Assistive Vision

Development Plan

Team Members: Venkat Yenduri, Pierre Tawfik, Sukrut Nadigotti, Sharefa Alshaary

Date	Description	Author	Comments
1/19/25	Version 1.0	Venkat Yenduri Sukrut Nadigotti Pierre Tawfik Sharefa Alshaary	Initial Draft
2/16/25	Version 2.0	Venkat Yenduri Sukrut Nadigotti Pierre Tawfik Sharefa Alshaary	Revised Plan
3/31/25	Version 3.0	Venkat Yenduri Sukrut Nadigotti Pierre Tawfik Sharefa Alshaary	Modified Development Plan to Match Updated Project Features

Contents

1 Project Overview

The Assistive Vision project is designed to address the challenges faced by blind and visually impaired individuals in navigating their surroundings. The solution provides real-time object detection with the use of image recognition and audio feedback, enabling users to

recognize objects and enhance their independence. The project incorporates advanced computer vision technology, an intuitive design modified for ease of access, an iPhone for object recognition, and a text-to-speech system to provide clear and descriptive feedback of objects in the environment.

2 Project Purpose, Scope, and Objective

The purpose of the Assistive Vision iOS application is to empower visually impaired individuals by providing real-time object recognition with audio feedback, allowing them to navigate their environment with greater confidence and independence. By leveraging iPhone's advanced camera, machine learning capabilities, and accessibility features, the app reduces reliance on human assistance and traditional mobility aids while enhancing situational awareness.

This project integrates hardware and software components to create a seamless and user-friendly assistive solution. The hardware consists of the iPhone's built-in camera, microphone, speakers, and Apple's Neural Engine, eliminating the need for external devices. The software utilizes CoreML and a custom YOLOv11m model for real-time object detection, with AVFoundation handling live camera feed processing. Audio descriptions of detected objects are delivered through Apple's Text-to-Speech (TTS) API, providing clear and customizable feedback through the iPhone's speakers or connected headphones.

The app is designed to be intuitive, requiring minimal user interaction through touch gestures and voice commands. It supports iOS accessibility features such as VoiceOver and Siri Shortcuts, ensuring seamless integration with existing tools for visually impaired users. Users can choose between Continuous and On-Demand scanning modes, customize detection settings, and enable priority alerts for important objects. The system operates efficiently in diverse environments, including homes, offices, and public spaces, adapting to varying lighting conditions for reliable performance indoors.

The objective of the Assistive Vision iOS application is to develop an intuitive and reliable mobile solution that enhances independence for visually impaired users. The app aims to provide fast, accurate, and context-aware object recognition with real-time audio feedback, enabling users to understand their surroundings more effectively. By leveraging on-device AI processing, the system ensures privacy, speed, and reliability without requiring internet connectivity.

The application's modular and scalable architecture allows for future enhancements, including expanded object recognition capabilities, improved navigation features, and personalized user settings. The ultimate goal is to create an adaptable and accessible tool that significantly improves the daily lives of visually impaired individuals by making real-time object detection simple, efficient, and effective.

3 Team Organization

3.1 Roles

Venkat Yenduri: Team Leader, Communications Lead, Software Developer

Pierre Tawfik: Scribe, Maintenance Lead, Software Developer Lead

Sukrut Nadigotti: Documentation Lead, Presentation Lead, Software Developer

Sharefa Alshaary: Quality Assurance Lead, Software Developer

3.2 Responsibilities

Team Leader

The team lead organizes all meetings with the team, GTA, and professor to ensure effective communication and alignment. They will assign tasks appropriately and reassign responsibilities if needed to maintain progress. In situations where the team encounters a problem, the team will effectively discuss both sides and then the team lead will make the final decision to resolve the issue. If necessary, the team lead may consult the GTA for additional guidance and support in decision-making.

Presentation Lead

The Presentation Lead is responsible for organizing and overseeing all weekly presentations. This includes structuring PowerPoints, developing consistent themes in terms of styling, and ensuring the content is clear and focused on key topics. They should delegate specific parts of each presentation to team members, ensuring everyone contributes while maintaining a cohesive narrative. Additionally, the Presentation Lead ensures that all materials are professionally formatted, succinct, and effectively communicate the intended message to the audience.

Documentation Lead

The Documentation Lead is responsible for managing all documentation related to the project. Their duties include developing the initial structure, assigning specific sections to team members, and determining the essential information to be included. They ensure that the content is comprehensive and accurate while maintaining proper grammar and an appropriate writing style.

Quality Assurance (QA)

The Quality Assurance (QA) role ensures the project meets high standards of functionality, accuracy, and user experience. Responsibilities include thorough testing to identify and resolve issues, validating that features meet requirements, and ensuring adherence to quality and coding standards. QA reviews deliverables for consistency and provides feedback to the team.

Scribe

The team scribe will take notes during lectures and keep track of what is discussed in team meetings. They will document key points, assigned tasks, and meeting minute times, ensuring all information is organized and accessible to the team. The scribe's role is to maintain accurate records of discussions and decisions to keep the team informed and aligned throughout the project.

Software Developer

Software developers' responsibilities focus on developing and implementing robust solutions using Swift and libraries such as TensorFlow, OpenCV, and gTTS to enable real-time object detection and audio feedback. This ensures that the software design aligns with accessibility goals, the tasks include creating a user-friendly interface, optimizing code for efficiency and scalability, and performing any testing required to address bugs and performance issues.

Communications Lead

Communications lead responsibilities involve facilitating clear and consistent coordination among team members, organizing meetings, and acting as a primary point of contact for Team members, GTA, and the professor. This includes providing timely updates and ensuring clarity in discussions. The use of platforms like Discord, Microsoft Teams, and OneDrive plays a key role in the team's collaboration and communication.

Maintenance Lead

Maintenance lead responsibilities include monitoring the project's performance, addressing issues as they arise, and updating hardware and software components based on testing results from the QA Lead and testing feedback. This ensures that updates are compatible with existing functionality and do not disrupt operations is essential. Additionally, maintaining detailed documentation of maintenance activities, including updates and resolved issues, ensures that the system remains reliable and adaptable for future use.

4 Problem Resolution Policies

Failure to Deliver:

- Team members missing deadlines or meetings without notice will receive a warning.
- Extensions of 3-7 days may be granted for emergencies. Repeat offenses will be escalated to the GTA.

Team Leader Accountability:

- If the team leader is at fault, a group discussion will occur on the issue and escalate to the GTA if unresolved.

Dispute Resolution:

- Technical disagreements will be mediated by the team leader. If unresolved, the GTA will be consulted.
- Workload imbalances or recurring misunderstandings will be addressed during team meetings

5 Project Plan

5.1 Iterations

The team will have three recurring meetings throughout the week to discuss their plans and update each other on their progress. They will meet with their assigned GTA once a week to review the application progress and ask questions. Below is the updated meeting schedule:

Weekly in-person team meeting at State Hall:

Monday from 2:00 PM - 4:00 PM EST

GTA Zoom Meeting:

Tuesday from 4:30 PM – 5:30 PM EST

Bi-weekly Team Discord Meeting:

Wednesday from 12:00 PM - 2:00 PM EST

5.2 Project Schedule

First Prototype: February 1st

Initial prototype featuring basic real-time object detection, audio feedback capabilities, and foundational support for voice command interaction. Incorporates preliminary UI structure and initial CoreML integration with YOLOv11m.

Software Requirements Specification (SRS): February 7th

Detailed documentation specifying functional and non-functional requirements, clearly outlining UI, interaction methods, voice command specifications, model accuracy benchmarks, and detailed integration plans for SwiftSpeech, Gemini AI, depth data processing, and specialized door detection features.

Design Document: February 21st

Comprehensive architecture detailing hardware (iPhone camera, depth sensors, audio components, Neural Engine) and software design including SwiftSpeech voice recognition, Gemini AI integration strategies, real-time depth estimation methodologies, multi-language support implementation, battery optimization approaches, and door detection module architecture.

Second Prototype: March 1st

Enhanced prototype demonstrating complete UI with functional speech recognition (SwiftSpeech), language switching, dynamic model selection (Quick, Optimal, Intensive modes), initial Gemini AI query handling, basic depth-based spatial feedback, and robust door detection integration.

Test Plan: March 14th

Robust testing approach addressing real-time object detection accuracy, voice command responsiveness, Gemini AI conversational reliability, depth-based spatial orientation precision,

multi-language audio feedback clarity, battery optimization effectiveness, and specialized door detection accuracy under diverse real-world conditions.

Third Prototype: March 29th

Fully integrated prototype featuring comprehensive notification system enhancements, advanced Gemini AI conversational interfaces for FAQ and Tutorials, accurate real-time depth feedback integration, extensive voice command interactions, door detection optimizations, improved multi-language support, and initial battery-saver functionalities for extended usability.

Final Application: April 11th

Finalized Assistive Vision application with seamless integration of all planned features, including advanced real-time object detection, refined audio and haptic feedback mechanisms, robust Gemini AI interactions, intuitive voice command capabilities, precise depth-based spatial guidance, specialized door detection, multi-language accessibility, comprehensive battery optimization, and full accessibility feature compatibility, ready for demonstration and deployment.

6 Configuration Management

The team will use Discord as the primary communication platform, with a single group chat for discussing project updates, asking questions, and sharing information. Additionally, weekly in-person meetings will ensure clear communication and collaboration. All documentation and shared resources will be stored in a Google Drive folder for easy access and collaboration among team members.

For version control, the team will use Git and host the project's codebase on GitHub. A master branch and a development branch will be maintained to organize the workflow effectively. Each member will have a specific branch where they work on their assigned tasks. This structure ensures changes are isolated and manageable.

Once a task is completed, the member will create a pull request and assign it to the QA Lead or a team member with expertise in the relevant area. The reviewer will conduct a detailed code review, checking for functionality, adherence to project requirements, and coding standards. Feedback will be provided as needed, and only code that has been thoroughly reviewed, tested, and approved will be merged into the development branch. Following further testing and final approval, changes will be merged into the master branch to ensure the project's integrity and functionality.

7 Technologies

7.1 Hardware

The Assistive Vision iOS application leverages the advanced hardware capabilities of modern iPhones to provide real-time object recognition with audio feedback. The iPhone's built-in rear camera serves as the primary sensor, capturing high-quality video for object detection. The application utilizes Apple's Neural Engine and A-series/M-series processors to efficiently run machine learning models, ensuring fast and accurate object identification.

For delivering clear and precise audio feedback, the app supports the iPhone's built-in speakers and Bluetooth-connected headphones or hearing aids. Users can also interact with the app using the iPhone's touchscreen for simple gesture-based navigation or the microphone for voice commands via Siri Shortcuts. Since continuous scanning can be battery-intensive, the app includes a Battery Saver mode, which optimizes power consumption by adjusting the frame capture rate and processing load.

7.2 Software

The software stack of the Assistive Vision application is optimized for real-time performance and accessibility on iOS devices. The app is developed using Swift and utilizes CoreML, Apple's machine learning framework, to deploy the custom YOLOv11m model for efficient on-device object detection. AVFoundation is integrated for managing live camera feed processing, ensuring smooth real-time analysis.

To deliver spoken object descriptions, the application utilizes Apple's Text-to-Speech (TTS) API, providing natural-sounding voice output with customizable options. The user interface is built using SwiftUI, ensuring a seamless and intuitive experience with full compatibility with iOS accessibility features, such as VoiceOver and Dynamic Type. Additionally, Siri Shortcuts enable hands-free operation, allowing users to activate object detection through voice commands.

Additional features include speech recognition implemented with SwiftSpeech, enabling detailed voice-command interactions, such as changing languages (English/Spanish), adjusting detection settings (confidence, IoU thresholds), and selecting detection modes

(Continuous/On-Demand). The app also integrates Google's Gemini AI, enhancing user interactions through conversational assistance, processing complex queries about detected objects and their surroundings.

Real-time depth data from the iPhone's camera is leveraged to accurately estimate object distances and spatial positions, providing detailed feedback about object proximity and orientation. A specialized door detection feature has also been added, clearly identifying and announcing doors, including their open or closed status.

Battery-saving features optimize app performance by reducing processing frequency during continuous scanning mode, ensuring prolonged usability.

All detection processing is performed locally on the device, ensuring user privacy by avoiding cloud-based AI processing. The combination of Apple’s powerful hardware, efficient machine learning frameworks, advanced AI integrations, real-time depth analysis, and accessibility focused software design ensures that the Assistive Vision iOS application provides reliable, real-time object recognition to assist visually impaired users in navigating their surroundings confidently.