Solution Design

Functional Requirements:

ID	Name	Description	Source	Priority
F01	Real-time Translation	System shall allow user to use application features in real-time via live video.	Trivial	MUST
F02	Hand/Gesture Identification	System shall identify the hand/gesture of the user.	Domain Analysis	MUST
F03	Hand Sign Helper	System shall provide a hand sign helper image to the user on the main window.	Stakeholder Analysis	MUST
F04	Gesture/Hand Feedback	System shall output message to the user when no hand is visible.	Developer Analysis	MUST
F05	Sign Recognition for Letters	System shall recognize sign language gestures for letters performed by the user.	Domain Analysis	MUST
F06	On-Screen Overlay Translation	System shall provide an on-screen overlay translation of the gestures to the user.	Stakeholder Analysis	MUST
F07	Interface Window	System shall display a user interface window when the application is executed.	Trivial	MUST
F08	Start Button	System shall display a start button on the startup page to initiate the translator.	Trivial	MUST
F09	Exit Button	System shall display an exit button on the startup page to exit the application.	Trivial	MUST
F10	Combo-Box Button	System shall provide a choice between alphabets and numbers for translation on the main interface screen.	Trivial	MUST

Non-Functional Requirements:

ID	Name	Description	Source	Priority
N01	Security	System must be secure and protect user data and prevent unauthorized access to systems.	Stakeholder Analysis	MUST
N02	Performance	System must respond quickly to user input and recognize the signs quickly and accurately in under 1 second.	Competitor Analysis	MUST
N03	Usability	System must provide a simple and intuitive interface that is user-friendly.	Domain Analysis	MUST
N04	Adaptability	System must be able to function in different lighting levels.	Developer Analysis	MUST
N05	Compatibility	System must be compatible with most Windows and Mac computers.	Stakeholder Analysis	MUST
N06	Security	System must adhere to user privacy laws and policies.	Stakeholder Analysis	MUST
N07	Responsiveness	System UI must be fluid and responsive.	Developer Analysis	SHOULD
N08	Error Handling	System must handle errors and provide clear error message to the user.	Developer Analysis	SHOULD
N09	Extensibility	System should be designed to accommodate additional features and new sign language data in the future.	Domain Analysis	SHOULD

N10	Modern User Interface	System shall have a modern user-friendly interface with appropriate elements for best user experience.	Stakeholder Analysis	SHOULD

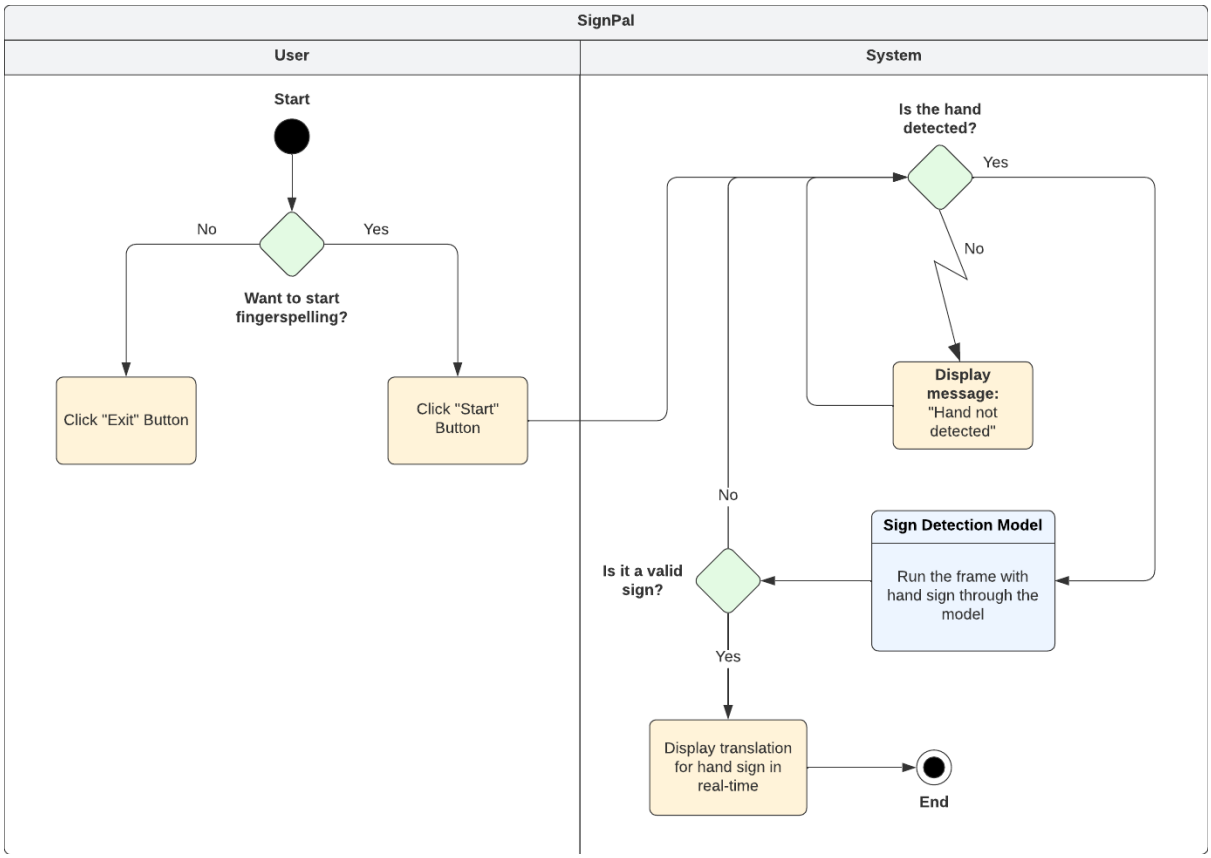
Hardware Interface Requirements:

ID	Name	Description
H01	CPU	**Minimum** - Intel Core i3 | AMD Ryzen 3 | Apple M1
H02	RAM	4GB RAM or more
H03	HDD/SSD	**Minimum** - 500MB for video analysis
H04	GPU	**Recommended** - Intel HD Graphics 520 or more | NVIDIA GeForce GT 635 | Radeon HD 8470D

Software Interface Requirements:

ID	Name	Description
S01	Operating System (OS)	**Minimum** - Windows 7 | MacOS Yosemite
S02	Communication app	Visual Studio Code | PyCharm
S03	Programming Language	**Required** - Python 8.0
S04	Libraries Used	numpy, pandas, keras, customtkinter, cvzone, cv2, Pillow (PIL), math, time, pytest

Solution Diagram:



Solution Description:

Introduction:

SignPal is a python application that allows users to make sign language gestures and receive real-time feedback. The system has a modern user-friendly interface and supports a variety of features such as Hand/Gesture Identification, Hand Sign Helper, Sign Language Recognition, and On-Screen Translation. The system complies with user privacy laws and policies and is designed to accommodate new features and sign language data in the future.

Solution Components:

Real-time Translation: This utilizes a video capture device, such as a webcam, that will record the user's movements and feed them into the system. This will be done using the *OpenCV* (PyPI, 2019) package.

Hand/Gesture Identification: This can be achieved by incorporating a computer vision package known as *CVZone* (CVZone, 2023) that will analyze the video frames and identify the position and orientation of the user's hand as well as the hand infrastructure of the user.

Hand Sign Helper: When the application is executed, a sign instructions image is displayed to the user on the main window in order to position and perform the sign language gestures correctly. This will be done using the *Pillow (PIL)* (Clark, n.d.) package.

Gesture/Hand Feedback: This will be done using exception handling that will detect when the user's hand is not visible and display an appropriate error message.

Sign Language Recognition (Letters): This will be done by passing the frames of the live video into a *Trained Model* that recognizes the sign language gestures used.

On-Screen Overlay: The *Trained Model* will output a translation when hand gestures frames are passed through it. The translation will be displayed as On-Screen Overlay above the user's hand.

Interface Window: An interface window that will appear upon running the application. *customtkinter* (Schimansky, 2022) module will be used to create a modern looking interface. It includes essential elements such as buttons, combo-box, and an image. These elements are designed to elevate user experience and provide optimal functionality for the application.

Solution Process:

Upon running SignPal, a user-friendly interface window appears containing various interactive elements. These include a "**Hand Sign Helper Image**", *buttons* for "**Start**" and "**Exit**", and a **combo-box** *button* to choose between **number** translation and **alphabet** translation. The user can exit SignPal by clicking the "**Exit**" button. Upon clicking the "**Start**" button, the system initializes the sign language translation process. This launches another window and the system's video capturing device is used to display the user's video in real-time. Finally, the user can begin executing sign gestures.

The system captures and processes the user's sign gestures in real-time while simultaneously displaying their corresponding translations. The translations will appear as an **On-Screen Overlay** which displays the translated word above the location of the user's hand in the live video feed.

Overall, SignPal provides a seamless and user-friendly solution for real-time sign language translation.

Real-World Example:

SignPal can be used in kindergarten to teach young deaf children how to fingerspell. Teachers could also use it to teach young children how each alphabet and number can be gestured in sign language, ultimately creating a strong foundation for them to communicate effectively. This could potentially sprout interest of sign language in other students, and they could also learn it alongside them to communicate with their friends.

SignPal has the potential to improve communication and facilitate social interactions for deaf individuals, even in situations where others do not know sign language.

Data Description:

Introduction:

To ensure that SignPal recognizes and translates the hand signs with utmost accuracy, the model used for it would need to be trained, tested, and validated using a dataset of images with their corresponding labels.

Dataset Description:

The dataset should contain images of a hand performing different sign language gestures along with its respective translation (labels) in English. Each hand sign should have multiple versions of it such as slightly twisted horizontally and vertically in both directions, zoomed in and zoomed out, and horizontally flipped since both hands can be used for fingerspelling (Price, 2009). This is to capture the variability that may be encountered in real-world scenarios. To ensure the dataset's quality, it should be balanced and contain an equal number of samples for each gesture and each variation, to prevent bias towards certain signs or versions. Hence, 50 images with central-axis rotation, 50 with vertical rotation, and 50 with horizontal rotation will be used for each hand, totalling up to 300 images per alphabet/number.

Splitting the Dataset:

The dataset will be split into **training** set, **validation** set, and **test** set. Initially, the ratio for it will be set to **80**:**10**:**10** respectively since it is a good practice to initialize with (Baheti, 2023). With further progression, this ratio can be modified for better optimization of the model.

- The **training** set will be used to train the sign language model by inputting the hand sign images and its corresponding translated labels using Google's Teachable Machine.
- The **validation** set will be used to fine-tune and optimize the model's hyperparameters.
- The **test** set will be used to evaluate the model's performance and accuracy after it has been fully trained and optimized. This set will **NOT** be used during the training or validation process to prevent bias in evaluation.

Solution Motivation:

The reasoning behind the proposed solution is as follows:

- The use of **OpenCV** (PyPI, 2019) module to capture video for real-time translation is a crucial feature and is the fastest and most effective method of getting input from the user. Moreover, providing translation instantaneously can be beneficial for the users in real-world situations, and can be a streamlined feature of the application.
- The **CVZone** (CVZone, 2023) computer vision package can be used for efficient hand/gesture recognition, as it provides swift hand detection and feedback. This eliminates the need to develop hand detection algorithms from scratch, thereby conserving time and resources that can be directed towards other mission-critical activities.
- Providing gesture feedback can improve user experience and ensure that the system is accessible to a broader range of users.
- Displaying a hand sign helper image using the **Pillow (PIL)** (Clark, n.d.) module is an essential feature to ensure that the system can accurately interpret the hand gestures. It also acts as notes that the user can refer to and fingerspell without errors. This could also contribute to user's learning.
- The system's ability to output translations and display them as On-Screen Overlay provides ease of use.
- The user of **customtkinter** (Schimansky, 2022) module for the user interface window improves user experience while encouraging user interaction, user productivity, and assuring value for time.
- The **pytest** module is used to test whether the code is error free and functional.

References

Price, M. (2009) 'The left brain knows what the right hand is doing', Monitor on Psychology, 40(1). Retrieved from https://www.apa.org/monitor/2009/01/brain.

Baheti, P. (2023) 'Train Test Validation Split: How To & Best Practices [2023]', *V7Labs*, 2 March. Available at: https://www.v7labs.com/blog/train-validation-test-set (Accessed 10 April 2023).

PyPI. (2019). opencv-python. [online] Available at: https://pypi.org/project/opencv-python/.

CVZone (2023). CVZone. [online] GitHub. Available at: https://github.com/cvzone/cvzone.

Clark, A. (n.d.). Pillow: Python Imaging Library (Fork). [online] PyPI. Available at: https://pypi.org/project/Pillow/.

Schimansky, T. (2022). CustomTkinter UI-Library. [online] GitHub. Available at: https://github.com/TomSchimansky/CustomTkinter.