Vocal Maturity Coding (VMC) Guide to Administrative Functionality

V1.0

10/01/2020

Prepared by Alex Gheith

email: gheith@purdue.edu

**VMC Guide to Administrative Functionality**

This document explains a set of administrative functions that should only be performed by a person designated as the VMC Admin. These functions may affect how the program works for all users, so it is essential to be executed carefully and only when needed. Below is a set of instructions that describe these functions, and how to set up a computer to be able to perform them.

Note that, similar to the VMC Application itself, you only require Python and a set of libraries to installed. The following steps describe the process of installing all the requirements for Windows and Mac machines.

# Installation Steps:

## Prerequisites

### Terminal Window

The instructions below show some command lines that you need to run from a terminal window. Here is how to start a terminal window on your machine:

* Mac:  
  Open the Terminal application (can usually be found in HD/Applications/Utilities/Terminal, or by searching for “Terminal” without the quotes).
* PC:  
  Type “**cmd**” without the quotes, in the “Search programs and files” text field. Once you see the “Command Prompt” app, choose “Run as administrator” option next to the app. If you do not see any options, right-click on the “Command Prompt,” and you will see them there.

### Xcode Installation (Mac Only)

Some Python libraries will need a compiler to be installed correctly. Hence, you need to install the Xcode on your machine. You can do that by going to the AppStore and searching for and installing the latest version of Xcode. Alternatively, you can run the following command from the Terminal:

$ xcode-select --install

## Installing Python

The first step is to install the desired version of Python, v3.7. Note that multiple Python versions can exist side-by-side, so if you have a different version of Python already installed, this installation will not affect it. (If you already have the required version, you may skip this step.) To install the desired version of Python, you should obtain it from the [Python download page](https://www.python.org/downloads/release/python-377/). Here are links for [Mac OS](https://www.python.org/ftp/python/3.7.7/python-3.7.7-macosx10.9.pkg) and [Windows](https://www.python.org/ftp/python/3.7.7/python-3.7.7-amd64.exe).

Notes (Windows Only):  
When following the installation wizard, you will need to choose the options to “install for all users” and to “Add Python to your PATH.” You also need to keep a note of the Python Installation location, which may be something like “C:\Program Files\Python\37\”. We will refer to this location as **(Python Path)**, and when seeing **(Python Path)** in the commands below, you must type the full path.

## VMC Codebase Folder

You should obtain a copy of the VMC Codebase from GitHub, or from another admin. The codebase contains all the files that make up the VMC Application. These files contain some important security credentials that allow the application to connect to remote resources. It is important to understand that these information should be guarded and not exposed.

In the Codebase Folder, there are three files that are of interest:

cloud.config

requirements-PC.txt

requirements-Mac.txt

These files are needed to setup your environment for the admin access.

## Python Virtual Environment Setup

We need to create a “custom location” that houses a Python executable, and all the necessary libraries in a folder other than the primary Python location. This technique allows for having any number of Python applications without worrying about requirement clashing. This custom location is referred to as a “Python Virtual Environment.” Please do the following to create a Virtual Environment, and then activate it so that we work entirely within that environment:

1. In these instructions, we chose the location to be on the desktop, but you can certainly choose a different folder, and use it instead of the suggested folder.
2. Open a Terminal window and navigate to the Desktop by typing:

* Mac:

$ cd ~/Desktop/

$ python3.7 -m venv PythonVirtualEnvironment

* PC:

$ cd %UserProfile%\Desktop

$ (Python Path)\python -m venv PythonVirtualEnvironment

1. Activate the virtual environment, in the same terminal window, by typing:

* Mac:

$ source PythonVirtualEnvironment/bin/activate

* PC:

$ PythonVirtualEnvironment\Scripts\activate

1. If you have successfully activated the virtual environment, your command prompt should now start with (PythonVirtualEnvironment). If not, then you have not yet activated the environment, and you may be working in an incorrect location.
2. Update the package manager in the environment.

* Mac & PC:

$ python -m pip install pip setuptools --upgrade

## Installing Python Libraries

Now that we have the virtual environment set and activated, we need to install the required libraries for the applications. These libraries are listed in the files “**requirements-PC.txt**” and “**requirements-Mac.txt**.” In the same window, with the environment activated:

1. In the same terminal window, navigate to the location of your VMC Codebase Folder.
2. Type the following line to install the necessary libraries:

* Mac:

$ python -m pip install -r requirements-Mac.txt

* PC:

$ python -m pip install -r requirements-PC.txt

## Storing Cloud Credentials

In order to be able to apply the admin functions, you will be using the access credentials provided to you by the application developer. Open the file “**cloud.config**” using any text editor, and add the information to the relevant lines without quotes. When done, the file should look like:

[AWS]

AccessKeyID = YOUR\_ACCESS\_KEY\_ID

SecretAccessKey = YOUR\_SECRET\_ACCESS\_KEY

Save the file and close the editor.

## Updating the Data Folder Path

Many functions will require access to the “Box” folder where the data is stored:

Box/[L3 HIPAA Int] NDFLab\_L3/LENA

Given that the folder location will vary from one machine to aonther, you will need to set the location in the program manually. Open the file “**VmcLoader.py**” using any text editor and search for the variable **DataFolder**, then modify it to include the full path of the folder. Here is an example of how it may look like:

DataFolder = r"~/Box/[L3 HIPAA Int] NDFLab\_L3/LENA"

Save the file and close the editor.

# Running Admin Functionality:

In order to run any of the following functionality, you will first need to:

1. Open a terminal window.
2. Activate the Python Virtual Environment.
3. Navigate to the Codebase Folder location.

## Adding new Participants

The list of Participants in the VMC System is located in the file:

Box/[L3 HIPAA Int] NDFLab\_L3/LENA/Participant.csv

When this file is updated, you will need to update the database as well. Note that the data in the file must be properly formatted, or the command will not run successfully.

You can do so by simply running the following command:

(PythonVirtualEnvironment) $ **python AddParticipantsToDB.py**

Here are the formatting requirements for each entry in the file:

* **ChildID**: This can be any text, but must be unique to each entry.
* **Sex**: Male, or Female.
* **DateOfBirth**: Date text formatted as MM/DD/YYY, e.g. 02/15/2019.
* **GeneticRisk**: Must be one of the risk types stored in the DB already, and the name should be verbatim. If a new type is needed, please contact the DB Admin to add that type *before* you try to add the new participants.

Note that all of the fields are required, and cannot be empty.

## Adding new Recordings

The list of Reocrdings in the VMC System is located in the file:

Box/[L3 HIPAA Int] NDFLab\_L3/LENA/Recording.csv

When this file is updated, you will need to update the database as well. Note that the data in the file must be properly formatted, or the command will not run successfully.

You can do so by simply running the following command:

(PythonVirtualEnvironment) $ **python AddRecordingsToDB.py**

Here are the formatting requirements for some entries. Note that any optional entry must either follow the requirements, or be left empty. Also, note that most of these entries are obtained from LENA without the need to change them:

* **ChildID**: Must match one of the values from the **Participant.csv** file.
* **RecordingType**: Home, or Assessment
* **BaseFileName**: Must NOT contain file extensions, like ".its" or ".csv".
* **ScrubTimes, NapTimes**: Must follow the following requirements:
* For each timestamp, both the hour and the minutes must be present, e.g. 3:00, not just 3.
* For each timestamp, the AM/PM designator must be present.
* A single time range is two timestamps separated by a dash, and the dash is preceded by, and followed by, a single space, e.g. 9:00 AM - 12:30 PM
* Ranges are separated by commas, not semicolons, with a space after each comma, e.g. 10:00 AM - 11:00 AM, 12:00 PM - 2:00 PM
* At any point time ranges are NOT finalized, please mark the row as "Skip".
* **IsValid**: Must be 1 or 0.
* **FirstEntry, SecondEntry, ThirdEntry**: If present, must match a username in the DB, not an abbreviation.

# Running Admin Functionality from DirectAccess:

The file **"DirectAccess.py"** contains many tasks that can be run individually. Any of these tasks must first be enabled, and the data modified via a text editor, to be run successfully, where only one task must be enabled at a time. Moreover, the task must be disabled when completed.

Note that the code is written in Python, which means two things need to kept in mind:

1. Python is space-sensitive, which means the code lines must align, or the file will not run.
2. The Comment/Uncomment must be done for each line separately, i.e. avoid using block comments.

Once a certain task is enabled and modified, you can save the file and run it as:

(PythonVirtualEnvironment) $ **python DirectAccess.py**

Below are some of the tasks that are included in the file.

## Adding a New User to the Database

Populate the list of variables with the information about the new user as shown in the associated comments, as well as the user type, and then run the file. Note that you can add only a single user at a time.

## Updating a User’s Password in the Database

If a user forgets his/her password, it cannot be retrieved from the database for security reasons. Hence, it can only be changed using this task. Populate the user name, which must exist in the database, as well as the new password, then run the file.

## Creating a new Coding Batch

A coding batch consists of three recordings that must be coded together. The recordings IDs must be present in the database, and then you can run the file. Note that this task can take several minutes to run, as it will load the large audio files, and process them to extract thousands of small utterance files.

## Generating the Utterance Consensus Report

This tasks extracts all the recordings that have been coded already in the database. You must provide the location for saving the new report, as well as the report name, then you can run the file.

# Running Database SQL Queries:

The following SQL Queries can be run to perform some database maintenance tasks and obtain quick piece of information from the database tables. Contact your DB Admin if you do not have access to the DB server.

## Obtaining the Number of Utterances Available to Code

This queries shows the number of current utterances that can be coded by users. It only checks the **UtteranceSamplePool** table. You can run this queries any number of times

**SELECT** **COUNT**(\*)

**FROM** "Main"."UtteranceSamplePool"

**WHERE** "CoderID" **IS** **NULL**

## Ophaned Utterances Available to Code

This queries shows the number of utterances that have been started but not finished by the users. It checks the **UtteranceSamplePool** table for any utterance that has been loaded an hour before the check and has not been coded.

**SELECT** \*

**FROM** "Main"."UtteranceSamplePool"

**WHERE** "IsProcessing"

**AND** "ModifiedOn" <= (**NOW**() - **interval** '1 hour')

**ORDER** **BY** "ModifiedOn"

Note that these orphaned entries must be cleared regularly, e.g. once a day, using the following query

**UPDATE** "Main"."UtteranceSamplePool"

**SET** "IsProcessing" = **FALSE**

**WHERE** "IsProcessing"

**AND** "ModifiedOn" <= (**NOW**() - **interval** '1 hour')

## Log Entries Used for Debugging

This queries shows the number of entries in the **LogEntry** table.

**SELECT** **COUNT** (\*)

**FROM** "Main"."LogEntry"

Note that these entries should not be kept indefinitely, so it is recommended to keep only the last month worth of entries. The following query clears all entries older than a month before the current time at which the query is run.

**DELETE**

**FROM** "Main"."LogEntry" **AS** l

**WHERE** l."TimeStamp" <= (**NOW**() - **interval** '1 month')