Mint Vision Application Architecture

03 June 2021

Created by Peter Reid

1 Introduction

This document outlines the Architecture of the Mint II Vision Product

The Mint II Vision Product combines a Universal Windows application with Azure Cognitive computing, storage and processing to leverage the best of both worlds. The full architecture of the system is described below.

All the components built below are Mint's Intellectual Property. For components that reside in Azure, such as the CosmosDB and Cognitive Services, these are Microsoft's Services but the manner in which our code consumes these services is our Intellectual Property.

2 Current Architecture

The diagram below represents the current architecture of the application.

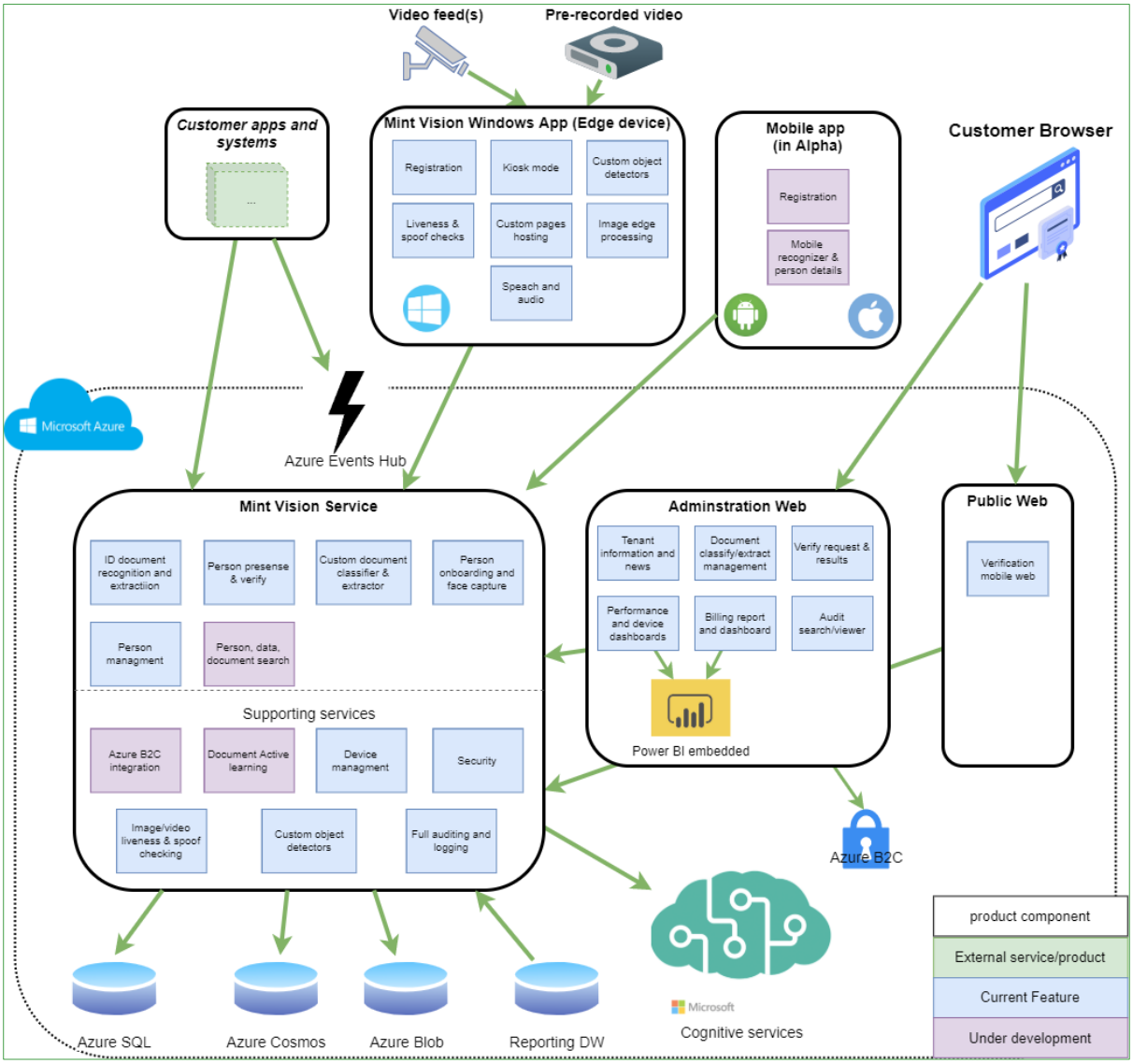

Figure 1: Mint II Vision Application Architecture

The individual components are described below.

2.1 Client Device

The Client Device is a Universal Windows Application. This application runs in Kiosk Mode on any Windows 10 Device, including embedded devices such as Raspberry Pi's.

The application has several components that are noteworthy:

- The application can run in online or offline mode
- The application can bind to streaming or usb cameras
- The application has a local storage solution that is then synchronized to the cloud

- The application can embed custom pages such as web pages, which are used to extend the application into several different use-cases

2.2 Azure

The application relies on several technology layers in Azure. Each layer is indicated below.

2.2.1 Storage

2.2.1.1 Images

Images that are captured of each face event are stored as images in Azure. This means that minutes after a capture event, the images and audit entries are stored in the cloud and retrievable from anyhere in the world.

2.2.1.2 Data

Data for the application, such as users, audit trails and settings, are stored in Azure CosmosDB. They are stored as documents and synchronized regularly to the devices themselves.

Cognitive Services

The application uses Microsoft Cognitive Services for

- Face Detection
- Face Recognition
- Similar Face Matching

2.2.1.3 IoT Hub

The IoT Hub is used to co-ordinate messages to and from the devices. For example, messages can be sent to each device to force an upgrade to a newer version of the application. We also send a “HeartBeat” message to the IoT Hub to facilitate detection of when a device is down.

2.2.1.4 Cognitive Services

The Vision Application uses Cognitive Services to do face recognition actions. In addition, the OCR service is used for processing Drivers’ licenses and ID documents.

2.3 Camera Streaming Service

In order to consume the stream from network cameras, a separate streaming server is deployed with the application. This is a service that is deployed using containers either on-premises, independently in Azure or within a Kubernetes cluster. The service consumes streaming camera in a multitude of formats, applies any necessary image correction, and then publishes the stream to the application in order to be consumed by the system.

2.4 Mint Vision Service

The Vision Processing Service is a REST-based, container-distributed service that performs Image Processing services such as Object Detection, Object Recognition, and similar

machine-learning-based services. This processing engine is external to the application so that powerful GPU-based servers can process the images.

2.5 Queue Service

Hosted external to the application (often as a service in Azure) is the Queue Management service. The Queue Management Service allows for people queueing, such as at a branch at a bank or other such use cases.

2.6 Extension Services

The application is designed to be extended to several different use-cases. The extension part of the application combines two pluggable items:

- Custom Pages. For example, pages that serve content for touchless purchasing in a canteen differ vastly to pages that dispense medication
- Custom integration. For example in a university setting the application connects to the Student Database to synchronize student details
- Custom Triggers. When using the application for example as an Access Control mechanism, the triggers open or close doors. When in a canteen, the triggers can cause vending machines to dispense, etc.

2.7 Edge Device

2.7.1.1 UWP Extension Pages

UWP Extension Pages are embedded in the application, and represent Single Page Applications that can be loaded into he application once a face is recognised.

2.7.1.2 Custom Trigger Component

When triggering hardware, such as doors or coffee machines, each trigger is slightly different. The externalised trigger mechanism allows for any trigger to listen to events on the application and trigger them as and when they occur.

2.7.1.3 Custom Local Image Pipeline

The application uses the execution of ONNX models to perform custom vision processing. The custom local image pipeline is a pluggable pipeline that uses the image processing engine to detect and classify objects and sequences.

2.7.1.4 Web Connector

The web connector serves as a bridge between the face detection routines and the Custom Extension Pages.

2.8 Queue Service

The Queue Service performs queueing of people, objects (for example on a conveyor belt, or medicine) and exposes this as an API to the end-user.

2.9 External Services

2.9.1.1 ID Service

We validate South African ID's through a third-party service. Although this is implemented in a pluggable architecture, at present there is a single provider that we use.

2.9.1.2 Reporting

All reporting in the system is implemented through PowerBI, using our Cosmos and SQL databases as a data source.

3 Sample Reports and Dashboards

3.1 Reports and Dashboards

Standard Crowd-Tracking reports are available out-of-the-box. Extension reports are available to be customised on request.

(Note: Some elements on the *example* dashboard that relate to the full user journey, as illustrated below, are only applicable in some use-cases.

3.1.1 Sample Crowd Tracking Report (1)

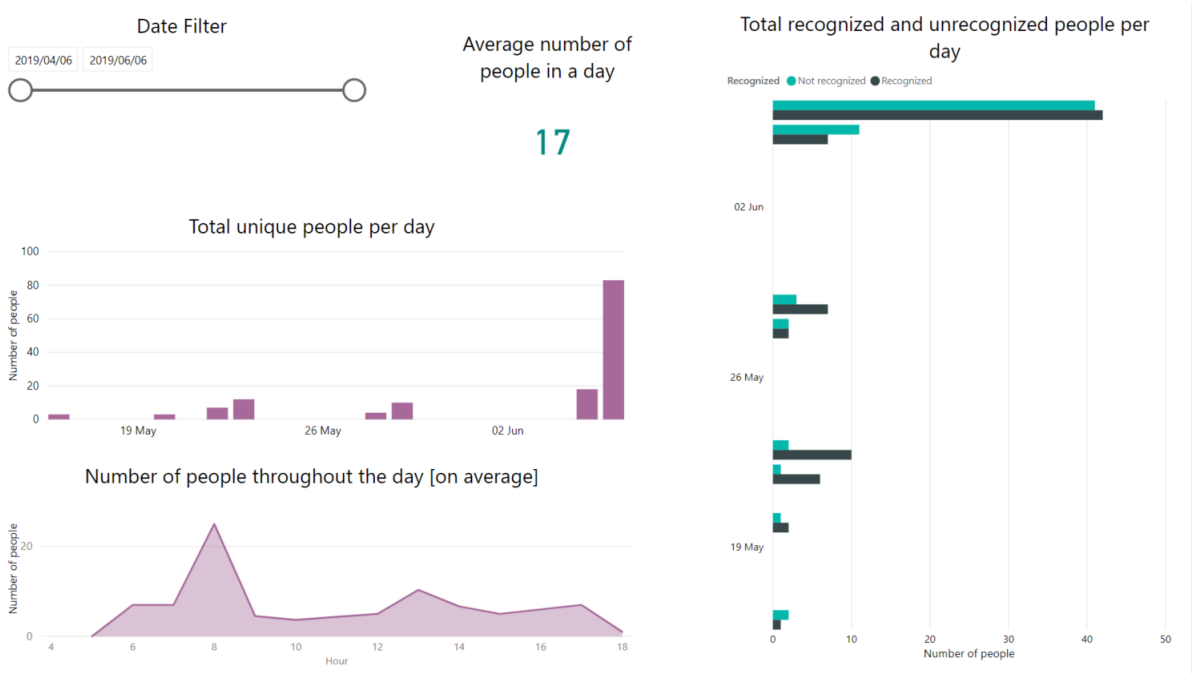

Figure 2: Sample Crowd Tracking Report

3.1.2 Sample Crowd Tracking Report (2)

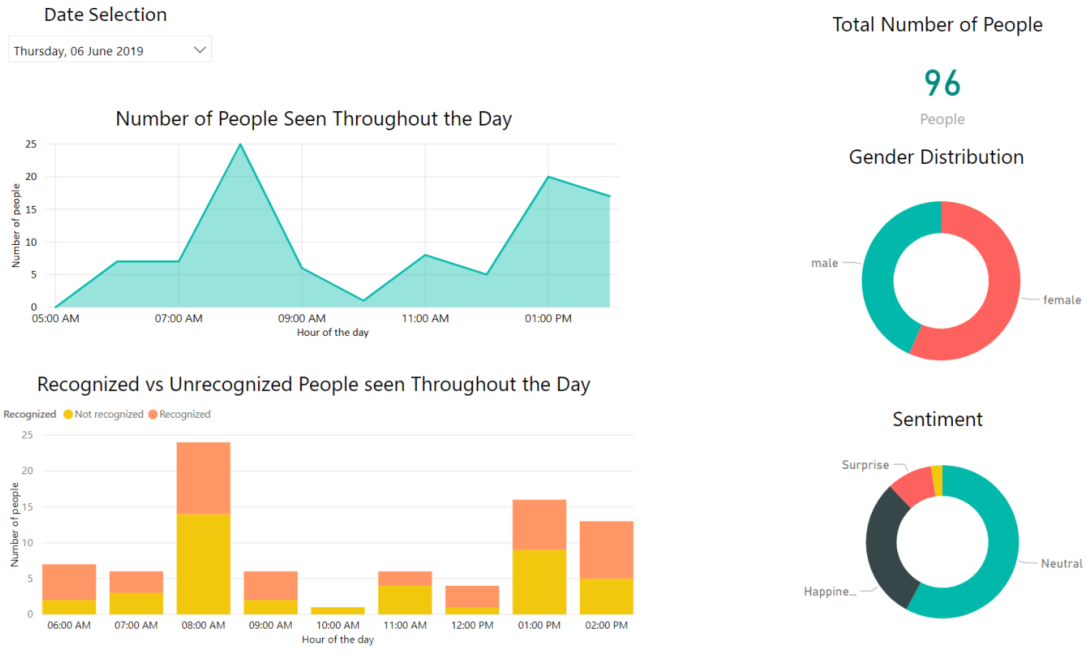

Figure 3: Sample Crowd Tracking Report

3.1.3 Sample Queue Report

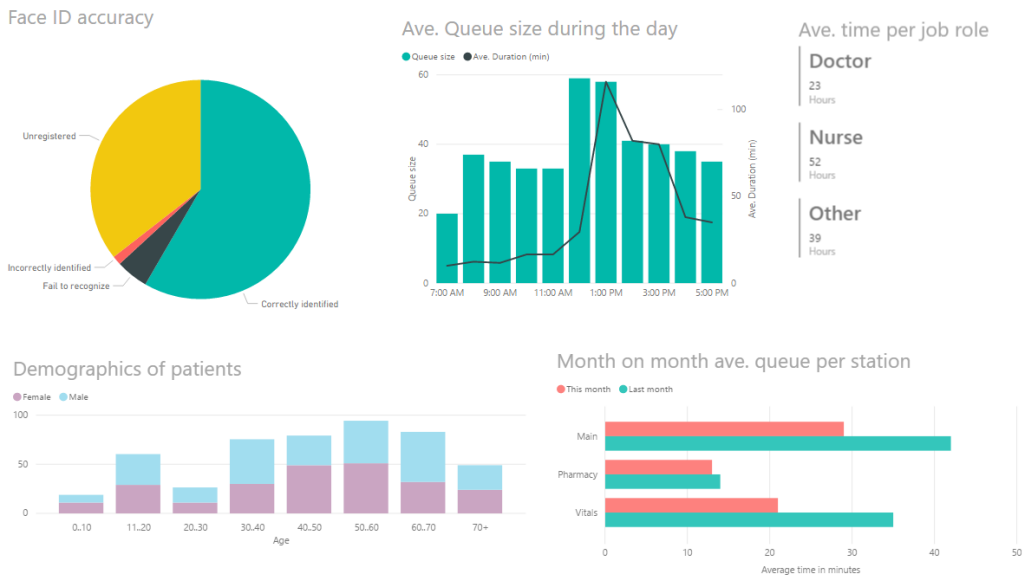

Figure 4: Sample Queue Report