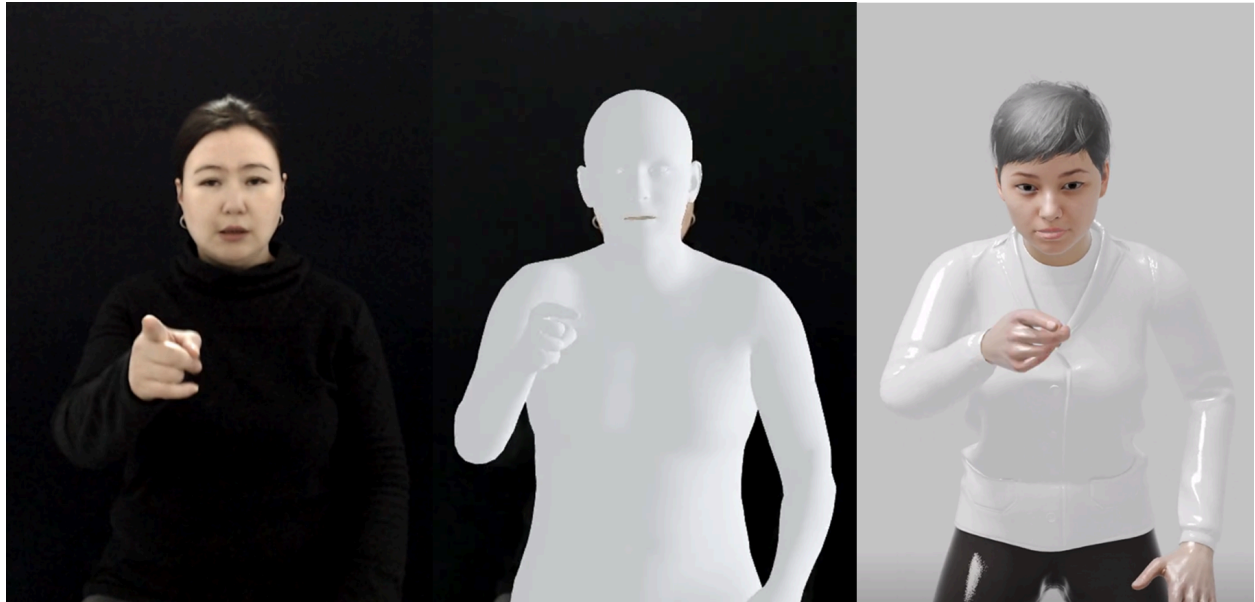


Figure 5: AI-YM Avatar Video Regeneration

6.3 Team Management

Since our team consisted of five members, we were able to distribute responsibilities across different components of the system while maintaining collaboration throughout the development process. From the beginning, we agreed that all team members would have a general understanding of both frontend and backend aspects of the system. However, specific functionalities were primarily assigned to individual members based on their focus areas.

The responsibilities were divided as follows:

- Ayaulym: UI/UX design using Figma, marketing and outreach activities, and development of the avatar generation pipeline
- Anel: mobile frontend development of Practice and Home modules in Flutter, integration and coordination of frontend components, App Store deployment, and backend-related tasks
- Aliya: frontend development of the Learn and Dictionary modules in Flutter
- Nargiza: animation system implementation, database migrations, and backend development

- Dana: frontend development of the Translation module in Flutter

Despite this division, team members frequently collaborated when working on interconnected features, especially during integration of frontend and backend components.

The team maintained regular communication through scheduled meetings and online communication tools, allowing continuous progress tracking and quick resolution of technical issues. Task management tools were used to organize development, assign responsibilities, and monitor the status of ongoing work.

7. Evaluation

To evaluate whether the proposed solution addresses the problem identified in the introduction, a user-based evaluation was conducted. The purpose of this evaluation was to determine how well the AI-YM application solves the problem of fragmented access to Kazakh Sign Language learning and communication tools. The evaluation focused on user perception of the system rather than technical debugging.

A total of 10 participants took part in the evaluation (5 students, 3 adults, 2 teenagers).

After interacting with the prototype deployed via TestFlight, participants completed a questionnaire consisting of scaled and open-ended questions.

7.1 Overall usability and intuitiveness

The overall results show that the application is easy to use and understandable for most users.

- As shown in **Figure 1**, all participants rated the overall usability positively.
- 60% of respondents gave the application a score of 4.
- 40% of respondents gave the application a score of 5.

These results indicate that the application already provides a clear and accessible user experience.

The intuitiveness of individual features is shown in **Figure 2**.

- The **navigation menu** received the strongest ratings.
- The **quiz process** was also rated as highly intuitive.
- **Word search in the dictionary** was perceived as clear and easy to use.
- Lower and more varied ratings were observed for the translation-related features.

Overall, this suggests that the general structure of the application is logically organized and user-friendly.

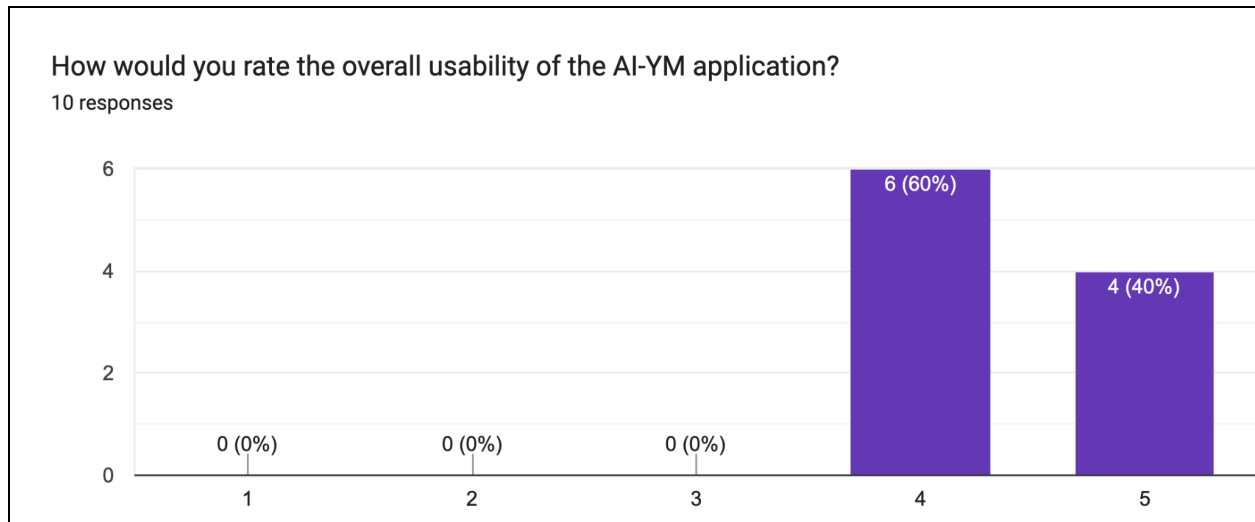


Figure 1. Overall usability rating

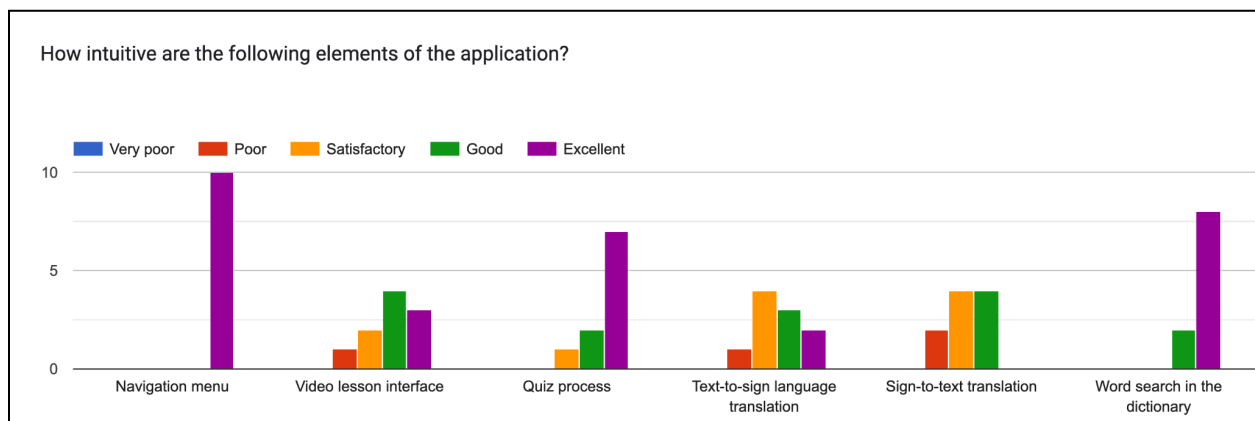


Figure 2. Intuitiveness of application features

7.2 Usefulness of features

The usefulness of the main modules is presented in **Figure 3**.

The most useful feature was identified as:

- Translation – 80%

Other highly valued features were:

- Video lessons – 70%
- Dictionary/search – 70%
- Quizzes – 70%

These findings support the main goal of the project: to combine translation, learning, and practice tools in one centralized platform.

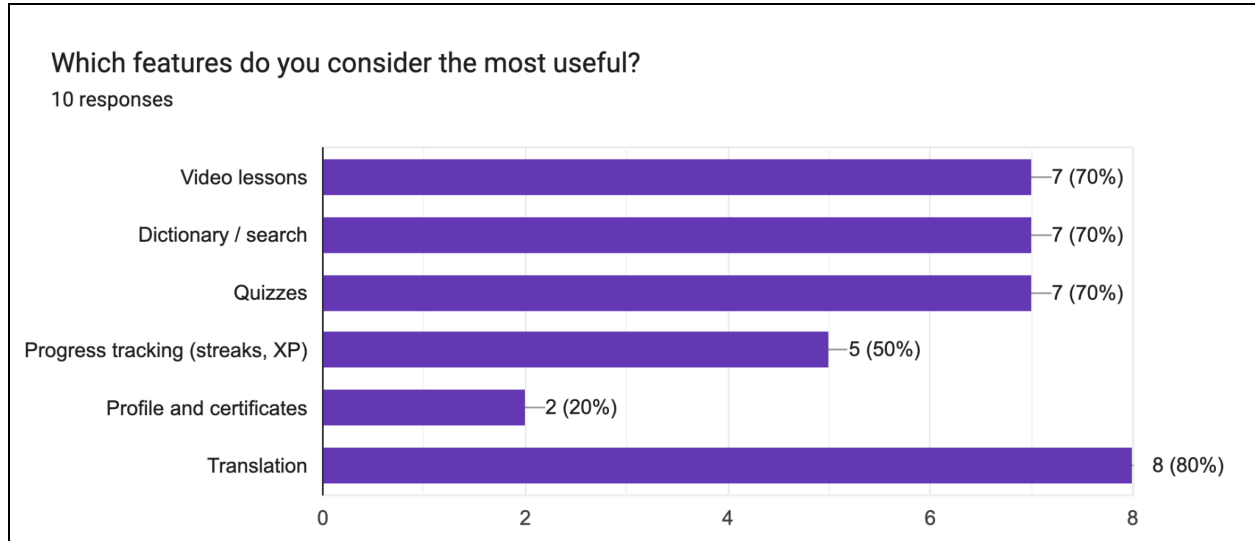


Figure 3. Most useful features.

7.3 Features requiring improvement

The features that users found unclear or in need of improvement are shown in **Figure 4**.

The main issues identified were:

- Translation – selected by 100% of participants
- Video lessons / Learn module – selected by 70% of participants

This shows that the strongest concerns are related mainly to the translation module and, to a lesser extent, the Learn module.

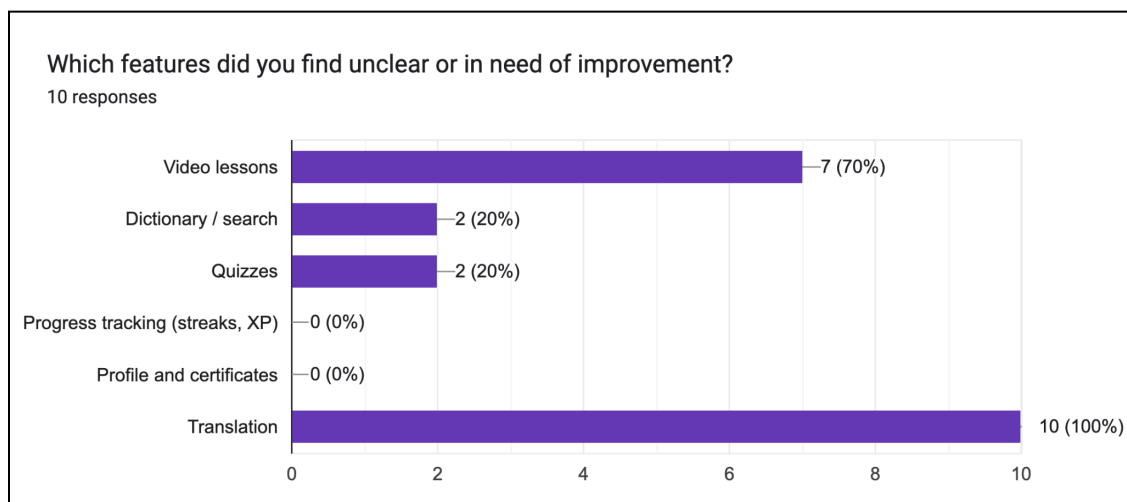


Figure 4. Features requiring improvement

7.4 Evaluation of courses, dictionary, and quizzes

More detailed results for the learning-related modules are presented in **Figures 5 and 6**.

Courses and dictionary (Figure 5)

The results show that:

- Video lessons were generally considered useful for learning signs.
- Dictionary search was rated as easy and convenient.
- The lesson interface received more mixed responses compared to the dictionary.

This suggests that the content of the learning module is valuable, but the video interface could be improved.

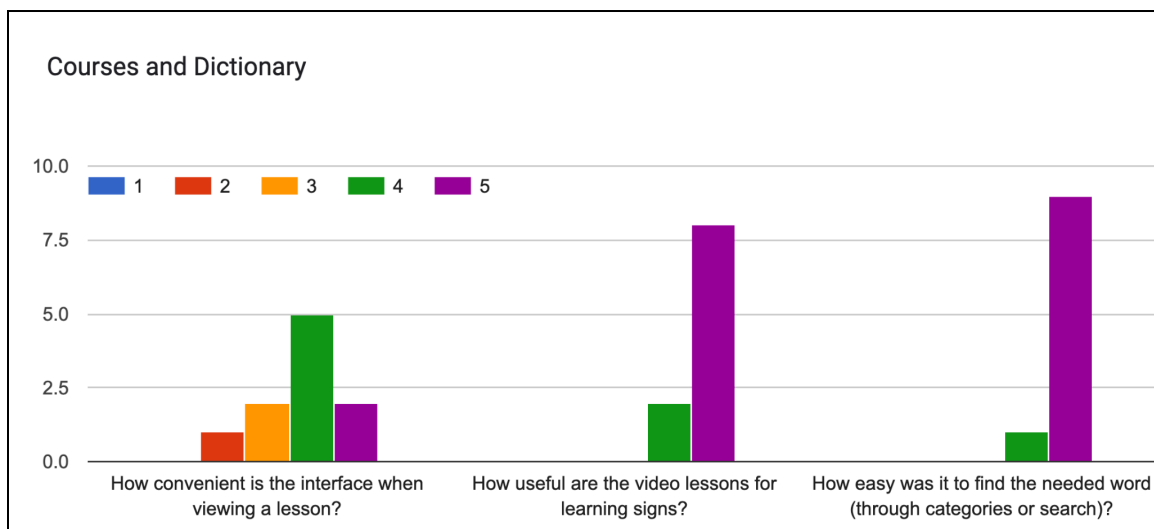


Figure 5. Evaluation of courses and dictionary

Quizzes (Figure 6)

The quiz module received strong ratings.

- Most users found it easy to select answers.
- The streak and XP system was viewed as motivating.
- The results were concentrated in the higher rating categories.

These findings indicate that the Practice module is one of the strongest parts of the application.

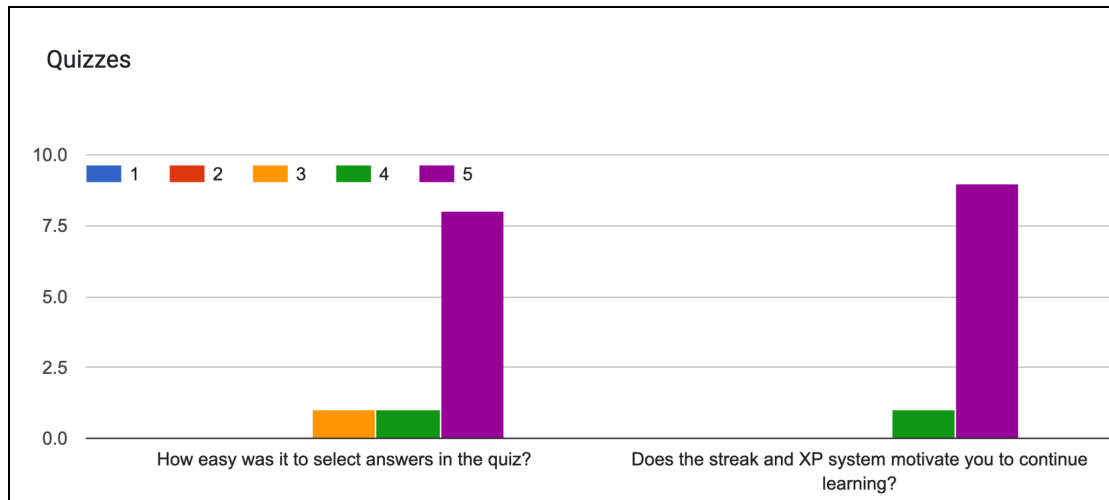


Figure 6. Evaluation of quizzes

7.5 Translation module evaluation

The most detailed limitations were observed in the translation module, as shown in **Figures 7 and 8**.

Text-to-sign translation (Figure 7)

The main findings were:

- Entering text for translation was generally easy.
- Ratings for response speed were moderate to positive.
- Ratings for clarity of generated sign animation were mixed.

This means that the feature is accessible from the user side, but the visual output still needs improvement in terms of clarity and precision.

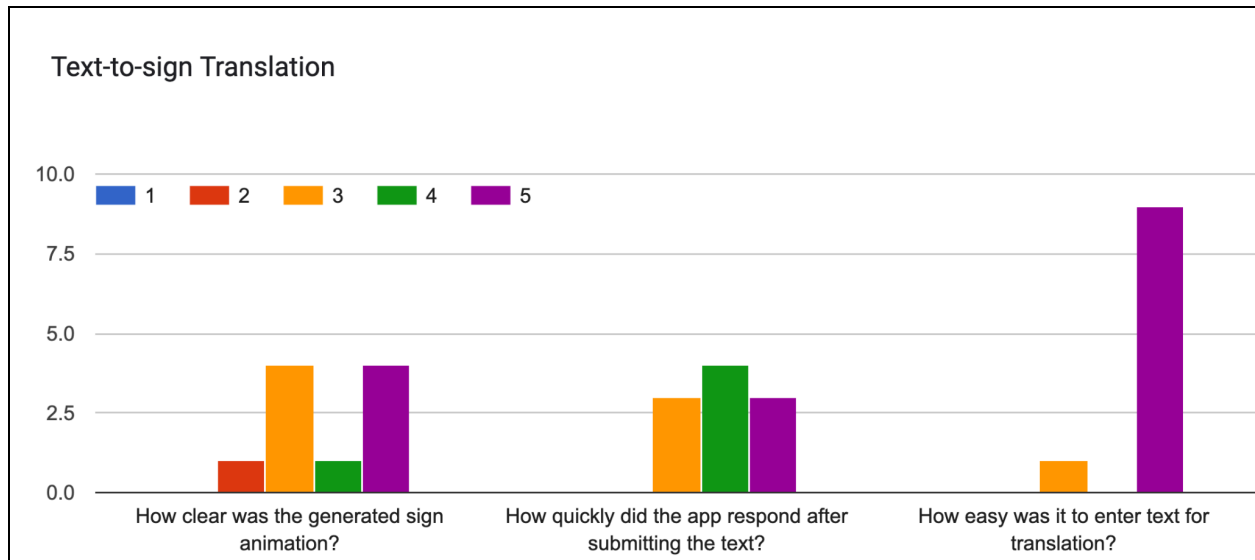


Figure 7. Text-to-sign translation evaluation

Sign-to-text translation (Figure 8)

The results show more noticeable variation.

- Gesture recognition accuracy received moderate ratings.
- Translation display speed was not consistently rated highly.
- The Start / Stop / Reset buttons were not fully clear to users.

These results suggest two major issues:

- a technical issue, related to recognition accuracy
- a usability issue, related to control clarity

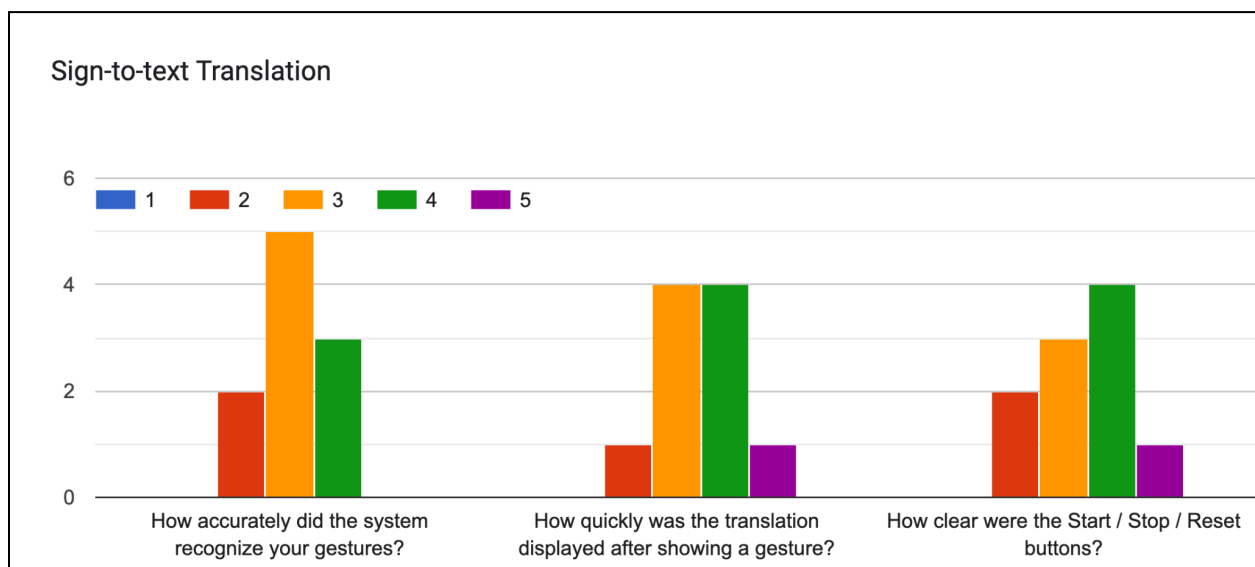


Figure 8. Sign-to-text translation evaluation

7.6 Evaluation summary

Overall, the evaluation shows that the AI-YM application successfully addresses the problem identified in the introduction, especially as a centralized platform for learning Kazakh Sign Language.

The strongest validated aspects of the system are:

- overall usability
- dictionary/search
- quizzes
- motivation through XP and streaks
- usefulness of video lessons

At the same time, the main limitations are concentrated in the translation module:

- insufficient gesture recognition accuracy
- unclear sign animation in some cases
- limited vocabulary coverage
- unclear translation controls

Therefore, the evaluation shows that the project already works well as a learning and practice platform, while the translation functionality still requires further refinement.

7.7 Future improvements

Based on the evaluation results, future work should focus on:

- improving sign-to-text recognition accuracy
- enhancing text-to-sign animation clarity
- expanding the dictionary vocabulary
- redesigning the translation control buttons
- improving the video lesson interface with more convenient playback features

These improvements would increase both the technical quality of the system and overall user satisfaction.

8. Conclusion and Future Work

The development of the AI-YM mobile application could successfully provide a single, high-performance mobile platform designed to support sign language translation and learning. Over the last two semesters we could adjust an existing web-based learning platform into an accessible mobile application built on Flutter Framework. This would allow users to access the app on both Android and iOS mobile devices. Key contributions to the project include the implementation of the essential Text-to-Sign, Sign-to-Text translations, Dictionary module, Learning module and new gamified Quizzes module for practice.

Overall, the integration of learning and practicing materials within a single application makes it more practical and accessible for all hearing, hard-of-hearing individuals, and those who are interested in learning sign languages. Although our current mobile version already provides solid all-in-one features needed for sign language learning, there are still some changes that need to be made in future. Particularly, expanding the dataset for Kazakh Sign Language will significantly increase the translation accuracy and strengthen system performance. Also, we are planning to expand the list of sign languages to reach a broader audience. Consequently, this would require extending the dataset and adapting translation models. In addition, improving offline working functionality would certainly make the application more accessible. For now, our pages heavily rely on backend services and stable internet connection for smooth work, which is not always possible in real-world conditions. Providing offline usage capability ensures continuous access for everyday learning. Our future work will also focus on developing more realistic avatar animations for precise hand gestures and facial expressions. After completing the TestFlight of the application we will proceed with the formal release on App Store and Play Market to introduce the app within the community.

In conclusion, AI-YM mobile application is a structured solution for bridging the communication gap between hearing and deaf individuals that combines vocabulary, learning, practice and translation support in one single place.

9. References

- ASL Bloom. (n.d.). *ASL Bloom* [Mobile app]. App Store. <https://www.aslbloom.com/>
- Bragg, D., Verhoef, T., Vogler, C., Morris, M., Koller, O., Bellard, M., Berke, L., Boudreault, P., Braffort, A., Caselli, N., Huenerfauth, M., & Kacorri, H. (2019). Sign language recognition, generation, and translation: An interdisciplinary

- perspective. *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 16–31. <https://doi.org/10.1145/3308561.3353774>
- David, D., Alamoodi, A. H., Albahri, O. S., Garfan, S., Albahri, A. S., Zaidan, B. B., & Chen, J. (2024a). Sign language mobile apps: A systematic review of current app evaluation progress and solution framework. *Evolving Systems*, 15(2), 669–686. <https://doi.org/10.1007/s12530-023-09494-0>
- David, D., Alamoodi, A. H., Albahri, O. S., Zaidan, B. B., Zaidan, A. A., Garfan, S., & Malik, R. Q. (2024b). Landscape of sign language research based on smartphone apps: Coherent literature analysis, motivations, open challenges, recommendations and future directions for app assessment. *Universal Access in the Information Society*, 23(2), 687–702. <https://doi.org/10.1007/s10209-022-00966-9>
- Falvo, V., Scatalon, L. P., & Barbosa, E. F. (2020). The role of technology to teaching and learning sign languages: A systematic mapping. *2020 IEEE Frontiers in Education Conference (FIE)*, 1–9. <https://doi.org/10.1109/FIE44824.2020.9274169>
- Imashev, A., Oralbayeva, N., & Sandygulova, A. (2024). An exploratory user study towards developing a unified, comprehensive assessment apparatus for deaf signers, specifically tailored for signing avatars evaluation: Challenges, findings, and recommendations. *Multimedia Tools and Applications*, 1–38. <https://doi.org/10.1007/s11042-024-20365-x>
- Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C., Yong, M., Lee, J., Chang, W., Hua, W., Georg, M., & Grundmann, M. (2019). *MediaPipe: A framework for building perception pipelines*. arXiv. <https://doi.org/10.48550/arXiv.1906.08172>
- Mukushev, M., Ubingazhibov, A., Kydyrbekova, A., Imashev, A., Kimmelman, V., & Sandygulova, A. (2022). FluentSigners-50: A signer independent benchmark dataset for sign language processing. *PLOS ONE*, 17(9), e0273649. <https://doi.org/10.1371/journal.pone.0273649>
- Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A. A. A., Tzionas, D., & Black, M. J. (2019). Expressive body capture: 3D hands, face, and body from a single image. *Proceedings of the IEEE/CVF Conference on Computer Vision and*

Pattern Recognition (CVPR), 10975–10985.

<https://doi.org/10.1109/CVPR.2019.01123>

Pavlakos, G., Shan, D., Radosavovic, I., Kanazawa, A., Fouhey, D., & Malik, J. (2024).

Reconstructing hands in 3D with transformers. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9826–9836.

<https://doi.org/10.48550/arXiv.2312.05251>

The Chinese University of Hong Kong. (2024, November). Meet SignTown handbook:

AI-powered online sign language learning platform. *CUHK Faculty of Arts News*.

<https://www.arts.cuhk.edu.hk/web/news/meet-signtown-handbook-an-ai-powered-online-sign-language-learning-platform>

World Health Organization. (2025, February 26). *Deafness and hearing loss*.

<https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>

Zholshiyeva, L., Zhukabayeva, T., Baumuratova, D., & Serek, A. (2025). Design of QazSL sign language recognition system for physically impaired individuals.

Journal of Robotics and Control, 6(1), 191–201.

<https://doi.org/10.18196/jrc.v6i1.23879>