Text to Sign Language Translator –Two Implementations

Durgadevi Yenuganti
Southeast Missouri State University, Cape Girardeau, Missouri, USA

Nikhitha Kasha
Southeast Missouri State University, Cape Girardeau, Missouri, USA

Pavan Subhash Chandrabose Nara
Southeast Missouri State University Cape Girardeau, Missouri, USA

Suhair Amer
Southeast Missouri State University, Cape Girardeau, Missouri, USA

ABSTRACT
This paper presents a comparative analysis of the projects of two students projects focused on developing text-to-American Sign Language (ASL) finger spelling translation systems. Both projects successfully convert English text into corresponding ASL hand shape images, but they differ in their technological approaches and implementation complexities. Project 1 utilizes PHP for a simpler implementation, while Project 2 employs Python and Flask for a more robust and scalable solution. The comparison highlights the diverse approaches and technologies that can be employed for sign language translation, emphasizing the importance of user-centered design and evaluation in developing accessible technologies for the Deaf community. The evaluations of both projects, while differing in methodology, reveal positive user experiences and identify areas for improvement, such as handling special characters and incorporating additional features. The students completed the development in one month as an end of semester project.

Keywords
Sign Language Translation, Finger spelling, American Sign Language (ASL), PHP, Python, Flask, User Interface, Evaluation, Accessibility, Assistive Technology

1. INTRODUCTION
Communication is a fundamental human need, yet it presents a significant challenge for those who are deaf or hard of hearing. Sign language serves as the primary means of communication for many individuals within the Deaf community, with American Sign Language (ASL) being the most prevalent in the United States and parts of Canada [1]. Bridging the communication gap between sign language and spoken/written language is crucial for fostering inclusivity and accessibility for all.

The development of technology has opened new possibilities for facilitating communication between sign language and spoken/written language users. Sign language translation systems, which aim to convert sign language to spoken/written language or vice versa, have garnered increasing attention in recent years [2]. These systems have the potential to break down communication barriers and empower deaf or hard-of-hearing individuals to interact more seamlessly with the hearing world.

Various approaches have been explored in the development of sign language translation systems, including rule-based systems, statistical machine translation (SMT), and neural machine translation (NMT) [2, 3]. Rule-based systems rely on predefined grammatical rules and dictionaries to map between sign language and spoken/written language [4]. SMT systems learn statistical relationships between the two languages from large parallel corpora [3]. NMT systems leverage deep learning models, such as recurrent neural networks (RNNs) and transformers, to learn complex mappings between sign language and spoken/written language [2]. Despite the advancements in technology, sign language translation remains a challenging task due to the linguistic complexity of sign languages, the scarcity of parallel data for training, and the variability in signing styles across individuals and regions [5, 6]. Sign languages involve a combination of hand shapes, facial expressions, and body movements, making it difficult to capture and translate accurately [7]. The limited availability of parallel data for training translation models further hinders progress in this area [3]. Additionally, variations in signing styles and regional dialects pose challenges for developing systems that generalize well [6].

This paper summarizes the experience of two students developing a text-to-sign language translation system that focuses on finger spelling. The project was completed in one month as an end of semester project. Finger spelling is a method of representing letters of the alphabet using hand shapes, and it is often used in sign language to spell out words or names that do not have a corresponding sign [1]. By focusing on finger spelling, the availability of resources for individual letter representations in ASL can be leveraged and allows the creation of a system that is relatively simple to implement and deploy.

The systems provide a user-friendly interface for inputting English text and generating corresponding ASL finger spelling images. This tool can be valuable for educational purposes, for facilitating basic communication between deaf and hearing individuals, and as a foundation for developing more comprehensive sign language translation systems in the future.

2. BACKGROUND
Sign language is a visual language that uses hand shapes, facial expressions, and body movements to convey meaning. American Sign Language (ASL) is the primary language used by many deaf individuals in the United States and parts of Anglophone Canada [1]. ASL is a complex language with its

own grammar and syntax, which differs significantly from spoken English [8]. Developing effective tools for ASL translation is crucial for enhancing communication accessibility for the Deaf community.

Efforts to improve communication between Deaf and hearing individuals have led to the development of sign language translation systems, which aim to bridge the communication gap by converting sign language into spoken or written language or vice versa. These systems can be broadly classified into two categories:

- Sign Language Recognition: These systems analyze sign language input, typically captured via video, and translate it into spoken or written language [2].
- Sign Language Generation: These systems take spoken or writ-ten language as input and generate sign language output, often in the form of animations or videos [9].

Various methodologies have been employed in the development of sign language translation systems:

- Rule-based Systems: These systems use predefined grammatical rules and dictionaries to map between sign language and spoken/written language [4].
- Statistical Machine Translation (SMT): SMT systems learn statistical relationships between sign language and spoken/written language from large parallel corpora [3].
- Neural Machine Translation (NMT): NMT systems utilize deep learning models, such as recurrent neural networks (RNNs) and transformers, to learn complex mappings between sign language and spoken/written language [2].

Despite significant advancements, sign language translation remains a challenging task due to several factors:

- Linguistic Complexity: ASL involves complex grammatical structures, visual elements, and hand shapes, which are difficult to capture and translate [8].
- Data Scarcity: There is limited parallel data available for training sign language translation systems compared to spoken languages [3].
- Variability: Sign language can vary significantly across individuals, regions, and dialects, making it difficult to develop systems that generalize well [6].

Previous research has focused on a variety of approaches to overcoming these challenges, such as:

- Zhou et al. (2021) introduced a sign back-translation approach to improve sign language translation using monolingual data. This approach aids in reducing the dependence on parallel data, which is scarce in the field [3].
- Camgoz et al. (2020) proposed Sign Language Transformers, a joint end-to-end model for sign language recognition and translation, advancing the use of transformer-based architectures in sign language translation [2].
- Bagus et al. (2019) developed an English-to-sign-language translation system specifically for Android, aiming to make ASL translation more accessible through mobile technology [10].

The work presented in this paper focuses on a simple yet effective approach to sign language translation. It focuses on translating English text into ASL finger spelling, an approach that avoids the complexity of full sentence level translation by utilizing letter-based representations. This method allows for easier data acquisition and implementation, providing a practical tool for communication between Deaf and hearing individuals. By leveraging existing resources for ASL finger spelling, an accessible solution is provided for educational purposes and basic communication needs.

3. COMPARING PROJECT APPROACHES

Comparing the implementations of two students in a project or assignment can provide several benefits, both for the students involved and for instructors. Next are some advantages:

- Diverse Problem-Solving: Students may approach the same problem from different angles. By comparing their solutions, they can learn new techniques, methodologies, or strategies that they hadn't considered before.
- Exposure to Multiple Perspectives: Each student brings their unique perspective and creativity to the project. This comparison exposes students to alternative ways of thinking and helps broaden their approach to solving problems.
- Evaluation and Reflection: Comparing implementations requires students to critically assess the strengths and weaknesses of each approach. This process encourages deeper reflection on the logic, design, and functionality of their work.
- Problem Identification: When analyzing different implementations, students can identify areas for improvement in their own work and recognize potential flaws in other approaches, fostering critical thinking.
- Concept Reinforcement: By comparing different implementations, students reinforce their understanding of course concepts, algorithms, or tools. Seeing how others use the same concepts in different ways can strengthen their grasp of the material.
- Clarifying Misunderstandings: Students may recognize gaps or misunderstandings in their own approach when comparing it to a peer's implementation. This can help them clarify concepts and improve their understanding of the topic.
- Knowledge Sharing: Comparison often leads to discussions between students, where they can share ideas and insights. This collaborative exchange can deepen their learning and help them develop new skills.
- Peer Learning: When students compare their implementations, they can explain their reasoning to each other, reinforcing their own understanding while teaching others. This peer-to-peer learning is a powerful tool for academic growth.
- Identifying Optimal Solutions: By comparing implementations, students can identify which approach is more efficient, robust, or scalable. This helps them understand the trade-offs between different methods and choose the best solution.
- Learning from Mistakes: If one student's implementation contains errors or inefficiencies, comparing it to another's may provide insight into how to avoid similar mistakes in the future.
- Motivation: Knowing that their work will be compared with others can motivate students to put more

effort into their projects, leading to higher-quality outcomes.

- Inspiration for Improvement: Seeing a peer's well-executed project can inspire students to strive for improvement in their own work and push their boundaries further.
- Self-Evaluation: Comparing their work with a peer's allows students to assess their own strengths and areas of improvement, fostering self-awareness and continuous development.
- Constructive Feedback: When comparing implementations, students can give and receive constructive feedback, which is crucial for refining their skills and enhancing their final product.
- Explaining Concepts: In discussions that arise from comparing implementations, students must articulate their thought processes clearly. This enhances their ability to explain complex ideas and engage in technical discussions.
- Debate and Justification: Comparing implementations often leads to debates where students must justify their choices. This can help them practice defending their ideas and reasoning, a valuable skill in both academic and professional settings.
- Acknowledging Progress: By comparing different implementations, students can recognize how they've improved over time and what areas still require development. This reinforces the idea that learning is a continuous process.
- Accepting Constructive Criticism: Students learn how to receive and give constructive criticism, which is essential for personal and academic growth.
- Exposure to Best Practices: In a comparison, students can identify and adopt best practices from one another, improving the overall quality of their future work.
- Refinement of Skills: As students analyze peer work, they may notice techniques or methods they could implement in their own projects, ultimately refining their skill set.
- Real-World Simulations: In the workplace, it's common to compare and evaluate different approaches to a problem. By practicing this in a classroom setting, students better prepare for future professional environments, where collaboration and comparison are routine.

In summary, comparing the implementations of two students helps deepen understanding, improves critical thinking and problem-solving skills, and promotes peer learning and collaboration. It encourages a growth mindset and prepares students for real world professional challenges. The process of discussing, evaluating, and learning from each other's work can lead to enhanced outcomes for all involved.

4. METHODOLOGY

The text-to-sign language translation system utilizes a combination of web technologies and image processing techniques to convert English text into corresponding ASL finger spelling representations. The system architecture comprises two main components: a frontend interface for user interaction and a backend processing engine for text analysis and image generation.

4.1 Frontend Interface

The frontend interface is built using HTML, CSS, and JavaScript. It provides a user-friendly web page where users can input English text and initiate the translation process. The interface design prioritizes simplicity and ease of use, ensuring accessibility for a wide range of users.

HTML: HTML (Hyper Text Markup Language) is used to structure the content of the web page, including input fields, buttons, and image display areas.

CSS: CSS (Cascading Style Sheets) is used to style the visual presentation of the web page, ensuring an attractive and intuitive user experience.

JavaScript: JavaScript is used to handle user interactions, such as capturing text input, triggering the translation process, and dynamically displaying the generated sign language images.

4.2 Backend Processing Engine

The backend processing engine is responsible for analyzing the input text and generating the corresponding ASL finger spelling images. It utilizes Python and image processing libraries to perform these tasks. In specific,

- Python: Python is a versatile programming language well-suited for text processing and image manipulation. It provides a wide range of libraries and tools for string manipulation, file handling, and image processing.
- Image Processing Libraries: Python libraries such as OpenCV and Pillow are used to handle image loading, manipulation, and display. These libraries enable efficient processing of image data, ensuring smooth and accurate generation of sign language representations.

4.3 System Workflow

The overall workflow of the system can be summarized as follows:

- Text Input: The user enters English text into the input field on the frontend interface.
- Text Processing: The backend processing engine receives the input text and splits it into individual words and characters.
- Image Retrieval: For each character, the system retrieves the corresponding ASL finger spelling image from a pre-compiled image dataset.
- Image Display: The retrieved images are dynamically displayed on the frontend interface in a sequential manner, representing the finger spelling of the input text.

4.4 Technology Integration

The frontend and backend components of the system are integrated using a web framework, such as Flask (Python) or PHP. The web framework facilitates communication between the frontend and backend, enabling seamless transfer of data and processing requests.

Flask/PHP: Flask (Python) or PHP is used to handle web requests, route data between the front-end and back-end, and manage the overall application logic.

By combining these technologies and employing an efficient workflow, our text-to-sign language translation system provides a user-friendly and effective tool for converting English text into ASL fin-ger spelling representations.

5. PROJECT 1 IMPLEMENTATION

This section explains the details of the first text-to-sign language translation project, referred to as Project 1. This project focuses on converting English text to American Sign Language (ASL) finger spelling using a combination of web technologies and image processing.

5.1 Approach

Project 1 adopts a straightforward approach to translate English text to ASL finger spelling. The system takes the input text, processes it to extract individual characters, and then maps each character to its corresponding ASL handshape image. This approach leverages the availability of standardized ASL finger spelling images, simplifying the translation process.

The design of Project 1 prioritizes simplicity and user-friendliness. The user interface (UI) is implemented using HTML and Bootstrap, providing a clean and intuitive interface for users to input text and view the translated sign language output. The UI consists of a text box for input, a submit button to initiate the translation, and an area to display the generated sign language images.

5.2 Design

The design of Project 1 prioritizes simplicity and user-friendliness. The user interface (UI) is implemented using HTML and Bootstrap, providing a clean and intuitive interface for users to input text and view the translated sign language output. The UI consists of a text box for input, a submit button to initiate the translation, and an area to display the generated sign language images

5.3 Implementation

Project 1 is implemented using PHP, a server-side scripting language well-suited for web development. The implementation involves the following key steps:

(1) Input Collection: The system retrieves the text entered by the user in the text box.
(2) Text Processing: The input text is converted to lowercase and split into individual characters.
(3) Image Mapping: Each character is mapped to its corresponding ASL finger spelling image file.
(4) Image Display: The mapped images are dynamically displayed on the web page, forming the translated output.

5.4 Code Snippets

The following snippets illustrate key aspects of the implementation:

Text Processing

```
<?php
// Input
$phrase = $_POST[’phrase’];
// Converting the string to lowercase $phrase =
strtolower($phrase);
// Splitting the string input into an array of characters $phrase
= str_split($phrase);
?>
```

(2) Image Mapping and Display

```
<?php
foreach ($phrase as $character) {
// Fetching jpg file based on the character $character =
$character . ".jpg";
// Loop to collect all the jpg files for given input if
(file_exists($character) == 1) {
$results .= "<img src=$character height=72>"; } else {
// DO NOTHING }
} ?>
```

5.5 Technologies Used

Project 1 utilizes the following technologies:

HTML: For structuring the web page content. —Bootstrap: For styling and enhancing the UI.
PHP: For server-side processing and image mapping.
Visual Studio Code: As the development environment.

5.6 Evaluation

Project 1 was evaluated by a group of five subjects who rated the system on various aspects, including clarity of goal, ease of input, UI design, accuracy of translation, and overall performance. The evaluation results were positive, with high ratings for UI design and ease of use. However, limitations were identified, such as the lack of support for special characters and shortcuts.

In general, project 1 demonstrates a simple and effective approach to text-to-ASL finger spelling translation using basic web technologies. While it has limitations, it provides a foundation for future development and highlights the potential of technology to facilitate communication for the Deaf community.

6. PROJECT 2 IMPLEMENTATION

This section describes the second text-to-sign language translation project, referred to as Project 2. This project also focuses on converting English text to ASL finger spelling, but it utilizes a different set of technologies and design choices compared to Project 1.

6.1 Approach

Project 2 employs a similar approach to Project 1, where the input text is processed to extract individual characters, and each character is then mapped to its corresponding ASL handshape image. However, Project 2 differs in its implementation, utilizing Python and Flask to handle the backend processing and HTML, CSS, and JavaScript for the front-end interface.

6.2 Design

The design of Project 2 emphasizes a minimalistic and user-friendly interface. The front-end is implemented using HTML and JavaScript, providing a clean and intuitive platform for users to in-put text and view the translated sign language output. The interface consists of a text box for input, a convert button to initiate the translation, and an area to display the generated sign language images.

6.3 Implementation

Project 2's implementation leverages Python and Flask for backend processing and HTML, CSS, and JavaScript for the front-end interface. The implementation involves the following:
(1) Text Input: The user enters the English text in the text box on the front-end interface.
(2) Backend Processing: The Flask backend receives the input text, splits it into words, and searches for matching signs in the dataset.
(3) Image Retrieval: The backend retrieves the relevant image translations for the identified signs.
(4) Front-end Display: The retrieved images are sent back to the front-end and displayed in a row-wise format.

6.4 Code Snippets

The following code snippets showcase key aspects of the implementation:

(1) Backend Processing (Python)

```
from flask import Flask, render_template, request
app = Flask(__name__, template_folder='templates')
@app.route('/', methods=['GET']) def index():
return render_template('index.html')
if __name__ == "__main__": app.run(host='0.0.0.0',
port=8080)
```

(2) Front end Interaction (JavaScript)

```
function callBack() {
let content = document.getElementById("input").value; // ...
(rest of the code for processing and displaying images)
}
```

6.5 Technologies Used

Project 2 utilizes the following technologies:

Python: For backend processing and logic.
Flask: As a web framework for handling requests and routing.
HTML, CSS, and JavaScript: For creating the front-end interface and user interaction.
Visual Studio Code: As the development environment.

6.6 Evaluation

Project 2 was evaluated by a developer, a homemaker, and a tester. The evaluation focused on usability aspects such as ease of use, interface attractiveness, and overall functionality. The feedback received was positive, highlighting the user-friendly interface and efficient translation process. However, some limitations were noted, such as the inability to handle special characters and the lack of prompts for missing words.

Overall, Project 2 presents an alternative implementation of a text-to-ASL finger spelling translator using Python, Flask, and web technologies. The project shows a user-friendly interface and efficient backend processing. While it has certain limitations, it offers valuable insights into different technological approaches for sign language translation.

7. COMPARISON AND ANALYSIS

This section provides a comparative analysis of Project 1 and Project 2, highlighting their similarities, differences, and respective strengths and weaknesses.

7.1 Similarities

Both projects share the fundamental goal of translating English text to ASL finger spelling, utilizing image-based representations of ASL hand shapes. They both employ a character-by-character translation approach, where the input text is processed to extract individual characters, which are then mapped to corresponding ASL images. Additionally, both projects prioritize user-friendliness by implementing simple and intuitive web-based interfaces for user interaction.

7.2 Differences

Despite their shared goal and approach, Project 1 and Project 2 exhibit several key differences:
Technology Stack: Project 1 utilizes PHP for backend processing, while Project 2 employs Python and Flask. On the frontend, Project 1 uses HTML and Bootstrap, whereas Project 2 relies on HTML, CSS, and JavaScript.
Implementation Complexity: Project 1's implementation is relatively simpler, relying on basic string processing and image mapping techniques. In contrast, Project 2 involves more complex backend logic with Flask handling web requests and routing.
Features: Project 2 demonstrates a slightly more advanced feature set, including dynamic image generation and display using JavaScript. Project 1, on the other hand, relies on pre-generated image files and basic HTML for display.
Evaluation Methodology: Project 1 was evaluated by a group of friends, while Project 2 involved a developer, a homemaker, and a tester. This difference in evaluation methodology reflects the varying perspectives considered for each project.

7.3 Interfaces

Figure 1 showcases the home screen of Project 1, presenting a clean and simple interface with a text box for user input and a submit button.

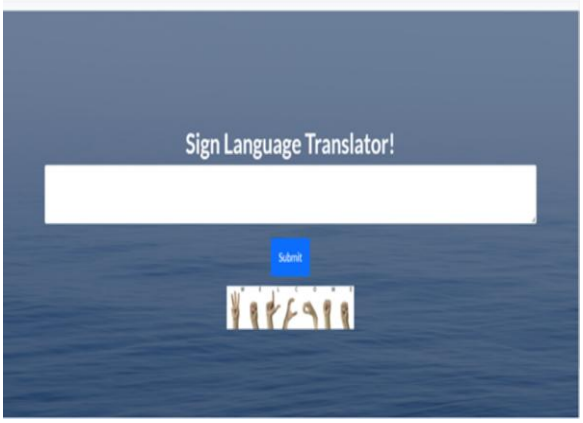

Fig 1. Project 1 Home Screen

Figure 2 displays the output screen of Project 1, showing the translated sign language images corresponding to the input text.

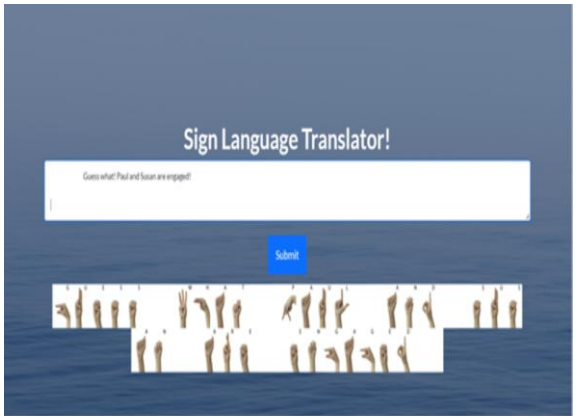

Fig 2. Project 1 Output Screen

Figure 3 illustrates the home screen of Project 2, featuring a text box for input and a convert button.

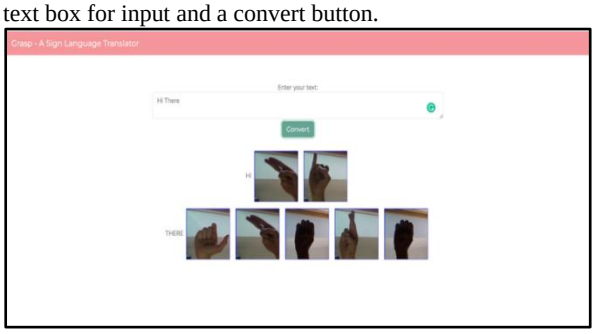

Fig 3. Project 2 Home Screen

Figure 4 presents the output screen of Project 2, displaying the translated sign language images for the given text.

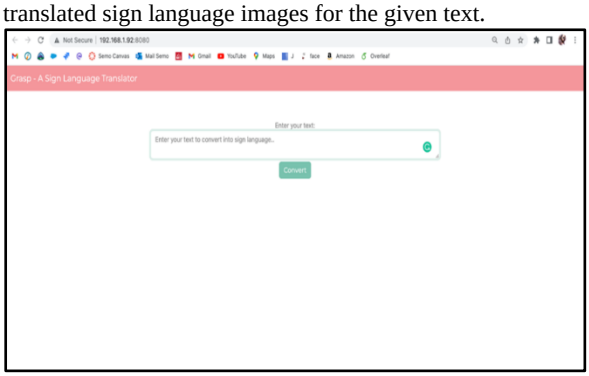

Fig 4. Project 2 Output Screen

7.4 Code Snippets Comparison

Next is an example of how text Processing is handled in Project 1 using (PHP):

```
<?php
// Input
$phrase = $_POST[’phrase’];
// Converting the string to lowercase $phrase =
strtolower($phrase);


        // Splitting the string input into an array of characters
$phrase = str_split($phrase); ?>
```

Next is an example code from Project 2 using (JavaScript):

```
function callBack() {
let content = document.getElementById("input").value; let
items = content.split(" ");
items = items.map((item) => item.trim().split("")); // ... further
processing and image display
}
```

Both projects perform similar text processing, but Project 2’s JavaScript implementation includes additional steps like splitting the input into words and trimming whitespace.

7.5 Image Handling

Project 1 (PHP): Relies on pre-generated image files and uses a simple img tag for display.

```
$results .= "<img src=$character height=72>";
```

Project 2 (JavaScript): Dynamically generates img elements and sets their attributes.

```
let img = document.createElement("img"); img.src =
"../static/images/alpha/" + element[j].toUpperCase() +
"_test.jpg"; img.className = "m-2";
img.style = "max-height: 120px";
imageDiv.appendChild(img);
```

Project 2’s approach allows for more flexibility and control over image display.

7.6 Comparative Table

Table 1 summarizes the key differences between Project 1 and Project 2:

Table 1: Comparing the key differences between projects 1 and 2.

Characteristics	**Project 1**	**Project 2**
Backend Technology	PHP	Python, Flask
Frontend Technology	HTML, Bootstrap	HTML, CSS, JavaScript
Implementation Complexity	Simpler	More Complex
Features	Basic image mapping and display	Dynamic image generation
Evaluation	Friends	Developer, Homemaker, Tester

7.7 Analysis

Project 1’s simplicity makes it easier to understand and implement, particularly for those familiar with PHP and basic web development. Project 2, with its use of Python and Flask, demonstrates a more robust and scalable approach, suitable for handling larger datasets and more complex functionalities. The dynamic image generation in Project 2 offers a more interactive and engaging user experience.

The evaluation of both projects reveals positive feedback regarding usability and effectiveness. However, the different evaluation methodologies provide unique perspectives. Project 1’s evaluation by friends highlights its user-friendliness and accessibility to a general audience. Project 2’s evaluation by individuals with varying technical expertise showcases its robustness and potential for broader applications.

Both Project 1 and Project 2 successfully demonstrate the feasibility of text-to-ASL finger spelling translation using web technologies. Their comparative analysis reveals valuable insights into different technological approaches, implementation complexities, and evaluation methodologies.

These insights can inform future development efforts and contribute to the advancement of accessible communication tools for the Deaf community.

8. COMPARISON OF EVALUATION

While both Project 1 and Project 2 aimed to achieve similar goals, their evaluation methods and the participant groups differed, leading to varied results and insights. This section compares the evaluation results of both projects, highlighting key findings and areas for improvement.

Project 1's evaluation, conducted with five master's students, focused on a goal-based approach. Participants rated the system on five key aspects: clarity of goal, ease of input, UI design, accuracy of translation, and overall performance. The results showed high ratings for UI design and ease of input, indicating a user-friendly interface and a clear understanding of the system's purpose. However, the lack of support for special characters and shortcuts was identified as a limitation.

Project 2's evaluation involved three participants with diverse backgrounds: a developer, a homemaker, and a tester. The evaluation utilized a usability testing approach with a questionnaire focusing on ease of use, interface attractiveness, and overall functionality. The results were generally positive, highlighting the system's user-friendly interface and efficient translation process. However, limitations such as the inability to handle special characters and the lack of prompts for missing words were noted.

Comparing the results of both projects reveals interesting insights. Project 1's evaluation by a homogenous group of master's students suggests that the system is well-suited for educational purposes and users with a basic understanding of technology. On the other hand, Project 2's evaluation by individuals with varying technical expertise indicates its potential for broader applications and user groups. Both evaluations identified limitations related to handling special characters, suggesting a common area for improvement in future development. Additionally, Project 2's evaluation highlighted the need for more informative feedback mechanisms, such as prompts for missing words, to enhance user experience.

The comparison of evaluation results reveals valuable insights into the strengths and limitations of each project. Project 1 demonstrates strong user-friendliness and clarity of purpose, while Project 2 showcases broader applicability and potential for diverse user groups. The identified limitations provide a road map for future development, guiding the refinement and enhancement of sign language translation systems to better serve the needs of the Deaf community.

9. CONCLUSION

This paper examined two distinct implementations of a text-to-ASL finger spelling translation system, Project 1 and Project 2, developed by students. Both projects successfully achieved their core functionality, converting English text into corresponding ASL fin-ger spelling images. However, their approaches differed in terms of technologies used and implementation complexities. Project 1 utilized PHP for a simpler implementation, while Project 2 employed Python and Flask for a more robust and scalable solution.

The comparison of these projects revealed valuable insights. Firstly, it highlighted the diverse approaches and technologies that can be employed for sign language translation, offering flexibility for future development. Secondly, it emphasized the importance of user centered design and evaluation in developing accessible technologies for the Deaf community. The evaluations, while differing in methodology, revealed positive user experiences and identified areas for improvement, such as handling special characters and incorporating additional features. This feedback is crucial for refining and enhancing future iterations of sign language translation systems.

In conclusion, both Project 1 and Project 2 contribute valuable insights into the development and implementation of text-to-ASL finger spelling translation systems. Their comparative analysis highlights the flexibility in technological choices, the importance of user-centered design, and the value of diverse evaluation approaches. These findings can guide future research and development efforts, ultimately contributing to more accessible and effective communication tools for the Deaf community. It is also important to note that the development of these two projects were completed in one month as an end of semester project.

10. REFERENCES

[1] Wikipedia contributors. American sign language, 2025. URL https://en.wikipedia.org/wiki/American_Sign_Language. Accessed: 2025-02-25.

[2] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and R. Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10020–10030, 2020. URL https://api. semanticscholar.org/CorpusID:214728269.

[3] Hao Zhou, Wen gang Zhou, Weizhen Qi, Junfu Pu, and Houqiang Li. Improving sign language translation with monolingual data by sign back-translation. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1316–1325, 2021. URL https://api. semanticscholar.org/CorpusID:235195736.

[4] Sandra Baldassarri, Eva Cerezo, and Francisco Royo-Santas. Automatic translation system to spanish sign language with a virtual interpreter. volume 5726, pages 196–199, 08 2009.ISBN 978-3-642-03654-5. doi: 10.1007/978-3-642-03655-2 23.

[5] Daiga Straupeniece, Dina Bethere, and Elza Ozola. Sign lan-guage of the deaf people: A study on public understanding. Education. Innovation. Diversity., 2:109–114, 12 2023. doi: 10.17770/eid2023.2.7356.

[6] Hicham Abdelouafi. The necessity for research on indigenous sign languages to enrich africa's deaf communities. 11 2024.

[7] Kirsten Bergmann. The production of co-speech iconic ges-tures: Empirical study and computational simulation with vir-tual agents. 2012. URL https://api.semanticscholar. org/CorpusID:36814279.

[8] Clayton Valli and Ceil Lucas. Linguistics of American Sign Language: an introduction. Gallaudet University Press, Washington, D.C., 3rd ed. edition, 2000. URL https: //cmc.marmot.org/EbscoAcademicCMC/ocm57352333. eBook, Available Online, Ebsco Academic (CMC).

[9] Parteek Bhatia, Sugandhi Verma, and Sanmeet Kaur. Sign language generation system based on indian sign language grammar. ACM Transactions on Asian Language Information Processing, 01 2020.

[10] Ida Bagus, Nyoman Wairagya, Putu Wira Buana, and I Made Sukarsa. Development of english-to-sign-language trans-lation system on android. 2019. URL https://api. semanticscholar.org/CorpusID:212584311.