TABLE OF CONTENTS

1 REQUIREMENTS

1.1 Required Status and Mods

This system is designed to adapt to different scenarios that visually impaired individuals face in daily life. The situations for this application, which is a Software Partial Element (SPE), are defined in Table 1:

Status/Mode Name	**Description**
Ready	After the system is started, it completes the initial checks for cameras and models and is ready to receive images from the user.
Active	Images are taken with the camera and these images are transferred to Object Detection and Depth Estimation models. Audio feedback is given.
Degraded	This is when the system is partially operational due to reasons such as the camera being unreachable, the model malfunctioning or hardware inadequacies. The user is notified of this situation audibly.
After Use	When the system is shut down by the user or after a period of inactivity, it switches to this mode, freeing up resources.

Table 1: Case Names and Descriptions

The system constantly switches between "ready" and "active" modes. Degraded and after-use modes are the system's responses to external factors.

Some of the system requirements according to these modes are associated in Table 2:

Requirement	**Related Mode/State**
Image Acquisition	Ready/Active
Object Detection	Active
Depth Estimation	Active
Voice Feedback	Active
Error Detection and Reporting	Degraded
Closing Sources	After Use

Table 2: Requirement-State Mappings

1.2 YKE Functional Requirements

This system is organized around three main functionalities: **object detection**, **depth estimation** and **audio feedback**. For each functionality, the functions and technical requirements that the system must fulfill are

explained below.

1.2.1 Object Detection Functionality

The system should periodically take images of the environment through the camera of the mobile device and detect whether there are objects in the images. This is done with a machine learning model that is pre-trained and optimized to run on the mobile device.

Functional Requirements:

- The system should capture images from the environment at least every 2 seconds.
- Image processing time should be maximum 1 second.
- Detected objects should be presented with class label and location information (right, left, up, down, across).
- The system should only consider objects with a confidence score of 60% and above.
- Up to 5 objects can be processed at the same time.
- For invalid or blurred images, the system should not give an error, but should produce the message "No object detected".
- This functionality should be optimized to run continuously and use mobile device resources efficiently.

1.2.2 Depth Estimation Functionality

The system must estimate the distance between the detected object and the device. This is done through a model that estimates the depth. The model is run only when the object is detected.

Functional Requirements:

- Depth estimation should take place within 1 second after object detection is complete.
- Distance information must be provided with an error tolerance of ±20 cm.
- In case of multiple object detection, the closest object should be prioritized.
- For low resolution images, the system should try to produce distance estimates even if the quality degrades.
- If the depth estimation is unsuccessful, the system should give the warning "Failed to estimate distance".
- The model should be optimized to run on a mobile device (e.g. TensorFlow Lite conversion).

1.2.3 Voice Feedback Functionality

The system should transmit the detected object and estimated distance information to the user audibly. In this way, the visually impaired individual gains environmental awareness.

Functional Requirements:

- Audio feedback should start no later than 1 second after object and distance information is obtained.
- Message format: "{Object} is {Distance} meters away from {Party}."
- Message duration should not exceed a maximum of 3 seconds.
- The previous message must be completed before a new detection is made; voice notifications should not overlap.
- If an error occurs in the Text-to-Speech engine, the system should voice a default error message (such as "Voice response failed").
- Voice responses should flow naturally and without delay; holds should be minimized.

1.3 Requirements for YKE External Interface

In this section, the system's connections to the outside world are defined and interface requirements are detailed. In the project, the interactions of the ACE with the user, hardware and system APIs are described below.

1.3.1 Interface Definition and Diagrams

The diagram of the interface to be defined is given in Diagram 1.

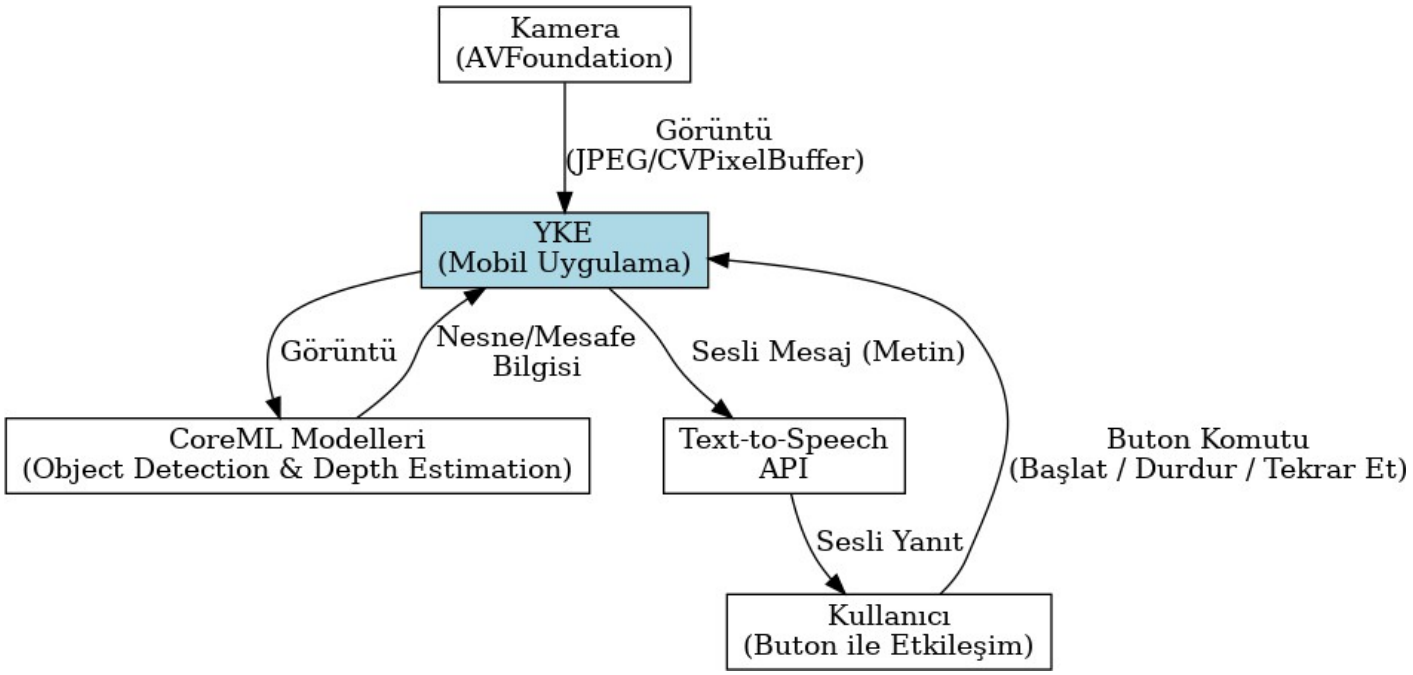

Diagram 1: External Interface Requirements

The system exchanges data with the following external interfaces:

- **Camera Interface (AVFoundation - iOS)**
  The iPhone camera is used to take images of the surroundings at regular intervals.

- API AVFoundation
- Input Camera image (RGB)
- Format: JPEG, 640x480 resolution
- Frequency: 1 frame every 2 seconds
- Security Requires camera access permission (iOS Privacy Setting)

- **Voice Output Interface (Text-to-Speech)**
  The detected object and distance information is transmitted to the user audibly.
  - API: iOS Text-to-Speech
  - Data Type: Text→ Voice
  - Format: English natural speech
  - Maximum Duration 3 seconds
  - Error Status: Voice prompt if feedback cannot be given ("Voice response failed")

- **User Interface (UI)**
  Apart from the voice output of the application, there may be limited but necessary user interaction (e.g. start/stop or repeat command).
  - Interaction Type: Button, voice command
  - UI Structure: Simple interface, accessibility friendly
  - Priority Real-time response (within 1 second)

- **Model Interfaces (Core ML - Object Detection & Depth Estimation)**
  Images are transferred to two Core ML-based models running on iOS:
  - Model Input: Image (CVPixelBuffer)
  - Model Output: Class, location and distance information
  - Model Formats: .mlmodel (pre-trained and optimized)
  - Time Constraint: Inference operations should not exceed 2 seconds in total

1.3.2 User Interface

- **Description:** The user interface i s activated upon opening the application. The user is informed with audio feedback.

- **Data Elements:**
  - Command: "Start" / "Stop" / "Repeat"
  - Output: Audio information

- **Timing:** All commands must be responded to in less than 1 second.
- **Accessibility:** The interface must fully comply with accessibility guidelines (such as iOS VoiceOver support).

1.3.3 Camera Interface

- **Description:** AVFoundation API to retrieve images from the device camera.
- **Data:**
    - Type: Image (RGB)
    - Format: JPEG / CVPixelBuffer
    - Frequency: 0.5 Hz (every 2 seconds)
- **Priority:** Ensure high, continuous and up-to-date image flow.
- **Security:** The app must request camera access permission from the user.

1.3.4 Voice Output (Text-to-Speech)

- **Definition:** Gives the system output to the user by voice.
- **Data:**
    - Type: Text→ Speech
    - Format: English, natural tone
    - Maximum duration 3 seconds
- **Security:** iOS should be used in accordance with access permissions; private data should not be read aloud.
- **Communication Path** CoreML result→ Text message→ TTS engine→ Voice

1.3.5 Model Interface (CoreML)

- **Description:** Image data is fed into two models integrated with Core ML.
- **Data Flow:**
    - Input Image (camera output)
    - Output 1 (Object Detection): Class, coordinate, score
    - Output 2 (Depth Estimation): Distance in meters
- **Format:** .mlmodel, converted from TensorFlow Lite
- **Timing:** Each inference process should not exceed 1 second

- **Communication:** In-script function calls, runs locally; no network connection required

1.4 Requirements for YKE Internal Interface

In this system, all components of the ACE (camera data processing, object detection, depth estimation, audio feedback) work within the same application and exchange data internally. The internal interfaces between these components and their requirements are described below. The diagram of the interface whose requirements will be discussed is given in Diagram 2.

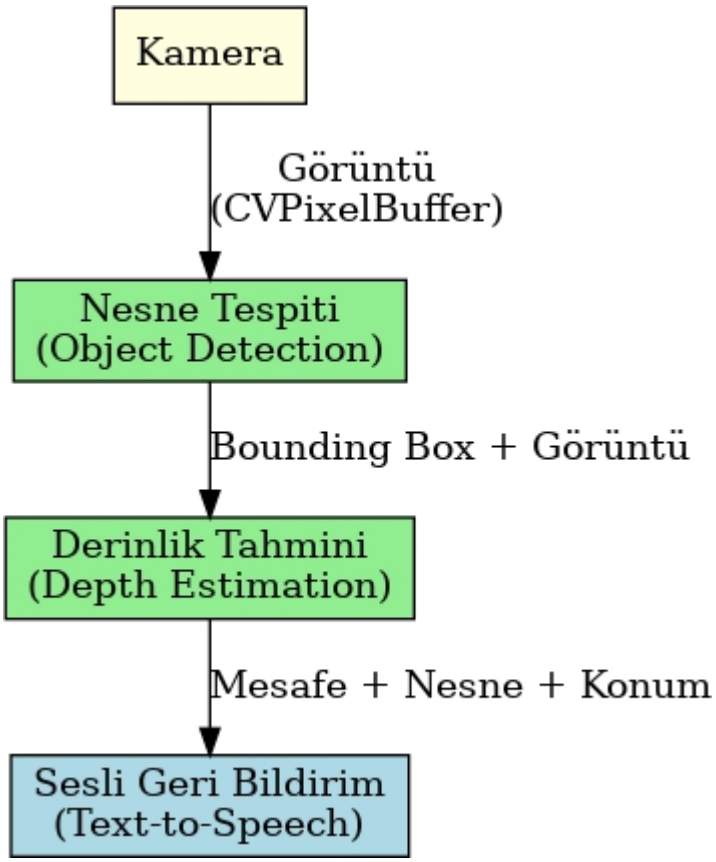

Diagram 2: Internal Interface Requirements

1.4.1 Camera→ Object Detection Interface

- **Data Type:** Image (RGB)
- **Format:** CVPixelBuffer (CoreML compliant), resolution 640x480 (subject to change, depending on the dimensions accepted by the object detection model)
- **Frequency:** every 2 seconds
- **Workflow**
    - The camera component takes images at regular intervals.
    - This image is converted into the input format of the object detection model and passed to the model.

- From the position of the object relative to the camera, the position of the object is marked as right, left, etc.

- **Restrictions:** Images should be marked if blurry in low light; filters can be applied before processing.

1.4.2 Object Detection→ Depth Estimation Interface

- **Data Type:** Bounding box of the detected object and original image
- **Format:**
  - Location information (x, y, w, h - integer)
  - Image (CVPixelBuffer), resized
- **Timing:** This data transfer must occur no later than 500ms after object detection is complete.
- **Restrictions:**
  - If there is more than one object, only the closest object should be transferred.
  - If detection is not possible, depth estimation should be disabled.

1.4.3 Depth Estimation→ Voice Feedback Interface

- **Data Type:** Distance information (float)+ Object class (string)+ Object location (string)
- **Format:**
  - Distance: float, unit= meters (e.g. 1.75)
  - Object class: alphanumeric string ("chair", "table", etc.)
  - Object position: alphanumeric string ("right", "up", etc.)
- **Timing:** All data must be transferred to voice feedback within 1 second.
- **Workflow**
  - The distance, object position and object class are combined into a sentence ready for voice notification.
  - The sentence is transmitted to the Text-to-Speech engine.

1.4.4 General Requirements

- **Data Formats:** All inter-component data transfer should be done with fixed data structures (e.g: ObjectInfo { class: string, bbox: [int, int, int, int, int], confidence: float })
- **Timing:** Total processing time (camera→ audio feedback) should not exceed 4 seconds.
- **Error Management:** If each component fails to receive data, it should generate a default response and notify other components

must be warned.

- **Security:** Internal components should only access data that belongs to their domain; memory leakage or collision should be avoided.

1.5 YKE Internal Data Requirements

This system does not use an external database or persistent data file. However, there are some internal data structures and model files that are needed during the operation of the system. These data items are described below and the related requirements are specified.

1.5.1 Artificial Intelligence Model Files

- **Object Detection Model**
    - File type: .mlmodel
    - Size: Maximum 50 MB
    - Format: CoreML compliant, runnable on mobile device
    - Location: Built-in within the app (offline)
    - Security File must be closed to external access
    - Version: v1.0
- **Depth Estimation Model**
    - File type: .mlmodel
    - Size: Maximum 100 MB
    - Format: CoreML-compliant, depth-extracting structure over a single image
    - Location Embedded in the application
    - Security: Not shared with other applications, accessed only by the system
    - Version: v1.0

1.5.2 Temporary Image Data (Buffer)

- **Input Type:** Camera image (RGB)
- **Format:** CVPixelBuffer
- **Resolution** 640x480 pixels
- **Storage:** Temporary only, kept in RAM
- **Security:** Images are not saved in any way, only kept in memory for the duration of the process

- **Cleaning:** Automatic wiping after each processing

1.5.3 Voicemail Templates

- **Template Format:** {Object} is {Distance} meters away from {Location}.
- **Tip:** String
- **Language:** English
- **Storage** Hardcoded in code
- **Extensibility:** Different language and sentence structures can be added

1.5.4 Configuration Parameters

- **Confidence Score Threshold:** 0.6 (objects detected with over 60% confidence are considered)
- **Maximum Number of Objects:** 5
- **Data Logging:** This system does not do any permanent data logging (logging, user registration, etc.)

1.6 Adaptation Requirements

This system includes some adaptive parameters according to different device environments and user profiles. The following describes the parameters that can be changed depending on the installation environment and operation of the system.

1.6.1 Audio Feedback Level

- Based on iOS audio access permissions, the default volume is initialized at 70% of the device.
- The user can adjust the volume manually if desired.

1.6.2 Camera Configuration

- The app uses the device's available camera (primarily the rear camera).
- If the device has more than one camera, the user can choose.

1.6.3 Change due to Geographical Location (If applicable)

- There is no location-based differentiation in this version.
- However, in future versions the definition of environmental objects can be differentiated depending on the GPS data (e.g. different priorities for street/home environment).

1.6.4 Operating Modes (Developable)

- The app can be adapted to the following modes in the future:
    - Silent Mode: Vibration instead of audible alert
    - Training Mode: Working with test data to teach the user how the system works
    - External Mode: Operation of the model via external server (currently all operations take place inside the device)

1.7 Safety Requirements

Since this system is used as an auxiliary tool to increase the environmental awareness of visually impaired individuals, there is a risk of physical harm if the user is given incorrect, incomplete or late information. Therefore, the software should provide the following measures for safety:

1.7.1 False Positive Information Prevention

- The system should not provide audible information about objects that do not have a sufficient confidence score.
- Objects with a confidence score below 60% are labeled as "unsafe detection" and are not included in the audio output.
- This reduces the risk of misdirection to the user.

1.7.2 False Negative Information Prevention

- If the camera image has been successfully acquired and the system is unable to detect the object, the user should be given clear and honest feedback such as "no object detected in front of you".
- Instead of remaining silent, the system should notify the user and make them aware of the situation.

1.7.3 Timely Notification Obligation

- There should be a maximum of 4 seconds between object detection and audio notification.

1.7.4 Safe Mode for Critical Bugs

- If the model fails (e.g. the inference process freezes), the system stops all operations and warns the user that "detection failed".
- In this case, the system will not continue to run silently.

1.7.5 Warning on Hardware Restrictions

- If camera access is blocked, the system warns the user audibly at startup.
- If the microphone/audio output is inaccessible, the user is notified by vibration.

1.7.6 Safety Feedback Tests

- The correct and timely output to the user for each voice notification scenario should be tested before the application is delivered.
- False/missing information scenarios should be simulated to verify the software's responses.

1.8 Security and Privacy Requirements

This system works to increase the environmental awareness of visually impaired individuals and does not record or share any personal data of the user. However, due to the camera, microphone and artificial intelligence models used, the following requirements must be met in terms of security and privacy.

1.8.1 Working Environment Safety

- The system is designed to work completely **on the device (offline)**. No data is shared with an external server over any network connection (Wi-Fi, cellular data).
- User data (image, voice, location information) is not processed, stored or transmitted by the app.

1.8.2 Camera and Microphone Access

- Camera and microphone access is only active when the system is running, all access is terminated when the application is closed.
- The privacy permissions defined in the iOS system are used. This hardware is not accessed without explicit permission from the user.

1.8.3 Data Recording and Storage

- The system **does not record** any visual, audio or text data.
- All data is stored in temporary memory (RAM) and deleted after processing.
- All temporary data is cleared before the application closes.

1.8.4 Risks and Precautions

- **Risk:** Violation of system permissions on jailbroken devices
    - **Precaution:** iOS security standards require the system to be packaged so that it can only be downloaded from the official App Store.

1.8.5 Compliance with Security Policy

- The Application takes care not to collect and process personal data within the framework of **GDPR (General Data Protection Regulation)** and **KVKK (Turkish Personal Data Protection Law)**.
- Camera and voice action permissions only work during use and with user approval.

1.8.6 Authorization and Certification

- The App must be signed with an **Apple Developer Certificate in** accordance with iOS security policies at the time of distribution and published in the App Store.
- The code of the application can only be updated by authorized developers and integrity checks should be performed against external intervention.

1.9 YKE Environment Requirements

This system is an application designed to run on iOS-based mobile devices. The following hardware and operating environment requirements must be met for the system components to function correctly.

1.9.1 Operating System Requirements of

- Supported platform: **iOS**
- Minimum operating system version: **iOS 15.0**
- The system should work on devices that support Apple's Core ML, AVFoundation and Text-to-Speech APIs.
- The app must be packaged so that it can only be downloaded from the official Apple App Store.

1.9.2 Hardware Requirements

- **Camera:** The device must have at least one rear camera. Camera resolution must be minimum 640x480 pixels.
- **Processor:** ARM-based Apple processor (A10 Fusion and above recommended)
- **Memory** Minimum 2 GB RAM
- **Storage:** At least 200 MB free space for application installation and models
- **Audio Equipment:** Audio feedback via built-in speaker or headphone output

1.9.3 Sensor and Connection Requirements

- The device **does not require an internet connection**, the system works completely offline.
- **GPS or location sensor is not mandatory** but may be supported for future versions.

1.9.4 Environment Conditions

- The system works more efficiently **in daylight or well-lit environments**. Performance may be reduced in low light.
- In extremely noisy environments, the audible notification may not be understood; in this case, the user is provided with an alternative warning

methods (e.g. vibration) should be provided.

1.10 Computer Source Requirements

This system is designed to run on mobile devices with iOS operating system and requires certain hardware and software resources. The requirements for these resources are given below.

1.10.1 Computer Hardware Requirements

- **Processor:** Apple A10 Fusion or higher (ARM-based architecture)
- **Memory (RAM):** Minimum 2 GB
- **Storage Space:**
    - Minimum 200 MB free space for application installation and model files
- **Camera:**
    - At least one rear camera, minimum resolution: 640x480 pixels
- **Speaker / Audio Output:**
    - Audio feedback should be provided via built-in speaker or connected headset
- **Battery:**
    - Since the app will run continuously, it must be resistant to battery saving modes.

1.10.2 Computer Hardware Resource Utilization Requirements

- **Processor Utilization:**
    - Up to 60% processor utilization during object detection and depth estimation
- **Memory Usage:**
    - Instant memory usage is approximately 300-400 MB
- **Storage**
    - No additional data is stored outside the application; temporary data is kept in RAM and deleted
- **I/O Usage:**
    - Continuous image acquisition is performed with the camera. Camera streaming rate is limited to a minimum of 0.5 fps (1 frame in 2 seconds)

1.10.3 Computer Software Requirements

- **Operating System:**
    - iOS 15.0 and above
- **Development Environment:**
    - Swift 5, Xcode 14 or above
- **Required Libraries and APIs:**
    - CoreML (model execution)
    - AVFoundation (camera access and image processing)
    - Text-to-Speech API (voice feedback)
- **Test Software:**
    - User testing can be done with Apple TestFlight

1.10.4 Computer Contact Requirements

- This system does not require any network or internet connection to operate.
- **Data Transfer:**
    - Since all processing takes place inside the device, data transfer is done only in memory.
- **Network Connectivity:**
    - Not required. Offline operation is based on.
- **Security:**
    - There is no data transfer to the outside world, so there is no network security risk.

1.11 Software Quality Factors

The developed software must comply with the following quality factors. These quality features are critical for the safety, comfort and sustainability of the system for visually impaired individuals who are the target audience of the project.

1.11.1 Functionality

- The software must fully fulfill all defined functions (object detection, distance estimation, audio feedback).
- Since all functions are interdependent, the system must work consistently from end to end.

1.11.2 Reliability

- The application must deliver accurate and consistent results.
- False positive and false negative outputs should be reduced and the system should warn when it is unstable.
- All operations must be terminated safely in case of errors.

1.11.3 Availability

- The application should be usable for visually impaired users with **audio guidance** and a simple interface.
- The user should be able to use the application without any training to get used to the system.

1.11.4 Availability (Availability)

- The app **should work without requiring an internet connection** as long as it is installed on the device.
- If camera and audio access is available, it must be accessible at all times.

1.11.5 Flexibility

- It should be easy to integrate different models based on data from feedback.

1.11.6 Portability

- The software must be able to run on different iOS devices and versions (iOS 15 and above).
- The application should run smoothly on different devices without recompiling.

1.11.7 Testability

- Each function (image acquisition, detection, audio output) must be independently testable.
- Feedback output is observable (audible), making verification easy.

1.11.8 Sustainability Maintainability

- The code structure should be designed to be modular and easy to add a new model or feedback format.
- Model files should be kept separately and version control should be possible.

1.11.9 Re Availability

- The object detection and depth estimation modules are developed in isolation and independently so that they can be used in other applications.

1.12 Design and Implementation Constraints

There are certain constraints and standards that must be taken into account during the design and realization of this software. These constraints have been determined by considering both the target user group of the project and the environment in which the system must operate.

1.12.1 Addiction to the platform

- The app is designed to work only on the **iOS operating system**.
- Therefore, it is mandatory to use **the Swift programming language**.
- **The Xcode** IDE provided by Apple should be used as the development environment.

1.12.2 Hardware Dependent Library Usage

- The system uses platform-specific Apple APIs such as **AVFoundation** and **Text-to-Speech API** for camera and audio feedback functions.
- **The CoreML** library is mandatory for image processing and model execution.
- Deep learning models should be in .mlmodel format, direct integration with external frameworks like TensorFlow or PyTorch is not possible.

1.12.3 Offline Working Obligation

- The system should perform all its functions **without an internet connection**.
- This prohibits the use of cloud-based models, data registration or external API calls over the network.
- Model and other process files should be **embedded in the device**.

1.12.4 Accessibility and User Audience Restrictions

- As the app is developed for visually **impaired individuals**, it must be designed in accordance with **accessibility guidelines** (e.g. Apple VoiceOver compatibility).
- All feedback to the user should be presented audibly**, not visually**.
- Visual feedbacks such as text, icons and colors are optional and secondary.

1.12.5 Extensibility and Versionability Limitations

- The application should be flexible enough to add new AI models or multi-language support in the future.
- For this reason, model files and audio templates **should not be hard-coded**, but should be loadable as separate modules.

1.12.6 Academic Standards

- Since the project is carried out within the scope of **BM314 Software Engineering** course, software lifecycle documents such as SPMP, SRS, SDD and STD should be prepared and submitted in accordance with certain templates.

1.13 Personnel Requirements

This system is designed to be used by end users who do not require technical expertise, such as visually impaired individuals. The system should be able to be operated with voice guidance and simple interactions without any training. It is also aimed to avoid the need for support staff during the use of the system.

1.13.1 User Profile and Number of

- Users include visually impaired individuals and the system is **accessibility-oriented.**
  designed.

- The system has been developed to be used by only one user at a time.

- Users interact directly with the system; no intermediate operator is required.

1.13.2 Training and Resident Assistance Requirement

- The system is designed to require no user training.

- When the app is first opened or when requested, a help mode should be activated that explains the basic steps of use **with voice instructions**.

- The system should include "repeat" or "help" commands that can be executed at the request of the user.

1.13.3 Human Factors Engineering Requirements

- The user interface should focus on **audio feedback rather than visual feedback**.

- Critical information (e.g. "camera inaccessible", "object not detected") **is communicated loudly and clearly**
  must be notified to the user.

- The tone and intensity of vocal warnings should be adjusted according to the severity of the warning.

- **Designs that are dependent on visual information** (e.g. button layout, colored icons) should be optional and supportive during system use; the main interaction should be provided by audio feedback.

- In areas where human error is common, the system should offer the user a retry or alternative path. For example:

    - When the image cannot be received: "Camera image could not be received, please redirect the device."

1.13.4 Support Personnel Requirement

- The system is developed to be used **independently** by the user.

- No support staff is required during application installation, use and shutdown.
- Technical maintenance or software updates are performed only by the development team.

1.14 Education Related Requirements

Since this system was developed to be used by non-technical users, such as visually impaired individuals, it was designed **to not require any external training**. The user interface and functionality of the application is simple, clear and based on voice guidance.

1.14.1 Help with Built-in Audio

- The app should audibly explain the basic operation to the user on first use.
- There should be a **built-in help mode** that can be activated if the user needs it.
- The help mode should provide fictional guide content with audio answers to questions such as "How does it work?", "What to do?".

1.14.2 Educational Software Requirement

- There is no need to provide any **separate courseware** within or outside the ACE.
- Ease of use is ensured by voice commands and natural language feedback.
- The user is not expected to receive any special training to get used to the system.

1.15 Logistics Related Requirements

Since the developed system is designed to work on portable mobile devices, it does not require any special logistical support. The application is easily installable, usable and portable on individual devices of visually impaired users. Installation, maintenance and updating of the system can be performed remotely by the software development team.

1.15.1 Deployment and Installation

- The system will be packaged for distribution through the **Apple App Store**.
- Users can download the app directly to their devices, requiring no physical installation or technical staff support.
- After installation, the system is ready to run.

1.15.2 Maintenance and Updates

- Software updates will be made through the App Store and can be downloaded automatically, requiring no user interaction.
- Updates may include model improvements or bug fixes.

- Since the data on the device is temporary, there is no data loss during updates.

1.15.3 Portability

- The system is completely portable as it runs on the user's **own personal mobile device**.
- The use of the app does not require any physical infrastructure or external hardware.

1.15.4 Hardware and Facility Impacts

- The system uses only the mobile device's own hardware (camera, speaker).
- No **external hardware, physical installation space or special facilities** are required.
- It does not put extra burden on existing device infrastructure and should be optimized to be battery and performance friendly.

1.16 Other Requirements

1.16.1 Vibrating Alert (Redundant Notification)

- In cases where voice feedback is not possible or not understood, the device should be able to warn by **vibration**.
- This is especially important in quiet environments or for dual-impaired users with hearing impairments.

1.16.2 Supervised Access Support

- The system must be compatible with iOS's "Guided Access" feature.
- This can prevent stepping out of the app, making it easier for visually impaired users to control.

1.17 Packaging Requirements

The developed YKE (mobile application) is designed for digital distribution only. There are no physical packaging, labeling or packaging requirements for the distribution of the application.

1.17.1 Digital Distribution

- The application will be digitally packaged in .ipa (iOS App Package) format.
- Distribution to the user will be done via manual upload or Apple App Store for the duration of the project.
- The deployment file must clearly identify the application name, version and developer information.

1.17.2 Physical Packaging Requirement

- No physical tape, disk, USB, box, label or physical packaging element is required.
- The system is completely software based and will be installed and updated digitally.

1.18 Priority of Requirements and Criticality

1.18.1 Critical Requirements (High Priority)

If these requirements are not met, the system b e c o m e s a safety hazard to use or is unable to perform its basic functions.

- **Safety requirements:** Combating false positive and negative outputs, warning on error
- **Voice feedback function:** Obligation to notify the user by voice
- **Timing requirements:** Feedback to be given within a maximum of 4 seconds
- **Camera access and model operability:** The system must stop and warn if it cannot perform basic operations

1.18.2 Key Requirements (Medium Priority)

Meeting these requirements increases the availability, flexibility and long-term success of the system, but their absence does not completely prevent the system from working.

- **Offline operation:** The system can work without an internet connection
- **Modular design:** Model files can be easily modified
- **Language support:** Ability to provide voice feedback in different languages (for future versions)

1.18.3 Supporting Requirements (Low Priority)

It increases the comfort, user satisfaction or expandability of the system. However, its absence does not affect the basic functions.

- **Vibrating warning support**
- **Help mode and repeat function**