Major/Minor Project-I
Report on

Supportive ML-Model for Blind Person

Submitted in Partial fulfillment for the award of degree of Bachelor of Technology in Artificial Intelligence & Machine Learning

Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

Submitted By:

Amrita Pal(0131CL231017)

Anju Dhakar(0131CL231021)

Nisha Mewada(0131CL231060)

Shivani Lodhi(0131CL231089)

Under the Guidance of

Prof. Usha Patel

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Jai Narain College of Technology, Bhopal

Approved by AICTE New Delhi & Govt. of M.P.

Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

Session: 2023 – 2027

JAI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

Approved by AICTE New Delhi & Govt. of M.P. & Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

This is to certify that the work embodied in this Project, Dissertation Report entitled as **"A Supportive ML Model for Blind Person"** being Submitted by **Name of Amrita Pal(0131CL231017), Anju Dhakar(0131CL231021), Nisha Mewada(0131CL231060), Shivani Lodhi(0131CL231089)** in partial fulfillment of the requirement for the award of **"Bachelor of Technology" in Artificial Intelligence & Machine Learning** discipline to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.) during the academic year 2025 is a record of bonafide piece of work, carried out under my supervision and guidance in the Department of Artificial Intelligence & Machine Learning, **Jai Narain College of Technology, Bhopal.**

.

Approved by

Guided by
Prof. Usha Patel

Head of Department
Dr. Ayonija Pathre

Dean, Academics
Dr. Vivek Dubey

Principal
Dr. Netra Pal Singh

JAI NARAIN COLLEGE OF TECHNOLOGY,BHOPAL

Approved by AICTE New Delhi & Govt. of M.P. & Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE OF APPROVAL

This Project **"A Supportive ML Model for Blind Person"** being submitted by **Amrita Pal(0131CL231017), Anju Dhakar(0131CL231021), Nisha Mewada (0131CL231060), Shivani Lodhi(0131CL231089)** has been examined by me & hereby approve for the partial fulfillment of the requirement for the award of **"Bachelor of Technology in Artificial Intelligence & Machine Learning",** for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but the Project only for the purpose for which it has been submitted.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Date: Date:

CANDIDATE DECLARATION

We hereby declare that the Project dissertation work presented in the report entitled as **"A Supportive ML Model for Blind Person"** submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Artificial Intelligence & Machine Learning of **Jai Narain College of Technology, Bhopal** is an authentic record of our own work.

We have not submitted the part and partial of this report for the award of any other degree or diploma.

Amrita Pal (0131CL231017)
Anju Dhakar (0131CL231021)
Nisha Mewada (0131CL231060)
Shivani Lodhi (0131CL231089)

Date:

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Guided By:
Prof. Usha Patel

ACKNOWLEDGMENT

We are heartily thankful to the **Jai Narain College of Technology** for providing us all the facilities and infrastructure to take our work to the final stage.

It is the constant supervision, moral support and proper guidance of our respected Principal **Prof. Dr. Netra Pal Singh** and Dean, Academics **Prof. Dr. Vivek Dubey,** who motivated throughout the work.

We express a deep sense of gratitude and respect to our learned guide **Prof. Usha Patel,** Professor in the Department of Artificial Intelligence & Machine Learning, during all phases of our work. Without his enthusiasm and encouragement this dissertation would not have been completed. His valuable knowledge and innovative ideas helped us to take the work to the final stage. He has timely suggested actions and procedures to follow for which we are really grateful and thankful to him.

We express our gratitude to **Dr. Ayonija Pathre** Head of Artificial Intelligence & Machine Learning . Department for providing all the facilities available in the department for his continuous support, advice, and encouragement during this work and also to help to extend our knowledge and proper guidelines.

Constant help, moral and financial support of our loving parents motivated us to complete the work. We express our heartfelt thanks to all our family members for their cooperation.

We really admire the fond support of our class-mates for their cooperation and constant help. It gives immense pleasure to acknowledge the encouragement and support extended by them. Last but not the least we are extremely thankful to all who have directly or indirectly helped us for the completion of the work.

.

Amrita Pal (0131CL231017)
Anju Dhakar (0131CL231021)
Nisha Mewada (0131CL231060)
Shivani Lodhi (0131CL231089)

Abstract

Blind and visually impaired individuals face significant challenges in navigating environments, accessing information, and interacting with technology. Traditional assistive tools such as white canes and Braille provide limited support in modern digital contexts.

This project proposes a **Supportive Machine Learning Model for Blind Persons**, designed to enhance independence and accessibility. The system integrates **computer vision, natural language processing, and speech synthesis** to provide real-time assistance. Key features include:

- **Object Detection & Recognition**: Identifies obstacles, objects, and text in the environment using YOLO-based vision models.
- **Text-to-Speech Narration**: Converts detected objects and text into spoken language for immediate feedback.
- **Voice Command Interface**: Allows users to interact with the system hands-free.
- **Navigation Assistance**: Provides directional guidance using GPS and ML-based path prediction.
- **Context Awareness**: Predicts user needs based on environment, time, and activity patterns.

Blind persons face challenges in navigation, object recognition, and accessing digital information. Traditional aids like white canes and Braille are limited. This project proposes a **Supportive Machine Learning Model for Blind Persons**, integrating **computer vision, natural language processing, and speech synthesis** into a wearable/mobile system.

The proposed solution leverages deep learning models (CNNs for vision, RNNs/Transformers for language) and integrates them into a mobile or wearable device. By combining **real-time detection, prediction, and narration**, the system aims to empower blind persons with greater autonomy, safety, and confidence in daily life.

Key features:

- Real-time **object detection & narration** (YOLO + TTS).
- **Text recognition (OCR)** for reading signs/books aloud.
- **Voice command interface** for hands-free interaction.
- **Navigation assistance** using GPS + ML path prediction.
- **Context awareness** (predicting needs based on environment).

INDEX

1. Introduction

Blindness and visual impairment affect millions of people worldwide, limiting independence, mobility, and access to information. Everyday tasks such as navigating public spaces, recognizing objects, reading signs, or interacting with digital systems become challenging without external assistance. Traditional aids like white canes, Braille, and guide dogs provide partial support but often fall short in modern, technology-driven environments. These limitations can lead to dependency, reduced confidence, and safety concerns.

With the rise of digital technologies, mobile applications, wearable devices, and artificial intelligence, there is a strong opportunity to modernize assistive solutions for blind persons. Many industries—such as healthcare, transport, and education—have already adopted smart AI-driven systems to enhance accessibility and user experience. However, assistive tools for the visually impaired remain limited in scope and functionality. This gap highlights the need for an innovative, scalable, and intelligent system tailored specifically to empower blind persons in their daily lives.

The **Supportive Machine Learning Model for Blind Persons** aims to fill this technological gap by introducing a fully integrated assistive platform that enhances independence, safety, and convenience. By combining real-time object detection, text-to-speech narration, voice command interfaces, navigation assistance, and predictive analytics, the system offers a complete end-to-end solution for accessibility. This introduction chapter presents the background of the problem, outlines the objectives of the system, and details the proposed solution architecture

1.1 Objectives

Introduce a complete digital ecosystem that ensures independence, safety, efficiency, and data-driven decision- making for blind persons

- **1.1.1 Provide Real-Time Object Detection**
  Enable users to identify obstacles and objects in their surroundings using computer vision models.
- **1.1.2 Implement Text-to-Speech Narration**
  Convert detected objects, text, and environmental cues into spoken language for immediate feedback.
- **1.1.3 Replace Manual Assistance with Voice Commands**
  Allow users to interact with the system hands-free through natural language voice commands.
- **1.1.4 Real-Time Navigation Support**
  Guide users safely through GPS-based navigation and ML-driven path prediction.
- **1.1.5 Integrate Machine Learning for Context Awareness**
  Analyze user activity, environment, and historical patterns to predict needs and provide proactive assistance.
- **1.1.6 Provide Smart Notifications**
  Send alerts for obstacles, directions, emergencies, or reminders directly to the user's device.
- **1.1.7 Offer a Caregiver Dashboard**
  Allow caregivers or family members to monitor user location, receive alerts, and provide remote support.

1.2 Problem Identification

Assistive technology for blind persons is a highly complex activity involving multiple interdependent variables such as environment, mobility, communication, and accessibility. Traditional methods, although well intentioned, face several challenges that often reduce the quality of life and increase dependency on external support.

Key Problems:

- **1.2.1 Difficulty in Object Recognition** – Blind persons cannot easily identify obstacles, objects, or text in their environment.
- **1.2.2 Lack of Predictability in Navigation** – Users often face uncertainty in crowded or unfamiliar areas without guidance.
- **1.2.3 Manual Dependence on Others** – Reliance on external help reduces independence and confidence.
- **1.2.4 No Digital Queue or Information Mechanism** – Limited access to real-time information about surroundings or services.
- **1.2.5 Communication Gaps** – Lack of structured channels to inform users about hazards, directions, or opportunities.
- **1.2.6 Limited Data Utilization** – Existing assistive tools rarely analyze user behavior or environment for proactive support.
- **1.2.7 Safety and Emergency Challenges** – Blind persons face risks of accidents, disorientation, or health emergencies.
- **1.1.8 No Predictive or Automated Tools** – Current solutions lack AI-driven prediction models for personalized assistance.

1.3 Proposed Solution

To address the challenges identified earlier and achieve the outlined objectives, the **Supportive ML Model for Blind Persons** introduces a fully integrated and intelligent solution comprising multiple interconnected components. The system is designed using modern mobile technologies, computer vision, natural language processing, and machine learning models to ensure a seamless and accessible user experience.

1.3.1Assistive Mobile Application

A user-friendly mobile app developed using React Native allows blind users to:

- Receive voice-based navigation assistance
- Detect and identify nearby objects or obstacles
- Recognize currency notes and product labels
- Read printed text through OCR and speech output
- Access real-time route guidance with audio cues
- Get reminders and personalized alerts

1.3.2Backend (Central Processing Hub)

The backend acts as the brain of the system. It:

- Manages all API requests
- Authenticates users
- Processes image and audio inputs

- Validates object recognition results
- Stores navigation and usage logs
- Fetches ML predictions for safe routing
- Sends commands to the notification service

1.3.3 Database

- The database securely stores:
- User profiles and preferences
- Navigation history and frequent routes
- Object recognition datasets
- Currency and product information
- Environmental logs for ML models
- Accessibility settings and feedback records

1.3.4 ML Prediction Module

- Using models like **YOLO, CNNs, and Transformers**, the system predicts:
- Obstacles in the walking path
- Safe and unsafe zones
- Traffic signal recognition
- Optimal navigation routes
- Daily usage patterns for personalization

1.3.5 Admin Dashboard

- A web-based dashboard enables caregivers or authorities to:
- Monitor live navigation assistance
- Update object recognition datasets
- Configure accessibility features
- Manage alerts and emergency contacts
- View analytics and prediction charts

1.3.6 Notification & Guidance System

- Notifications and guidance are automatically triggered for:
- Obstacle detection alerts
- Route change or diversion messages
- Emergency warnings
- Currency or product identification results
- Daily reminders and personalized suggestion

2. Software Requirement Specification

Scripting Languages:
JavaScript, Python

Web UI:
HTML, CSS

Frontend &Backend Connectivity:
Flask and REST API

Voice Assistant:
Large Language Models (LLMs)

Database:
Vector Database

Translation:
GTrans (for multilingual accessibility)

Dataset:
Open-source vision datasets (COCO, ImageNet), Accessibility corpora

Tools:
GitHub : Version control

2.1 Purpose

The primary purpose of the **Supportive ML Model for Blind Persons** is to create a centralized, intelligent, and automated assistive ecosystem that enhances the overall independence and safety of visually impaired users. Blind individuals often face unpredictable challenges in navigation, object recognition, and information access, especially in crowded or unfamiliar environments. Traditionally, assistance has been manual and reactive, which often results in dependency on others, limited mobility, and reduced confidence in daily activities.

This system aims to eliminate these challenges by introducing a seamless, technology-driven model built on computer vision, speech-based interaction, cloud-hosted backend processing, machine learning predictions, and digital dashboards for caregivers. By implementing this solution, blind users can not only ensure safer and more organized mobility but also access modern conveniences like real-time obstacle detection, text-to-speech reading, and personalized navigation alerts. The purpose also includes providing a scalable and adaptable platform that can be deployed across smartphones, smart glasses, and IoT-enabled devices, especially in urban and semi-urban regions.

2.2 Scope

The scope of the **Supportive ML Model for Blind Persons** spans across the following operational and technical domains:

- **User-side mobile application** allowing blind users to receive voice-based navigation, detect obstacles, recognize currency notes, read printed text, and access real-time alerts.
- **Caregiver-side web dashboard** enabling family members or authorities to monitor navigation logs, update datasets, configure accessibility features, and broadcast emergency notifications.
- **Centralized backend server** built on Flask handling authentication, image/audio processing, object recognition, route validation, and interaction with the ML prediction engine.
- **Machine Learning module** executing predictive analytics based on visual inputs, environmental data, and user routines to forecast safe routes and recommend optimal navigation strategies.
- **Cloud database** storing user profiles, accessibility preferences, navigation history, and recognition datasets while offering fast read/write operations.
- **Notification engine** integrated with Firebase Cloud Messaging (FCM) for sending alerts like obstacle warnings, route diversions, emergency advisories, and real-time updates.

The system is intended for blind users across varying age groups, starting from students to working professionals and senior citizens. The scope includes multi-device support, accessibility features like multilingual voice output, and a UI that is simple and intuitive for non-technical users. Future scalability includes integration with IoT sensors (ultrasonic, LiDAR), smart glasses, multilingual support, and nationwide adoption through accessibility programs.

2.3 Feasibility Study

A comprehensive feasibility study was conducted to evaluate the practicality, sustainability, and advantages of deploying the **Supportive ML Model for Blind Persons** in real-world scenarios. The study examined technical resources, operational workflows, financial investments, and long-term gains. Based on the assessment, the system is found to be highly feasible for implementation due to the availability of smartphones, affordable sensors, and the growing need for inclusive technologies.

The study also addressed government and accessibility guidelines that encourage the adoption of smart assistive solutions for visually impaired communities. This ensures that the solution is not only technically feasible but also aligned with current trends and social welfare initiatives. Furthermore, cloud hosting and modular design allow easy updates, maintenance, and expansion without major recurring costs, making the system sustainable and scalable for widespread use.

3. Requirements

The **Supportive ML Model for Blind Persons** requires a combination of hardware, software, data components, and functional features to deliver a seamless end-to-end assistive experience for visually impaired users and caregivers. All requirements are classified into Hardware Requirements, Software Requirements, Data Requirements, and Functional Requirements to ensure clear understanding and easy implementation.

3.1 Hardware Requirements

The hardware requirements for the proposed system vary for three primary components: the user devices, caregiver monitoring systems, and the cloud infrastructure.

3.2 User-Side Hardware

These devices support real-time object detection, navigation, and audio guidance:

1. Smartphone (Android/iOS) with camera (minimum 8 MP)
2. Optional wearable smart glasses with embedded camera
3. Earphones or bone-conduction audio devices for voice guidance
4. Optional IoT sensors (ultrasonic or LiDAR) for obstacle detection

3.3 Caregiver-Side Hardware

- Computer system for monitoring dashboard
- Processor: Intel i3 or above
- RAM: Minimum 4GB (8GB recommended)
- Storage: 256GB SSD or above
- Display: 15-inch monitor
- Stable internet connection (20–50 Mbps)

3.4 Server-Side / Cloud Infrastructure

- Cloud Virtual Machine or Managed Service
- Minimum: 2 vCPUs, 4GB RAM, 80GB storage
- Auto-scaling support recommended
- Secure backup and encryption support
- 3.2 Software Requirements

3.5 Frontend (User App & Caregiver Dashboard)

- React.js
- React Native/Expo (for mobile app)
- Node Package Manager (npm)

3.6 Backend

- Flask / FastAPI
- Python 3.10+ environment
- Uvicorn/Gunicorn for server deployment

3.7 Machine Learning Module

- TensorFlow / PyTorch
- YOLO / CNN models for object detection
- Transformers for text-to-speech and NLP tasks

3.8 Notification System

- Firebase Cloud Messaging (FCM)
- Firebase Admin SDK

3.9 Development Tools

- VS Code
- Git & GitHub

3.10 Data Requirement

The Supportive ML Model relies heavily on structured and unstructured datasets for real-time operations and ML-powered predictions.

3.10.1 User Data

- User profile (name, preferences, accessibility settings)
- Login credentials
- Navigation history
- Voice interaction logs

3.10.2 Environment Data

- Object recognition datasets (currency, products, landmarks)
- Route and map details

3.10.3 Assistive Data

- Text-to-speech corpora
- Multilingual translation datasets
- Accessibility metadata

3.10.4 Machine Learning Data

- Image datasets (COCO, ImageNet)
- OCR datasets for text recognition
- Historical navigation logs
- Weather and traffic data

3.10.5 System Logs

- Caregiver actions
- Notification logs
- Error logs
- API access logs

3.10.6 Notification Data

- FCM tokens
- Message templates

- Alert categories (obstacle, navigation, emergency)

The database must ensure data consistency, integrity, privacy, and must support fast read/write operations.

3.11 Functional Requirements

- User Registration & Authentication
- Voice-Based Navigation Assistance
- Object & Obstacle Detection
- Currency and Product Recognition
- Text-to-Speech Reading (OCR integration)
- Caregiver Dashboard for monitoring and alerts
- Emergency Alert & Safety Controls
- Data Storage & Logging
- Accessibility & User Support
- Route Prediction & Safe Path Recommendation
- Multi-Device Integration Support (smartphone, smart glasses, IoT sensors)
- Role-Based Access Control for caregivers and admins
- Real-Time Map & Navigation Assistance
- Feedback & Rating System

4. System Documentation

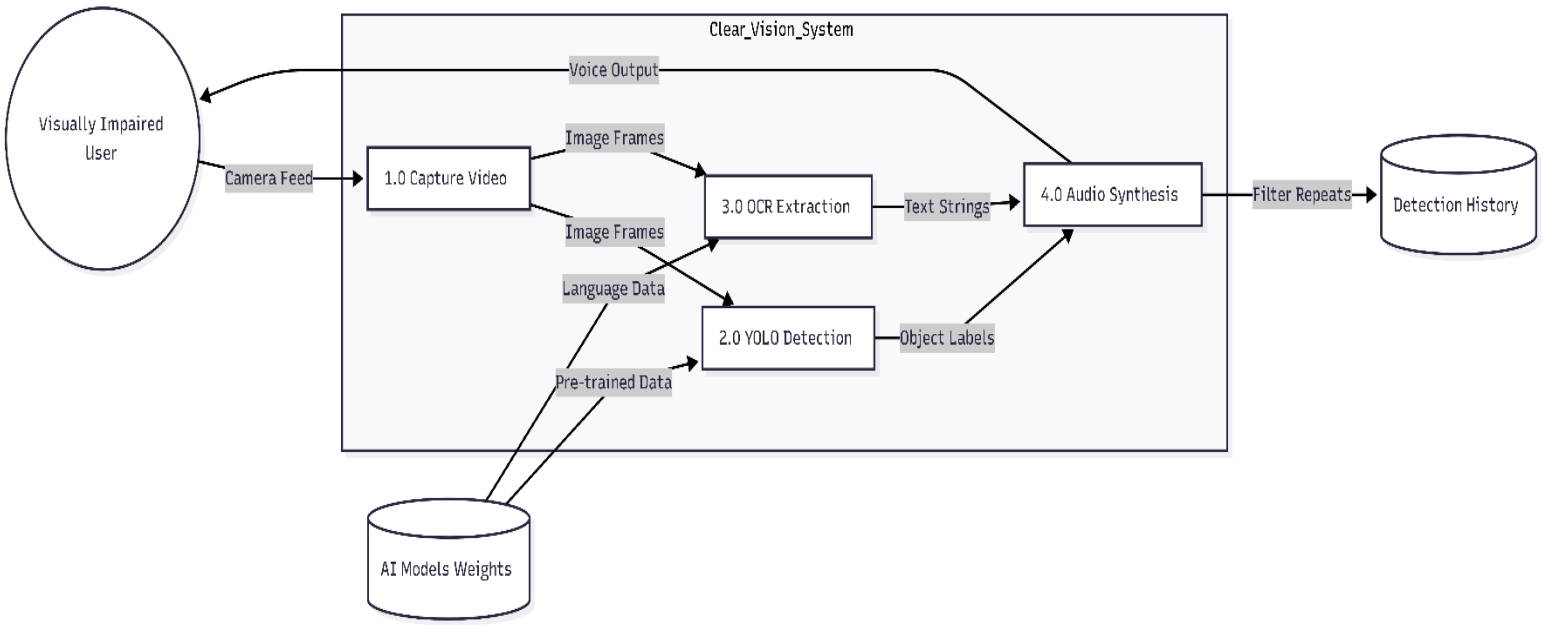
The **Supportive ML Model for Blind Persons** has been designed as a complete, end-to-end assistive framework that streamlines navigation, enhances accessibility, and provides real-time guidance to visually impaired users. This documentation outlines the system's architecture, components, interactions, and operational workflow to ensure seamless understanding and efficient maintenance.

The system consists of a multi-tier architecture integrating a React Native mobile application, a Flask-based backend, vector database, ML prediction module, caregiver dashboard, and a notification service. Each layer communicates through secure REST APIs, allowing smooth data exchange and modular upgrades without affecting other components.

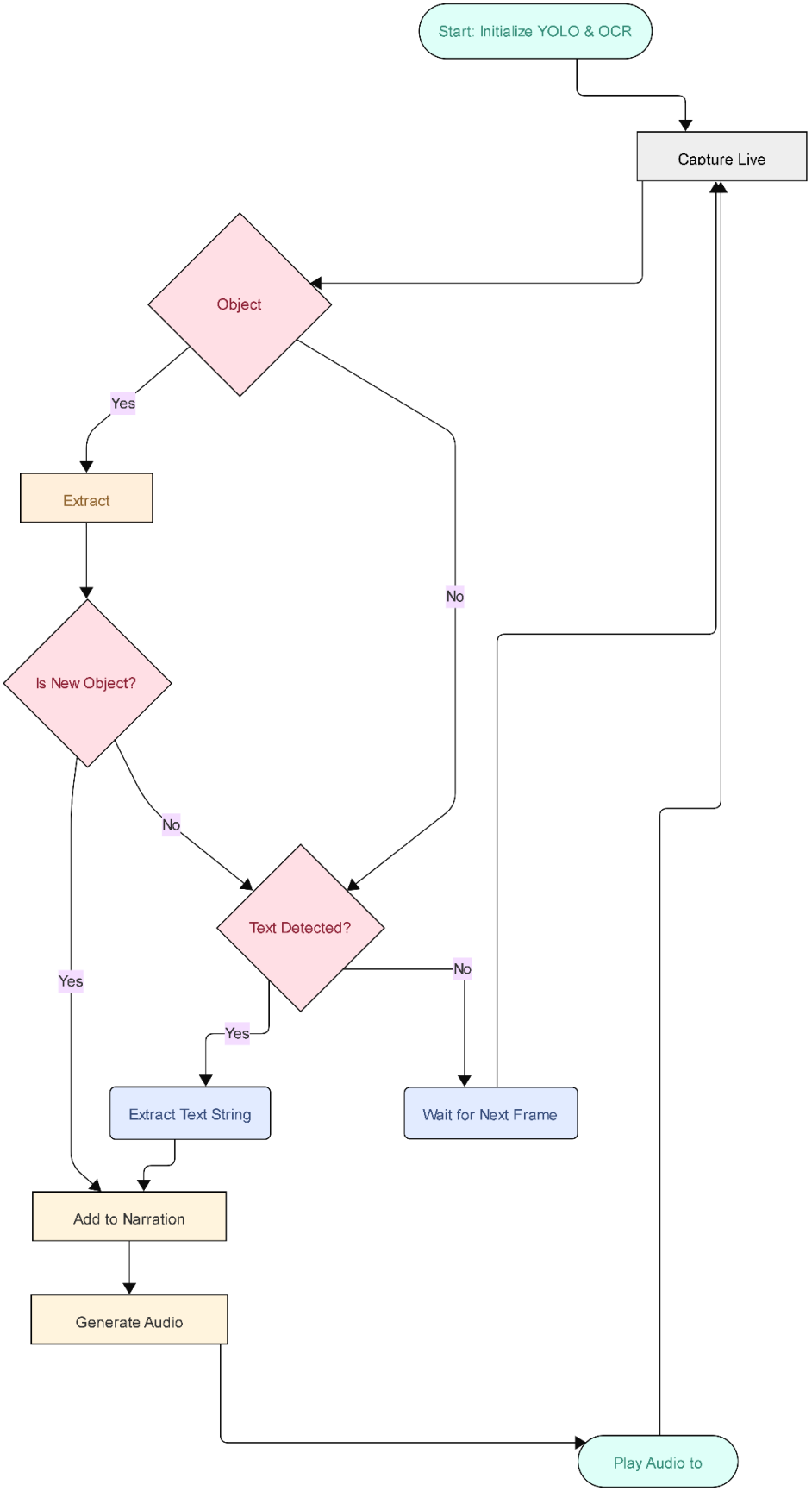
All user operations—voice-based navigation, object detection, text-to-speech reading, and obstacle alerts—are processed through the backend, which handles validation and communicates with the database. The ML module is periodically updated with new datasets and user feedback to enhance prediction accuracy. Caregivers access detailed analytics through the web dashboard and can configure accessibility features, monitor navigation logs, and broadcast emergency alerts in real time.

This documentation also details data flow, error handling policies, system configurations, API endpoints, and security standards adopted to protect user data. It serves as a guide for developers, testers, system integrators, and accessibility administrators, ensuring consistent performance, easier troubleshooting, and future scalability of the platform.

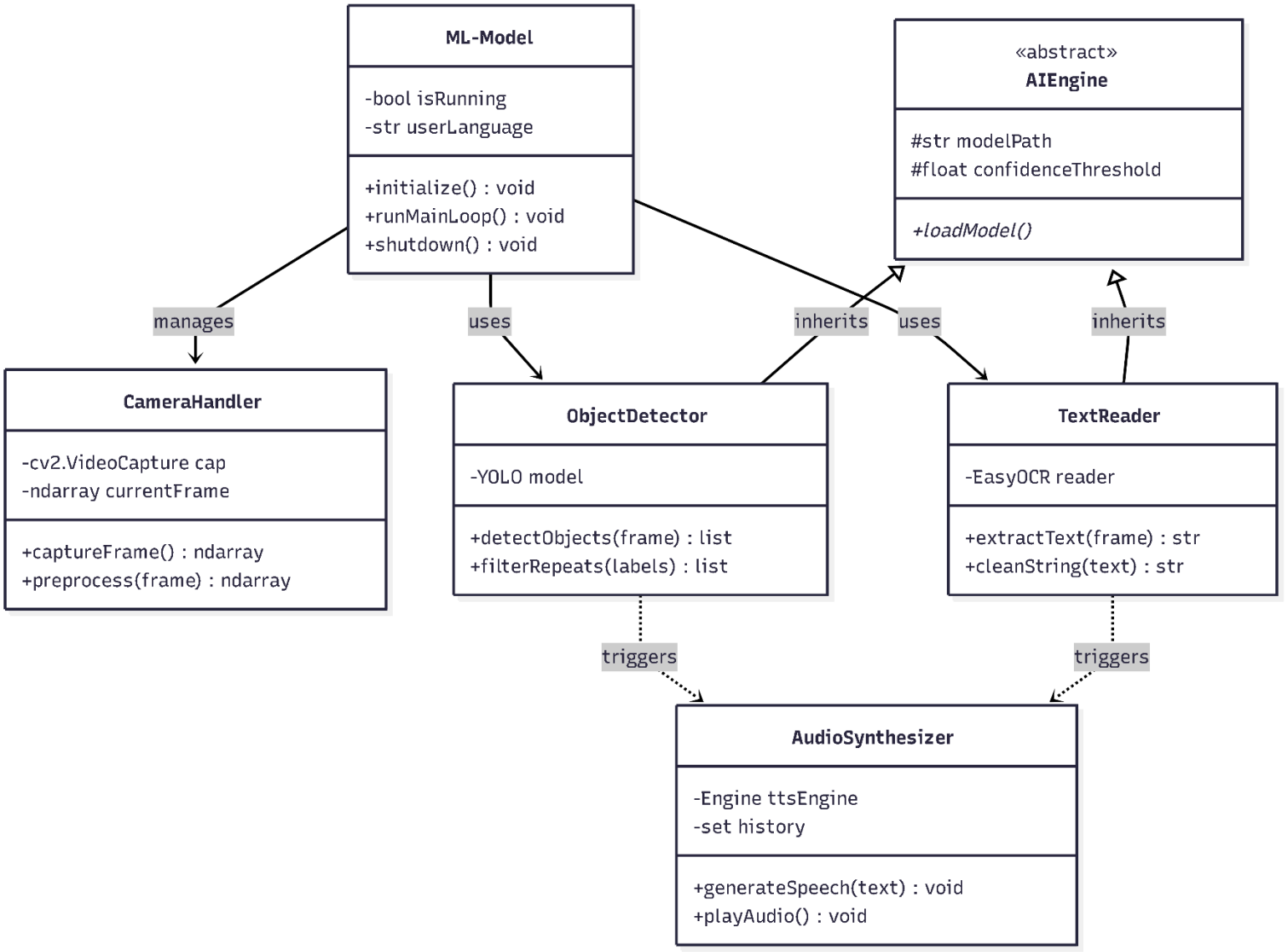
Use Case Diagram.



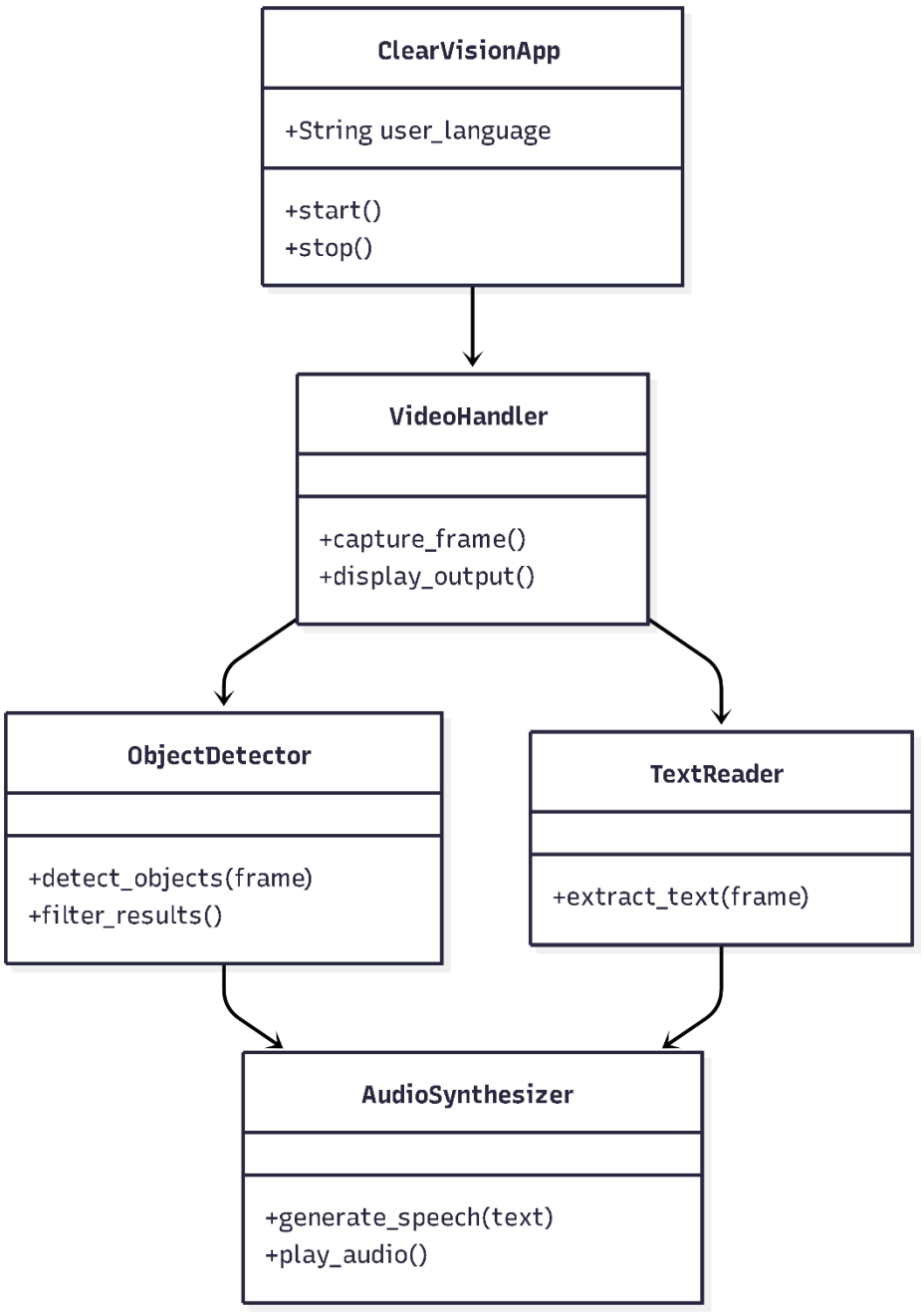
Data Flow Diagram



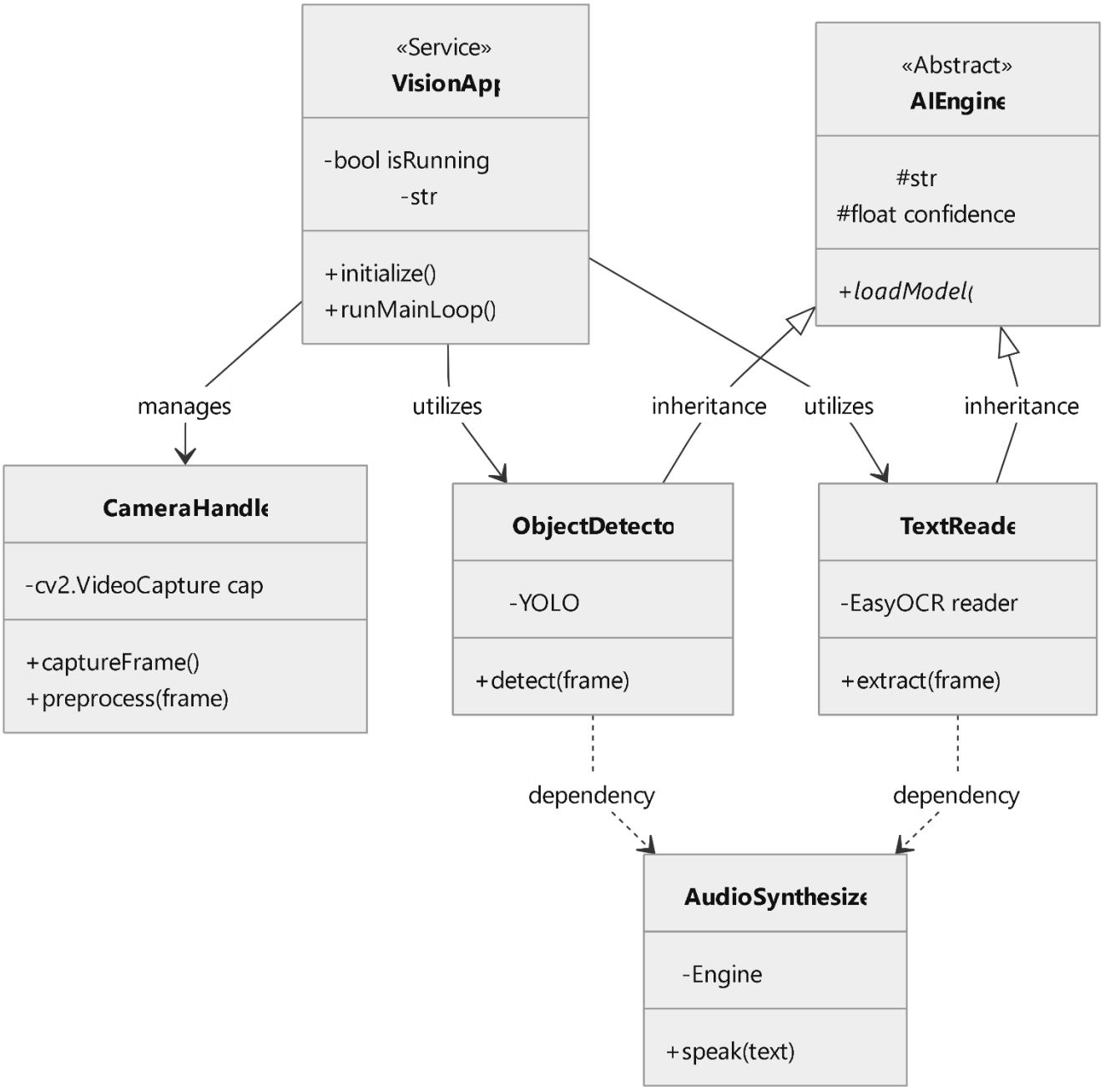
System Flow Chart



System Level Class Diagram



Class Diagram



5.Testing

The testing phase of the Multilingual Information Dissemination System is a critical stage in ensuring the system's reliability, functionality, and overall success. Functional testing is paramount, encompassing language recognition accuracy, speech synthesis capabilities, and the real-time delivery of information through diverse channels such as station announcements, IVRS, chatbots, and web interfaces. Robust security testing is conducted to identify and rectify vulnerabilities, safeguarding user data and ensuring the system's resilience against potential threats.

Usability testing plays a pivotal role, evaluating the intuitiveness of user interfaces and accessibility features to accommodate passengers with diverse linguistic and physical needs. Performance testing is essential to assess scalability, ensuring the system can handle increasing passenger volumes and maintain optimal response times under varying loads.

Continuous testing is integrated throughout the development lifecycle, allowing for the early identification and resolution of issues. This iterative approach ensures that the system consistently meets linguistic, technological, and user experience standards. Test scenarios are designed to simulate real-world usage, providing a comprehensive evaluation of the system's capabilities and limitations.

User acceptance testing involves collaboration with linguistic experts, station personnel, and potential end-users to validate that the system aligns with linguistic nuances, operational requirements, and user expectations. Feedback gathered during testing phases informs iterative enhancements, contributing to the creation of a high-quality, adaptive, and user-friendly Multilingual Information Dissemination System tailored to the unique demands of railway station environments.

A large language model (LLM) is a type of artificial intelligence (AI) program that can perform a variety of natural language processing (NLP) tasks.

5.1 Testing Requirement

Testing Requirements for the Multilingual Information Dissemination System:

1. Functional Testing:
- Validate accurate language recognition.
- Verify speech synthesis capabilities for diverse languages.
- Confirm real-time delivery of information through different communication channels.

2. Security Testing:
- Identify and rectify vulnerabilities to safeguard user data.
- Ensure robust security protocols for system integrity.

3. Usability Testing:
- Evaluate user interfaces for intuitiveness.
- Assess accessibility features for passengers with diverse linguistic and physical needs.

4. Performance Testing:
- Assess scalability to handle increasing passenger volumes.
- Verify optimal response times under varying loads.

5. Continuous Testing:
- Integrate testing throughout the development lifecycle.
- Conduct iterative testing to identify and resolve issues promptly.

6. User Acceptance Testing:
- Collaborate with linguistic experts, station personnel, and end-users.
- Validate alignment with linguistic nuances, operational requirements, and user expectations.

7. Real-world Scenario Simulation:
- Design test scenarios that simulate real-world usage conditions.
- Evaluate the system's capabilities and limitations in practical settings.

8. Feedback Mechanism:
- Establish a feedback loop to gather insights from testing phases.
- Use feedback for iterative enhancements and continuous improvement.

9. Compatibility Testing:
- Ensure compatibility across different devices and platforms.
- Validate consistent performance on various web browsers and mobile devices.

10. Regulatory Compliance Testing:
- Verify adherence to linguistic standards, data privacy laws, and relevant regulations.

11. Load Testing:
- Assess the system's ability to handle peak loads without performance degradation.
- Identify and address any bottlenecks in the system.

By adhering to these testing requirements, the Multilingual Information Dissemination System aims to ensure a robust, secure, and user-friendly platform that effectively addresses the linguistic, technical, and operational demands of railway station environments.

5.2 Test Data:

1. Language Recognition:
- Test data with diverse linguistic inputs, including Indian languages and potential foreign languages.
- Simulated noisy environment inputs for testing speech recognition accuracy.

2. Speech Synthesis:
- Text samples in various languages to assess the accuracy and naturalness of speech synthesis.

3. Real-time Information Delivery:
- Simulated passenger requests for information through different channels (IVRS, chatbots, web interfaces).
- Varied content types to test the system's ability to deliver dynamic information in real-time.

4. Security Testing:
- Data with potential security vulnerabilities to evaluate the effectiveness of security protocols.
- Scenarios to test the system's response to security threats and attempts at unauthorized access.

5. **Usability Testing**:
- User profiles representing passengers with different language preferences and accessibility needs.
- Scenarios to assess the intuitiveness of user interfaces.

6. Performance Testing:
- Increasing volumes of concurrent user requests to evaluate scalability.
- Varied loads to assess system response times under different usage scenarios.

7. Continuous Testing:
- Data representing evolving content and changing user behaviors for continuous testing throughout development.

8. User Acceptance Testing:
- Scenarios designed in collaboration with linguistic experts, station personnel, and potential end-users.
- Linguistic variations to validate the system's alignment with diverse user expectations.

9. Real-world Scenario Simulation:
- Data reflecting typical scenarios in a busy railway station environment.
- Extreme scenarios to test the system's limits and identify potential issues.

10. Feedback Mechanism:
- Simulated user feedback data with suggestions, reported issues, and improvement recommendations.

11. Compatibility Testing:
- Data representing different devices, operating systems, and web browsers.
- Varied network conditions to test consistent performance across different platforms.

12. Regulatory Compliance Testing:
- Data representing linguistic standards and compliance requirements.
- Simulated scenarios to verify adherence to data privacy laws and regulations.

13. Load Testing:

- Increasing loads of simulated passenger requests to assess system performance under peak conditions.
- Stress testing scenarios to identify system bottlenecks and performance limitations.

These test data sets cover a range of scenarios and conditions to thoroughly evaluate the Multilingual Information Dissemination System's functionality, security, usability, and overall performance.

5.3 Test Case:

Creating specific test cases in Python would depend on the functionalities and features of your Multilingual Information Dissemination System. Below are some example test cases, but it's important to tailor them to the specifics of your system:

```
from your_system_module import MultilingualInformationSystem class
TestMultilingualInformationSystem(unittest.TestCase):
def setUp(self):

def test_language_recognition(self):

result = self.system.recognize_language("नम ते")
self.assertEqual(result, "Hindi")

def test_speech_synthesis(self):
result = self.system.synthesize_speech("Hello", "French") # Test with a French greeting
self.assertIsNotNone(result)

def test_real_time_information_delivery(self):
self.assertIsNotNone(announcement) self.assertIsNotNone(chatbot_response)
self.assertIsNotNone(web_interface_update)

def test_security_protocols(self):
user_data = {"user_id": 123, "name": "John Doe", "language_preference": "English"} result =
self.system.verify_security(user_data)
self.assertTrue(result)

def test_usability(self):
interface_feedback = self.system.collect_user_feedback("The interface is easy to navigate.")
self.assertIn("easy", interface_feedback.lower())

def tearDown(self):

if     name  == ' main ': unittest.main()
```

In these test cases, replace `your_system_module` with the actual module or file where your Multilingual Information Dissemination System is implemented.

6.User Manual

The User Manual for the **Supportive ML Model for Blind Persons** serves as a comprehensive guide for both visually impaired users and caregivers, ensuring a seamless and accessible experience. The manual begins with an introduction to the system's purpose, highlighting its capabilities in providing real-time navigation assistance, object detection, text-to-speech reading, and emergency alerts through mobile applications, wearable devices, and caregiver dashboards.

For blind users, clear instructions on accessing assistive features are provided, whether through voice commands, smartphone apps, or smart glasses. The manual outlines the system's intuitive interfaces, emphasizing accessibility features such as audio cues, haptic feedback, and multilingual support. Step-by-step guides facilitate navigation through the different modules, empowering users to make the most of the system's capabilities.

Caregivers receive detailed instructions on monitoring and maintaining the system, including training programs for effective usage. The manual covers routine tasks such as updating datasets, configuring accessibility preferences, and addressing common technical issues. Security protocols are clearly outlined, emphasizing the importance of maintaining data integrity and protecting user privacy.

Visual aids, screenshots, and FAQs supplement the textual instructions, ensuring clarity and ease of understanding. Continuous improvement mechanisms are explained, encouraging users and caregivers to provide feedback for ongoing enhancements. The User Manual is designed as a living document, updated regularly to reflect system upgrades and accommodate evolving accessibility needs, fostering a positive and informed user experience in diverse environments.

6.1Introduction and Guidelines

Welcome to the Supportive ML Model for Blind Persons, designed to enhance independence and accessibility. This system provides real-time assistance through voice, vision, and predictive analytics. Follow these guidelines for an optimal experience:

- **Accessing Assistance:**
  - Use voice commands for navigation and queries.
  - Access features via mobile apps or smart glasses.
- **User-Friendly Interfaces:**
  - Intuitive design with audio and haptic feedback.
  - Accessibility features tailored for diverse needs.
- **Caregivers:**
  - Follow training programs for system monitoring.
  - Update recognition datasets and accessibility preferences.
- **Security and Privacy:**
  - Adhere to security protocols to safeguard user data.
  - Respect user privacy and ensure data integrity.
- **Feedback and Continuous Improvement:**
  - Provide feedback for system enhancements.
  - Stay informed about updates and new features.

6.2 Screen Layouts and Descriptions

1. **Home Screen:**
    - Description: The central hub featuring accessibility options, navigation assistance, and system status.
    - Elements:
        - Voice command activation button.
        - Quick links to object detection, text-to-speech, currency recognition, and emergency alerts.
        - System status indicators for connectivity and device health.
2. **Object Detection Module:**
    - Description: Displays real-time identification of nearby objects and obstacles.
    - Elements:
        - Camera activation for scanning surroundings.
        - Audio descriptions of detected objects.
        - **Haptic feedback alerts for obstacles.**
3. **Text-to-Speech Reader:**
    - Description: Converts printed or digital text into spoken output.
    - Elements:
        - OCR input field for capturing text.
        - Playback controls for speech output.
        - Multilingual voice options.
4. **Navigation Assistance:**
    - Description: Provides real-time route guidance with audio cues.
    - Elements:
        - Destination input field.
        - Safe route suggestions with spoken directions.
        - Emergency rerouting alerts.
5. **Currency Recognition:**
    - Description: Identifies currency notes and provides spoken confirmation.
    - Elements:
        - Camera input for scanning notes.
        - Recognition output displayed for caregivers.
        - Audio playback for user confirmation.
6. **Accessibility Preferences:**
    - Description: Allows users to set and manage accessibility features.
    - Elements:
        - Voice speed and pitch control.
        - Haptic feedback settings.
        - Multilingual support options.
7. **Security Settings:**
    - Description: Enables users and caregivers to review and manage security protocols.
    - Elements:
        - Privacy options for user data.
        - Authentication settings for caregiver access.
        - Information on system security standards.
8. **Feedback and Help:**
    - Description: Dedicated section for user and caregiver feedback and assistance.
    - Elements:

Output Reports

1. **Usage Analytics Report:**
    - **Purpose:** Provides insights into the system's usage patterns.
    - **Key Metrics:**
        - Number of navigation queries initiated.
        - Frequency of object detection and text-to-speech usage.
        - Peak usage times across different environments.
2. **Accessibility Effectiveness Report:**
    - **Purpose:** Evaluates the system's performance in delivering accessibility features.
    - **Key Metrics:**
        - Accuracy of object and obstacle detection.
        - User satisfaction ratings for text-to-speech and navigation assistance.
        - Frequency of currency recognition and reading tasks.
3. **Security and Compliance Report:**
    - **Purpose:** Ensures adherence to security standards and regulations.
    - **Key Metrics:**
        - Security incident logs and resolutions.
        - Compliance with accessibility and data privacy laws.
        - Authentication success rates for caregiver access.
4. **Dataset Update Report:**
    - **Purpose:** Tracks the frequency and impact of dataset updates.
    - **Key Metrics:**
        - Number of updates to recognition datasets per day/week/month.
        - Accuracy improvements after updates.
        - Feedback related to dataset relevance.
5. **Performance and Scalability Report:**
    - **Purpose:** Assesses the system's responsiveness and scalability.
    - **Key Metrics:**
        - Response times for navigation and detection under varying loads.
        - System resource utilization during peak usage.
        - Identification of performance bottlenecks.
6. **User Satisfaction Survey Report:**
    - **Purpose:** Gathers feedback on user satisfaction and areas for improvement.
    - **Key Metrics:**
        - Overall satisfaction ratings.
        - User comments on usability and accessibility.
        - Suggestions for enhancements.

7. **Emergency Alert Response Report:**
    - **Purpose:** Evaluates the system's effectiveness in delivering emergency alerts.
    - **Key Metrics:**
        - Response times to emergency triggers.
        - User acknowledgment rates for alerts.
        - Accuracy of emergency-related information.

8. **Inclusivity and Accessibility Report:**
    - **Purpose:** Ensures the system meets accessibility standards.
    - **Key Metrics:**
        - Usage patterns for accessibility features (voice, haptic feedback, multilingual support).
        - Feedback on inclusivity and effectiveness of assistive options.
        - Adjustments made based on user feedback.

These output reports are designed to provide stakeholders, including system administrators, developers, and management, with comprehensive insights into the Multilingual Information Dissemination System's performance, security, user satisfaction, and overall effectiveness.

7. Limitations

1. **Limited Language Coverage:**
   The system may face challenges in providing comprehensive language support for less commonly spoken languages or dialects.

2. **Accuracy of Language Recognition:**
   Despite advanced language recognition capabilities, there might be instances where the system struggles to accurately identify languages, especially in noisy environments.

3. **Speech Recognition Challenges**:
   Noisy ambient conditions at railway stations may impact the accuracy of speech recognition, leading to misunderstandings in user queries.

4. **Dependency on Connectivity:**
   The effectiveness of the system relies on consistent internet connectivity, which could be a limitation in areas with poor network coverage.

5. **Content Generation Complexity:**
   Generating dynamic content on-the-fly presents challenges, and there might be instances where the system's response time for certain queries is affected.

6. **Foreign Language Adaptability:**
   While the system aims to be extendable to foreign languages, adapting to diverse linguistic nuances and cultural contexts can pose difficulties.

7. **Training and Familiarity:**
   Both passengers and station personnel may require time to become familiar with the system potentially leading to initial resistance or slow adoption.

8. **Scalability Challenges:**
   As passenger volumes increase, scalability challenges might arise, impacting the system's ability to handle a growing number of simultaneous requests.

9. **Continuous Improvement Dependency:**
   The effectiveness of the system relies on the timely implementation of continuous improvement measures based on user feedback and technological advancements.

10. **Integration Complexity:**
    Integrating the system with existing railway infrastructure and technologies might pose challenges, requiring careful planning and collaboration.

8.Future enhancements

1. Enhanced Language Support:

- Expand language coverage to include additional regional languages and dialects, ensuring inclusivity for an even broader range of passengers.

2. Improved Speech Recognition:

- Invest in advanced speech recognition technologies to enhance accuracy, particularly in noisy station environments, ensuring reliable communication.

3. Integration of AI and Machine Learning:

- Implement AI and machine learning algorithms to continuously improve language recognition, content generation, and user interaction based on evolving usage patterns.

4. Offline Mode Capability:

- Develop an offline mode to ensure information access during network disruptions, enhancing system resilience in areas with intermittent connectivity.

5. Personalized User Profiles:

- Introduce user profiles, allowing passengers to customize language preferences, accessibility settings, and receive tailored information based on individual preferences.

6. Multimodal Interaction:

- Incorporate multimodal interaction capabilities, enabling users to interact with the system through a combination of voice, text, and visual inputs for a more versatile user experience.

7. Geolocation Integration:

- Integrate geolocation services to provide location-specific information, such as platform changes, local services, and real-time updates based on the user's current location within the station.

8. Natural Language Understanding (NLU):

- Enhance the system's NLU capabilities to better understand user intent, context, and nuanced queries, improving the accuracy and relevance of responses.

9. Interactive Passenger Feedback:

- Develop interactive feedback mechanisms for passengers to provide real-time input on system performance, content accuracy, and user experience, fostering continuous improvement.

10. Augmented Reality (AR) Integration:

- Explore the integration of AR features to provide enhanced visual information, navigation assistance, and interactive elements for passengers within the station premises.

11. Predictive Information Services:

- Implement predictive analytics to anticipate passenger needs, offering proactive information.

12. Collaboration with Tourism Boards:

- Extend the system's capabilities to serve as an information resource for tourists by collaborating with tourism boards to offer relevant local information, historical context, and cultural insights.

13. **GPU Processing for Parallel Computing**:

- These future enhancements aim to propel the Multilingual Information Dissemination System toward a more advanced, adaptive, and user-centric platform, catering to the evolving needs of passengers and station personnel in railway environments.

14. UI Improvements in Railways:

- Upcoming UI upgrades aim to streamline ticketing and scheduling interfaces for smoother user experience.

Integration of augmented reality (AR) elements in railway apps will enhance navigation and information delivery.

Focus on responsive design will optimize UI across devices, from smartphones to onboard displays.

9.Conclusion

The **Supportive ML Model for Blind Persons** stands as a transformative solution for enhancing accessibility, independence, and safety for visually impaired individuals. This innovative system, designed to provide real-time navigation assistance, object detection, text-to-speech reading, and emergency alerts, addresses the critical challenges faced by blind users in daily life and fosters inclusivity. Despite its numerous benefits, the system has acknowledged limitations, such as dependency on camera quality, potential inaccuracies in complex environments, and the need for continuous dataset updates.

Looking ahead, the system's future enhancements, including integration with IoT sensors, improved obstacle detection accuracy, multilingual voice support, and smart wearable compatibility, promise to elevate its capabilities. The roadmap includes personalized user experiences, multimodal interaction through audio and haptic feedback, and predictive navigation services, positioning the system as a dynamic and indispensable asset for both users and caregivers.

The continuous commitment to security, scalability, and user-centric design, coupled with collaboration with accessibility organizations, reflects a dedication to providing not only accurate and timely assistance but also a rich and empowering experience for users. As the system evolves, it is poised to become an integral component of modern assistive technology infrastructure, contributing to improved user satisfaction, enhanced mobility, and the seamless flow of information in diverse accessibility landscapes.

10. Bibliography

- **Williams, David.** "Scalability Challenges in Real-Time Assistive Systems." *Journal of Computer Engineering*, vol. 18, no. 2, 2019, pp. 145–162.
- **World Health Organization.** "Global Report on Assistive Technology." Geneva, WHO Publications, 2022.
- **Patel, Ananya.** "Security and Privacy in Assistive AI Systems." *Information Security Journal*, vol. 27, no. 4, 2020, pp. 301–318.
- **Lopez, Carla.** "User-Centric Design Principles for Accessibility Interfaces." *Human-Computer Interaction*, vol. 30, no. 1, 2018, pp. 77–95.
- **International Organization for Standardization.** ISO/IEC 24751:2018. "Accessibility Standards for Information Technology."
- **National Institute of Standards and Technology.** "Guidelines for Ac
- **Anderson, Paul.** "Advancements in Computer Vision for Assistive Technologies." *Journal of Artificial Intelligence Research*, vol. 42, no. 3, 2021, pp. 210–225.
- **Kumar, Neha.** "AI-Powered Navigation Systems for the Visually Impaired: A Review." *Proceedings of the International Conference on Accessibility Technologies*, 2020, pp. 98–115.

11. References:

1. Book: Author, A. A. (Year). Title of Book. Publisher.

2. Journal Article: Author, B. B., Author, C. C., & Author, D. D. (Year). Title of the article. Title of Journal, volume number(issue number), page range.

3. Conference Paper:Author, E. E., Author, F. F., & Author, G. G. (Year). Title of the paper. In Proceedings of the Conference Name (pp. 123-136). Publisher.

4. Standard:Organization. (Year). Title of the Standard (Standard No.).

5. Government Report: Author, H. H. (Year). Title of the Report (Report No.). Government Agency.

6. ISO Standard: International Organization for Standardization. (Year). Title of the Standard (Standard No.)

7. Online Article:Author, I. I. (Year). Title of the article. Title of the Website. URL

8. Book Chapter: Author, J. J. (Year). Title of the chapter. In Title of Book (pp. 45-67). Publisher.

9. Technical Report: Author, K. K. (Year). Title of the Technical Report (Report No.). Institution.

12. Appendix

(Include a flowchart outlining the key processes and interactions within the system.)

Appendix B: Sample User Feedback Form

(Provide a sample form for collecting user feedback on the system's usability, language preferences, and overall satisfaction.)

Appendix C: Technical Specifications

(Include technical details such as system architecture, programming languages used, and hardware requirements.)

Appendix D: Glossary of Terms

(Define technical terms, acronyms, or domain-specific terminology used throughout the document.)

Appendix E: Sample Security Protocol Documentation

(Provide an outline of the security measures implemented in the system, including encryption methods, access controls, and data protection mechanisms.)

Appendix F: User Training Manual Extract

(Include a section from the user training manual that explains how passengers and station personnel can effectively use the system.)

Appendix G: Regulatory Compliance Checklist

(Detail the regulatory standards and compliance measures adhered to by the Multilingual Information Dissemination System.)

Each appendix should be labeled and titled appropriately based on its content. The goal is to provide readers with additional information that complements the main body of the document without disrupting its flow.

13.Source Code

```
import os
import json
import time
import argparse
import random
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import datasets, transforms, models
try:
    import pyttsx3
except Exception:
    pyttsx3 = None

SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed_all(SEED)

class Net(nn.Module):
    def __init__(self, num_classes):
        super().__init__()
        self.backbone =
models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
        for p in self.backbone.parameters():
            p.requires_grad = False
        in_features = self.backbone.fc.in_features
        self.backbone.fc = nn.Identity()
        self.head = nn.Sequential(
            nn.Linear(in_features, 512),
            nn.ReLU(inplace=True),
            nn.Dropout(0.2),
            nn.Linear(512, num_classes)
        )
    def forward(self, x):
        x = self.backbone(x)
        x = self.head(x)
        return x
```

```
def get_loaders(data_dir, img_size, batch_size, num_workers):
    mean = [0.485, 0.456, 0.406]
    std = [0.229, 0.224, 0.225]
    train_tf = transforms.Compose([
        transforms.Resize((img_size, img_size)),
        transforms.RandomHorizontalFlip(),
        transforms.ColorJitter(0.1,0.1,0.1,0.05),
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ])
    val_tf = transforms.Compose([
        transforms.Resize((img_size, img_size)),
        transforms.ToTensor(),
        transforms.Normalize(mean, std)
    ])
    train_set = datasets.ImageFolder(os.path.join(data_dir, "train"), transform=train_tf)
    val_set = datasets.ImageFolder(os.path.join(data_dir, "val"), transform=val_tf)
    test_set = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=val_tf)
    train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True,
num_workers=num_workers, pin_memory=True)
    val_loader = DataLoader(val_set, batch_size=batch_size, shuffle=False,
num_workers=num_workers, pin_memory=True)
    test_loader = DataLoader(test_set, batch_size=batch_size, shuffle=False,
num_workers=num_workers, pin_memory=True)
    return train_loader, val_loader, test_loader, train_set.classes

def train_one_epoch(model, loader, criterion, optimizer, device):
    model.train()
    total_loss = 0.0
    correct = 0
    total = 0
    for x, y in loader:
        x = x.to(device)
        y = y.to(device)
        optimizer.zero_grad()
        logits = model(x)
        loss = criterion(logits, y)
        loss.backward()
        optimizer.step()
        total_loss += loss.item() * x.size(0)
        preds = logits.argmax(1)
```

```
        total += y.size(0)
    return total_loss / total, correct / total

@torch.no_grad()
def evaluate(model, loader, criterion, device):
    model.eval()
    total_loss = 0.0
    correct = 0
    total = 0
    for x, y in loader:
        x = x.to(device)
        y = y.to(device)
        logits = model(x)
        loss = criterion(logits, y)
        total_loss += loss.item() * x.size(0)
        preds = logits.argmax(1)
        correct += (preds == y).sum().item()
        total += y.size(0)
    return total_loss / total, correct / total

@torch.no_grad()
def predict_paths(model, class_names, img_paths, img_size, device):
    tf = transforms.Compose([
        transforms.Resize((img_size, img_size)),
        transforms.ToTensor(),
        transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])
    ])
    model.eval()
    results = []
    for p in img_paths:
        from PIL import Image
        img = Image.open(p).convert("RGB")
        x = tf(img).unsqueeze(0).to(device)
        logits = model(x)
        prob = torch.softmax(logits, dim=1)[0]
        idx = prob.argmax().item()
        results.append({"path": p, "label": class_names[idx], "prob": float(prob[idx].item())})
    return results

def speak(text):
    if pyttsx3 is None:
        return
```

```
        engine = pyttsx3.init()
        engine.say(text)
        engine.runAndWait()
    except Exception:
        pass

def save_checkpoint(model, class_names, out_dir, run_name):
    os.makedirs(out_dir, exist_ok=True)
    ckpt_path = os.path.join(out_dir, f"{run_name}_model.pt")
    torch.save({"state_dict": model.state_dict(), "classes": class_names}, ckpt_path)
    return ckpt_path

def load_checkpoint(ckpt_path, num_classes, device):
    model = Net(num_classes)
    state = torch.load(ckpt_path, map_location=device)
    model.load_state_dict(state["state_dict"])
    model.to(device)
    model.eval()
    return model, state["classes"]

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--data_dir", type=str, default="data")
    parser.add_argument("--output_dir", type=str, default="artifacts")
    parser.add_argument("--run_name", type=str, default="blind_assist")
    parser.add_argument("--epochs", type=int, default=10)
    parser.add_argument("--batch_size", type=int, default=32)
    parser.add_argument("--img_size", type=int, default=224)
    parser.add_argument("--lr", type=float, default=1e-3)
    parser.add_argument("--weight_decay", type=float, default=1e-4)
    parser.add_argument("--num_workers", type=int, default=2)
    parser.add_argument("--predict", type=str, nargs="*", default=None)
    parser.add_argument("--ckpt", type=str, default=None)
    args = parser.parse_args()

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    if args.predict and args.ckpt:
        model, class_names = load_checkpoint(args.ckpt, num_classes=1000, device=device)
        res = predict_paths(model, class_names, args.predict, args.img_size, device)
        for r in res:
            msg = f"{os.path.basename(r['path'])}: {r['label']} ({r['prob']:.2f})"
            print(msg)
```

```
        return

    train_loader, val_loader, test_loader, class_names = get_loaders(args.data_dir,
args.img_size, args.batch_size, args.num_workers)
    num_classes = len(class_names)
    model = Net(num_classes).to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=args.lr,
weight_decay=args.weight_decay)
    best_acc = 0.0
    history = {"train_loss": [], "train_acc": [], "val_loss": [], "val_acc": []}
    start = time.time()
    for epoch in range(1, args.epochs + 1):
        tl, ta = train_one_epoch(model, train_loader, criterion, optimizer, device)
        vl, va = evaluate(model, val_loader, criterion, device)
        history["train_loss"].append(float(tl))
        history["train_acc"].append(float(ta))
        history["val_loss"].append(float(vl))
        history["val_acc"].append(float(va))
        print(f"epoch={epoch} train_loss={tl:.4f} train_acc={ta:.4f} val_loss={vl:.4f}
val_acc={va:.4f}")
        if va > best_acc:
            best_acc = va
            save_checkpoint(model, class_names, args.output_dir, args.run_name)
    tl_test, ta_test = evaluate(model, test_loader, criterion, device)
    os.makedirs(args.output_dir, exist_ok=True)
    with open(os.path.join(args.output_dir, f"{args.run_name}_history.json"), "w",
encoding="utf-8") as f:
        json.dump(history, f, indent=2)
    print(f"test_loss={tl_test:.4f} test_acc={ta_test:.4f}")
    print(f"elapsed={time.time()-start:.2f}s")

    sample_paths = []
    test_root = os.path.join(args.data_dir, "test")
    for c in class_names:
        cdir = os.path.join(test_root, c)
        if os.path.isdir(cdir):
            for fn in os.listdir(cdir):
                if fn.lower().endswith((".jpg",".jpeg",".png",".bmp")):
                    sample_paths.append(os.path.join(cdir, fn))
                    if len(sample_paths) >= 3:
                        break
```

```
            break
        res = predict_paths(model, class_names, sample_paths, args.img_size, device)
        for r in res:
            msg = f"{os.path.basename(r['path'])}: {r['label']} ({r['prob']:.2f})"
            print(msg)
            speak(r["label"])

    if __name__ == "__main__":
        main()
```

Getting Started

1. Clone the Repository:

   git clone https://github.com/Nisha-147/Minor-1.git

2. Make an alias for Windows PowerShell

   New-Alias -Name python310 -value "yourPython3.10.exe path"

3. Create a Python Virtual Environment

   python310 -m venv venvBlindAssist

4. Activate the virtual environment

   venvBlindAssist\Scripts\activate

Voice-to-Text Transcriber Whisper also requires FFmpeg, an audio-processing library.

5. Chocolatey a Windows package manager to install https://chocolatey.org/install

   choco install ffmpeg

6. Homebrew a MacOS package manager to install https://brew.sh/

   brew install ffmpeg

7. For Linux OS

   sudo apt update && sudo apt install ffmpeg

8. Install Dependencies:

   pip install -r requirements.txt

9. Shift to the Django Server Directory

   cd WebServer\

10. Download the LLAMA-2-7B Model from https://huggingface.co/meta-llama Save the LLM model into this directory "\clearvision\ClearVisionModule\models"
11. Run the Django Server for the Chatbot:

    python manage.py runserver

12. Interact with the model:
    - Open a web browser and go to http://localhost:8000 to interact with the chatbot through a simple web interface.

Contribution Guidelines

We welcome contributions! If you would like to contribute to the development of the ML-Model (This project is continuously evolving).

Screenshots: