ABSTRACT

This project presents a real-time currency recognition system designed to assist visually impaired individuals in identifying Indian currency notes and determining their total value. The system utilizes computer vision and machine learning techniques to detect and classify currency denominations from a live video feed. The recognition model is built using YOLOv10 (You Only Look Once), OpenCV, and Python, ensuring high-speed detection and accuracy. The dataset for training the model is created using RoboFlow, optimizing the system's ability to accurately distinguish different currency notes. The project also integrates cloud-based processing and notification services, enabling automated real-time detection and audio feedback to users. To enhance accessibility, the system automates various tasks through the Automate App, allowing hands-free operation via IP camera connectivity, voice commands, and real-time cloud interactions. Additionally, robust server monitoring mechanisms are implemented to ensure uninterrupted functionality, automatically handling connectivity issues and reducing latency with GPU-supported cloud processing. This project demonstrates the practical application of machine learning and video processing technologies in assistive technology, providing a user-friendly, efficient, and scalable solution for visually impaired individuals to manage currency transactions independently. Future improvements may include higher-efficiency models, enhanced datasets, and more optimized real-time processing for increased reliability.

INTRODUCTION

A real-time currency recognition system is a software application designed to assist visually impaired individuals in accurately identifying and calculating the total value of Indian currency notes. This system leverages computer vision and machine learning to detect and classify currency denominations from a live video feed, providing an efficient and accessible solution for independent financial transactions. This project consists of several key components:

User Interface & Automation: The system is designed to operate with minimal user interaction, making it ideal for individuals who may not be accustomed to using smartphone applications. It integrates with the Automate App, enabling hands-free operation through IP camera connectivity, voice commands, and real-time cloud interactions. The detected currency values are delivered audibly through the smartphone's speakers, ensuring a seamless and user-friendly experience.

Currency Detection & Recognition: The system captures live video using an IP camera or smartphone camera and processes it in real-time. It employs YOLOv10 (You Only Look Once) object detection for recognizing different currency denominations. The dataset for training the model is developed using RoboFlow, ensuring optimized accuracy in detecting various Indian banknotes.

Transaction Processing & Cloud Integration: The detected currency values are processed and stored using a cloud-based server for enhanced reliability. Real-time notifications are sent to the user via cloud services, allowing immediate feedback on the recognized currency and its total value.

Security & Error Handling: The system includes robust security mechanisms, such as validating the camera connection and ensuring secure cloud communication. It also incorporates error-handling features to manage invalid inputs, camera failures, and connectivity issues. If the camera feed becomes unavailable, the system enters a waiting state and continuously attempts to reconnect, ensuring uninterrupted service.

Performance Optimization & Future Scope: The system is designed to minimize latency by leveraging GPU-supported cloud processing, ensuring fast and accurate real-time detection. Future improvements may include high-efficiency deep learning models, enhanced datasets, and better real-time processing for increased accuracy and usability.

By integrating these components, the currency recognition system not only enhances financial independence for visually impaired individuals, but also demonstrates the practical applications of machine learning, computer vision, and automation in assistive technology.

LITERATURE SUVEY

The need for automated currency recognition originated from the increasing demand for counterfeit detection, self-service banking solutions, and assistive technologies for visually impaired individuals. Early systems were designed to recognize banknotes using basic pattern-matching techniques and were primarily used in automated vending machines**,** ATMs**,** and currency counting devices. With the growth of computer vision and artificial intelligence**,** currency recognition has evolved into a sophisticated process that can be implemented on mobile devices, kiosks, and cloud-based platforms.

Traditional Image Processing Techniques

Initial currency recognition systems relied on image processing algorithms such as edge detection, histogram analysis, and template matching. Techniques like Canny edge detection, Hough transform, and Scale-Invariant Feature Transform (SIFT) were commonly used to extract distinguishing features of banknotes. However, these methods were highly sensitive to illumination changes, note folding, and background noise, making them unreliable for real-world applications.

Machine Learning-Based Approaches

To overcome the limitations of traditional methods, researchers started implementing machine learning algorithms such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forest for currency classification. These models used handcrafted features like color histograms, texture descriptors, and shape analysis to differentiate banknotes. While these approaches improved accuracy, they required manual feature engineering and struggled with real-time processing and diverse currency variations.

Deep Learning and CNN-Based Models

The introduction of deep learning revolutionized currency recognition by enabling models to learn features automatically. Convolutional Neural Networks (CNNs) have demonstrated superior accuracy in detecting and classifying banknotes. Popular architectures like YOLO (You Only Look Once), ResNet, and MobileNet have been widely adopted for their speed and efficiency. These models can recognize multiple denominations in real-time, making them ideal for applications designed for visually impaired users.

Integration of Assistive Technologies

Currency recognition has also played a crucial role in assistive technologies for blind and visually impaired individuals. Recent studies have focused on integrating text-to-speech (TTS), voice assistants, and smartphone applications to provide real-time audio feedback. Additionally, cloud-based AI services are now being used to enhance the accuracy and accessibility of such systems.

Challenges and Future Scope

Despite advancements, several challenges remain, including real-time processing constraints, counterfeit detection, environmental variations, and the ability to recognize multiple currencies. Future research aims to address these issues by integrating blockchain for secure transactions, AI-powered counterfeit detection models, and optimized mobile-based implementations. Additionally, the use of high-efficiency deep learning models and cloud-based computing is expected to improve system performance and scalability.

This literature survey highlights the evolution of currency recognition systems and the significance of deep learning, real-time automation, and assistive technologies in making these systems more efficient and accessible.

PROBLEM STATEMENT

Visually impaired individuals face significant challenges in handling currency independently, making daily financial transactions difficult. Traditional methods of currency recognition, such as tactile features on banknotes and third-party assistance, often prove inadequate, unreliable, or inconvenient. Existing technological solutions either lack accuracy, suffer from high latency, or fail to function effectively in diverse real-world conditions, such as varying lighting, folded or worn-out banknotes, and background noise.

This project aims to develop a real-time currency recognition system that leverages deep learning and computer vision to provide an accessible solution for visually impaired users. By employing a robust Convolutional Neural Network (CNN)-based model, specifically utilizing YOLO (You Only Look Once) for object detection, the system ensures accurate identification of Indian currency notes in various conditions. The dataset for training the model is developed using RoboFlow, ensuring optimized recognition performance.

The system integrates assistive technologies such as text-to-speech (TTS) and voice-guided responses to provide real-time auditory feedback to users. It is designed to work with minimal user interaction, making it highly user-friendly. The application supports live video processing through smartphone or IP cameras, enabling efficient recognition and response delivery.

Additionally, the system incorporates cloud-based storage for transaction records, ensuring reliability and accessibility. Security features such as encrypted data transmission and error-handling mechanisms enhance its robustness, preventing fraudulent activities and mitigating technical failures. The incorporation of GPU-accelerated processing further optimizes performance, reducing latency and enhancing real-time response.

Despite its advantages, challenges remain, including counterfeit detection, cross-currency recognition, and environmental adaptability. Future enhancements will focus on integrating blockchain for secure transactions, improving AI-driven counterfeit detection, and optimizing mobile-based implementations for broader accessibility.

By addressing these issues, the proposed currency recognition system aims to empower visually impaired individuals, providing them with greater financial independence and accessibility in daily transactions.

SYSTEM ANALYSIS

4.1 Analysis

1. Requirements Gathering

This phase involves understanding the needs and expectations of visually impaired users, as well as identifying technical requirements. Key requirements include:

- **Functionality:** Real-time currency recognition and total value estimation.
- **Automation:** Hands-free operation via IP camera and voice commands.
- **Security Features**: Secure server communication and error handling.
- **User Interface Preferences**: Audio-based feedback for ease of use.
- **Performance Expectations**: Low-latency and real-time processing.

2. Use Case Analysis

Use case analysis identifies different actors and their interactions with the system:

- **Primary Actor**: The visually impaired user.
- Use Case Example:
    - **Recognize Currency**:
        - The user places a currency note in front of the camera.
        - The system captures the live video feed and detects the denomination.
        - The total amount is calculated and announced via audio feedback.

3. Data Modeling

The system processes and stores currency recognition data, requiring structured data management:

- **Entities and Attributes**:
    - Currency Dataset: Contains labeled images of Indian banknotes.
    - User Transactions: Logs of recognized denominations and total amounts.
    - Server Communication Logs: Tracks real-time processing and error reports.
- **Relationships**:
    - The currency detection model interacts with the cloud server for processing.
    - The Automate App handles camera control and result delivery.

4. Functional Design

Functional design defines how the system processes user input and generates output:

- **Input**:
    - Live video feed from an IP camera or smartphone camera.
    - Voice commands for hands-free interaction.
- **Processing**:
    - The YOLOv10 model detects currency notes in the video feed.
    - The system calculates the total value of detected currency.
- **Output**:

- Audio feedback announces the detected amount.
- Cloud notifications send the results to the user.

5. Architectural Design

The system is designed with modularity and scalability to ensure smooth operation:

- Currency Recognition Module (YOLOv10 & OpenCV for detection).
- Cloud Processing & Storage (GPU-supported servers for real-time computation).
- User Interaction Module (Automate App for camera control & voice feedback).
- Error Handling & Security Layer (Handles camera failures & connection issues).

6. Interface Design

The user interface is designed to be accessible and intuitive for visually impaired users:

- Automated IP Camera Connection: The system automatically starts video capture.
- Voice-Controlled Interaction: Uses Automate App for hands-free commands.
- Audio Output: Delivers spoken feedback on detected currency values.

7. Security Analysis

To ensure a secure and reliable experience, the system incorporates:

- Secure Cloud Communication: Prevents data breaches and unauthorized access.
- Error Handling & Recovery Mechanisms: Detects camera failures and reconnects automatically.
- Continuous Monitoring: Ensures the server remains connected for real-time processing.

8. Performance Analysis

To optimize real-time currency detection, the system is evaluated for:

- Processing Speed: Uses GPU-supported cloud servers to reduce latency.
- Accuracy & Efficiency: The trained model ensures high precision in detection.
- Scalability: The system can be expanded to support more currencies and improved AI models.

By conducting a thorough system analysis, the currency recognition system ensures efficient, secure, and accessible functionality, providing visually impaired individuals with a reliable financial aid tool

4.2 Feasibility System

As of now, ready-made currency recognition systems specifically designed for visually impaired individuals are not widely available as off-the-shelf solutions. However, custom-built machine learning models and computer vision applications have been developed for similar purposes, leveraging technologies like YOLO (You Only Look Once), OpenCV, and cloud-based automation.

This project aims to provide a specialized, real-time currency recognition solution by integrating advanced object detection algorithms, automated camera functionality, and cloud-based processing to deliver an efficient and accessible financial assistance tool.

Technical Feasibility

- Uses YOLOv10 for real-time detection of Indian currency notes.
- Trained dataset created using RoboFlow to improve accuracy.
- The Automate App enables hands-free operation using voice commands and IP camera connectivity.
- Audio-based real-time feedback eliminates the need for screen-based interaction.
- GPU-supported cloud servers ensure low-latency detection.
- Continuous server monitoring helps maintain system stability and accuracy.

Operational Feasibility

- Designed specifically for visually impaired users, requiring minimal manual intervention.
- Seamless currency detection, audio output, and cloud notifications for an intuitive experience.
- Error-handling mechanisms ensure the system remains functional even in case of network failures or disconnections.

Economic Feasibility

- Cost-effective approach using open-source libraries like OpenCV, Python, and YOLOv10.
- Cloud services provide scalable and flexible deployment options.
- Reduces the need for expensive specialized hardware-based currency recognition devices.

Future Feasibility & Scalability

- The system can be expanded to recognize other global currencies.
- Future updates can incorporate more advanced deep learning models to further improve detection speed and accuracy.

- Potential integration with banking or digital payment systems for enhanced accessibility.

While existing currency recognition solutions do exist, this project offers a custom-built, AI-driven, real-time approach tailored specifically for visually impaired individuals, ensuring accessibility, accuracy, and ease of use.

4.3 Proposed System

User Interface

- Implement a user-friendly interface for visually impaired individuals to interact with the currency recognition system. This could include:
    - Voice commands for initiating the currency detection process.
    - Audible feedback to communicate the detected currency and total amount.
- Design the interface to:
    - Prompt users to start the camera feed for currency detection.
    - Announce the detected currency and total amount in real-time.

Currency Detection and Recognition

- Develop a system for real-time currency detection using YOLOv10 and OpenCV:
    - Train the model using a custom dataset created on RoboFlow to recognize Indian currency notes.
    - Implement real-time video processing to detect and classify currency notes in the camera feed.
    - Calculate the total amount of detected currency notes and provide the result audibly.
- Include validation checks to ensure:
    - Accurate detection of currency notes under varying lighting and orientation conditions.
    - Handling of multiple currency notes in the frame simultaneously.

Automation and Server Integration

- Automate the process of connecting the IP camera to the server and sending live video feeds for processing.
- Implement a cloud-based notification system to:
    - Send detected results (currency and total amount) back to the user’s device.
    - Deliver results audibly via the smartphone’s speakers.
- Ensure the system can handle server disconnections by entering a waiting state and attempting to reconnect automatically.

Security Features

- Implement security measures to protect user data and ensure privacy:
    - Encrypt sensitive data such as camera feeds and detection results.
    - Ensure the system operates locally on the user's device to minimize data exposure.
    - Automatically stop the camera feed after a period of inactivity to prevent unauthorized access.

Logging and Auditing

- Log system activities and detection results for auditing purposes:
    - Record details such as timestamps, detected currency notes, and total amounts.
    - Securely store logs and make them accessible only to authorized personnel for monitoring and analysis.

Testing and Validation

- Conduct thorough testing to validate the functionality, accuracy, and usability of the currency recognition system:
    - Test with various Indian currency notes under different conditions (e.g., lighting, angles, backgrounds).
    - Evaluate the system's performance in real-time scenarios, including edge cases like overlapping notes or partially visible notes.
    - Use both automated testing frameworks and manual testing techniques to identify and address bugs or issues.

Documentation

- Provide comprehensive documentation covering:
    - System architecture and design decisions.
    - Instructions for setting up and using the system.
    - Troubleshooting guidelines for common issues.
    - APIs and interfaces for integration with external systems or databases (if applicable).

Deployment and Maintenance

- Deploy the currency recognition system in a secure environment, ensuring:
    - Appropriate access controls and security measures are in place.
    - The system is optimized for performance on devices used by visually impaired individuals.
- Establish procedures for ongoing maintenance, including:
    - Regular updates to the dataset and model to improve accuracy.
    - Patches and enhancements to address evolving requirements and potential security vulnerabilities.

4.4 Hardware Configuration

- **Computer**: A modern computer (desktop, laptop, or Raspberry Pi) capable of running Python and handling real-time video processing using OpenCV and YOLOv10. The system should have at least 4GB of RAM and a GPU (for faster processing) if using PyTorch for model inference.
- **Camera**: An IP camera or smartphone camera to capture the video feed. The camera should be capable of providing a consistent and high-quality feed at a resolution suitable for currency detection (preferably 640x480 or higher).
- **Input Devices**: A keyboard (for CLI operation) or touchscreen/mouse (for potential GUI interaction, if needed).
- **Display**: A monitor or screen to display the detected frames (if applicable) or for debugging purposes during development and testing.
- **Internet Connection**: A stable internet connection for cloud communication to send the detected results to an automation service (via the requests API).
- **Processing Power**: If running the system on a smartphone or Raspberry Pi, ensure the device has sufficient processing power and memory to handle the real-time video processing, YOLOv10 model inference, and communication with the server.

4.5 Software Specification:

- **Programming Language**: Python will be used to develop the Currency Recognition system due to its simplicity, readability, and compatibility with libraries like OpenCV, YOLOv10, and PyTorch.
- **Development Environment**: The system will be developed within an Integrated Development Environment (IDE) such as PyCharm, Visual Studio Code, or Jupyter Notebook for easier coding and debugging.
- **Version Control**: Git will be used for version control to manage code changes and collaborate on development.

- **Key Libraries and Frameworks**:

    - **OpenCV**: For capturing and processing the video feed from the camera.
    - **YOLOv10 (Ultralytics)**: For object detection and recognizing currency notes in the video feed.
    - **Torch (PyTorch)**: For model inference and running the YOLOv10 model.
    - **supervision (sv)**: For annotating the detected bounding boxes around the currency notes.
    - **requests**: For sending the detected total amount to an external service (Automate) via a cloud-based API.
    - **time**: For handling delays and waiting periods between detection cycles.

- **Testing**:

    - **Unit tests**: To test individual components such as currency detection and amount calculation.
    - **Integration tests**: To ensure seamless integration between camera feed, model inference, and cloud communication.
    - **End-to-End tests**: To test the system as a whole, ensuring real-time detection, amount calculation, and successful communication with the automation service.

- **Deployment**:
    - The system should be deployed in a secure environment with appropriate access controls.
    - If deploying on a smartphone or Raspberry Pi, ensure the system runs efficiently with adequate memory and processing power.

- **Cloud Communication**:
    - The system will use an HTTP request to send detected amounts to an external service (Automate) using the requests module. The message includes the total currency detected, which is sent as a notification.
    - The server is expected to be connected to via a camera feed URL, and the system should wait for the camera feed to become available before starting the processing.

- **Security Features**:
    - The system does not currently include encryption, but it should ensure secure handling of camera data, especially when used in cloud communication.
    - The camera feed and detected results should be handled with minimal exposure to ensure privacy.

4.6 Future Improvements:

- **Model Training and Dataset Updates**: The model can be retrained with new currency images to improve its accuracy and handle various currency note conditions.
- **UI Improvements**: A graphical user interface (GUI) can be added to configure settings such as camera selection, network settings, or even to visualize the results before sending notifications.
- **Security Enhancements**: Future iterations may implement encryption for sensitive data, especially for camera feeds and detected results

SYSTEM DESIGN

5.1 Input Design:

The input design of the Currency Recognition System is focused on providing a seamless, user-friendly experience with a primary goal of ensuring easy interaction with visually impaired individuals. The system facilitates initiating currency detection, receiving results, and managing settings.

- **Currency Detection Interface:**
    - Users are prompted to position currency notes in front of the camera.
    - Clear audible instructions guide users throughout the detection process.
    - Voice commands are used to initiate the detection process and provide real-time feedback about the process.
- **Results Interface:**
    - The system audibly announces the detected currency and total amount.
    - Users are given feedback on the success or failure of the detection process.
    - If the system encounters issues (e.g., poor lighting, overlapping notes), it provides audible error messages and instructions to help the user resolve the issue.
- **Error Handling:**
    - The system addresses several scenarios that may interfere with the detection process, including:
        - Poor lighting conditions affecting the camera feed.
        - Overlapping or partially visible notes that might confuse the detection model.
        - Server disconnections if cloud-based processing is involved.
    - Audible error messages guide the user to correct any issues by repositioning currency notes or adjusting lighting.
- **Navigation:**
    - The system features intuitive navigation options, allowing users to use voice commands to:
        - Start or stop the camera feed.
        - Request assistance or instructions.
        - Repeat or stop the detected results if necessary.
- **Accessibility:**
    - Designed specifically for visually impaired individuals, the system offers:
        - Audible feedback for all user actions and results.
        - Simple voice commands for navigation, providing an intuitive way to interact with the system.
        - Compatibility with screen readers or other assistive technologies if a GUI interface is employed.

By focusing on clarity, simplicity, and error management, the input design enhances the system's usability for visually impaired users, ensuring an intuitive and effective currency recognition process.

5.2 System Architecture:

The Currency Recognition System adopts a modular architecture to support scalability, ease of maintenance, and integration of future features:

- **Camera Module**:
    - Captures the real-time video feed from the camera, ensuring consistent input for detection.
    - This module interfaces with OpenCV for video frame capture and processing.
- **Detection Module**:
    - Uses YOLOv10 and OpenCV for processing the video feed, detecting currency notes, and calculating the total amount based on predefined values.
- **Audio Module**:
    - Provides audible feedback to the user:
        - Instructions for positioning the currency notes.
        - Announcing the detected currency and total amount.
        - Communicating error messages for troubleshooting.
- **Server Module (Optional)**:
    - If cloud-based processing is used (through server interactions), the server module will handle the processing of the video feed and send results back to the user's device.
    - The server ensures minimal latency and maintains a stable connection for smooth operation.
- **User Interface Module**:
    - Features a voice command interface designed for visually impaired individuals to interact with the system.

- Ensures a seamless user experience by providing voice-activated controls for various functionalities.

5.3 System Configuration and Dependencies:

- **YOLOv10 Model**: The system requires a pre-trained YOLOv10 model to detect currency notes. The model will be loaded into the PyTorch device (either CPU or GPU) depending on the available hardware.
- **Cloud Integration**: The system is designed to send messages to a cloud service (Automate) to notify users of the detected currency amount. This is done using a POST request to a specified URL with JSON payload containing the total detected currency.
- **Error Handling**: The system has error handling for lost connections to the server, including retries and a waiting period before reattempting to reconnect.

5.3.1 Error Handling and Validation:

The system includes robust error handling mechanisms to maintain reliability and ensure smooth user experiences under various conditions:

- **Invalid Inputs**:
  - In cases where the system cannot detect the currency (e.g., poor lighting or overlapping notes), it provides audible error messages explaining the issue and offering corrective actions.

- **Server Disconnections**:
  - If the system uses cloud processing, it will automatically attempt to reconnect if the connection is lost. It enters a waiting state until the server becomes available again.

- **Edge Cases**:

  - The system is tested for various edge cases:
    - **Multiple currency notes in the frame**: The system detects all notes and calculates the total amount.
    - **Partially visible or folded notes**: The system attempts to detect even partially visible notes, with error feedback if the detection fails.
    - **Varying lighting conditions**: The system handles different lighting conditions by adjusting detection parameters for robustness.

5.4 System Flow Diagram (SFD)

The System Flow Diagram (SFD) or Data Flow Diagram (DFD) provides a visual representation of the data flow and the processes within the Currency Recognition System, helping to understand how data moves between different components or processes. This is essential for understanding system functionality and the interactions between various elements.

1. Main Processes:

- **Camera Feed Capture:**
    - This process captures the real-time video feed from the camera. It is essential for ensuring high-quality input, enabling the system to perform accurate currency detection.
- **Currency Detection:**
    - This process uses YOLOv10 (a deep learning model) and OpenCV to detect and classify currency notes in the video feed. Once the notes are detected, the system calculates the total amount by summing up the monetary values of the recognized notes.
- **Audible Feedback:**
    - After currency detection and amount calculation, the system provides audible feedback to the user, announcing the detected currency and total amount. It also provides instructions and error messages (e.g., on positioning the currency correctly) if issues arise during the detection process.
- **Error Handling:**
    - This process is responsible for managing errors that may arise, such as poor lighting conditions, overlapping notes, or server disconnections (in case of cloud-based processing). If the system encounters such issues, it offers audible error messages to guide the user on how to resolve the problem.

2. Data Stores:

Since the Currency Recognition System doesn't use a permanent database, all data is processed in real-time and discarded after the detection is complete. However, there are some temporary data stores used during the processing phase:

- **Camera Feed:**

    - This is the real-time video feed from the camera, serving as the primary input for the system.
- **Detection Results:**
    - Temporary storage of the detected currency notes and calculated total amount during processing.
- **Error Logs:**
    - Temporary storage of error messages and system logs for debugging purposes or for providing feedback to the user.

3. Data Flows:

- **User Input:**
    - The user positions the currency notes in front of the camera.
    - The user can also interact with the system via voice commands to control operations, such as starting/stopping the camera feed or requesting help.
- **System Output:**
    - The system provides audible feedback on the detected currency and the total amount.
    - If issues arise (e.g., poor lighting or overlapping notes), the system will deliver audible error messages to guide the user in troubleshooting.
- **Processing Data:**
    - The system processes the camera feed to detect and classify the currency notes.
    - The system also calculates the total amount of detected currency notes based on predefined values associated with different currency classes.

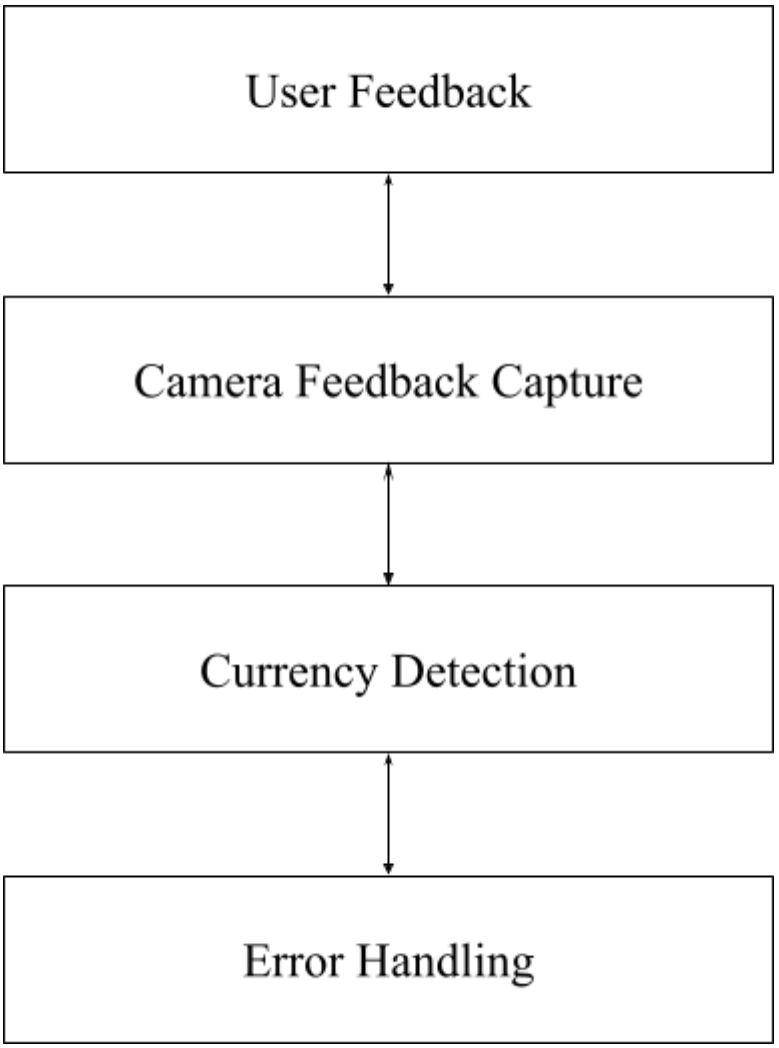


Fig 5.4.1 System Flow Diagram

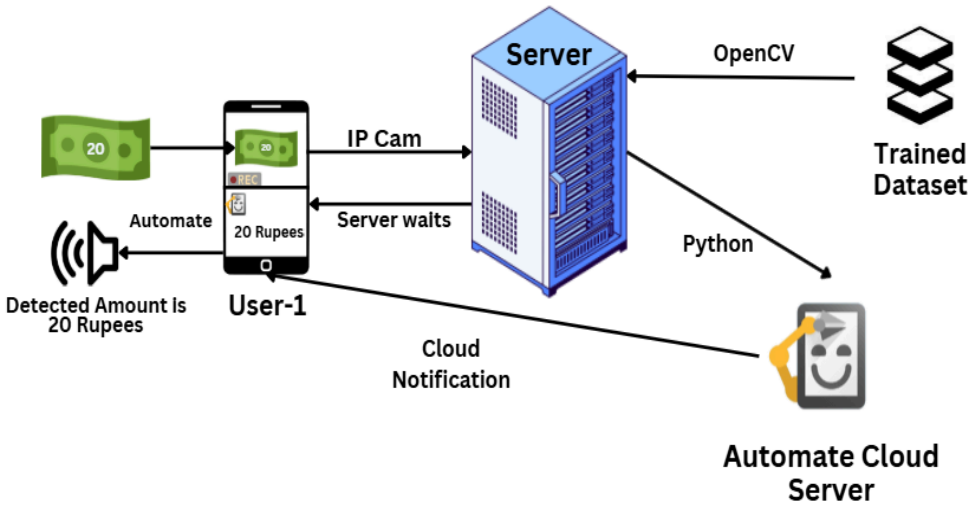


Fig 5.4.2 Workflow Diagram

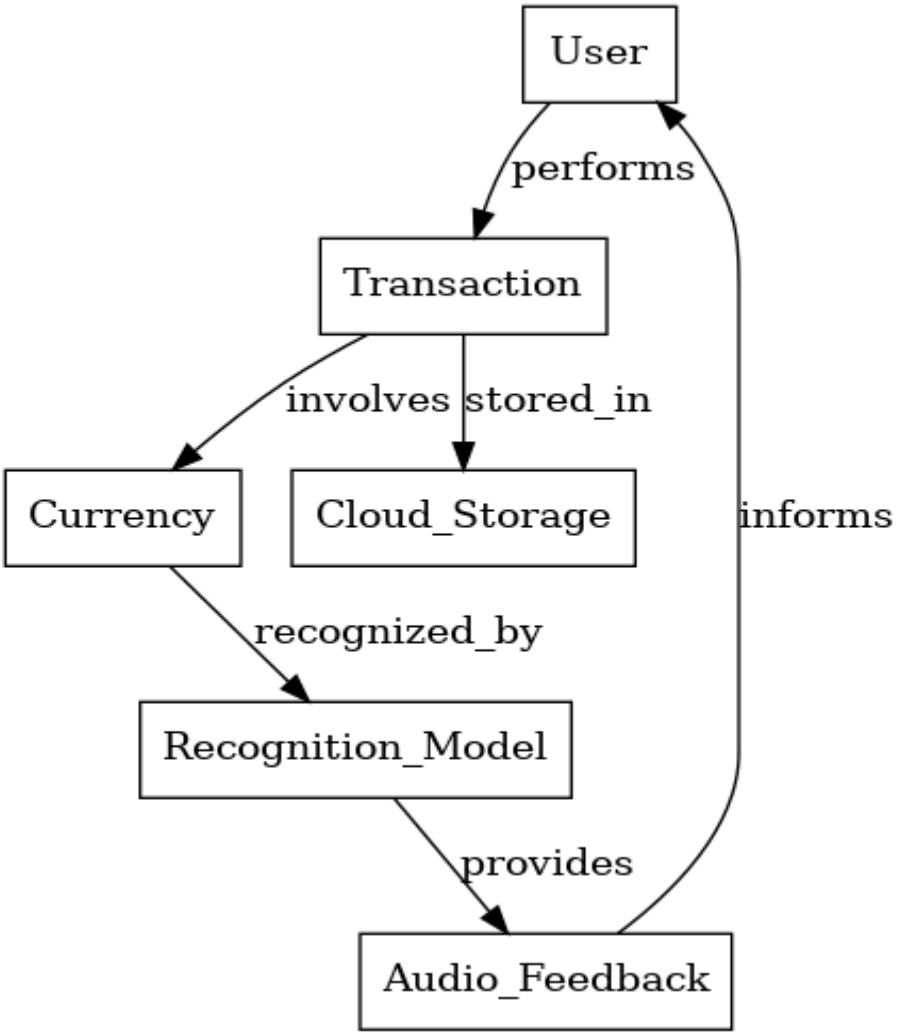


Fig 5.4.3 ER Diagram

SYSTEM DESCRIPTION

The Currency Recognition System is a Python-based application that assists visually impaired individuals in identifying and determining the total value of Indian currency notes. The system uses real-time video processing and machine learning techniques, specifically YOLOv10 and OpenCV, to detect and classify currency notes accurately. Below is a detailed breakdown of the system's features and functionality:

1. Currency Detection:

- **Real-Time Video Capture**:
  - The system captures a real-time video feed from the camera and processes it using machine learning algorithms.

- **Currency Classification**:
  - The system detects and classifies Indian currency notes by identifying key features of the notes. It uses YOLOv10 (a powerful deep learning model) and OpenCV (a computer vision library) for accurate detection and classification.

- **Total Value Calculation**:
  - After detecting the notes, the system calculates the total value of the detected currency notes based on predefined values for different denominations.

- **Custom Dataset**:
  - The system is trained on a custom dataset created using RoboFlow, ensuring high accuracy and effectiveness in recognizing the currency notes.

2. User Interaction:

- **Voice Command Interface**:

  - The system provides an accessible interface for users by allowing voice-based commands. The user can interact with the system by issuing simple commands like "start camera feed", "stop", or "help**"**.

- **Audible Feedback**:

  - The system gives real-time audible announcements of detected currency notes and the total amount, ensuring that the user can track the process even without visual feedback.

- **Guidance for Positioning Notes**:
    - The system provides clear audible instructions to guide users in positioning the currency notes correctly in front of the camera for accurate detection.

3. Error Handling:

- **Handling Poor Lighting**:
    - The system can detect poor lighting conditions and alert the user with an audible error message, such as "Please adjust the lighting and try again".
- **Overlapping or Partially Visible Notes**:
    - The system is capable of detecting overlapping or partially visible currency notes and provides error messages to guide the user in properly adjusting the notes for detection.
- **Server Disconnections**:
    - If cloud-based processing is used, the system has mechanisms to handle server disconnections by attempting to reconnect automatically, ensuring minimal disruption.

4. Internal Data Management:

- **Real-Time Processing**:
    - The system processes all data locally on the user's device, meaning no sensitive information (e.g., camera feed or detection results) is transmitted or stored externally, preserving privacy and security.
- **Temporary Data**:
    - The system processes temporary data during detection (e.g., detected currency and total amount) but discards this information immediately after the detection process is completed.

5. User-Friendly Interface:

- **Intuitive Design**:
    - The system's interface is designed to be easy to use even for individuals with limited technical knowledge. Clear and concise audible instructions ensure smooth operation.

- **Accessibility**:
    - The system is compatible with screen readers and other assistive technologies, ensuring it can be used by visually impaired users.
- **Simple Voice Commands**:
    - Users can control the system and receive feedback using voice commands, reducing the need for physical interaction.

6. Modularity and Extensibility:

- **Modular Architecture**:
    - The system is designed to be modular, allowing for easy integration of new features and enhancements. This design facilitates easy updates and scalability.
- **Future Enhancements**:
    - Future improvements may include:
        - Integration with cloud-based processing to improve accuracy and scalability.
        - Support for additional currencies or languages.
        - Integration with banking APIs for real-time balance updates.

7. Security and Privacy:

- **Local Processing**:
    - All data is processed locally on the user's device, ensuring privacy and preventing sensitive data from being sent over the internet.
- **No Data Storage**:
    - The system does not store sensitive information such as the camera feed or detection results. All data is discarded after processing.
- **Auto Stop Feature**:
    - The camera feed automatically stops after a period of inactivity, preventing unauthorized access.

8. Testing and Validation:

- **Testing Scenarios**:
    - The system undergoes thorough testing to ensure:
        - Accurate detection of different Indian currency notes under various lighting and orientation conditions.
        - Handling of edge cases, such as overlapping or partially visible notes.
        - Real-time performance evaluation to assess how the system performs under real-world conditions.

9. Deployment and Maintenance:

- **Deployment**:
    - The system is deployed on devices such as smartphones or Raspberry Pi that are used by visually impaired individuals. It is designed to work seamlessly on these platforms.

- **Regular Maintenance**:
    - The system undergoes regular updates to enhance its functionality, improve detection accuracy, and incorporate user feedback.

Key Features

1. **Real-Time Currency Detection**:
    - Detects and classifies Indian currency notes accurately in real-time.
2. **Audible Feedback**:
    - Provides real-time results and instructions audibly to help visually impaired users.
3. **Voice Commands**:
    - Allows users to interact with the system using simple voice commands to control the process.
4. **Error Handling**:
    - Handles issues like poor lighting or overlapping notes and provides guidance.

5. **Local Processing**:
    - Ensures data privacy by processing all data locally on the user's device.
6. **Accessibility**:
    - Designed to be intuitive and accessible for visually impaired individuals, with compatibility for screen readers and voice commands.

IMPLEMENTATION

7.1 Implementation of the System

1. Designing the System:

The first step in designing the Currency Recognition System involves defining the core functionalities and user interface, ensuring that it addresses the specific needs of visually impaired users.

- **Core Functionalities**:
    - **Real-Time Detection**: The system must accurately detect and classify Indian currency notes using video processing.
    - **Currency Value Calculation**: It should calculate the total value of the detected notes and provide this information audibly to the user.
    - **Audible Feedback**: The system should provide clear, real-time audible **feedback** on detected currency and the total value.
    - **Error Handling**: The system must be able to handle errors such as poor lighting, overlapping or partially visible notes, and other common challenges.
- **User Interface**:
    - The user interface will be voice-command-driven, with audible instructions and feedback to guide the user throughout the process.
    - The interface should be simple, intuitive, and fully accessible for visually impaired individuals.

2. Implementing the System Logic:

The implementation of the Currency Recognition System involves writing Python code to handle each of the system's core functionalities:

- **Real-Time Video Processing with OpenCV**:
    - Use OpenCV to capture the video feed from the camera in real-time, process each frame, and detect currency notes.
- **Currency Detection with YOLOv10**:
    - Use the YOLOv10 object detection model to detect and classify currency notes in the video feed. The model is trained specifically on a custom dataset of Indian currency notes.

- **Audible Feedback with Text-to-Speech**:
    - Use a text-to-speech library such as pyttsx3 or gTTS to provide real-time, audible feedback to the user. This includes instructions, detected currency, and the total value.

- **Error Handling**:
    - Implement error handling mechanisms to ensure the system can gracefully handle poor lighting conditions, overlapping notes, and other common detection errors.
    - Provide audible error messages to guide the user.

7.2 Overview Of Automate App

7.2.1 Introduction to the Automate App

The Automate App is a versatile and powerful automation tool designed for Android devices, available on the Google Play Store. It offers users the ability to create automation scripts without requiring programming skills, using an intuitive flowchart-based interface. The app simplifies complex automation tasks by visually connecting actions and conditions, making it user-friendly for beginners and advanced users alike.

With Automate, users can automate a wide range of smartphone functions, from simple tasks like adjusting device settings to complex workflows involving multiple triggers and conditions. The app's unique flow-based approach provides a clear representation of automation sequences, enabling users to visualize, modify, and debug their scripts with ease.

Key Features of the Automate App

1. **Flow-Based Automation:**
    - The core feature of the Automate App is its **flowchart-based automation**. Users can create automated workflows by dragging and connecting "blocks" representing various actions, conditions, and loops.
    - Each block is customizable, allowing users to specify actions such as enabling Wi-Fi, sending messages, or triggering system functions.
    - The flowchart design provides a clear and intuitive way to create, monitor, and troubleshoot complex automation scripts.

2. **Device Control and System Integration:**

- The Automate App grants extensive control over Android device functions, including:
  - Managing calls, sending SMS, and triggering notifications.
  - Adjusting system settings like brightness, volume, and connectivity options.
  - Interacting with apps, launching applications, and performing background operations.
- It also enables users to automate device-specific features, such as GPS location services, Bluetooth, and NFC.

3. **Event-Based Triggers and Real-Time Automation:**
   - Automation can be triggered based on specific events, such as:
     - **Location-based Triggers:** Activate scripts based on geographic coordinates (e.g., enable Wi-Fi when arriving home).
     - **Time-Based Triggers:** Execute workflows at scheduled times or recurring intervals.
     - **Battery and Charging Events:** Automate tasks when charging or on low battery.
     - **Network and Connectivity Changes:** React to Wi-Fi or mobile network status changes.
   - This event-driven approach ensures that workflows run automatically without manual intervention.

4. **Advanced Logic and Conditions:**
   - The Automate App offers **advanced scripting capabilities**, including:
     - Conditional Statements: Supports if-else conditions for dynamic decision-making.
     - Loops and Iterations: Allows repeat actions until specific conditions are met.
     - Variables and Expressions: Store and manipulate data within scripts.
     - Error Handling: Execute alternate actions when errors occur.

5. **Security and Privacy:**
   - The app includes comprehensive security features, allowing users to safeguard their scripts and control permissions.
   - Users can set password protection and encryption for sensitive workflows.

6. **User-Friendly Interface and Customization:**

- The flowchart editor offers drag-and-drop functionality for ease of use.
- The app provides pre-built templates for common tasks, reducing the need for manual setup.
- Users can personalize scripts and manage them efficiently within the app.

7.2.1 Why Choose the Automate App?

The Automate App stands out as a powerful automation tool for Android devices, offering versatility and functionality unmatched by traditional task managers. Its ability to create custom workflows, automate complex tasks, and provide real-time control makes it an essential tool for tech enthusiasts, productivity seekers, and developers. The app's visual scripting approach ensures that users can create comprehensive workflows effortlessly, reducing complexity while maintaining flexibility. Whether for personal convenience or advanced automation, the Automate App is a game-changer in the world of Android automation tools.

TESTING

Thorough testing is essential to ensure that the system works as expected in real-world scenarios. Various types of testing should be conducted to evaluate different aspects of the system:

- **Types of Testing**:
    - **Unit Testing**:
        - Test individual components or modules to ensure that each part of the system functions correctly.
        - Examples:
            - Test the currency detection module to confirm that it accurately detects and classifies different denominations.
            - Test the audible feedback module to ensure it clearly announces the detected currency and total value.
        - **Tools**: Python's built-in unittest framework or third-party libraries like pytest can be used.
    - **Integration Testing**:
        - Test how well the different components of the system work together.
        - Examples:
            - Test the integration between the camera feed capture and currency detection modules to ensure seamless data transfer and processing.
            - Test the integration between the currency detection and audible feedback modules to ensure correct information is relayed to the user.
    - **Functional Testing**:
        - Test the entire system to ensure it works as intended and handles different user scenarios.

        - Examples:
            - Test the system's ability to detect currency notes under different lighting conditions.
            - Test how the system handles overlapping or partially visible notes.
            - Test for real-time performance, ensuring the video feed is processed without significant delays.

    - **User Acceptance Testing (UAT)**:

        - Involve real users (especially visually impaired individuals) to test the system and provide feedback on its usability, accuracy, and accessibility.
        - This phase helps ensure the system meets user expectations and real-world needs.

    - **Security Testing**:

        - Check for vulnerabilities to ensure data security and privacy.
        - Since the system processes data locally on the device, ensure that no sensitive information (e.g., camera feeds) is stored or transmitted externally.

    - **Performance Testing**:

        - Evaluate the system's performance under various conditions.
        - Examples:
            - Test the system's real-time performance in different lighting conditions.
            - Test the system with multiple currency notes in the frame to evaluate how it performs with crowded inputs.
            - Ensure the system processes the video feed with minimal latency and no delays.

    - **Regression Testing**:

        - After making changes to the system (e.g., updating the detection model or adding new features), test the system to ensure that existing functionalities have not been broken.

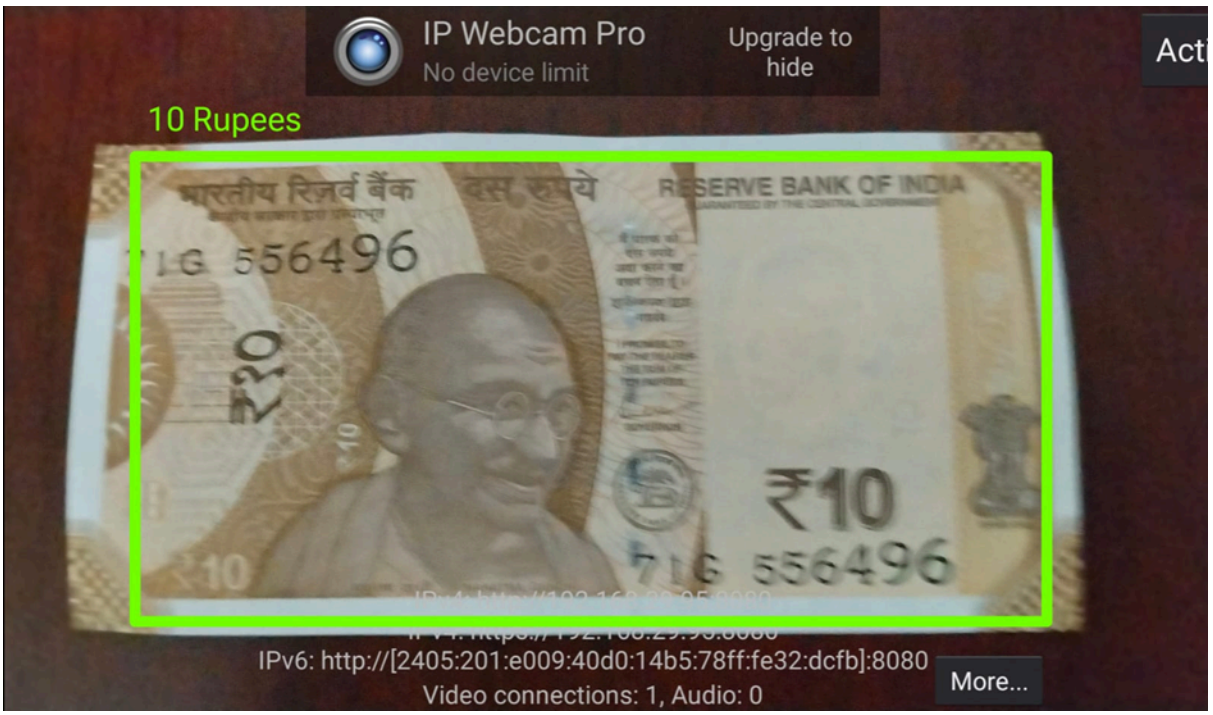
        - Examples:
            - Test the system after updating the currency detection model.
            - Test the system after adding enhancements like cloud-based processing.

- **Usability Testing**:
    - Evaluate the user interface to ensure it is intuitive and easy to use.
    - Examples:
        - Test the clarity, accessibility features and effectiveness of voice commands,compatibility with screen readers and audible feedback.
        - Measure the app's performance in recognizing currency in different lighting conditions and backgrounds to ensure real-time usability
        - Conduct surveys and usability tests with visually impaired individuals to gather feedback and refine the app for enhanced accessibility and efficiency.

OUTPUT SCREENSHOT

```
Server not available. Retrying in 5 seconds...
Server is up and running.

0: 384x640 (no detections), 68.5ms
Speed: 8.5ms preprocess, 68.5ms inference, 53.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 40.7ms
Speed: 3.0ms preprocess, 40.7ms inference, 1.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 39.3ms
Speed: 3.0ms preprocess, 39.3ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 40.1ms
Speed: 4.0ms preprocess, 40.1ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 39.9ms
Speed: 2.0ms preprocess, 39.9ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 40.1ms
Speed: 2.5ms preprocess, 40.1ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 37.0ms
Speed: 3.0ms preprocess, 37.0ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 39.4ms
Speed: 3.0ms preprocess, 39.4ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 (no detections), 72.5ms
Speed: 7.5ms preprocess, 72.5ms inference, 3.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 85.3ms
Speed: 6.6ms preprocess, 85.3ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 79.1ms
Speed: 7.3ms preprocess, 79.1ms inference, 3.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 74.0ms
Speed: 6.9ms preprocess, 74.0ms inference, 3.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 78.7ms
Speed: 6.2ms preprocess, 78.7ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 82.6ms
Speed: 8.0ms preprocess, 82.6ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 10, 68.1ms
Speed: 7.6ms preprocess, 68.1ms inference, 276.2ms postprocess per image at shape (1, 3, 384, 640)
Pausing for 10 seconds... Amount detected: 10
```


IP Webcam Pro
No device limit
Upgrade to hide
10 Rupees
भारतीय रिज़र्व बैंक
RESERVE BANK OF INDIA
₹10
IPv6: http://[2405:201:e009:40d0:14b5:78ff:fe32:dcfb]:8080
Video connections: 1, Audio: 0
More...

LIMITATION

1. **Speech Recognition Accuracy:**
   - The app's accuracy depends on background noise, pronunciation, and the clarity of the speaker.
   - It may struggle with detecting commands in noisy environments or if the speaker has a strong accent.
   - Continuous speech recognition might lead to increased battery consumption.

2. **Language and Dialect Support:**
   - The app is limited to specific languages supported by Android's Speech Recognition and TTS engines.
   - Multilingual users may face challenges if the app doesn't support language switching dynamically.

3. **Internet Dependency:**
   - Speech recognition often requires an internet connection to process commands using cloud services.
   - Offline mode might not support advanced features, reducing functionality.

4. **Limited Control Over System Calls:**
   - Android's security restrictions prevent the app from accessing or controlling certain system-level call functions.
   - Auto-answering or rejecting calls might be restricted by device policies.

5. **Battery and Resource Consumption:**
   - Continuous listening and processing consume significant battery power.
   - High CPU usage during recognition can affect overall device performance.

6. **Compatibility Issues:**
   - Some features may not work on older Android versions due to API limitations.
   - Device-specific variations can affect performance and functionality.

7. **Privacy Concerns:**

- Users may hesitate to use the app due to concerns about eavesdropping and data storage.
- Cloud-based speech recognition can raise security concerns if data is transmitted online.

8. **Speech Command Limitations:**
   - The app may struggle to interpret complex or multi-step commands.
   - Contextual understanding is limited, making it unsuitable for conversations.

9. **Error Handling and Misinterpretation:**
   - Misinterpreted commands may result in unintended actions.
   - Lack of error correction mechanisms can reduce reliability.

10. **Environmental Constraints:**
    - In noisy or crowded areas, recognition performance may degrade.
    - Speech recognition fails if the device's microphone is blocked or malfunctioning.

CONCLUSION

The Currency Recognition System offers a reliable and intuitive solution for visually impaired individuals to independently identify and calculate the total value of Indian currency notes. Utilizing real-time video processing combined with machine learning techniques such as YOLOv10 and OpenCV, the system provides high detection accuracy. The incorporation of voice commands and audible feedback ensures accessibility, while the robust error handling features guarantee a seamless user experience. Furthermore, local processing ensures data privacy and security, making the system a trustworthy tool for visually impaired users.

11.1 Future Scope:

1. **Enhanced Dataset and Model:**
    - Expand the dataset to include more diverse currency notes, enhancing the system's robustness and accuracy.
    - Handle edge cases like folded, partially visible, or crumpled notes by training on more varied data.

2. **Cloud-Based Processing:**
    - Integrate cloud-based processing to improve scalability and performance, especially for lower-end devices.
    - Use cloud servers for more advanced computations, ensuring the system works optimally across various devices.

3. **Biometric Authentication:**
    - Implement biometric authentication such as fingerprint or voice recognition to personalize and secure user interactions.
    - Libraries like face_recognition or fingerprint can help with these features.

4. **Multi-Language Support:**

    - Expand the system to support multiple languages, making it accessible to a wider range of users.
    - Use text-to-speech libraries (e.g., gTTS) for multilingual support in user feedback.

5. **Mobile Application Development:**

    - Develop a mobile app to extend the system's accessibility, allowing users to carry the solution on-the-go.
    - The app could offer offline detection, voice commands, and integrate with banking APIs for real-time account balance checks.

6. **Integration with Banking APIs:**

    - Connect the system to banking APIs to enable real-time updates of account balances and transaction histories.
    - This would offer users a comprehensive view of their financial status directly through the system.

7. **Data Analytics and Reporting:**

    - Introduce data analytics features, enabling users to analyze their spending patterns and set financial goals.
    - Provide users with visualized data and insights about their currency usage and transactions.

8. **Compliance with Regulatory Standards:**

    - Ensure that the system complies with relevant regulatory standards, including GDPR and PCI DSS, to ensure user data protection and privacy.
    - This will improve user trust and encourage widespread adoption of the system.

9. **Enhanced Accessibility Features:**

    - Introduce haptic feedback (vibrations) for users with both visual and hearing impairments, providing multi-sensory interaction.
    - Improve compatibility with screen readers and other assistive technologies to ensure a more inclusive experience for all users.
    -

10. **Advanced Error Handling:**

- Implement AI-based error handling to automatically adjust for poor lighting, overlapping notes, or other detection challenges.
- Provide more context-aware error messages to guide users when issues arise.

11. **Potential Advanced Features:**

    - **Enhanced Transaction Management:** Integrate functionalities for bill payments or fund transfers if linked with banking systems.
    - **Wearable Device Integration:** Extend the system to wearable devices like smart glasses, providing hands-free currency detection.
    - **Offline Mode:** Develop an offline mode that allows users to detect currency without requiring an active internet connection.

11.2 Final Thoughts:

The **Currency Recognition System** showcases the power of **machine learning** and **real-time video processing** in creating accessible solutions for visually impaired individuals. With continuous advancements in accuracy, usability, and security, the system has the potential to empower users, enhancing their independence and confidence in handling currency. By addressing future challenges and expanding its features, the system could become an essential tool for visually impaired users, promoting greater inclusivity and accessibility in everyday financial tasks.

BIBLIOGRAPHY

The following sources have been referenced for the research, development, and implementation of the Automated Currency Recognition System, covering topics such as computer vision, deep learning, assistive technology, automation, security, and future advancements.

Computer Vision and Currency Recognition OpenCV. (n.d.). Open Source Computer Vision Library. Retrieved from https://opencv.org/

Redmon, J., & Farhadi, A. (2018). YOLO: Real-Time Object Detection. Retrieved from https://pjreddie.com/darknet/yolo/

RoboFlow. (n.d.). Training Custom Object Detection Models for Currency Recognition. Retrieved from https://roboflow.com/

Assistive Technology and Accessibility American Foundation for the Blind. (n.d.). Accessible Mobile Apps. Retrieved from https://www.afb.org/blindness-and-low-vision/using-technology/assistive-technology-products/mobile-apps

ResearchGate. (n.d.). A Survey on the Use of Mobile Applications for People who Are Visually Impaired. Retrieved from https://www.researchgate.net/publication/318885754_A_Survey_on_the_Use_of_Mobile_Applications_for_People_who_Are_Visually_Impaired

Automate by LlamaLab. (n.d.). Automating Tasks and Workflows on Android. Retrieved from https://llamalab.com/automate/

TensorFlow Lite. (n.d.). AI-Powered Mobile App Development. Retrieved from https://www.tensorflow.org/lite

Future Technologies and Advancements Brain-Computer Interface Society. (n.d.). Brain-Computer Interface (BCI) Technology. Retrieved from https://bcisociety.org/

IEEE Xplore. (n.d.). Deep Learning-Based Currency Recognition for Visually Impaired Individuals. Retrieved from https://ieeexplore.ieee.org/
This bibliography and reference list provides an overview of the academic, technical, and industry sources used in the research and development of this Automated Currency Recognition System project.