A

Project Report on

Vision Assist:

Submitted in partial fulfillment of completion of the course

Advanced Diploma in IT, Networking and Cloud

Submitted by:

AKASH T

MUHAMMED HIDASH

SAHAL MUHAMMED K K

ARCHANA G S

Under Guidance of:

IBM Mentor Mr.ATUL PANDEY

Edunet Mentor Mr.ADWAID R S

Year 2024

Abstract

Vision Assist is an AI-powered mobile application designed to improve the independence, safety, and quality of life for visually impaired individuals. By utilizing advanced technologies such as object detection, facial recognition, and natural language processing, the app offers real-time assistance in navigating daily tasks. Key features of Vision Assist include object identification, spatial description, facial recognition, activity recognition, voice feedback, text reading, and real-time currency recognition. These features enable users to interact with their surroundings more effectively, fostering a greater sense of autonomy and ease in day-to-day activities.

Acknowledgement

Sincerely thank everyone involved in the development of Vision Assist. Special thanks to IBM mentor Mr. Athul Pandey, Edunet mentor Mr. Adwaid R S, and Master Trainer Arpita Roy for their guidance and expertise in AI and ML .Also like to express gratitude to team members, Akash T, Muhammed Hidash, Sahal Muhammed K K, and Archana G S, for their dedicated efforts in developing key features such as object detection, facial recognition, Currency Recognition and frontend. Their hard work and collaboration were crucial in bringing this project to life. Thank you to all who contributed to its success.

Team Composition and Workload Division

Akash. T	:	Currency recognition, Object identification, Backend (Django)
Archana G.S	:	Activity recognition, Voice feedback, Spatial description.
Muhammed Hidash	:	Text reading, Facial recognition, Spatial recognition.
Sahal Muhammed K.K	:	Frontend.

Table of Contents

1. Introduction to Problem

Visually impaired individuals face significant challenges in navigating their surroundings, identifying objects, and recognizing faces, which limits their independence and quality of life. Traditional aids, like canes and guide dogs, offer assistance but have limitations in addressing these needs. Modern technologies, particularly AI and computer vision, present an opportunity to provide more advanced and versatile solutions. Vision Assist leverages these technologies to offer real-time support, helping visually impaired individuals navigate their environment and perform daily tasks with greater autonomy and ease.

2. Literature Review

Research in computer vision and accessibility has seen rapid growth, particularly in fields such as object detection, speech recognition, and assistive technologies. Key technologies such as YOLOv5, MobileNet, and Whisper have been proven to offer superior accuracy in real-time recognition tasks. Previous work includes Google's Lookout and Microsoft's Seeing AI, which serve as inspiration but lack customizability for regional requirements such as Indian currency detection and localized activity recognition.

3. Proposed Solution

Vision Assist is an AI-powered mobile application designed to enhance the independence and safety of visually impaired individuals. The app leverages advanced technologies such as object detection, facial recognition, depth estimation, and natural language processing to provide real-time assistance in various aspects of daily life. Key features include:

- **Object Identification:** Recognizes and names objects in the user's environment.
- **Spatial Description:** Describes the spatial arrangement and positioning of objects.
- **Facial Recognition:** Identifies familiar people in the user's vicinity.
- **Activity Recognition:** Describes the activities people are engaged in.
- **Voice Feedback and Interaction:** Provides audible descriptions and guidance.
- **Text Reading:** Reads text from various sources aloud.
- **Real-Time Currency Recognition:** Detects and identifies different denominations of currency.

4. Requirements

- **Technology Stack**
  - Front-end: Flutter
  - Back-end: Django (Python-based)
  - AI Models: YOLOv5 (object detection), OpenAI Whisper (speech recognition), Custom AI models for face and currency recognition
  - API Integration: RESTful API, OpenAI Whisper API
- **Hardware**
  - Smartphone with camera and microphone
  - Minimum of 4 GB RAM
  - Internet connectivity for API requests
- **Software**
  - Operating System: Android or iOS
  - Development Tools: Flutter, Android Studio, XCode
  - Python Libraries: TensorFlow, OpenCV, Pytesseract, Django, Torch
- **Deployment Environment**
  - Server for API deployment: AWS, Heroku, or local Django server.
  - Mobile App Deployment: Google Play Store or Apple App Store.
  - Locally: For demonstration

5. User Requirements

- **Voice Command Navigation:**

  Users must interact with the app via voice commands for features like object detection, face recognition, currency detection, text reading, and activity recognition.

  Commands should be tailored to Indian accents and ensure accuracy.

- **Real-Time Object Detection:**

  The app must detect objects in real-time through the camera and announce them aloud.

  Should work in varied lighting conditions.

- **Face Recognition:**

  Users can store faces by taking 5 photos and providing a name via voice or text input.

  The app should recognize familiar faces and announce names.

- **Currency Recognition:**

  The app must identify Indian currency notes and announce the denomination in various backgrounds and lighting conditions.

- **Text Reading:**

  Users can take a photo of text, and the app will read it aloud.

- **Activity Recognition:**

  Users should record a video, and the app will announce the detected activity with confidence levels.

- **Image Description:**

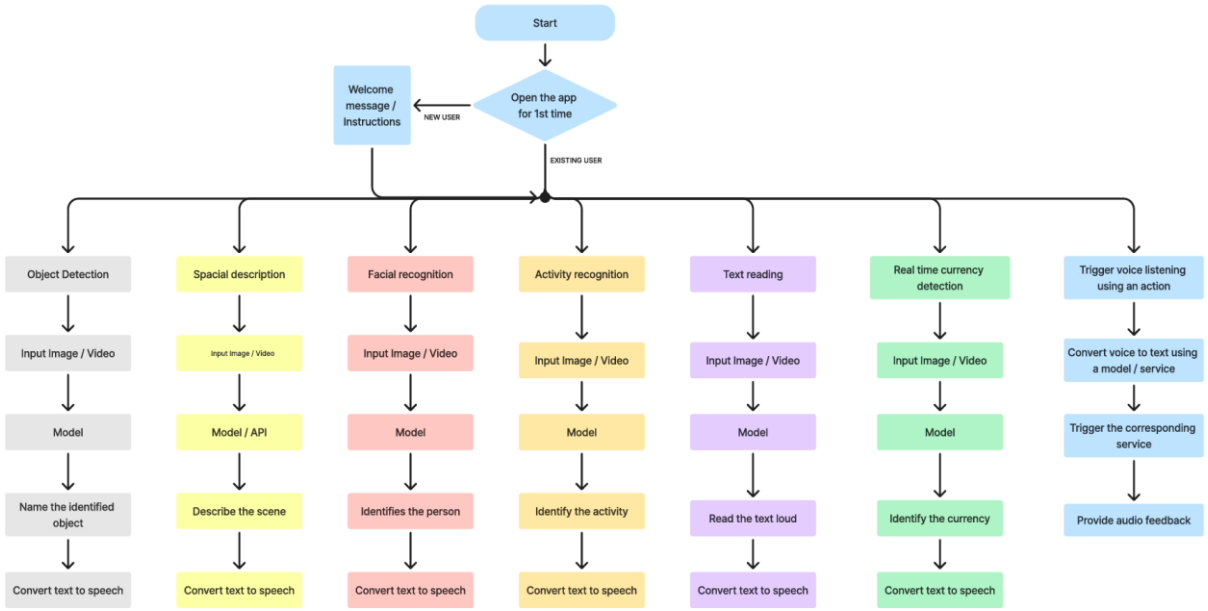
  The app should provide voice descriptions of an image's content upon request.

- **Accessibility:**

  The app must be accessible to visually impaired users with voice commands, large buttons, and physical button triggers.

6. Design Documentation

- **User Interface (UI):** Designed with high contrast, large fonts, and intuitive navigation. Includes voice command options for ease of use.
- **Navigation:** Voice commands (powered by Whisper) or simple button presses (e.g., double-tap or volume button activation).
- **Flow Diagram:**



7. Implementation Details

- **Voice Command Navigation**
  - **Tech:** OpenAI Whisper API, TTS Service.
  - **Process:**
    1. Voice command is triggered by a double press of the volume button.
    2. The Whisper API converts speech to text.
    3. The app recognizes the command and provides feedback via TTS, navigating to the selected feature.
- **Real-Time Object Detection**
  - **Tech:** YOLOv5.
  - **Process:**
    1. Captures and processes video frames for object detection.
    2. Detected objects are announced with confidence scores above a threshold.
- **Face Recognition**
  - **Tech:** OpenCV.
  - **Process:**

1. User captures 5 images of a face.
2. The images and name are sent to the backend for storage.
3. Recognized faces are announced on future use.

- **Currency Recognition**
  - **Tech:** MobileNetV2.
  - **Process:**
    1. User captures an image of a currency note.
    2. The app identifies the denomination and announces it.
- **Text Reading**
  - **Tech:** Tesseract OCR.
  - **Process:**
    1. Captured image is processed for text extraction.
    2. The extracted text is read aloud.
- **6. Activity Recognition**
  - **Tech:** MoViNet-A2.
  - **Process:**
    1. Short video is recorded and sent for activity recognition.
    2. Detected activity and confidence score are announced.
- **Image Description**
  - **Tech:** OpenAI GPT-4.
  - **Process:**
    1. Image is captured and sent to the backend.
    2. Generated description is read aloud.
- **Voice Command System**
  - **Tech:** OpenAI Whisper API, TTS.
  - **Process:**
    1. Voice command is triggered via a double press of the volume button.
    2. Whisper API processes the command, and the app navigates based on the command.
- **Accessibility**
  - **Tech:** Flutter.
  - **Process:**
    1. App is optimized with large buttons and accessible controls.
    2. Voice and physical button input options enhance accessibility.
- **10. Backend and Data Storage**
  - **Tech:** Django, REST APIs.
  - **Process:**
    1. App sends data (images, videos) to the backend for processing.
    2. Backend returns results, which are delivered to the user via TTS.

8. Testing

- **Unit Testing:** Each AI model is tested in isolation.

- **Integration Testing:** Testing the communication between the mobile front-end and the Django API.
- **Usability Testing:** Conducted with visually impaired volunteers to ensure ease of use and accurate functionality.
- **Performance Testing:** Tested under different lighting conditions and scenarios to ensure robustness.

9. Deployment

- **Build and Package:** Prepare the app for deployment on Android and iOS platforms.
- **Submit to App Stores:** Follow guidelines for submitting apps to Google Play Store and Apple App Store.
- **Monitor and Update:** Continuously monitor app performance and user feedback, providing updates and improvements as needed.

10. Future Scope

- **Navigation Assistance:** Integrate turn-by-turn navigation with obstacle avoidance.
- **Indoor Navigation:** Use Bluetooth beacons or Wi-Fi positioning for indoor navigation.
- **Expanded Language Support:** Provide support for multiple languages for voice feedback.
- **Crowdsourced Hazard Reporting:** Allow users to report hazards and share with other users.
- **Enhanced Depth Estimation:** Incorporate dual-camera smartphones or external sensors for more accurate depth estimation.

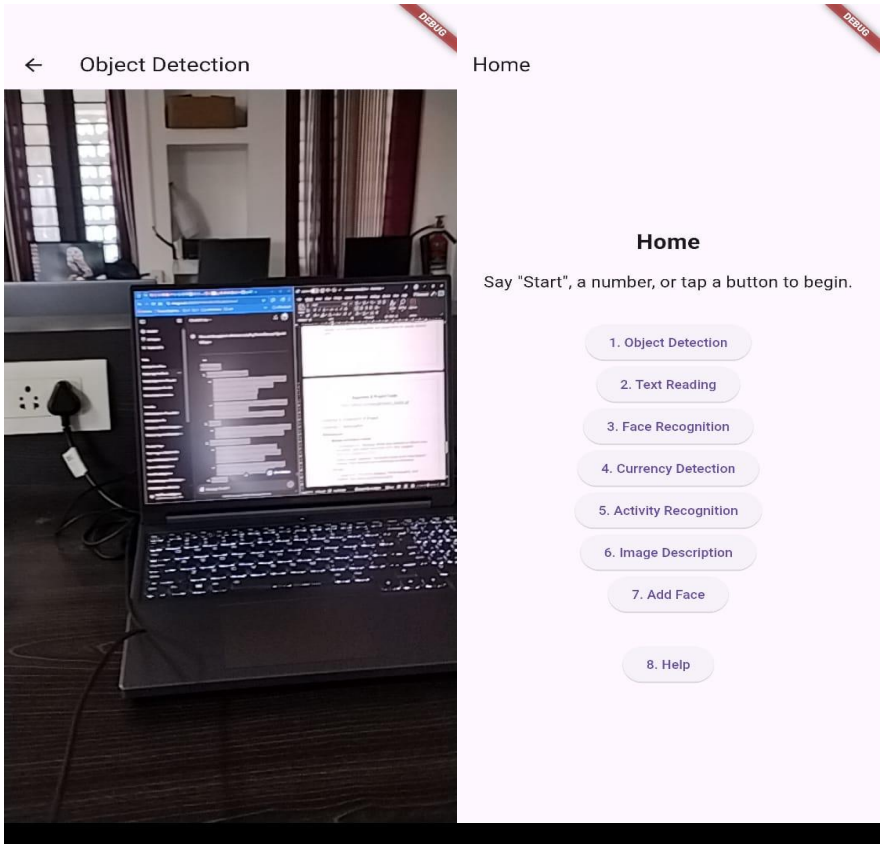
11. Conclusion

This project successfully developed a multi-functional mobile application that provides real-time assistance to visually impaired individuals through object detection, face recognition, currency recognition, and more. The use of state-of-the-art AI models like YOLOv5 and Whisper ensures high accuracy, making this a valuable tool in improving accessibility and independence for visually impaired users.

Appendix A Project Code

- https://github.com/akzgit/Vision_Assist.git

Appendix B Screenshot of Project



References

- **MoViNet and Kinetics Dataset**
  - A. Kondratyuk et al., “MoViNets: Mobile Video Networks for Efficient Video Recognition,” *arXiv preprint arXiv:2103.11511*, 2021. Available: https://arxiv.org/abs/2103.11511
  - Kinetics Dataset. DeepMind, "The Kinetics Human Action Video Dataset," Available: https://deepmind.com/research/open-source/kinetics
- **YOLOv5**
  - G. Jocher et al., "YOLOv5 by Ultralytics," GitHub Repository, 2020. Available: https://github.com/ultralytics/yolov5
- **OpenAI Whisper API & GPT-4**
  - OpenAI. "Whisper: OpenAI's Automatic Speech Recognition (ASR) System," OpenAI API, Available: https://openai.com/research/whisper

- OpenAI. "GPT-4 Technical Report," OpenAI API, 2023. Available: https://openai.com/research/gpt-4

➢ **TensorFlow**

- M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016. Available: https://www.tensorflow.org/

➢ **Face Recognition Library (OpenCV)**

- OpenCV Team, "OpenCV: Open Source Computer Vision Library," Available: https://opencv.org/
- A. Geitgey, "Face Recognition: A Python Library for Face Recognition," GitHub Repository, Available: https://github.com/ageitgey/face_recognition

➢ **MobileNetV2**

- M. Sandler et al., “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” *arXiv preprint arXiv:1801.04381*, 2018. Available: https://arxiv.org/abs/1801.04381