Engineering and Technology Journal e-ISSN: 2456-3358

Volume 11 Issue 05 May-2026, Page No.-9936-9945

DOI: 10.47191/etj/v11i05.28, I.F. – 8.595

Sign Language Interpreter: A Mobile-Based Communication Application for Bridging the Communication Gap Between Deaf and Hearing Individuals

Felise O. Marcaida[1], Althea D. L. Romero[2], Robert John V. Santos[3], Eliza Ayo[4]

[1,2,3,4]School of Science and Technology, Centro Escolar University – Manila, Philippines

ABSTRACT: Communication between deaf and hearing individuals remains a persistent challenge despite advances in mobile and machine-learning technologies. Existing sign language interpretation tools often suffer from limited vocabulary, dependence on specialized hardware, lack of integrated text-to-speech (TTS) functionality, environmental sensitivity, and the absence of offline capability.

This study aimed to design, develop, and evaluate a mobile-based Android application capable of (a) translating American Sign Language (ASL) gestures into text and synthesized speech in real time, (b) providing a built-in tutorial of the alphabet and commonly used words, and (c) operating offline on standard Android smartphones.

A descriptive-developmental research design was employed, complemented by an Agile software development life cycle (Define–Design–Develop–Demonstrate–Release). The application was implemented in Java using Android Studio, with hand-landmark detection performed by Google's MediaPipe framework. Quality assurance was conducted through automated tools (Sauce Labs, BrowserStack, WeTest, SonarQube) covering functionality, usability, performance, security, compatibility, reliability, and maintainability. Manual testing examined the influence of lighting, background, hand position, and hand size. User Acceptance Testing (UAT) was administered to 30 purposively sampled respondents (15 deaf, 15 hearing). Data were analyzed using percentages, weighted mean, and standard deviation.

Functionality testing yielded a 90.0% pass rate. Security testing revealed no high- or medium-risk vulnerabilities. Compatibility testing confirmed seamless operation on Android 12, 13, and 14 (rated Excellent) and acceptable performance on Android 11 (rated Good). Recognition accuracy was approximately 75% under controlled conditions, with optimal performance observed in natural light, against solid backgrounds, and with the hand placed frontally to the camera. UAT results across the eight ISO/IEC 25010 quality characteristics (Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability) ranged from 4.55 to 4.72 on a 4-point scale, all corresponding to a "Strongly Agree" verbal interpretation.

The developed mobile application addresses key gaps identified in prior literature by integrating real-time gesture-to-text translation, on-device text-to-speech synthesis, an embedded tutorial, and offline functionality on standard Android hardware. While the system performs reliably under controlled conditions, future work should focus on expanding the lexicon, accommodating dynamic gestures, improving robustness to lighting and background variability, and supporting cross-platform deployment.

KEYWORDS: *sign language recognition; mobile application; MediaPipe; text-to-speech; assistive technology; Android; ISO/IEC 25010*

1. INTRODUCTION

Communication is a fundamental human right, yet for individuals who are deaf or hard of hearing, everyday interactions with the hearing population frequently involve significant barriers. Sign language—a complete visual-gestural language that conveys meaning through hand shapes, facial expressions, and body movement—remains the primary means of communication for the Deaf community. However, the limited proficiency of the general public in sign language continues to constrain the social, educational, and economic participation of its users. The widespread availability of smartphones and recent advances in computer vision, machine learning, and natural language processing offer a promising avenue for closing this gap through accessible mobile-based assistive technologies.

Existing solutions for automated sign language interpretation include glove-based sensor systems, depth-camera devices such as Microsoft Kinect, and vision-based mobile applications.

While each approach has demonstrated some technical merit, persistent limitations include high cost, dependence on specialized hardware, restricted vocabulary, the absence of text-to-speech (TTS) output, environmental sensitivity, and reliance

on continuous internet connectivity. These limitations restrict the practical adoption of such technologies in real-world communication contexts—particularly in low-resource settings.

1.2 Research Problem and Gap in the Literature

A systematic review of the existing literature reveals several recurring gaps in mobile-based sign language interpretation systems. First, many models are restricted to alphabet recognition with little or no support for full words or commonly used phrases (Obi et al., 2023). Second, real-time text-to-speech functionality—an essential component for bidirectional communication—is frequently absent or implemented only as a desktop feature (Alam et al., 2021; Papastratis et al., 2021). Third, applications often degrade substantially under non-ideal lighting or with cluttered backgrounds (Abalos et al., 2021), and many depend on a constant network connection. Finally, integrated learner-oriented features, such as built-in tutorials, are rare in deployed mobile solutions (Brindha et al., 2022; Nyaga & Wario, 2018). Collectively, these gaps motivate the development of a more inclusive, lightweight, and self-contained mobile system.

1.3 Objectives of the Study

The study sought to design, develop, and evaluate a mobile-based sign language interpreter for Android devices. Specifically, the research aimed to: (a) identify the gaps in existing sign language interpretation systems through a structured review of literature and similar applications; (b) determine the appropriate hardware, operating system, software, and machine learning technologies for development; (c) design and implement a real-time gesture-to-text and text-to-speech application supplemented by a built-in tutorial; (d) evaluate the application using both automated quality assurance tools and manual testing across varied environmental conditions; and (e) assess user acceptance among deaf and hearing respondents using ISO/IEC 25010 product quality characteristics.

1.4 Hypotheses

H0. The developed mobile application does not significantly improve communication effectiveness and efficiency between deaf and hearing individuals despite the inclusion of a limited basic-word vocabulary, a text-to-speech function, and a built-in tutorial.

H1. The developed mobile application significantly improves communication effectiveness and efficiency between deaf and hearing individuals through the integration of a limited basic-word vocabulary, a text-to-speech function, and a built-in tutorial.

1.5 Significance of the Study

The findings of this study contribute to the growing literature on assistive communication technologies and inclusive design. The application is intended to serve members of the deaf and hard-of-hearing community as a portable, offline-capable communication aid; to provide non-signing users with a scaffolded entry point into sign language through embedded tutorials; and to inform educators, service providers, and policymakers regarding the role of mobile assistive technology in promoting inclusion. The work also offers a replicable framework for future researchers seeking to extend lightweight, vision-based recognition models to lower-resource settings.

1.*6 Scope and Delimitation*

The application was developed exclusively for the Android operating system (versions 11–14) and recognizes only static, single-handed ASL gestures corresponding to the alphabet and a limited set of commonly used words. Dynamic gestures, full-body sign recognition, multilingual sign systems, emotion recognition, and contextual lipreading were excluded. The application functions offline and does not implement user authentication or data-storage security features, as no sensitive personal information is collected. Optimal recognition assumes the hand is presented frontally to the camera against a uniform background.

2. REVIEW OF RELATED LITERATURE

The literature on automated sign language recognition (SLR) and mobile interpretation systems can be organized around four interrelated themes: (a) sensor- and vision-based recognition approaches, (b) machine-learning architectures for gesture classification, (c) integration of text-to-speech and tutorial features, and (d) deployment platforms and accessibility considerations.

2.1 Sensor-Based and Vision-Based Recognition Approaches

Two principal paradigms dominate sign language recognition research: sensor-based systems and vision-based systems. Sensor-based systems employ instrumented gloves, accelerometers, or radio-frequency signals to capture hand motion. Rajamohan and colleagues described a glove-based deaf–mute communication interpreter, while Rewari et al. (2018) implemented an Arduino-Uno-controlled interpreter using flex sensors, an SD-card module, and Bluetooth. Although such systems are robust to changes in illumination, they are intrusive, costly, and constrain natural articulation (Nandi, Shinde, & Itkarkar, 2017). Lee and Gao (2020) demonstrated that Wi-Fi Channel State Information processed through a dual-output two-stream convolutional neural network can recognize gestures non-invasively, achieving recognition accuracies of 99.13%, 96.79%, and 97.08% in home, laboratory, and combined environments, respectively.

Vision-based approaches use ordinary cameras coupled with image-processing or learned feature extractors. Shukor et al. (2015) emphasized the advantage of incorporating facial expressions and head gestures alongside hand shapes, while Raboy et al. (2015) developed an Android-based ASL alphabet translator that segmented the hand region and matched its contour against a trained dataset. Ardiansyah et al. (2021) reviewed the use of Microsoft Kinect, which combines color and depth imaging to improve foreground–background segmentation. More recently, Bisht et al. (2022) and Abalos et al. (2021) demonstrated that Google's MediaPipe framework

supports lightweight, real-time hand-landmark detection suitable for both web and mobile deployment.

2.2 Machine-Learning Architectures

A wide range of classifiers have been applied to sign language recognition. Support Vector Machines (SVMs) have been used for static-gesture classification with reported accuracies of approximately 97.13% on a 16-gesture ASL subset and 80.30% across the full 26-letter alphabet (Chong & Lee, 2018). Sood, Hernández, and Navdiya (2023) and Johnny and Nirmala (2021) reported that the k-nearest neighbors (kNN) algorithm outperformed neural networks and decision-tree classifiers in their experiments, providing an accessible benchmark for mobile deployment. Convolutional neural networks (CNNs)—including LeNet-5 architectures (Vishwanath & Yawer, 2019), CNNs trained on the Sign Language MNIST dataset (Mannan et al., 2022, achieving 99.67% accuracy), and PCANet variants (Mohsin, 2024, with 93.67% accuracy on ASL alphabets, numerals, and basic words)—have established the dominance of deep-learning methods for image-based gesture classification. GAN-based context-aware systems for continuous sign recognition (Papastratis et al., 2021) push toward sentence-level translation but remain computationally demanding.

2.3 Integration of Text-to-Speech and Tutorial Features

Despite the proliferation of recognition algorithms, fewer systems integrate the recognition pipeline with output modalities suitable for spoken communication. Chandragandhi et al. (2021) described a desktop CNN-based system that converts gestures to text and synthesized audio, but mobile deployments with comparable functionality remain scarce. Bisht et al. (2022) integrated MediaPipe with text-to-speech and speech-to-text features on a web platform, while Batte (2017) argued that mobile-based translation has the potential to expand deaf users' access to public services and reduce social isolation. The integration of structured tutorials—particularly those covering both the alphabet and frequently used words—has been highlighted as a means of fostering bidirectional communication, but is rarely implemented in deployed applications.

2.4 Deployment Platforms and Accessibility

Most prior work has been deployed either on workstations using Python or on specialized hardware such as Raspberry Pi (Alam et al., 2021). Although these implementations advance algorithmic understanding, they limit the everyday accessibility of the technology. The growing ubiquity of Android smartphones, combined with on-device machine-learning frameworks such as MediaPipe and TensorFlow Lite, makes mobile deployment increasingly feasible. Halawani (2008) and Parton (2005) emphasized that affordable, locally accessible technologies are essential for closing the global communication gap, particularly for under-served sign languages.

2.5 Synthesis and Position of the Present Study

Taken together, the literature establishes that (a) vision-based recognition using lightweight frameworks such as MediaPipe is a feasible foundation for mobile deployment; (b) machine-learning classifiers can achieve high accuracy on static gestures, with kNN offering a reasonable trade-off between accuracy and computational cost; (c) integration of TTS and tutorials remains under-developed in mobile contexts; and (d) offline-capable, environmentally-robust systems are still rare. The present study addresses these gaps by combining MediaPipe-based hand-landmark detection with a kNN-based classifier, integrating text-to-speech output, embedding a tutorial covering the alphabet and commonly used words, and deploying the system as a fully offline Android application.

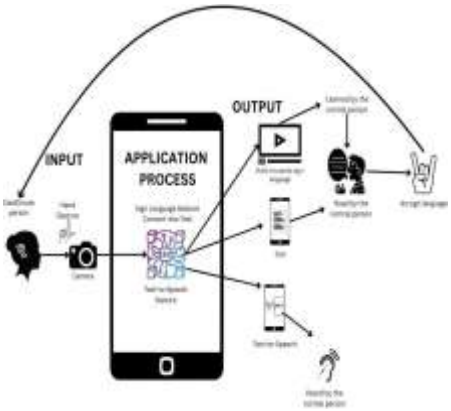

Fig. 1. ***Conceptual Framework***

3. METHODOLOGY

3.1 Research Design

The study employed a descriptive-developmental research design. The descriptive component characterized the present state of mobile-based sign language interpretation, including the gaps identified in the literature and in similar applications. The developmental component applied an Agile software-development life cycle—comprising the phases Define, Design, Develop, Demonstrate, and Release—to the iterative construction and refinement of the proposed application. A pragmatic research orientation guided the work, emphasizing real-world utility, user-centered iteration, and practical evaluation against established quality criteria.

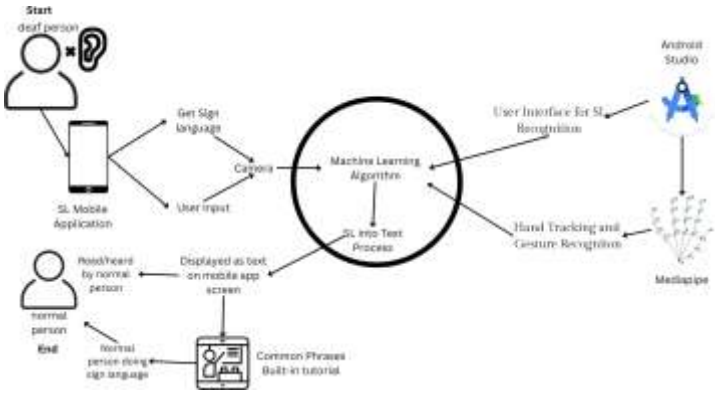

Fig.2. ***Process Flow***

3.2 Participants and Sampling

Purposive sampling was used to recruit thirty (30) respondents for the User Acceptance Testing (UAT) phase, comprising

fifteen (15) deaf individuals and fifteen (15) hearing individuals. Purposive sampling was selected because the evaluation required participants with direct relevance to the research aim—namely, deaf users who would benefit from the gesture-to-speech feature, and hearing users who would interact with deaf respondents through the application. Demographic data collected included full name, age, sex, mobile number, and (for deaf respondents) the number of years they had been deaf. Participants also indicated whether they had previously used any mobile application for sign language interpretation.

3.2 Participants and Sampling

Purposive sampling was used to recruit thirty (30) respondents for the User Acceptance Testing (UAT) phase, comprising fifteen (15) deaf individuals and fifteen (15) hearing individuals. Purposive sampling was selected because the evaluation required participants with direct relevance to the research aim—namely, deaf users who would benefit from the gesture-to-speech feature, and hearing users who would interact with deaf respondents through the application. Demographic data collected included full name, age, sex, mobile number, and (for deaf respondents) the number of years they had been deaf. Participants also indicated whether they had previously used any mobile application for sign language interpretation.

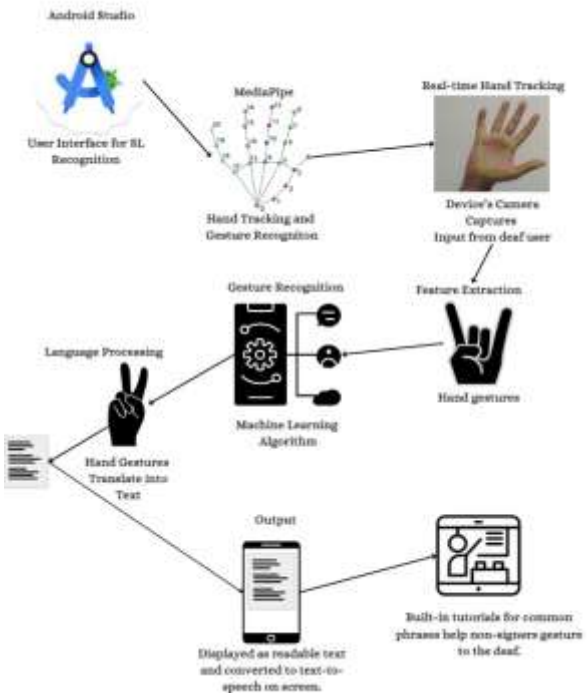

Fig. 3. ***Software and Hardware Requirement***

3.3 Hardware and Software Environment

Three hardware configurations were evaluated for development purposes: (a) a laptop equipped with an AMD Ryzen 5 3500U CPU, 8 GB RAM, integrated Radeon Vega graphics, and 256 GB SSD; (b) a desktop PC with an AMD Ryzen 5 3400G CPU, 32 GB RAM, an NVIDIA GTX 1650 GPU, and combined 480 GB SSD plus 2 TB HDD storage; and (c) a laptop with an Intel Core i3-1005G1 CPU, 12 GB RAM, integrated Intel UHD Graphics, and a 117 GB SSD. The desktop PC was selected as the primary development workstation. Application development was performed in Android Studio using Java; gesture recognition was implemented through Google's MediaPipe hand-tracking framework. Test deployment targeted Android smartphones running operating-system versions 11 through 14 with cameras of at least 16-megapixel resolution.

3.4 Software Development Methodology

Development followed the five-stage Agile model previously described. In the Define phase, functional and non-functional requirements were specified, prioritizing accessibility, recognition accuracy, and real-time responsiveness. In the Design phase, wireframes and user-interface mockups were prepared, and the integration of MediaPipe hand-tracking modules was planned. The Develop phase implemented the core gesture-recognition pipeline, the text-to-speech component, and the built-in tutorial; user feedback was solicited iteratively to refine performance. The Demonstrate phase consisted of stakeholder demonstrations to validate functionality and gather usability feedback. Finally, the Release phase produced the version subjected to formal evaluation.

Fig. 4. ***Software Methodology***

3.5 Instruments and Evaluation Procedures

Evaluation was conducted through three complementary procedures. First, automated quality-assurance testing was performed using established external platforms: Sauce Labs for functionality and performance testing, BrowserStack for usability evaluation, WeTest for security scanning, and SonarQube for source-code maintainability analysis. Manual compatibility testing was conducted on three Android devices (Samsung A13 running Android 14; POCO X4 GT running Android 13; and Redmi Note 13 Pro+ 5G running Android 13). Second, manual environmental testing assessed recognition performance under varying lighting conditions (low, natural, bright), background clarity (white, black, mixed colors), hand position (front, side, top), and hand size (small, medium, large). Third, User Acceptance Testing was conducted using a researcher-developed questionnaire aligned with the ISO/IEC 25010 product-quality characteristics, covering Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability.

3.6 Statistical Treatment

Quantitative data were analyzed using descriptive statistics. The percentage distribution was computed to summarize categorical respondent characteristics. The weighted mean was used to summarize Likert-scale responses, and standard deviation was reported as a measure of dispersion.

Performance metrics derived from automated testing included the Average Time per Message (ATM), Message Delivery Rate (MDR), Failure Rate, Process Capability Index, and Performance Index, defined as follows:

1. Failure Rate = (Number of failures) ÷ (Total operating time)
2. Process Capability Index = (USL − LSL) ÷ (6 × σ)
3. Performance Index = (Actual performance ÷ Expected performance) × 100%

where USL and LSL denote the upper and lower specification limits and σ denotes the standard deviation.

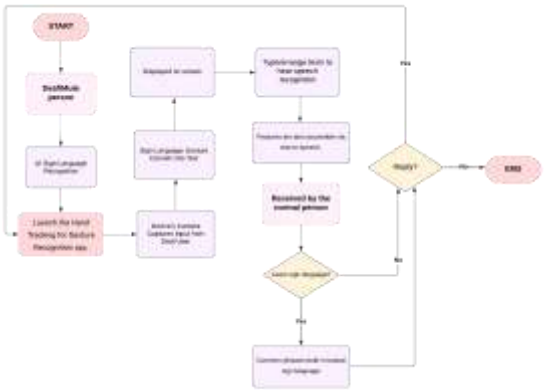

Fig.5. ***Flowchart***

3.5 Instruments and Evaluation Procedures

Evaluation was conducted through three complementary procedures. First, automated quality-assurance testing was performed using established external platforms: Sauce Labs for were examined for the literature review, and three commercially available applications were assessed for the comparative analysis.

functionality and performance testing, BrowserStack for usability evaluation, WeTest for security scanning, and SonarQube for source-code maintainability analysis. Manual compatibility testing was conducted on three Android devices (Samsung A13 running Android 14; POCO X4 GT running Android 13; and Redmi Note 13 Pro+ 5G running Android 13). Second, manual environmental testing assessed recognition performance under varying lighting conditions (low, natural, bright), background clarity (white, black, mixed colors), hand position (front, side, top), and hand size (small, medium, large). Third, User Acceptance Testing was conducted using a researcher-developed questionnaire aligned with the ISO/IEC 25010 product-quality characteristics, covering Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability.

3.7 Ethical Considerations

The study was conducted in accordance with the requirements of the Centro Escolar University Institutional Ethics Review Board (IERB). All participants provided informed consent prior to participation, with explicit acknowledgment that their responses would be used solely for research purposes and treated as anonymous and confidential. Participation was voluntary and uncompensated.

4. RESULTS

This section presents the results of the gap analysis, automated quality-assurance testing, manual environmental testing, and user acceptance testing of the Sign Language Interpreter: Mobile-Based Communication Application.

4.1 Identified Research Gaps

Tables 1 and 2 summarize the principal limitations identified in prior studies and existing similar applications. Six representative studies

Table 1. Gaps identified in the existing literature on mobile sign language interpretation.

Study	Identified Gap
Obi et al. (2023)	Limited to alphabet and a small word set; lacks text-to-speech and tutorial.
Alam et al. (2021)	Absence of text-to-speech functionality and offline operation; requires reliable network access.
Brindha et al. (2022)	Lacks tutorials; emphasizes noisy-image preprocessing rather than text-to-speech or real-time translation.
Papastratis et al. (2021)	Focused on GAN-based continuous-sign recognition; does not include tutorials, text-to-speech, or real-time translation.
Nyaga & Wario (2018)	Restricted gesture-recognition distance; no text-to-speech or tutorial; lacks real-time mobile translation.
Abalos et al. (2021)	Performance is sensitive to environmental factors such as lighting.

Table 2. Gaps identified in commercially available similar applications.

Application	Identified Gap
Sign Language AI (Paul, 2023)	Incomplete alphabet and inaccurate word recognition; recognition halts after 10 s if no gesture is detected.
Hand Sign (Azis, 2021)	Detects hand gestures in real time but does not translate them to text; users must follow predetermined sign sequences.
AI Sign (RockCat Studio, 2020)	Requires full-body sign recognition; lacks alphabet support and text-to-speech functionality; recognition accuracy is limited.

4.2 Hardware and Compatibility Evaluation

The desktop development workstation (AMD Ryzen 5 3400G, 32 GB RAM, GTX 1650, 480 GB SSD + 2 TB HDD) received an Excellent rating, providing seamless multitasking and real-time text-to-speech conversion. The two laptop configurations received Fair and Good ratings, respectively, owing to lower memory and storage capacity. Compatibility testing across Android versions is summarized in Table 3.

Table 3. Device compatibility ratings across Android versions.

Android Version	Rating	Observation
11	Good	Functions adequately; minor lags and reduced responsiveness compared to newer versions.
12	Excellent	Seamless performance with enhanced processing speed and hardware support.
13	Excellent	Fully optimized; accurate real-time gesture interpretation.
14	Excellent	Optimal performance leveraging the latest system and hardware integration.

4.3 Quality-Assurance Testing

Table 4 summarizes the results of automated quality-assurance testing across functionality, usability, performance, security, and maintainability. The functionality test (Sauce Labs) yielded a 90.0% pass rate over five test runs (total duration 11 m 11 s); a single test produced a minor issue not affecting core functionality. The usability test (BrowserStack) identified 40 issues, of which 22 concerned readable text and layout, 16 were minor, two were classified as serious, and none was critical. The performance test (Sauce Labs) passed with no errors. The security test (WeTest) detected zero high- or medium-risk vulnerabilities, two low-risk vulnerabilities, and two safety risks. Compatibility testing on the Samsung A13, POCO X4 GT, and Redmi Note 13 Pro+ 5G passed without failures.

Table 4. Summary of automated quality-assurance results.

Test Type	Tool	Outcome
Functionality	Sauce Labs	Pass rate 90.0%; error rate 10.0%; completion 90.0%.
Usability	BrowserStack	40 issues (16 minor, 22 moderate, 2 serious, 0 critical).
Performance	Sauce Labs	1/1 test passed; no errors.
Security	WeTest	0 high-risk, 0 medium-risk, 2 low-risk vulnerabilities, 2 safety risks.
Compatibility	Manual (3 devices)	Passed on Samsung A13, POCO X4 GT, Redmi Note 13 Pro+ 5G.

Reliability testing conducted over 10- and 15-minute sessions yielded an approximate 75% recognition success rate when the hand was correctly positioned in front of the camera; three of four scripted actions completed successfully, with one partial success and no technical failures. Maintainability analysis using SonarQube identified the following issue distributions: LandingActivity.java (2 low, 4 medium, 1 high), LearningActivity.java (no identified issues), LearningLettersActivity.java (0 low, 0 medium, 2 high), and MenuActivity.java (1 low, 5 medium, 1 high).

4.4 Manual Environmental Testing

Manual testing examined the influence of lighting, background, hand position, and hand size on recognition performance. The application achieved its highest performance under natural light, was rated Good under bright light, and Fair under low light. White and black backgrounds yielded Excellent performance, while mixed-color backgrounds yielded a Good rating. With respect to hand position, frontal placement produced Excellent performance, side placement produced Good performance, and top-down placement produced Fair performance. Across small, medium, and large hand sizes, performance was uniformly rated Excellent.

Table 5. Manual testing ratings across environmental conditions.

Condition	Levels Tested	Rating
Lighting	Low / Natural / Bright	Fair / Excellent / Good
Background	White / Black / Mixed	Excellent / Excellent / Good
Hand Position	Front / Side / Top	Excellent / Good / Fair
Hand Size	Small / Medium / Large	Excellent / Excellent / Excellent

4.5 User Acceptance Testing

Thirty respondents (15 deaf, 15 hearing; 100% of whom tested the BETA version) participated in the UAT. Mean ratings and standard deviations across the eight ISO/IEC 25010 quality characteristics are presented in Table 6. All composite means corresponded to a verbal interpretation of "Strongly Agree."

Table 6. User Acceptance Testing results across ISO/IEC 25010 quality characteristics (n = 30).

Quality Characteristic	Mean	SD	Verbal Interpretation
Functional Suitability	4.72	1.31	Strongly Agree
Performance Efficiency	4.55	1.50	Strongly Agree
Compatibility	4.62	0.97	Strongly Agree
Usability	4.67	1.82	Strongly Agree
Reliability	4.67	1.40	Strongly Agree
Security	4.65	0.98	Strongly Agree
Maintainability	4.65	1.90	Strongly Agree
Portability	4.65	1.43	Strongly Agree

5. DISCUSSION

5.1 Addressing Identified Gaps

The results indicate that the developed application addresses the principal gaps identified in the literature. By combining MediaPipe-based hand-landmark detection with a kNN-based classifier, integrating an on-device text-to-speech engine, and embedding a tutorial covering the alphabet and commonly used words, the system extends the functionality of comparable mobile applications such as those reported by Obi et al. (2023), Alam et al. (2021), and Brindha et al. (2022). The decision to deploy the application offline removes the dependence on network access that constrained earlier systems and aligns with calls for affordable, locally accessible technologies (Halawani, 2008; Parton, 2005).

5.2 Performance in Comparison with Prior Work

The 75% recognition accuracy observed under controlled conditions is lower than the 94.69% accuracy reported by Bisht et al. (2022) for a MediaPipe-and-random-forest combination, the 99.67% accuracy reported by Mannan et al. (2022) for a hyper-tuned CNN on the Sign Language MNIST dataset, and the 99.13–97.08% accuracies reported by Lee and Gao (2020) for a dual-output two-stream CNN using Wi-Fi signals. Several factors may account for the present study's lower accuracy: the use of a lightweight kNN classifier rather than a deep CNN; a real-mobile (rather than laboratory) deployment context; the use of a smaller and less curated training dataset; and the inclusion of dynamic recognition during natural use rather than carefully posed gestures. The trade-off is consistent with the priorities of the present work, which emphasized offline operation on low-end smartphones rather than peak laboratory accuracy.

5.3 Environmental Sensitivity

Manual testing confirmed the well-known sensitivity of vision-based recognition systems to environmental conditions, echoing the findings of Abalos et al. (2021). Performance was optimal under natural light and against high-contrast backgrounds, while low-light conditions and mixed-color backgrounds reduced recognition accuracy. The reduced performance observed for side and top-down hand placements likewise reflects the geometric assumptions of frontal-pose hand-landmark models. These findings suggest that future work should investigate adaptive preprocessing—e.g., brightness normalization, background subtraction, or self-supervised domain adaptation—to extend the range of conditions under which the system operates reliably.

5.4 User Perceptions and Acceptance

The uniformly high UAT ratings across the eight ISO/IEC 25010 characteristics suggest strong user acceptance among both deaf and hearing respondents. The highest-rated indicator (suitability for real-time translation, M = 4.83) is consistent with the system's design priorities, and the lowest-rated indicators (clarity of usage instructions and absence of perceptible delays, both M = 4.50) identify two clear targets for refinement: improved on-boarding documentation and further latency optimization. The convergence of high ratings across functional, performance, and usability dimensions also corroborates the literature's emphasis on integrated mobile solutions as a means of reducing communication barriers (Batte, 2017; Papatsimouli et al., 2023).

5.5 Limitations

Several limitations qualify the interpretation of the present findings. First, the sample of 30 respondents was modest, recruited through purposive sampling from a single institution, and may not represent the broader deaf or hearing population. Second, recognition accuracy was reported only for static, single-handed ASL gestures; dynamic gestures, two-handed signs, and grammatical features such as facial expression were not evaluated. Third, the application was tested only on three Android devices and only across versions 11–14; performance on lower-tier hardware and on older Android versions remains unknown. Fourth, the absence of a formal control condition prevents causal inferences about the application's effect on communication outcomes. Finally, the questionnaire relied on self-reported perceptions and was not psychometrically validated.

5.6 Implications for Future Research

The findings imply several directions for further investigation. Recognition robustness can be improved through deep-learning classifiers trained on a larger, more diverse gesture dataset, ideally using a hybrid MediaPipe-CNN pipeline. Adaptive image-processing modules could mitigate environmental sensitivity. Cross-platform deployment to iOS and the web would extend the reach of the system. Continuous sign-language recognition—encompassing sentence-level translation rather than isolated gestures—remains a compelling research target, as does the integration of speech-to-sign output for fully bidirectional communication. Larger-scale evaluation studies with diverse user populations and rigorous psychometric instruments will be essential to establish generalizable evidence of the system's effectiveness.

6. CONCLUSION

This study presented the design, development, and evaluation of a mobile-based sign language interpreter that integrates MediaPipe-based hand-landmark detection, a kNN-based gesture classifier, on-device text-to-speech synthesis, and a built-in tutorial of the ASL alphabet and commonly used words. The application is offline-capable and runs natively on Android smartphones. Automated quality-assurance testing demonstrated a 90.0% functionality pass rate, the absence of high- or medium-risk security vulnerabilities, and seamless compatibility across Android versions 12–14. Manual testing confirmed optimal recognition under natural light and against high-contrast backgrounds, and uniformly excellent performance across hand sizes. User Acceptance Testing among 30 deaf and hearing respondents yielded composite means between 4.55 and 4.72 across the eight ISO/IEC 25010 quality characteristics, all corresponding to a "Strongly Agree" verbal interpretation.

Although recognition accuracy under field conditions (≈75%) trails the laboratory benchmarks of recent deep-learning systems, the application demonstrates that an offline, lightweight, mobile-first approach can address several long-standing gaps in deployed sign language interpretation tools, including the absence of integrated text-to-speech output, the lack of embedded tutorials, and dependence on continuous internet connectivity. With further refinement of the recognition pipeline and broader empirical validation, the application has the potential to contribute meaningfully to the inclusion and communicative autonomy of deaf and hard-of-hearing users.

6.1 Recommendations

1. Expand the application's lexicon to include a broader vocabulary, idiomatic expressions, and culturally relevant signs.
2. Develop iOS and web versions to extend cross-platform accessibility.
3. Adopt deeper machine-learning architectures (e.g., transformer- or CNN-based models) to improve recognition accuracy and support continuous, dynamic gestures.
4. Integrate an in-app feedback mechanism to enable users to report errors and propose enhancements.
5. Improve robustness to non-frontal hand poses and to variable lighting and background conditions.

6. Add data-protection and privacy features in anticipation of future capabilities that may handle sensitive information.
7. Introduce optional internet-enabled features (e.g., over-the-air model updates and cloud-based vocabulary expansion)

REFERENCES

1. Abalos, D. M., Dacsil, J. C. B., & Raquedan, A. A. L. (2021). SignTalk: Real-time mobile for the hearing-impaired individuals through a MediaPipe Holistic-based approach.
2. Abraham, A., & Rohini, V. (2018). Real-time conversion of sign language to speech and prediction of gestures using artificial neural network. Procedia Computer Science, 143, 587–594. https://doi.org/10.1016/j.procs.2018.10.435
3. Adeyanju, I. A., Bello, O. O., & Adegboye, M. A. (2021). Machine learning methods for sign language recognition: A critical review and analysis. Intelligent Systems with Applications, 12, 200056. https://doi.org/10.1016/j.iswa.2021.200056
4. Alam, S., Hussain, M., Munir, M. B., Shalahuddin, M., Ishrak, S., & Islam, M. N. (2021). A machine learning based sign language interpretation system for communication with deaf-mute people. Proceedings of the International Conference on Computing Advancements. https://doi.org/10.1145/3471391.3471422
5. Ardiansyah, A., Hitoyoshi, B., Halim, M., Hanafiah, N., & Wibisurya, A. (2021). Systematic literature review: American sign language translator. Procedia Computer Science, 179, 541–549. https://doi.org/10.1016/j.procs.2021.01.038
6. Azis, S. N. (2021). Handsign: Learn a sign language with your camera. DEV Community.
7. Bachchhav, P., & Potgantwar, A. (2019). Sign language interpreter using Kinect motion sensor using machine learning. International Journal of Innovative Technology and Exploring Engineering, 8(12).
8. Batte, B. (2017). A mobile-based system for translating sign language and facilitating communication between deaf and non-deaf people. Social Science Research Network. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4553096
9. Bisht, D., Kojage, M., Shukla, M., Patil, Y., & Bagade, P. (2022). Smart communication system using sign language interpretation. In 2022 31st Conference of Open Innovations Association (FRUCT). IEEE. https://doi.org/10.23919/FRUCT54823.2022.9770914
10. Brindha, R., Renukadevi, P., Vathana, D., & Jayanthi, D. (2022). Sign language interpreter. In 2022 International Conference on Inventive Computation Technologies (ICICT) (pp. 1024–1028). IEEE.
11. Chandragandhi, S., R, A. R., Ml, M. S., Akhil, S., & Pt, P. (2021). Real-time translation of sign language to speech and text. International Advanced Research Journal in Science, Engineering and Technology, 8(4). https://doi.org/10.17148/IARJSET.2021.8412
12. Chong, T. W., & Lee, B. (2018). American sign language recognition using Leap Motion controller with machine learning approach. Sensors, 18(10), 3554. https://doi.org/10.3390/s18103554
13. Halawani, S. M. (2008). Arabic sign language translation system on mobile devices. International Journal of Computer Science and Network Security, 8(1), 251–256.
14. Harkude, A., Namade, S., Patil, S., & Morey, A. (2020). Audio to sign language translation for deaf people. International Journal of Engineering and Innovative Technology, 9(10), 30.
15. Johnny, S., & Nirmala, S. J. (2021). Sign language translator using machine learning. SN Computer Science, 3(1). https://doi.org/10.1007/s42979-021-00896-y
16. Lee, C., & Gao, Z. (2020). Sign language recognition using two-stream convolutional neural networks with Wi-Fi signals. Applied Sciences, 10(24), 9005. https://doi.org/10.3390/app10249005
17. Mannan, A., Abbasi, A., Javed, A. R., Ahsan, A., Gadekallu, T. R., & Xin, Q. (2022). Hypertuned deep convolutional neural network for sign language recognition. Computational Intelligence and Neuroscience, 2022, 1–10. https://doi.org/10.1155/2022/1450822
18. Mohsin, S. (2024). Deep-learning approaches for static sign language recognition: A review. [Bibliographic details require verification].
19. Nandi, U., Shinde, A., & Itkarkar, R. R. (2017). Static and dynamic hand gesture recognition: A comparative study. International Journal of Engineering Research and Technology, 6(8), 23–27.
20. Nyaga, C., & Wario, R. (2018). Sign language gesture recognition through computer vision. In 2018 IST-Africa Week Conference (IST-Africa). IEEE.
21. Obi, Y., Claudio, K. S., Budiman, V. M., Achmad, S., & Kurniawan, A. (2023). Sign language recognition system for communicating to people with disabilities. Procedia Computer Science, 216, 13–20. https://doi.org/10.1016/j.procs.2022.12.106
22. Papastratis, I., Chatzikonstantinou, C., Konstantinidis, D., Dimitropoulos, K., & Daras, P. (2021). Artificial

intelligence technologies for sign language. Sensors, 21(17), 5843. https://doi.org/10.3390/s21175843

23. Papastratis, I., Dimitropoulos, K., & Daras, P. (2021). Continuous sign language recognition through a context-aware generative adversarial network. Sensors, 21(7), 2437. https://doi.org/10.3390/s21072437
24. Papatsimouli, M., Sarigiannidis, P., & Fragulis, G. F. (2023). A survey of advancements in real-time sign language translators: Integration with IoT technology. Technologies, 11(4), 83. https://doi.org/10.3390/technologies11040083
25. Parton, B. S. (2005). Sign language recognition and translation: A multidisciplined approach from the field of artificial intelligence. The Journal of Deaf Studies and Deaf Education, 11(1), 94–101. https://doi.org/10.1093/deafed/enj003
26. Raboy, L., Canlas, J., Renejane, C., Mojica, C., & Bandiala, A. (2015). American sign language alphabet translator Android application: Hand shapes into text. Social Science Research Network. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3097464
27. Rajasekhar, N., Yadav, M., Vedantam, C., Pellakuru, K., & Navapete, C. (2023). Sign language recognition using machine learning algorithm. In 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS). IEEE. https://doi.org/10.1109/ICSCSS57650.2023.10169820
28. Rewari, H., Dixit, V., Batra, D., & Hema, N. (2018). Automated sign language interpreter. ResearchGate. https://www.researchgate.net/publication/328912139
29. RockCat Studio. (2020). AI Sign: Sign language [Mobile application].
30. Sanaullah, M., Ahmad, B., Kashif, M., Safdar, T., Hassan, M., Hasan, M. H., & Aziz, N. (2022). A real-time automatic translation of text to sign language. Computers, Materials & Continua, 70(2), 2471–2488.
31. Shukor, A. Z., Miskon, M. F., Jamaluddin, M. H., Bin Ali, F., Asyraf, M. F., & Bin Bahar, M. B. (2015). A new data glove approach for Malaysian sign language detection. Procedia Computer Science, 76, 60–67. https://doi.org/10.1016/j.procs.2015.12.276
32. Sood, K., Hernández, A., & Navdiya, B. (2023). American sign language interpreter: A bridge between the two worlds. In 2023 International Conference on Artificial Intelligence and Smart Communication (AISC). IEEE. https://doi.org/10.1109/AISC56616.2023.10085066
33. Vishwanath, S., & Yawer, S. S. (2019). Sign language interpreter using computer vision and LeNet-5 convolutional neural network architecture. International Journal of Innovative Science and Research Technology.