Ocularis

Empowering the visually impaired

# The team

Ehsan Mirsaeedi, Shirin Abdolahi

# The Concept

The promise of technology and all its advancements is to bring a better world through solving inherent problems of human society and its individuals. The predominance of smart devices is obvious to an extent that it is impossible to imagine a world without them around. However, today technology is heavily biased towards people with no impairment. There are some explanations to tendency to normal people such as smaller market size of impaired people and their feeble collective voice for speaking of their needs. Apart from non-technical impediments, there are also some major technical challenges in providing these people with high quality services on par with natural human capabilities. Though, we believe that recent remarkable achievements in Machine Learning has put us in a unique era of human history that we can easily overcome many of those challenges, especially in case of visually impaired people.

Visual Impairment is a decreased ability to see to a degree that causes problems not fixable by usual means, such as glasses [1]. As a result, they need to rely on others for even their basic needs and questions such as knowing who they are talking to, what is their location, what’s in front of them, knowing the time and date, etc. We believe we can address a lot of their needs and confusions using Microsoft Cognitive Services and Windows IoT core cognitive capabilities along with a cheap hardware called Ocularis – Eye in Latin - that we intend to design and build. Ocularis is an intelligent buddy who can understand requests of its user via voice commands and provide its user with thoughtful responses through her voice. Ocularis tries to find answers to questions by gathering information from its built-in knowledge, internet, or from the environment that she sees through a high quality built-in camera. Ocularis is a small light-weight device that is supposed to get install easily in the forehead of its customer. In other words, we want it to be a buddy in the form a third eye.

1-2 paragraphs about your project. What problem are you trying to solve, and how are you solving it?

Make sure you answer the following questions:

* Does the project address a clear need, problem, or opportunity and is the solution clearly explained?
* Is the project’s purpose and basic functionality easily understood?

## Target Audience or Market:

According to World Health Organization, in 2018, 217 million people have moderate to severe vision impairment and 36 million people are blind [2]. Unfortunately, this number is increasing because of population growth and ageing problems, and consequences of unhealthy life styles.

Explain who your audience is. Consider targeted platforms, geography, and demographics. If you can estimate the size of your target audience please do so and cite your source for the estimated data.

### Personas

Create one or two personas who represent your likely users.

## Feedback

Pitch your core concept to a few people who would be in your audience or who have expertise in the kind of project you are making. Note here what you learned from them. The more sources of early feedback you include, the better.

# How it works

The hearth of Ocularis is empowered by a **UWP** application hosted on **Windows IoT Core**. This software operates a light-weight device which is supposed to place conveniently on the forehead of its user via a hat or band. Ocularis consists of a Camera, Audio I/O Module, Wi-Fi Module, Bluetooth Module, Cellular Module, a low consumption processor, and 1Gb of ram. These modules enable the device to interact with its user via voice and delegate the processing needs to the cloud. The variety of connectivity modules ensure us that Ocularis is always connected to internet. Having high quality internet connection is crucial to some functionalities because Ocularis is heavily dependent on **Microsoft Azure Cognitive Services** such as **Computer Vision** and **Bing** to bring an incredible value to its customer.

Apparently, internet connection introduces latency and uncertainty to our product. We have taken two measures to mitigate this problem. First, Ocularis tries to return the answer to its user’s request by going through **Ocularis Internal Knowledge Base** which is sufficient for some basic needs such as asking about time, date, calendar, or anything that is accessible from user’s phone like messages, emails, contacts. Secondly, we try to minimize the number of requests that we need to make to Azure Cognitive Services by using a server that acts as a mediator between Ocularis and Azure. Without this mediator, we would need several back and forth between Ocularis and Azure Cognitive Services to be able to serve some requests. However, by putting this Mediator in place, Ocularis just needs to send one request to the Mediator and then Mediator will be responsible to handle the required conversation with Azure.

In addition to **Ocularis Internal Knowledge Base**, this device has a built-in **Internal Image Processor** in order to alarm the user in case of detecting potential danger of falling down, getting out of a sidewalk, or getting close to an obstacle. This functionality is not supposed to be a replacement to guide dogs but a complementary to them. It helps visually impaired people to have a better quality of life and independence. It should be mention that, existing Depth and Obstacle Detection approaches do not guarantee 100% accuracy [,,], but we aim to implement and train the model which provides the most confidence and deterministic behavior []. This process needs to be done by the device and cannot be delegated to cloud because we need a near real-time alarming system to increase its reliability quality.

Lastly, to make the conversations between user and Ocularis seem intelligent and pleasant, Ocularis attaches a context to user requests and provide a response based on this new context and previous contexts that have changed the state of conversation. In other words, Ocularis does not interpret user’s commands in isolation but it takes the context of conversation and its state into consideration. As an example, when a user asks a question like “Who is the current prime minister of Canada?” and Ocularis notifies its user that she has found 5 relevant information, Ocularis initiate a state machine (context) to the pair of request and response. So, when the user asks for further questions such as “give me all your findings”, “give me the first relevant one”, or “repeat it again please”, Ocularis knows how to respond to the request according to the state of the conversation. **Ocularis State Manager** is responsible for transitioning through the possible set of states and even memorizing the state previous requests. As an example, a user can ask a question like “who **was** in front of me?” after getting the required information about Canada Prime Minister and since device has answered to the question “who **is** in front of me?” just before the question about Prime Minister, she can respond back quickly without sending requests to Azure Cognitive Services.

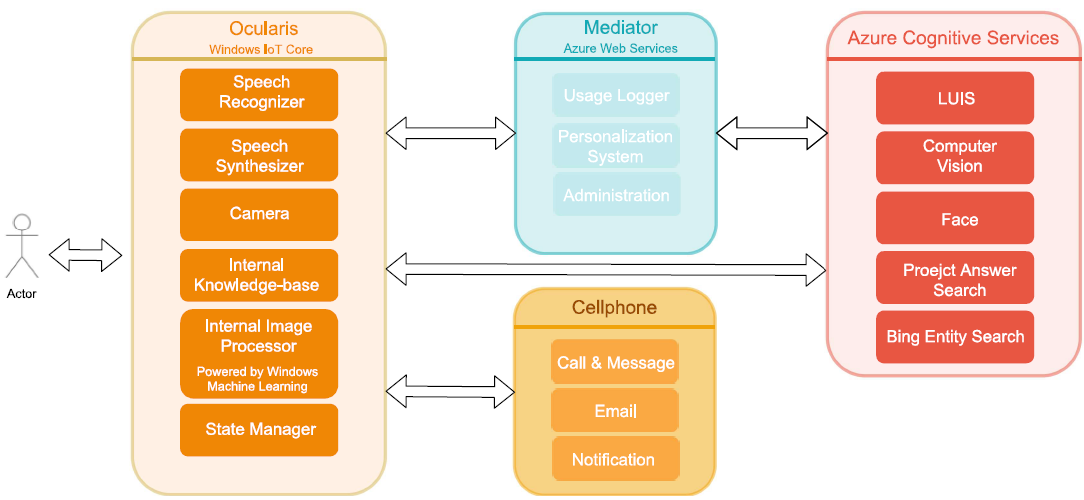


Figure 1 The Whole Architecture of Ocularis with its main components

## Interactions

Ocularis notifies the user or responds via its voice which is constructed by the built-in **Speech Synthesizer** of Windows IoT Core. Since Windows IoT Core supports **Speech Synthesis Markup Language**, Ocularis voice would be very similar to a real voice in terms of pronunciation, volume, pitch, rate and emphasis. User can hear the voice using a headset.

There are two major scenarios that Ocularis notifies a user of an event. First, when it detects the user is approaching a danger like an obstacle or stairs and secondly when the connected cellphone publishes a notification such as incoming call, email or text message. In either way, Ocularis, notifies the user in a clear voice and user can ask for further information such as “read it for me” or “I want to answer it back”.

At the same importance, user can ask several types of questions from Ocularis and wants him to answer back. When user asks a question, Ocularis maps the voice to one of intents that it supports.

## Intent Detection

Intents are building blocks of Ocularis which define its core capabilities. In other words, user requests get mapped to one of the built-in intents of Ocularis and that selected intent will be responsible for gleaning the information that user has asked for. Intents have the logic required for querying the information from intern and cloud knowledge sources.

We have designed two ways for analyzing user voice in order to detect the actual intent the user has. First, we use **Speech Recognition Grammar Specification (SRGS)** which is supported by UWP applications by default. A SRGS Grammar file gives us a great control over defining a complex set of commands and Windows IoT Core built-in **Speech Recognizer** is capable to capture semantics of a voice based on the SRGS file and map it to an intent. If offline recognition cannot be successful in capturing the intent, we use **Microsoft Language Intelligent Understanding (LUIS)** as the last resort. This service requires an active internet connection and only accepts texts as an input. So, we must convert the user’s voice to text before passing it to LUIS which requires another http connection and results in a long latency. Fortunately**, Bing Speech API** has a service that can help us with keeping the required HTTP connections at one. Therefore, we submit the user’s voice to Bing Speech API to ask it to convert the voice to text and query LUIS afterwards. Bing Speech API returns the text along with the extracted intent in its response.

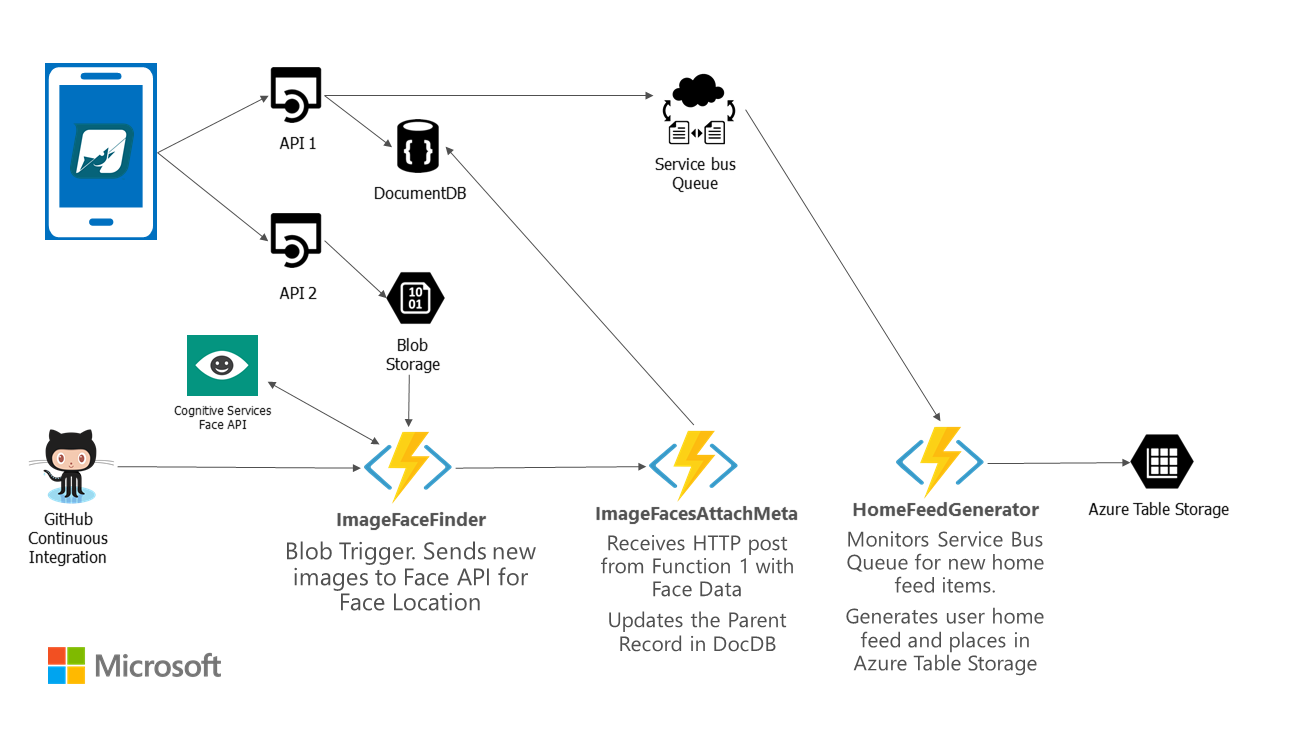
By capturing the intent, our next step is to serve the user by running the responsible intent handler.

## Intent Handling

## intelligence

Use this section to talk about how you built your product, how have Microsoft technologies made it feasible?

Include an architecture diagram in this section (example given below). If you need Microsoft-icons, you can [download them here](https://docs.microsoft.com/en-us/azure/architecture/resources/diagrams).



Think about the following questions, when writing this section:

* Does the project make effective and appropriate use of the major features of its chosen platform(s)?
* Were there significant platform features or even platforms the project could have benefitted from but failed to utilize?
* Does the project include innovations in technical design and/or implementation?
* Does the project include innovations in user experience?
* Does the project have a professional degree of production in terms of performance, user interface, visuals, and audio?

## Core Technologies

This includes your key platform(s) as well as specific technologies such as the Kinect SDK, the Unity game engine, or other elements that you believe will be critical to your project’s success.

# The Business Plan:

You don’t need to have a fully fleshed out business plan, we just want to know that you’ve put time into thinking about it.

## Competition:

What products, companies, or services will you directly compete with for your audience? How do you expect to differentiate your project from your competitors?

Think about the following questions in this section:

* Does the project create a new category of product or service?
* Does the project clearly and meaningfully innovate beyond existing products or services?

## Business Model

How will your project make money? If it provides an ongoing service, how will that service be sustainable? If there are critical partnerships you must make for your business model to work, describe those here as well.

Think about the following questions in this section:

* Does the team have a credible plan for getting their project to market in terms of business model, any required partnerships, or other factors?
* Does the project have a reasonable chance of success in its appropriate market given the team’s existing plan?

# Additional Information:

Add any links or information here. Any information you want to share is welcomed!