SOFTWARE DESIGN DOCUMENT

TABLE OF CONTENTS

1 SCOPE

1.1 Definition

This document is the software design description (SDD) of the "Object Detection for the Visually Impaired" mobile application developed within the scope of BM-314 Software Engineering course.

The system includes an artificial intelligence-supported mobile application that detects objects in the environment using the mobile device camera, estimates their distances and transfers this information to the user with audio feedback in order to increase the environmental awareness of visually impaired individuals.

The project targets the iOS operating system and Swift language, CoreML, AVFoundation and Text-to-Speech APIs were used in the development process.

- **Project Name:** Object Detection Application for the Visually Impaired
- **Abbreviation** GENET
- **Version Number** 1.0
- **Publication Number:** SDD-001
- **Target Platform:** iOS 15.0 and above
- **Developers:** Furkan Egecan Nizam, Dilay Ece Maral
- **Publication Date:** 14.05.2025

This document aims to explain in detail the overall structure of the system, subsystem designs, data flows and interactions between components.

1.2 System Overview

This document is the software design document of the "Object Detection for the Visually Impaired" mobile application developed within the scope of BM-314 Software Engineering course.

The main purpose of the system is to enable visually impaired individuals to detect objects in their environment, to obtain the location and distance information of these objects and to communicate this information to them through audio feedback. The system aims to help the user gain environmental awareness and help them to move independently in their daily life activities.

The system is developed to run on iOS-based mobile devices and is designed to work completely offline. Environmental images are captured through the mobile device camera; object detection and depth estimation are performed with CoreML-based artificial intelligence models. The information obtained is transferred to the user by voice using the iOS Text-to-Speech API.

The system development process was managed within the framework of the Scrum model. For operation and maintenance processes, software updates and version improvements are planned after the App Store distribution of the application.

Currently, the system is developed for individual use only, but expandability features such as multi-language support, external data integration and support for new model versions are planned in the future.

Interested parties:

- **Client** BM-314 Software Engineering Course Instructor Prof.Dr.Hacer Karacan
- **User** Visually impaired individuals
- **Developer:** Furkan Egecan Nizam, Dilay Ece Maral
- **Support Unit** Project developer team (for maintenance and updates)

Current and planned functional components:

- Object Detection Module
- Depth Estimation Module
- Voice Feedback Module
- Camera Interface
- User Interface (start/stop commands)

Other related documents:

- Software Project Management Plan (SPMP)
- Software Requirements Specification (SRS)
- Software Test Documentation (STD)

1.3 Document Overview

This document is prepared to describe in detail the software design (Software Design Description - SDD) of the "Object Detection for the Visually Impaired" mobile application. The document comprehensively presents the overall architecture of the system, the functions of the subcomponents, the data flows between the components, the technologies used and the design decisions.

The purpose of the document is to create a common reference source for all team members, maintenance personnel and relevant stakeholders in the software development process, and to ensure that the system is developed in an accurate, consistent and secure manner. It is also intended to facilitate understanding of the structure of the system for future updates, improvements or bug fixes.

The design described in this document has been prepared with special safety precautions in mind to ensure the safety of visually impaired individuals. In particular

- All system operations are designed to be performed offline, protecting data confidentiality.
- Minimum confidence scores have been set and error management mechanisms integrated to prevent false positive or false negative detections.
- Safe mode mechanisms that alert the user in critical error situations are included in the document.

This document also covers in detail the definition of system components, their functional distinctions, data flow within the system, external interfaces and internal interfaces, and clearly outlines design decisions to support the safe and accessible operation of the system.

2 RELATED DOCUMENTS

No	Document Title	Publication Number	Amendment Number	History	Source
1	Software Project Management Plan (SPMP)	SPMP-001	0	19.03.2025	Project Developer Team
2	Software Requirements Specification (SRS)	SRS-001	0	16.04.2025	Project Developer Team
3	Software Design Description (SDD)	SDD-001	0	30.04.2025	Project Developer Team
4	Software Test Description (STD)	STD-001	0	Planned	Project Developer Team

Explanations:

- The documents were sourced by Furkan Egecan Nizam and Dilay Ece Maral, the project development team.

3 YKE-WIDE DESIGN DECISIONS

This chapter describes the system behavior of the "Object Detection for the Visually Impaired" mobile application from the user's point of view and the key decisions that affect the selection and design of software units. The design decisions w e r e made to ensure that the system is safe, fast, accessible and sustainable.

3.1 Input and Output Related Design Decisions

- Images obtained from the camera of the mobile device are used as system input.
- Images are acquired at regular intervals (every 2 seconds) using the AVFoundation API and converted into CoreML-compliant format.
- The outputs are presented to the user in the form of voice feedback. Through the Text-to-Speech API, the object's location, object name and distance information are combined to create a spoken message.
- In the first version, no visual output is used, only audio information is provided to the user.

3.2 Design Decisions on ACE Behavior Based on Inputs and Conditions

- If camera access is not available, the system notifies the user audibly and switches to safe mode.
- In cases where no object is detected, the user will receive the message "No object detected".
- If the depth estimation fails, the system gives an audible warning "Distance not estimated".
- Objects with a confidence score above 60% are considered to reduce the probability of false positives and false negatives.
- In critical error situations (for example: model freeze), the system stops processing and informs the user.

3.3 Database / Data File Related Design Decisions

- The system does not store persistent data, but only operates in temporary RAM memory.
- Artificial intelligence model files (in .mlmodel format) are embedded in the application.
- The image data received from the user is not stored in the device in any way and is deleted from the memory when the process is completed.

3.4 Approach to Meeting Safety, Security and Privacy Requirements

- All operations are performed offline; the system does not share data with the outside world.

- Camera and microphone access is only active when the user gives permission.
- Compliance with data protection regulations such as GDPR and KVKK is ensured; user data is not collected or stored.
- In critical error moments, the system is put into safe mode by giving an audible warning to the user.

3.5 Approach to Ensure Flexibility, Availability and Maintainability

- AI models are integrated in a modular way; different models or improved versions can be easily added in the future.
- The application has been developed in Swift language for the iOS platform and is configured so that all updates can be distributed through the App Store.
- Since the user interface is designed with minimal and voice interaction, accessibility and ease of use are prioritized.
- The system design has been optimized to run on devices with low processing power (e.g. TensorFlow Lite transformations).

4 STRUCTURAL DESIGN OF YKE

4.1 Components of the ACE

a) Software Units and Identifiers

The application consists of the following basic software units:

Software Unit	Identifier Code	Description
Camera Management Module	YKE-CAM	It allows to capture images from the camera of the mobile device.
Object Detection Module	YKE-OBJ	Detects objects from images (uses CoreML model)
Depth Estimation Module	YKE-DEP	Estimates the distance of the detected object (uses CoreML model)
Voice Feedback Module	YKE-TTS	Transmits voice information to the user (Text-to-Speech API)
Control and Coordination Module	YKE-CTRL	Coordinates the data flow between modules and the overall operation of the system.

b) Purpose of Software Units and Related ACE Requirements

- **YKE-CAM (Camera Management Module):**
  This module provides access to the mobile device's camera and captures images at set intervals (e.g. every 2 seconds). It transfers the captured images to the Object Detection Module for processing. Camera permission, image resolution and capture frequency are managed by this module.
  Related Requirements: Camera Interface (1.3.3), Internal Camera→ Object Detection Data Stream (1.4.1).

- **YKE-OBJ (Object Detection Module):**
  This module performs object detection on images taken by the camera. Using a pre-trained CoreML model, it determines whether there is an object in the image, its class (e.g. chair, table) and its position (right, left, opposite).
  Only objects above a 60% confidence score are considered.
  Related Requirements: Object Detection Functionality (1.2.1), Model Interface (1.3.5).

- **YKE-DEP (Depth Estimation Module):**
  This module estimates the distance of detected objects from the device. Using the Depth Estimation model, the approximate distance of the object (e.g. 1.5 meters) is calculated. The calculated distance is passed to other modules to prepare for the voice notification.
  Related Requirements: Depth Estimation Functionality (1.2.2), Model Interface (1.3.5).

- **YKE-TTS (Voice Feedback Module):**
  This module combines the detected object and distance information and presents it to the user as a voice. The specified message format (such as "There is a chair 2 meters away on the right side.") is spoken with the iOS Text-to-Speech API.
  Related Requirements: Voice Feedback Functionality (1.2.3), Voice Output Interface (1.3.4).

- **YKE-CTRL (Control and Coordination Module):**
  It ensures synchronization of all modules within the system. It manages the camera, object detection, depth estimation and voice notification steps to work in the correct order. It also detects error conditions, activates safe mode mechanisms and informs the user.
  Related Requirements: Management of Internal Interfaces (1.4), Safety Requirements (1.7).

c) Development Status / Type

Software Unit	Development Status	Explanations
YKE-CAM	New Design	Developed with Swift and AVFoundation API.
YKE-OBJ	Intra-application Use of the Existing Model	The pre-trained CoreML object detection model (v1.0) is used.
YKE-DEP	Intra-application Use of the Existing Model	The pre-trained CoreML depth estimation model (v1.0) is used.
YKE-TTS	Platform API Usage	iOS Text-to-Speech API is used.

YKE-CTRL	New Design	A structure that coordinates all modules has been developed using Swift language.

Current Used Models:

- Object Detection Model: Version 1.0, CoreML format, offline use, documentation available in the project folder.
- Depth Estimation Model: Version 1.0, CoreML format, offline use.

d) Use of Computer Hardware Resources

Source Type	**Planned Use**
Processor (CPU):	During object detection and depth estimation, processor utilization is expected to be between 50-60%.
Memory (RAM):	An average memory usage of 300-400 MB is foreseen.
Input/Output:	The camera will capture images every 2 seconds and audio outputs will be provided in real time.
Storage	Approximately 200 MB of permanent storage is required for application and model files.
Communication:	Since the application is designed to work offline, there is no need for a network connection.

4.2 General Concept of Execution

This section describes the execution sequence and dynamic relationships between software units in the "Object Detection for the Visually Impaired" application.

The system operates synchronously between the modules in real time through data flow and control flow. The working concept is explained in detail below:

Execution Flow

1. **System Initialization:**
    - The Control Module (YKE-CTRL) becomes active when the user starts the application.
    - Performs system startup checks (such as camera permission, model loading).
2. **Camera Data Collection:**
    - The Camera Management Module (YKE-CAM) is started.
    - Every 2 seconds a new image is taken from the device's camera.
3. **Object Detection:**
    - The received image is transmitted to the Object Detection Module (YKE-OBJ).
    - The model detects objects on the image.
    - If one or more objects with a confidence score above 60% are detected, the location and object class information is extracted.
4. **Depth Estimation:**
    - If the object is detected, the same image is transferred to the Depth Estimation Module (HDE-DEP).
    - The depth model estimates the distance of the object to the device.
5. **Voice Feedback:**
    - The name, position and distance of the object are transmitted to the Voice Feedback Module (YKE-TTS).
    - This information is translated into sentences using the Text-to-Speech API and spoken to the user.
6. **Continuity:**
    - All these steps are repeated every 2 seconds and the system runs live all the time.

Dynamic Relationships and Situations

- **Control Flow:**
    - The YKE-CTRL module triggers all modules in sequence: first the camera, then object detection, then depth estimation and finally voice notification.

- **Data Flow:**
    - Data is transferred sequentially between Camera → Object Detection → Depth Estimation → Voice Notification modules.
- **State Transitions:**
    - The system operates between 4 basic modes:
        - **Ready Mode:** Initial checks are performed.
        - **Active Mode:** Object detection and audio notification loop active.
        - **Error Mode:** If the camera is not accessible or the model does not work, the user will receive an error message.
        - **Closing Mode:** All modules are closed when the application is closed.
- **Administration of Exceptions:**
    - If the camera image cannot be acquired or the model estimation fails, the system switches to safe mode and the user is notified ("Camera inaccessible", "Object not detected", etc.).
- **Timing Relationships:**
    - All operations must be completed within a maximum of 4 seconds after each image capture (camera → detection → distance → audio output).
- **Concurrency**
    - The main working loop processes sequentially; multi-threading is not used directly.
    - iOS asynchronous operations at the system level (e.g. camera data flow and TTS) are managed.

Operation Diagram (Simplified Flow Diagram)

Figure 1 shows the working diagram of the Object Detection Application for the Visually Impaired.

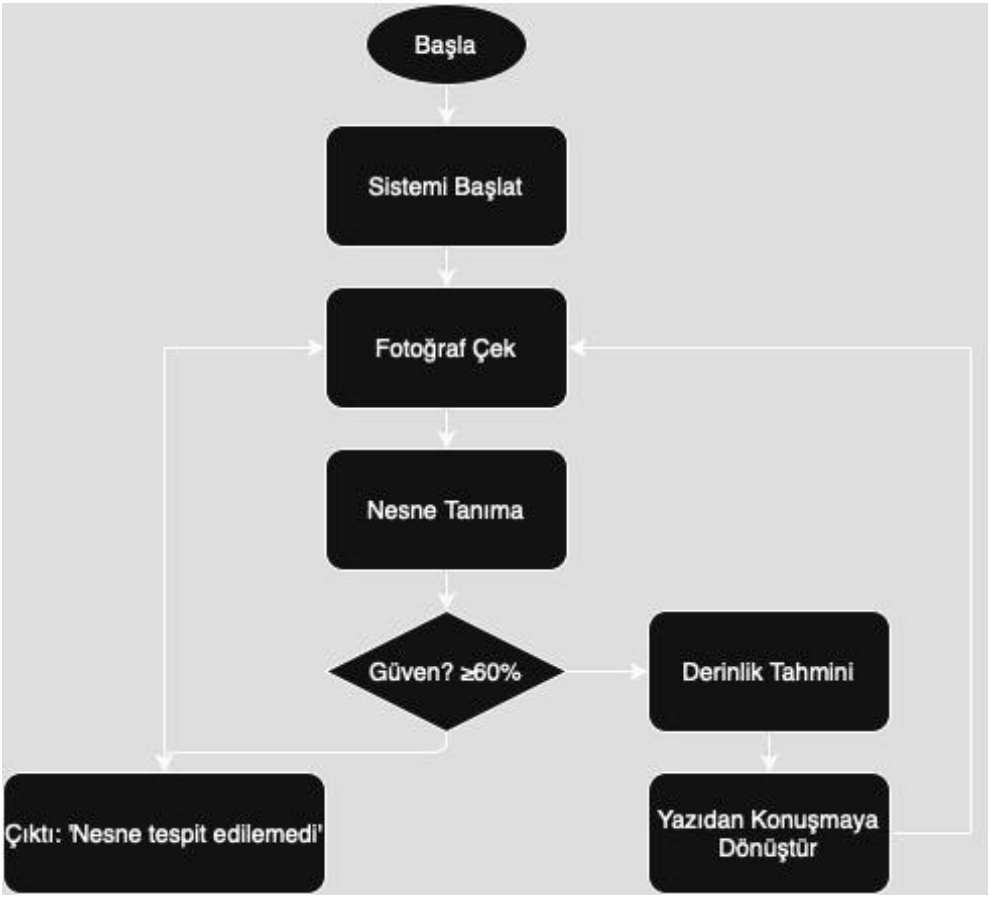

Figure 1 - Working Diagram of Object Detection Application for the Visually Impaired

Special Considerations

- **Dynamic Memory Usage:**
  Images are temporarily stored in RAM and deleted after processing.

- **Dynamic Object Management:**
  Each detected object is processed individually and information is managed for up to 5 objects.

- **Timing and Priority:**

  - Priority ranking: Camera image acquisition> Object detection> Distance estimation> Audio feedback.

  - Voice notification duration is limited to a maximum of 3 seconds.

- **Error Management:**
  In the event of critical errors, the system stops operations and provides the user with a secure error message and voice notification.

4.3 Interface Design

In this section, the design of the interfaces in the "Object Detection for the Visually Impaired" mobile application is described.

For each interface, a project-specific identifier is assigned, indicating which system, module or user the interface interacts with. It is also indicated whether the interfaces use existing APIs or were developed within the scope of the project.

Interface List and Definitions

Interface Identifier	Interface Name	Elements (Related Modules)	Status (Existing / Developed)	Version	Related Document
INT-001	Camera Interface	YKE-CAM ↔ Camera Hardware	Available (AVFoundation API usage)	iOS 15+	SRS 1.3.3
INT-002	Camera Interface	YKE-CAM → YKE-OBJ	Developed	v1.0	SRS 1.4.1
INT-003	Depth Estimation Interface	YKE-OBJ → YKE-DEP	Developed	v1.0	SRS 1.4.2
INT-004	Voice Feedback Interface	YKE-DEP → YKE-TTS	Developed	v1.0	SRS 1.4.3
INT-005	User Interface (Voice Commands)	YKE-TTS ↔ User	Available (supported with iOS Text-to-Speech API)	iOS 15+	SRS 1.3.2

Descriptions of Interfaces

- **INT-001 Camera Interface:**
  Mobile device from his camera image intake is done. AVFoundation API is implemented using It is a ready-made (existing) iOS API, not developed.
- **INT-002 Object Detection Interface:**
  The image taken by the Camera Module is transferred to the object detection module. This interface system

is a data transfer and formatting process (newly developed).

- **INT-003 Depth Estimation Interface:**
  The object information marked by the Object Detection Module is transferred to the Depth Estimation Module. This internal interface was specially developed within the scope of the project.

- **INT-004 Voice Feedback Interface:**
  Depth estimation results are transmitted to the Voice Feedback Module for voice transmission to the user. This interface was developed for data formatting and voice output within the application.

- **INT-005 User Interface (Voice Commands):**
  It is an interface that provides voice information to the user. iOS is built using the Text-to-Speech API. An existing system API was used, no development was required.

Special Notes:

- **Available Interfaces:** iOS system APIs such as AVFoundation and Text-to-Speech (unchanged).

- **Developed Interfaces:** Inter-module data exchange (INT-002, INT-003, INT-004) was designed from scratch and integrated into the application.

- **Version Information:** The first version (v1.0) has been accepted for the interfaces developed in the project.

5 YKE DETAILED PLAN

In this section, detailed planning is made for each software unit defined in the "Object Detection for the Visually Impaired" mobile application.

For each software unit, design decisions, technologies used, data flows, processing logic and special cases are described.

5.1 YKE-CAM (Camera Management Module)

a) Unit Design Decisions

- It was decided to take an image from the device camera every 2 seconds using the AVFoundation library.

- The image format is set to CVPixelBuffer (for CoreML compatibility).

b) Restrictions / Limitations

- User permission is required for camera access.

- Image quality may be reduced in poor lighting conditions.

c) Programming Language

- Swift programming language is used. AVFoundation library is optimized for Swift.

d) Functional Commands

- Camera startup, image acquisition and shutdown functions are provided by AVFoundation functions:
    - AVCaptureSession, AVCaptureDeviceInput, AVCaptureVideoDataOutput.

e) Data Flow

- Input Camera image (RGB format).
- Output: CVPixelBuffer data structure passed to the object detection module.

f) Logic and Dynamic Controls

- The camera runs in a continuous loop when turned on.
- A new image is acquired and processed at the specified shooting interval.
- If an error occurs in the camera connection, the user will be warned audibly.

5.2 YKE-OBJ (Object Detection Module)

a) Unit Design Decisions

- A pre-trained CoreML model (with mobile optimization) is used for object detection.
- Objects below the 60% confidence score threshold are ignored.

b) Restrictions / Limitations

- The model supports input size 640x480 resolution.
- Up to 5 objects can be processed simultaneously.

c) Programming Language

- Swift language is used and CoreML framework is integrated.

d) Functional Commands

- Model loading and inference:
    - MLModel, VNCoreMLModel, VNCoreMLRequest.

e) Data Flow

- Input CVPixelBuffer Image.
- Output: Object class, location, confidence score information.

f) Logic and Dynamic Controls

- The model is run on each new image.
- Results above a 60% confidence score are taken, others are discarded.
- If the detection fails, an error message is generated.

5.3 YKE-DEP (Depth Estimation Module)

a) Unit Design Decisions

- A separate CoreML model was used for depth estimation.

b) Restrictions / Limitations

- The model only works on image segments with detected objects.

c) Programming Language

- Swift language+ CoreML usage.

d) Functional Commands

- Depth prediction functions:
    - Loading MLModel and estimating depth with VNCoreMLRequest.

e) Data Flow

- Input: Cropped image of the detected object.
- Output: Distance estimate (in meters).

f) Logic and Dynamic Controls

- The closest object is prioritized.
- If the depth estimation fails, the system generates an error message.

5.4 YKE-TTS (Voice Feedback Module)

a) Unit Design Decisions

- Natural voice output is produced using the Text-to-Speech API.

b) Restrictions / Limitations

- The maximum message length should not exceed 3 seconds.
- Voice responses must be uninterrupted.

c) Programming Language

- Swift language+ AVSpeechSynthesizer API.

d) Functional Commands

- Voice message creation and playback commands:
    - AVSpeechUtterance, AVSpeechSynthesizer.

e) Data Flow

- Input: Object name, location and distance information (string).
- Output: Voice message.

f) Logic and Dynamic Controls

- The current message is completed before a new voice message starts.

- In case of error, the default audible error message is played.

5.5 YKE-CTRL (Control and Coordination Module)

a) Unit Design Decisions

- Data flow and sequence control between all modules is provided by this module.

b) Restrictions / Limitations

- Delays between modules should be maximum 1 second.
- In case of error in any module, the system switches to safe mode.

c) Programming Language

- Swift language.

d) Functional Commands

- Module triggers and error handling:
    - DispatchQueue, Error Handling structures.

e) Data Flow

- Camera→ Object Detection→ Depth→ Voice Notification order is maintained.

f) Logic and Dynamic Controls

- When each cycle of the system is completed, a new cycle is started automatically.
- Safe shutdown of the system is ensured when an error is detected.

6 TRACEABILITY OF REQUIREMENTS

In this section, the traceability of each software unit defined within the scope of the Software Design Document (SDD) in the "Object Detection for the Visually Impaired" application with its allocated ACE requirements is provided.

In addition, traceability from Software Configuration Elements (SCE components) to the relevant software units is also presented in tables.

6.1 Traceability from Software Unit to ACE Requirements

Software Unit	Allocated ACE Requirements
YKE-CAM (Camera Management Module)	SRS 1.3.3 Camera Interface, 1.4.1 Internal Camera → Object Detection Interface
YKE-OBJ (Object Detection Module)	SRS 1.2.1 Object Detection Functionality, 1.3.5 Model Interface
YKE-DEP (Depth Estimation Module)	SRS 1.2.2 Depth Estimation Functionality, 1.3.5 Model Interface
YKE-TTS (Voice Feedback Module)	SRS 1.2.3 Audio Feedback Functionality, 1.3.4 Voice Output Interface
YKE-CTRL (Control and Coordination Module)	SRS 1.4 Internal Interfaces, 1.7 Safety Requirements

6.2 Traceability from Software Configuration Element to Software Unit

Software Configuration Staff	Allocated Software Unit
Camera Interface (AVFoundation API)	YKE-CAM
Object Detection CoreML Model	YKE-OBJ
Depth Estimation CoreML Model	YKE-DEP
Text-to-Speech API (iOS)	YKE-TTS
Control Flow Between Modules	YKE-CTRL

7 NOTES

This chapter contains various general information, terms used, abbreviations and explanations in order to provide a better understanding of the Software Design Document (SDD) of the "Object Detection for the Visually Impaired" mobile application.

7.1 Abbreviations

Abbreviation	Description
YKE	Software Configuration Staff
YTD	Software Design Document
SRS	Software Requirements Specification
SDD	Software Design Description
SPMP	Software Project Management Plan
STD	Software Test Documentation
API	Application Programming Interface
TTS	Text-to-Speech (Text-to-Speech Conversion)
AVFoundation	iOS Camera and Sound Framework (Apple Framework)
CoreML	Apple Machine Learning Model Run Framework

7.2 Terms and Definitions

Term	Definition
Object Detection for the Visually Impaired	The process of detecting objects in the environment using the mobile device camera, estimating their distance and providing audio feedback to the user.
Software Configuration Element (SCE)	A software unit that is a structural part of the system and can be developed and managed independently.
CoreML Model	Optimized artificial intelligence model file (.mlmodel extension) that can perform machine learning inference on mobile devices.
Text-to-Speech (TTS)	Technology that converts written text into audio output in natural human voice format.

AVFoundation	Built-in iOS API library provided by Apple, used for image and sound processing.
Trust Score	The probability value (e.g. 60%) that the model thinks an object is correct as a result of an object detection.
Safe Mode	When errors are detected in system operation a special operating mode that gives the user a safe notification and stops the system before it can take risky action.
Inference	The process of a machine learning model making predictions on a new data input.

7.3 Background Information

- This project was realized within the scope of **BM-314 Software Engineering** course.
- The app is designed to work only on **iOS 15.0 and above**.
- **Scrum** method was adopted in the development process, and iterative development and continuous testing principles were applied.
- The entire system design has been developed with a focus on **offline operation**, **privacy**, **accessibility** and **security**.

8 APPENDICES

There is no additional content for this document.