**Compassionate Officer Program (COP)**

**Technical Design Document**

This document assumes that you have read **COP- Overview document** which describes the desired *end state* of this project. This is a living document. As the project progresses, additional information will be added to this document.

This document is targeted for a technical reader. It is **NOT** expected that the reader is fully versed in all the technologies listed below. Team members can all contribute to the direction of this project with the skills they have..

# Goals of this Document

1. Explain the phases of development
2. Discuss general architecture directions
   1. Key requirements
3. Provide technical reference information

**Terms used in this document.**

* **NPC** = Non-Player Character. This is the computer-generated personality.
* **Trainee** = The person doing the training with the software.
* **NPC model** = This is the specific embodiment of the NPC that generates a particular response to the trainee. There a several possible technologies to embody the NPC, such as pattern matching, artificial intelligence, chatbot technologies or other solutions.
* **Prompt** or **NPC prompt** = The programmed prompt that the trainee should respond to. (e.g. “I feel scared when you talk like that.”)
* **User input** or **Trainee input** = This is the trainee’s response. (e.g. “Do you need safety?)
* **Conversational Unit** = It is the NPC prompt AND the trainee’s response. Several scenarios can be linked together to create a conversation.
* **Contributor** = This is a person (like you!) that wants to contribute to this project
* **Cookie** = It is a small piece of data that is stored while a trainee is practicing. The cookie data keeps track of the current state of the game.

# 1.) Phases of Development

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | Title | Description | Where | When |
| 1 | 1 step conversations | A simple computer game where the trainee is given a single prompt (e.g. “You’re going bowling again?!?”) and the trainee needs to type the single correct needs as a response. | Normal computer or mobile device | March |
| 2 | Multi-step conversations | Like above but a more complex version that can allow multiple steps (feelings and needs) in a conversation | Normal computer or mobile device | May |
| 3 | Simple VR version of game | Like above and create a VR environment where an NPC embodies the conversation | VR | Sept |
| 4 | Complex VR | Create more complex models based on AI or other technology | VR |  |

# General Architecture Directions

The architecture will evolve through several steps, but assumes a separation of NPC model and the user interface

User Interface

The NPC model

The current embodiment of the Needs Game, **run locally.**

Text input

**User Interface**

*Tech:* Flask / HTML

*Hardware:* PC browser

**The NPC model**

*Tech:* Python

*Approach:* Word/pattern matching

HTTP / API

Port 80

HTML response

The current embodiment of the Needs Game, **run on the cloud.**

**Internet**

**Google Cloud**

**AppEngine**

Text input

**User Interface**

*Tech:* Flask / HTML

*Hardware:* Any browser or mobile device

**The NPC model**

*Tech:* Python

*Approach:* Word/pattern matching

HTTP

API

Port 80

HTML response

For the future embodiment in a virtual reality environment that is voice activated by the trainee and communicating with an AI-based NPC might look something like this:

**Internet**

**User Interface**

*Tech:* Unity / Unreal Engine

*Hardware*: Oculus (or PC)

*Approach:* VR Environment

Voice/audio input

Biometric data\*

**Google Cloud**

**AppEngine**

**Model Processor**

*Tech:* Python

Custom app

Voice/audio response

System Info \*\*

**Voice-to-Text Service**

audio text

**Text-to-Voice Service**

Biometric data\*

audio text

System Info\*\*

**The NPC model**

*Approach:* AI

**Storage**

Vocabulary data

Audio Data

Logs -game play

Logs- system

Other data

\*Biometric data from the VR environment might include things like the head position of the trainee. (“Is he looking at the NPC?”)

\*\* System information return might provide other data to help the NPC act appropriately. For example the NPC could move away/toward trainee to the NPC could express a pre-programmed emotional response, such as shaking in fear.

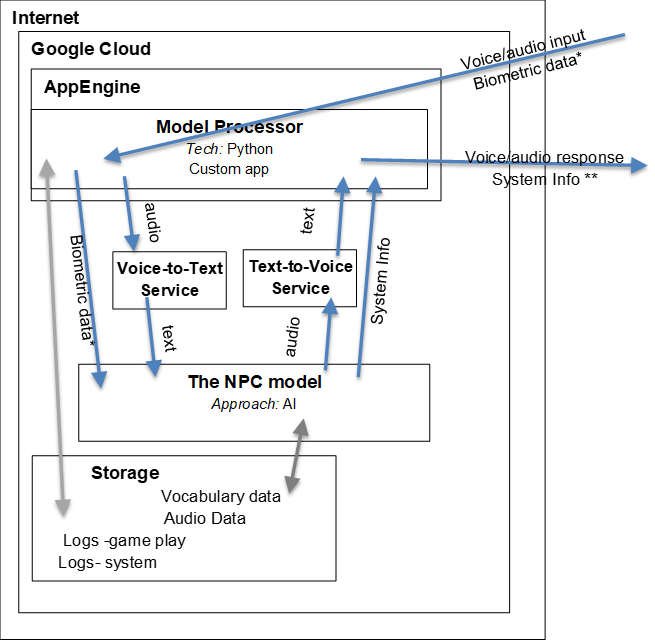
## **Why this design?**

There are several reasons for this design:

1. It gives us the ability to ‘tune’ characters without revising the VR software. We can tweak a NPC’s language, or how it responds to a particular prompt, For example, a character's communications can be changed to accommodate recent events in gender, race, orientation or other sensitive groups, without any updates to the VR software.
2. The ability to easily add additional models. For example, we can ask a university professors to help us build models that accurately model an autistic person, or a person with some other mental or behavioral issue. It is envisioned in the long term that there might be multiple models built to create responses for NPCs. For example, there might be a ‘scripted version’, a chatbot version, and an ‘ai version’ of an ‘Autistic NPC’.
3. We can build very complex AI models that could not be built in typical game software platforms for Unity and the Unreal Engine
4. Create a rich ecosystem of content from multiple authors. If we can decouple the game programming from the character responses, we can enable many people to help build the NPC models without authors needing to have VR programming skills.
5. We are ready for tomorrow. This is the most important aspect. Consider below

**User Interface**

**Robot**



In about 8 years we might find it more reasonable to train with robots in reality, rather than with NPCs in VR. When that is true, we will already have the NPC models in the cloud to inform the robot’s actions. **A goal of the project is to create a set of de-escalation nonviolent training that can be technology agnostic.**  *In fact, the models, themselves, might evolve into their own product that we sell to other companies that would like to incorporate de-escalation training in their programs.*

## **Anticipated Technical Issues**

* **Internet connectivity?** - It won’t work without the Internet. We recognize that some customers, such as the United States Department of Defense (1st responders), may not let the internet into their facilities for training. This product is not for them in the current iteration.
* **Lag time on the API calls** - The design above would require multiple API calls for a single cycle of an NPC’s response. It is expected that most of these API calls would be returned in 1-2 seconds max. However, it is reasonable to expect delays that might take as much as 8-10 seconds in exceptional cases. We can hide these delays from the trainee by having some in-game NPC actions that fill the time. Such as the NPC could moan, start shaking or some other action that might be typical of that character. Actions in the scene might also fill the time, such as having a car or cat go by.
* **The NPC models might be hard to build** - True, good models will be hard to build, but technology gets better all the time. Building a chatbot or an ML language model is something a junior programmer does now. Simple [tutorials for language processing are abundant](https://cloud.google.com/natural-language/docs/tutorials)
* **New costs associated with hosting content on the cloud**. - Yes, there will be costs associated with maintaining the data on the cloud, that would not normally be part of game development costs. Generally cloud costs are low and some providers, such as Google, have a ‘pay only for what you use’ pricing strategy, which could easily be passed on to the consumer. (It should also be noted that there might be a cost savings caused by fewer updates to the VR game software because the NPC models are separate.)
* **Can we really make the NPC models ‘technology agnostic’?**  In theory ‘yes’, but in practice ‘probably not completely’. I recognize that we will need to carefully create the API handoff between the NPC Model API and the game software. As needed, the NPC Model API may send Unity specific commands back to the game software that would not apply to other technologies. The goal is to try to be as technology agnostic as possible, so we can reduce re-work as this product grows.
* **Potential hardware issues with voice capture -** I’m assuming that if the Oculus can capture voice for the avatars in the metaverse, then there is probably a way to capture the voice data. I have a gap in knowledge here, but believe this can be remediated

# More

Briefly, the first version of the software will be implemented with a Python/Flask app that is deployed to the Google cloud’s AppEngine to make it available on the internet. Source code for the first version of the ‘Needs Game’ can be found here <https://github.com/MikeRobertsIsHappy/NeedsGameApp>

## 