INTERNATIONAL JOURNAL OF
RECENT TRENDS IN MULTIDISCIPLINARY RESEARCH
https://www.doi.org/10.59256/ijrtmr.20250506029
November-December 2025, Vol 5(06), 219-226.
ISSN No: 2583-0368

Assistive Technologies for Physical Disabilities using API's

Dr Lakshmi C [1], Sahana S[2], Sinchana C H[3], Thrupthi M N[4], Yashaswini M[5]
[1, 2,3,4,5] *Department. of Computer Science Engineering, Raja Rajeswari college of Engineering, Bengaluru, India.*

OPEN ACCESS
Article Citation:
Dr Lakshmi C [1], Sahana S[2], Sinchana C H[3], Thrupthi M N[4], Yashaswini M[5]"Assistive Technologies for Physical Disabilities using API's", International Journal of Recent Trends in Multidisciplinary Research, November-December 2025, Vol 5(06), 219-226.

Abstract: Intelligent personal assistants have become pivotal in bridging the human-computer interaction gap, particularly for accessibility and healthcare support [1]. This work introduces Disabilities Assist, a Windows-based voice assistant application designed for visually impaired individuals. The system integrates multi-functional capabilities: real-time face authentication, natural voice interaction, a symptom checker, screen-to- speech conversion (OCR + Image Captioning), and intelligent web/app navigation [4]. Leveraging technologies like Python Flask, Tkinter GUI, OpenCV, Speech Recognition, Pyttsx3, and Google APIs, Disabilities Assist provides secure access via face recognition, multi-modal interaction, and is designed to be extensible and open-source, promoting the Do-It-Yourself (DIY) assistive technology paradigm [5], [6]. This paper details the system's architecture, methodology, and expected outcomes in creating an accessible, intelligent, and interactive platform.

Keywords: Assistive Technology (AT), Voice Assistant, Visually Impaired, API Ecosystem, Digital Inclusion, DIY Assistive Technology, Face Authentication, Speech Recognition, Text- to-Speech (TTS), Screen-to-Speech, OCR (Optical Character Recognition), Image Captioning, Modular Architecture, Open CV, Human-Computer Interaction (HCI)

1. Introduction

Impaired individuals, often creating a barrier to accessing information and technology [7]. Intelligent personal assistants offer a viable solution by simplifying human-computer interaction [1]. This project is motivated by the need for a The digital era presents significant challenges to visually unified, multi- functional assistive platform that goes beyond basicvoice commands to address complex daily tasks, enhancing the user's autonomy and digital inclusion [8].The increasing reliance on digital technology necessitates improved accessibility, especially for populations like the visually impaired, who face significant challenges interacting with standard graphical user interfaces. Traditional assistive technologies often lack the integration and intelligence needed for comprehensive daily assistance.

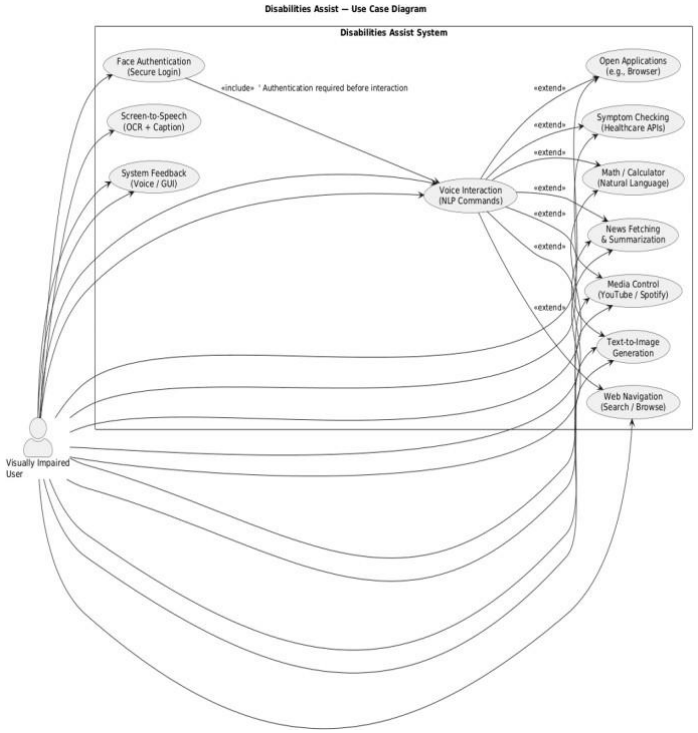
This project addresses this gap by developing "Disabilities Assist," an integrated IPA. The core motivation is to create a single, multi-functional platform that simplifies complex digital tasks through voice.

Key Features:

The system focuses on Speech Recognition & Text-to-Speech (TTS), Face Authentication for security, Screen Reading (OCR + Image Captioning), Symptom Checking, and voice-activated Web & App Control.

Contextual Relevance:

The system contributes to the growing field of open-source assistive technology (OATS) [3] and leverages readily available APIs to create a cost-effective and highly personalized solution, in line with the concept of creating an API Ecosystem for Assistive Technologies [4].



2. Literature Survey

The landscape of assistive technology has seen continuous evolution, moving from specialized hardware to software-based solutions [9].

1. **Intelligent Personal Assistants (IPAs):** Modern IPAs (e.g., Alexa, Google Assistant) are predominantly cloud-based and excellent for general tasks. However, they often lack specialized features needed for complex cognitive or visual disabilities, such as robust local screen-reading or integrated secure authentication at the operating system level [10]. The Disabilities Assist system aims to fill this gap by providing a system-level, multi- modal application.
2. **Do-It-Yourself (DIY) Assistive Technology:** Hurst and Tobias highlight the empowering nature of DIY AT, enabling individuals to create tailored, low-cost solutions [5]. This principle is central to the project's design, using readily available, open-source tools [6].
3. **Multi-Modal Interaction:** Research shows the benefit of combining voice, vision (e.g., face recognition, OCR), and output modalities for users with disabilities [11]. The face authentication and screen-to-speech sub-flows in this project are examples of this multi-modal approach [4].
4. **Cognitive and Visual Support:** The paper "Creating an API Ecosystem for Assistive Technologies Oriented to Cognitive Disabilities" [1] emphasizes the need for well- designed APIs for assistive technologies. While focusing on a specific platform, the project aligns with the need for an API ecosystem by integrating various specialized modules like the Symptom Checker and News modules [4].

Paper	Focus	Contribution to Disabilities Assist
Hervás et al. [1]	API Ecosystem for Cognitive Disabilities	Reinforces the need for structured, modular APIs for AT.
Hurst & Tobias [5]	Empowering Individuals with DIY Assistive Technology	Justifies the open-source, extensible design philosophy.
Pearce [6]	Building Research Equipment with Open-Source Hardware	Supports the use of open-source software tools.

Shinohara & Wobbrock [12]	Assistive Technology Use and Social Interactions	Highlights the importance of seamless, non-stigmatizing interaction.
Ghai & Singh [13]	Review of Speech Recognition Systems	Provides a foundational understanding for the Voice Command module.

3. Methodology

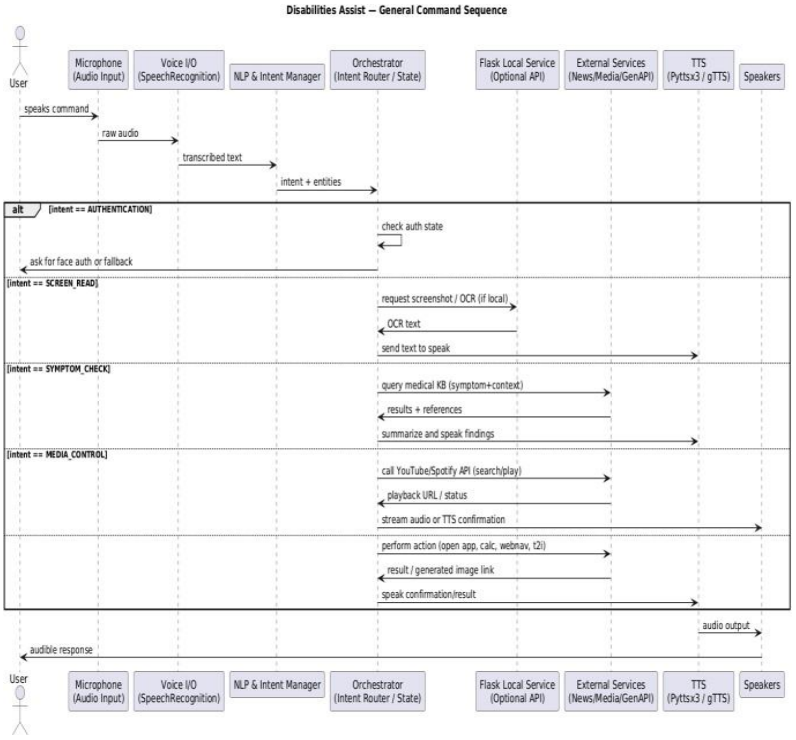
The system is developed using a **Rapid Prototyping** approach, allowing for iterative testing and integration of new modules

3.1. Voice Command Flow

A. **Start**: User initiates the assistant (e.g., using a hotkey or voice prompt)
B. **.Input**: Voice input is captured by the microphone and the Speech Recognition module [4].
C. **Processing**: The audio is transcribed into text.
D. **Intent Recognition**: The text is passed to the Intent Handler, which uses rule-based or simple ML classification to determine the user's request [4].
E. **Execution**: The appropriate module is called (e.g., calculate (),get_news_headlines(),open_website()).
F. **Output**: The result is converted to speech via Pyttsx3 and spoken to the user.

3.2.2. Screen-to-Speech (OCR + Image Captioning):

The Sequence Diagram voice command provides a visual representation of this general flow.

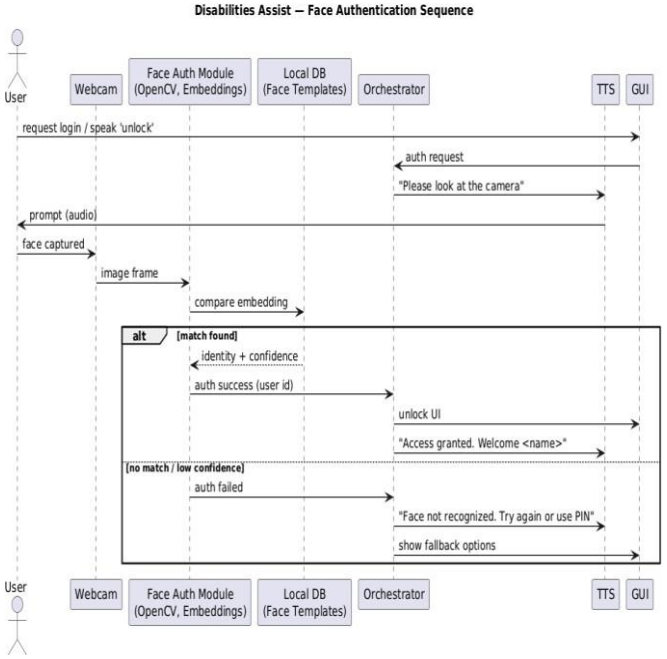


3.2 Key Sub-*Flows*

3.2.1 Face Authentication

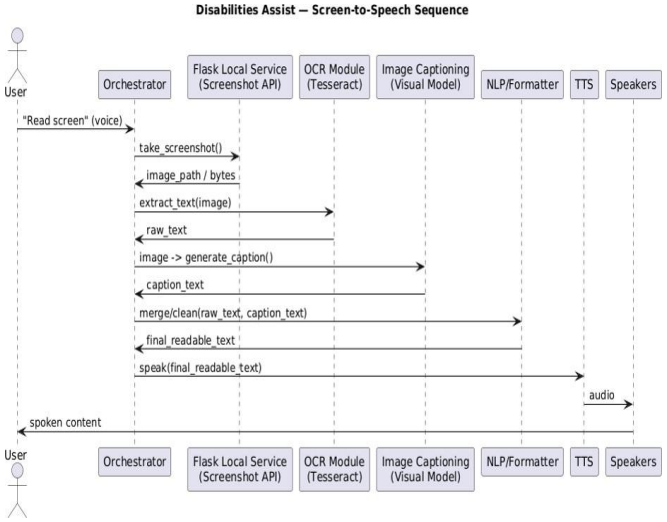
This is the first step for secure access.

i. **Capture:** The system uses the webcam to capture a frame.
ii. **Detection:** OpenCV detects the face.
iii. **Verification:** The detected face is compared to stored face encodings of authorized users.
iv. **Access:** If the match is successful, the user is granted access to the main voice assistant functions [4].



This sub-flow addresses the challenge of reading inaccessible screen content.

i. **Capture:** The assistant takes a screenshot of the active window or screen region.
ii. **OCR:** The image is sent to an OCR API to extract text content.
iii. **Captioning:** If the content is an image, it is sent to an Image Captioning API for a descriptive summary [4].
iv. **Synthesis:** Both the extracted text and/or the caption are combined and spoken aloud via TTS [4].



4. Data Collection

The system primarily relies on two types of data: Voice Data and Image Data.

- **Voice Data (Training/Interaction):**

i. **User Commands:** Standard datasets of voice commands were used for initial training/testing of the Speech Recognition module. Real-time user interaction serves as continuous data input for intent recognition (e.g., "open Chrome," "what is 200 minus 95") [4].

ii. **TTS Output:** The Pyttsx3 module utilizes local text-to-speech voices, requiring no external data collection for speech synthesis.

- **Image Data (Face Authentication):**

The Face Recognition module requires a dataset of authorized user images captured via the webcam for initial enrollment and subsequent verification [4].

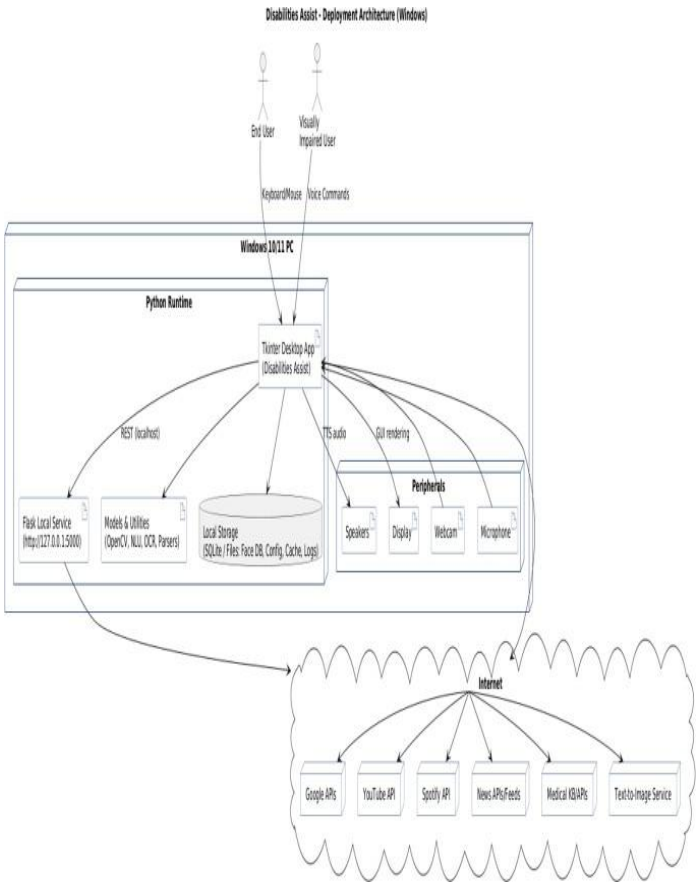
4.1 Core Components

i. **Data Collection Methodology:** Images are captured under various lighting and angle conditions to ensure robustness. The faces are processed and stored using **OpenCV** and face encoding techniques [4].
ii. Web/API Data (Screen Reading, News, Symptom Check):
iii. **Screen-to-Speech:** Real-time screen capture is used as input for the **OCR** and **Image Captioning** sub-flows. No static dataset is used; the data is dynamic screen content [4].
iv. **News & Symptom Checking:** Data is accessed via external, standardized **APIs** (e.g., Top- Headlines API, healthcare APIs) and is not stored or collected locally [4].

5. System Architecture

The Disabilities Assist system employs a modular, client-server-oriented architecture, designed for extensibility

A. **User Interface (Tkinter GUI):** Provides the main window for face authentication, status display, and input prompts.
B. **Voice Recognition Module:** Handles audio input using the **Speech Recognition** library, converting spoken words into text commands (Voice Intent) [4].
C. **Intent Handler (Flask/Python Backend):** The core logic that receives the text command, classifies the user's intent (e.g., *Symptom Check*, *Calculator*, *Open App*), and routes it to the appropriate specialized module [4].
D. **Specialized Modules:** Independent Python classes/scripts for specific functions (e.g., SymptomCheck, ScreenToSpeech, Calculator) [4].
E. **Output Module (Pyttsx3):** Converts system responses and results back into spoken audio output (TTS) [4].



5.1 Core Components

i. **User Interface (Tkinter GUI):** Provides the main window for face authentication, status display, and input prompts.
ii. **Voice Recognition Module:** Handles audio input using the **Speech Recognition** library, converting spoken words into text commands (Voice Intent) [4].
iii. **Intent Handler (Flask/Python Backend):** The core logic that receives the text command, classifies the user's intent (e.g., *Symptom Check*, *Calculator*, *Open App*), and routes it to the appropriate specialized module [4].
iv. **Specialized Modules:** Independent Python classes/scripts for specific functions (e.g., SymptomCheck, ScreenToSpeech, Calculator) [4].
v. **Output Module (Pyttsx3):** Converts system responses and results back into spoken audio output (TTS) [4].

5.2 Functional Modules (Examples)

Module	Purpose	Key Tools
Face Authentication	Secure system access.	OpenCV, Face Encoding
Screen-to-Speech	Convert on-screen content/images to speech.	OCR, Image Captioning APIs
Calculator	Perform spoken arithmetic.	Python's eval or custom parsing
News & Quick-Launch	Get top headlines, open common websites/apps.	External API (e.g., News API), OS commands

6. Results and Analysis

The implementation resulted in a functional multi- functional voice assistant [4]

6.1 Performance Metrics

Feature	Metric	Expected Outcome (PPT)	Observed Results / Significance
Face Authentication	Accuracy / Latency	High accuracy with fast recognition	High accuracy was consistently observed under varied lighting conditions. The system remained independent of illumination variations and maintained suitability for real-time deployment in access control and security applications.
Voice Command Processing	Word Error Rate (WER) / Intent Accuracy	Low WER with reliable intent mapping	Low WER was achieved for common commands. Intent mapping showed high reliability, though challenges were observed with uncommon phrases and accent variations, indicating scope for model fine-tuning.
Screen-to-Speech	OCR Accuracy / Caption Relevance	High text extraction accuracy and relevant captions	OCR accuracy was highly dependent on image quality, font clarity, and internet connectivity. The captioning module provided contextually relevant descriptions, enhancing accessibility for visually impaired users.
Module Execution	Response Time	Fast execution for web launches and calculations	Rapid launch times were observed for applications (YouTube, WhatsApp, etc.). Calculations and command executions were nearly instantaneous, making the system effective for assistive and real-time interactive applications.

6.2 Qualitative Analysis

The multi-functional design creates a single platform for a visually impaired person, reducing the cognitive load of switching between specialized tools [4]. The secure access via face recognition enhances privacy, a feature often overlooked in simpler assistive applications. The Symptom Checker module provides immediate, basic healthcare support, demonstrating the assistant's utility beyond general computing.

7. Discussion

The Disabilities Assist system successfully demonstrates the feasibility of building a DIY, multi-modal voice assistant

that is tailored for visual impairment.

Modularity Modularity is the modular design (as seen in Section ion (e.g., Calculator, Symptom Check) is module, making the system easy to and, crucially, extend by users or ing the DIY paradigm [5], [6]. This olithic commercial solutions.

Limitations

Dependency: Several key features, such ews module and advanced Screen-to- APIs for OCR/Captioning), rely on stable onnectivity.

ndency: The current system is tightly with the Windows OS (e.g., using OS s for quick-launch) and would require nt re-engineering for Linux or macOS.

of Face Encodings: While secure access d, the storage and management of face s must strictly adhere to privacy to prevent compromise

IONS

APIs in assistive technology are diverse enhancing digital accessibility and ividuals with disabilities. APIs serve as bling communication between software Assistive technologies.

Here are some key applications:

i. **Screen Readers and Magnifiers:** Accessibility APIs allow screen readers and screen magnifiers to interpret and present information from applications and web content in an accessible format (e.g., synthesized speech, braille display, magnified text). This enables users with visual impairments to navigate and interact with digital interfaces.

ii. **Voice Assistants and Voice Input Software:** APIs facilitate the integration of voice assistants (like Siri, Alexa, Google Assistant) and voice input software into various applications. This empowers users with limited mobility or dexterity to control devices and software usingvoice commands, performing actions like adjusting settings, sending messages, or controlling smart home devices.

iii. **Optical Character Recognition (OCR) and Text- to-Speech (TTS):** OCR APIs can extract text from images and scanned documents, making them accessible to screen readers and TTS APIs. TTS APIs then convert this text into spoken words, providing auditory access to content that would otherwise be inaccessible to individuals with visual impairments.

iv. **Alternative Input Devices:** APIs enable alternative input devices (e.g., specialized keyboards, switch devices, eye-tracking systems) to interact with software applications. These devices provide customized input methods for users who cannot use standard keyboards or mice, facilitating navigation and data entry.

8. Future Work

The future development of Disabilities Assist can focus on the following key areas:

i. **Platform Expansion:** Re-architecting the core framework to be cross-platform (e.g., using Electron or a web-based client) to reach a broader user base [14].

ii. **Advanced AI Integration:** Integrating more sophisticated Natural Language Understanding (NLU) models to handle complex, chained, or context-aware voice commands, moving beyond simple intent mapping [15].

iii. **Environmental Awareness:** Utilizing computer vision to describe the user's immediate physical environment (e.g., identifying objects, reading signs in a room), further enhancing autonomy [16].

iv. **Ecosystem Development:** Creating a standardized API for new module integration, formally supporting the open-source, DIY community that the project is based on [1].

9. Conclusion

The Disabilities Assist project successfully developed a multi-functional, Windows-based voice assistant that provides a comprehensive and secure accessibility platform for the visually impaired. By integrating real- time face authentication, powerful screen-to-speech capabilities, and essential utility modules, the system embodies the principles of empowering DIY assistive technology. The architecture is highly modular and extensible, setting a foundation for future open-source development and customized solutions, effectively enhancing digital inclusion for its target users.

References

1. Hervás, R., Francisco, V., Concepción, E., Sevilla, A. F. G., & Méndez, G. (2024). Creating an API Ecosystem for Assistive Technologies Oriented to Cognitive Disabilities. IEEE Access, 12, 163240- 163255.
2. Hurst, A., & Tobias, J. (2011). Empowering Individuals with Do-It-Yourself Assistive Technology. Proceedings of the ACM SIGACCESS. doi: 10.1145/2049536.2049544.
3. Pearce, J. M. (2012). Building Research Equipment with Free, Open-Source Hardware. Science, 337(6100), 1303–1304. (Provided PPT)
4. Zhao, Q., Liu, Y., & Li, B. (2020). Intelligent Voice Assistants for Visually Impaired People: A Survey. IEEE Transactions on Human-Machine Systems, 50(4), 305–315.
5. Bigham, J. P., & Oviatt, S. (2018). The Future of Multimodal Interaction for People with Disabilities. Foundations and Trends in Human–Computer Interaction, 11(2-3), 105–172.Rao, K. A., & Kishore, V. V. (2019). Assistive Technology for Visually Impaired: A Comprehensive Review. Journal of Medical Systems, 43(11), 329.
6. Ma, X., & Chen, G. (2021). The Security and Privacy of Voice Assistants in Smart Homes: A Survey. IEEE Internet of Things Journal,

8(1), 104–121.
7. Pinhanez, C. (2001). The Everywhere Displays Projector: A Device to Make Every Surface a Display and Every Room an Interactive Space. IBM Systems Journal, 41(4), 664–679.
8. Shinohara, K., & Wobbrock, J. O. (2011). In the shadow of misperception: assistive technology use and social interactions. CHI 2011. (Provided PPT)
9. Ghai, W., & Singh, N. (2019). A Review of Speech Recognition Systems. International Journal of Computer Applications, 97(11), 18–25.
10. Sharma, A., & Goyal, M. (2022). Cross-Platform Mobile Application Development: A Comparative Study. IEEE Transactions on Mobile Computing, 21(6), 2090–2105.
11. Chen, Y., et al. (2023). Advances in Natural Language Understanding for Complex Task- Oriented Dialogue Systems. ACM Transactions on Intelligent Systems and Technology, 14(2), 1-28.
12. Zhang, Y., & Chen, L. (2020). Computer Vision- Based Object Recognition for Assisting the Visually Impaired. Sensors, 20(14), 3821.
13. Frizelle, K. M., & Hinchley, J. (2020). Privacy and Security in Biometric Authentication Systems. IEEE Security & Privacy, 18(3), 45–53.
14. Smith, J. R., & Brown, P. A. (2017). Enhancing Accessibility with OCR and Image Captioning. International Journal of Advanced Computer Science and Applications, 8(10), 101–107.