cover_logos.tifcover_logos.tifiOS Voice Command Idea

# Problem & Solution

## Dietary Intake Problem

Fact, astronauts need to take track of their dietary intake to be studied. The current way to track it is using the FFQ (Food Frequency Questionnaire) where they declare the food they have eaten. The main issues with this method are the time it takes to be filled, the difficulty to remember what they have eaten certain times, among others.

This generates the need to create a solution that replaces and improves this method through an iPad application considering the problem key factors and constraints.

## Idea Overview

Taking a look to the big picture, I think the application should be very *friendly* overall. With friendly I mean the application should behave in such a way to provide the feeling of having another friend in the station that cares about your healthy state of body and mind. It’s a known fact that eating makes us happy (feel better scientifically speaking). If we make astronauts eat in a happy environment, with a *friend a side* and no distracting tasks (remember, type, etc) we can manage to ease the FFQ process.

Considering the way of living on a weightless environment (also short spacing), astronauts should feel as happier as possible, avoiding uncomfortable situations such as filling forms, remember what they ate during a week, etc (*the problem*). If the problem is also approached from a psychological stand point it would benefit the whole tracking process overall.

For a rough start, *I’d recommend naming the application very friendly* something like “Commander Yummy”, “Dr. Meal” – a friend with authority – rather than a serious and scary “Food Frequency Questionnaire” (it represents what they are uncomfortable with already). I think having a friend to talk to would create a powerful habit to solve part of the problem. Imagine this scenario where you to talk to your Doctor, someone who cares about your health. And maybe astronauts start talking between them: “Hey did you *yum* the meal day already?”, “I’d like to have a juice now. I’ll tell this to my *commander*”.

Personal Note

From my personal experience when I worked in a software development company, they asked all developers to fill out online forms with very detailed information about what you did every day. You were requested to fill time lapses, type of activity, description, etc. Overall, it was very annoying, it’s hard to remember all that and also fill them out, waste of time. I would have been very thanked if I could speak to a friend about what I did and helped me process it, instead of talking to my girlfriend everyday about what I didn’t want to do (fill the form!).

## Voice recognition Features

Regarding to the voice recognition system, it should be an important part of the application to achieve the friend-environment level I’m proposing. My approach focuses on making the application another friend on the station by talking to the crew in certain moments, keeping an easy voice interaction process when the user wants to enter the information and also including a polite behavior when talking to the user. A good sample of this polite behavior could be seen in the Apple’s famous Siri.

The voice recognition system for iPad should cover the following features to successfully solve *the problem* and as an extra to fulfill my friend-environment approach.

* **Interactive friend-oriented conversations**; the application should talk to the user in a polite and probably fun way sometimes to work with the mood (important factor when eating).  
  *Quick sample*: An astronaut enters the FFQ home screen and the app immediately interacts “App: Welcome John, are you ready to tell me what you have eaten? John: Yes. App: Excellent! You want to tell me what to eat today? John: No, yesterday. App: Great. What delicious meal did you have yesterday? John: I had carrots, bread and orange juice. App: Sounds yummy. What time did you have this meal?” … and so on.
* **In-time food registry “Menu Style”**; usually it’s easy to talk to someone about what you want to eat at certain moment. If we add an in-time registry option combined with a sort of meal notification (see notifications below) it could save time and the difficulty of remembering a whole week meals for not filling the form right on time. It would be like talking to the waitress what you want to have for eating. The astronaut opens the app, this one says hi, after confirmation for in-time registry the app asks “What do you want to eat today John?” and goes on. It would be like talking to your waitress in some way. There’s also the possibility to add pre-loaded menus.
* **Voice recognition log in**; the voice recognition framework should provide the capability to identify each astronaut (voice-printing) previously to filling the FFQ.
* **Ambient noise low sensitivity**; it’s a known fact that there could be noise distractive elements in the environment. The framework should provide filters or just be efficient managing this issue.
* **In-app processing**; there is not access to internet for this application. The framework should be able to provide an off-line wide dictionary.
* **Wide vocabulary**; English will be the only language for this application (assumed). But if any other language is needed it would be nice to evaluate technologies that offer multiple and wide dictionaries.
* **System notification reminders**; notifications can become a powerful resource. There should be system notifications from the app for time-to-eat (should be configurable), after the end of the day (if someone didn’t fill anything in the day), end of the week (last call to fill the FFQ).

# Voice Recognition Frameworks-iOS Review

There are many frameworks out there for voice recognition in iOS applications. We can classify them onto three basic types: Text-To-Speech SDK’s, Speech Recognition SDK’s and Full Voice Support SDK’s (speak and listen).

I’ve reviewed them and collected the most useful ones (my favorites, marked with stars). Most of them are on-line depending, especially because it requires very big computational and memory resources (terabytes of memory) to implement an off-line library, and thus it's impossible to do that in the iPad or other device. My top favorite and recommended technology to use is **OpenEars** (the only off-line solution, considered the major constraint), followed by **Dragon SDK** and **iSpeech** (full voice support the three of them). Let’s study *the why* of these particular picks.

comparison.tif

*Key Features Comparison*

## OpenEars

OpenEars is a shared-source iOS framework for voice recognition and speech synthesis (TTS). It lets you easily implement round-trip English language speech recognition and text-to-speech on the iPhone and iPad and uses the open source CMU Pocketsphinx, CMU Flite, and CMUCLMTK libraries, and it is free to use in an iPhone or iPad app.

This is the only framework I implemented/tested; it was very easy to set up. I have to say it has some difficulties to interpret the voice commands in very crowded places (I did the testing in a coffee shop with my laptop mic – not recommended); however it can be improved with proper coding and use of existing algorithms.

There is a small delay before recognizing the voice (less than one second). There are paid extensions that help to improve response speech and add different fluid voices. The core technology is free of use for any purpose.

In Details:

* Listen continuously for speech on a background thread, while suspending or resuming speech processing on demand, all while using less than 4% CPU on average on an iPhone 4 (decoding speech, text-to-speech, updating the UI and other intermittent functions use more CPU).
* Use any of 9 voices for speech, including male and female voices with a range of speed/quality level, and switch between them on the fly.
* Change the pitch, speed and variance of any text-to-speech voice.
* Know whether headphones are plugged in and continue voice recognition during text-to-speech only when they are plugged in.
* Dispatch information to any part of your app about the results of speech recognition and speech, or changes in the state of the audio session (such as an incoming phone call or headphones being plugged in).
* Deliver level metering for both speech input and speech output so you can design visual feedback for both states.
* Get n-best lists with scoring.
* Test existing recordings.
* Be easily interacted with via standard and simple Objective-C methods.
* Control all audio functions with text-to-speech and speech recognition in memory instead of writing audio files to disk and then reading them.
* Drive speech recognition with a low-latency Audio Unit driver for highest responsiveness.
* Be installed in a Cocoa-standard fashion using an easy-peasy already-compiled framework.
* In addition to its various new features and faster recognition/text-to-speech responsiveness, OpenEars now has improved recognition accuracy.
* OpenEars is free to use in an iPhone or iPad app.
* Website: <http://www.politepix.com/openears/>
* Tutorial (very easy to implement): <http://www.politepix.com/openears/tutorial/>

## Dragon Speech sdk

It’s a powerful framework that uses the known capabilities of the Nuance Dragon voice-text applications engine (recently open to developers).

It technology is *tremendously powerful*; it’s the most complete one on the market, it’s expensive though. It allows to do voice-printing (identify persons by voice), speech to text, text to speech, voice recognition. It communicates to a web server to perform the speech tasks though, so it made me give it the 2nd place for my favorites. If internet connection wasn’t an issue this one would be my top favorite.

See this video of *Nina*, a virtual assistant (similar to Siri) application that uses most of the features of this SDK, something really cool: <http://www.youtube.com/watch?v=O_2JPXUUsOA> (pay attention at 1:15).

In Details:

* Supports Apple iOS 3.0 and iOS 4.0.
* Text-to-speech support for more than 35 languages.
* Minimal integration effort.
* Small footprint SDK with simple API.
* Highly accurate and fast speech recognition results.
* Rapid deployment of speech recognition apps.
* Enhancement of mobile application functionality.
* Licensing prices go from $399 to $3,000 and more (there’s a free trial).
* Website: <http://dragonmobile.nuancemobiledeveloper.com/public/index.php?task=home>

## ispeech

This technology provides a SDK for multiple platforms allowing performing text to speech and speech recognition (needed). It’s easy to implement and deploy. It’s fully documented and has lots of resources easy to find online. Once again, it doesn’t work for in-app processing; it needs to connect to a web server to process the data.

In Details:

* Save Time With a User Friendly SDK.
* Voice Commands.
* Web Search.
* Note Taking.
* Data Field Entry.
* Narration.
* Virtual Assistants.
* Listen to Any Text-based Article.
* Speech-to-Speech Translation.
* License pricing is related to the amount of words you need to use. It goes from $200 to $5,000 or more.
* Website: <https://www.ispeech.org/developers>

## Watson AT&T

This is a sleek solution supported by AT&T technology. They recently release an API RESTful based to operate with the AT&T contextual engine which make this the easiest one to implement (and the most efficient), not only for iOS, also for any kind of technology. In my opinion, it’s really well designed from architecture stand point. It offers a way more functionalities not required by this iPad app, but the core ones can be used still.

This can’t be used to “talk to the astronaut” unfortunately. It doesn’t provide text to speech functionality. However, with some tweaking could be manage a way to add voice to the application using any of the top two favorite frameworks (OpenEars, Dragon Speech SDK).

Watson can learn different accents, speech patterns and dialects over time, as well as distinguish between speaker and platform variations and background environments (plus). Watson takes input, analyzes it, performs one or more services and returns a result. Input can be audio files, speech, gestures, face recognition and text.

See a video explaining how the technology works: <http://www.youtube.com/watch?v=uDI6ZacK8ok>

In Details:

* Web Search Speech to Text.
* Business Search Speech to Text.
* Voicemail to Text.
* SMS Speech to Text.
* Question and Answer Transcription.
* TV Speech to Text.
* Generic Speech to Text.
* Pricing is directed related to API calls amount: [http://developer.att.com/developer/pricing](http://developer.att.com/developer/forward.jsp?passedItemId=13400484)
* Website: <http://developer.att.com/developer/forward.jsp?passedItemId=12500023&_requestid=18593>
* Tutorial: <http://developer.att.com/developer/forward.jsp?passedItemId=13200102&parentItemId=13200100>

## CeedVocal SDK

This framework is very similar to OpenEars. It provides speech recognition functionality only. We have the internet connection issue again for this technology; it requires connectivity in order to operate properly.

It’s not well documented; it would make the implementation process a bit slow due to this missing resource.

See *Vocalia*, an application implemented using the CeedVocal SDK.

<https://itunes.apple.com/us/app/vocalia/id291683886?mt=8>

In Details:

* It is provided in the form of a static library with header files. The programming language of the SDK is Objective-C.
* It is a multi-locutor Automatic Speech Recognition SDK, meaning that no training is required.
* It supports isolated words recognition from a user provided list of up to 5,000 words (can take from 0 to 2 seconds to output result, depending on word count).
* It supports keyword spotting. No grammar based recognition at this time.
* It is optimized for fast launch times (about 3 seconds for loading acoustic model, excluding phonetizations)
* It provides a phonetizer which is sufficient for common words, but won't phonetize neither abbreviations nor numbers, and won’t give access to phoneme sequences.
* It is available in 6 languages: English, French, German, Dutch, Spanish and Italian.
* It provides Voice Activity Detection (VAD), but may need manual cut in noisy environments.
* It’s provided as a Universal Binary library (armv6/armv7/i386) for easier setup within the Simulator (iOS 5.0 SDK required for development but deployment possible on iOS 4 devices).
* Supports iOS 4 and 5.
* Commercial license required. For a single application it’s €2,000.
* Website: <http://www.creaceed.com/ceedvocal/about>
* Tutorial: <http://www.creaceed.com/ceedvocal/example>