Dashboard Summary

This is part of the Applied Project that was submitted for my MSIT degree. The project culminates many of the various aspects of Information Technology that were covered throughout the degree program. Students are expected to come up with their own idea for the project and complete it with very little direction and support. The three milestones for the project were planning, implementation, and deliverables. This idea was chosen for its uniqueness, equipment that was already on hand, its real world application, and incorporating one of my main interests of data acquisition, analysis and visualization.

The dashboard depicts various metrics that can be used to gauge the watchability and reliability of over the air digital television channels in the Phoenix area. Data was gathered from a network connected tuner through its command line interface via a custom Python program and the manufacturer's proprietary library. Output of the CLI is in textual format and data is extracted into key-value dictionaries by using regular expressions with pattern matching and stored in a text file. Each dictionary represents a one minute test for each channel where errors are cumulative and other readings such signal strength and signal to noise quality are averaged throughout the duration of each test. A channel scan is performed prior to each round of channel testing to populate a channel test list so time isn't wasted by testing channels that are either not being used for the particular service area or beyond the reception range. There are instances when channel reception is borderline where the channels show up in the scan, but are unavailable for the channel test. The dataset generated with the Python application consists of 73,271 records across 22 television channels.

```
SCANNING: 563000000 (us-bcast:29)
LOCK: 8vsb (ss=72 snq=85 seq=100)
TSID: 0x0E7B
PROGRAM 3: 39.1 KTAZ-DT
PROGRAM 4: 39.2 TeleX
PROGRAM 5: 39.3 COZI
PROGRAM 6: 39.4 NBC LX
PROGRAM 7: 35.3 GetTV
SCANNING: 557000000 (us-bcast:28)
LOCK: none (ss=28 snq=0 seq=0)
SCANNING: 551000000 (us-bcast:27)
LOCK: none (ss=63 snq=0 seq=0)

tun: ch=8vsb:545000000 lock=8vsb:545000000 ss=75 snq=89 seq=100 dbg=-5825/1178
dev: bps=19251200 resync=0 overflow=0
ts:  bps=7315456 te=0 crc=0
net: bps=7311680 pps=626 err=0 stop=0
```

Sample CLI output from tuner

```
    scan_string = re.sub(r'(?<=\))\|\n(?=LOCK:)|(?<=\))\|\n(?=TSID:)|(?<=\s0x\w{4})\|\n(?=PROGRAM)'
                         , ' ', scan_output)
    scan_string = re.sub(r'\|\nPROGRAM\s[^\|\n]*(?=\n)|\)|\|\n\|','\|,\sNone','', scan_string)
    scan_string = re.sub(r'\|\n', '\|\n', scan_string)
```

```
    scan_pattern = re.compile(r'SCANNING: (?P<freq>\d*)0{6} \(us-bcast:(?P<ch>\d{1,2}) '
                              'LOCK: (?P<mod>\w{4}) \(ss=(?P<ss>\d*) snq=(?P<snq>\d*) '
                              'seq=(?P<seq>\d*)(\sTSID: (?P<tsid>\w{6}))?'
                              '( PROGRAM\s\d{1,4}: (?P<v_ch>\d{1,2}))?'
                              '(\.\d{1} (?P<call_sign>[-\w ]{3,7}))?', re.M)
```

```
    debug_pattern = re.compile(r'tun: ch=auto:(?P<ch>\d*) lock=(?P<mod>\w*)(:(?P<freq>\d*)?0{6})? '
                               'ss=(?P<ss>\d*) snq=(?P<snq>\d*) seq=(?P<seq>\d*).*'
                               'te=(?P<te>\d*) crc=(?P<crc>\d*)')
```

Regex patterns for extracting data from CLI output

{'debug_timestamp': 1644383683, 'scan_timestamp': 1644382668, 'ch': 29, 'mod': '8vsb', 'freq': 563, 'ss': 72, 'snq': 94, 'seq': 100, 'te': 0, 'crc': 0, 'tune_time': 3.16, 'test_dur': 60, 'v_ch': 39, 'call_sign': 'KTAZ-DT', 'tsid': '0x0E7B'}

Key-value dictionary resulting from regex pattern matching and substitution of the channel scan and debug commands combined with other elements that were generated within the Python code

Each individual chart was produced using Python with Pandas data analysis and Bokeh visualization libraries. Bokeh was chosen for the dashboard due to its compatibility with Microsoft Windows, MacOS, and Linux operating systems and open source software licensing where it can be used at no cost. Data from the key-value dictionaries are loaded into a Pandas DataFrame so value can be added to the data through error counts and means of the various other readings for each channel. Additional attributes have been added to the DataFrame such as call sign, transmitter power, transmitter location, and virtual channel to help make the data easier to interpret or relate to. Interactivity has been added using Bokeh's radio group and select widgets; tools and clickable legends for the line plot, and ability to sort the data in the datatable by clicking on the column headers. The error and signal bar charts have also been linked together so they are both sorted according to the radio button selection. Axis labels and titles also change to reflect radio button and select widget selections. Bokeh's HoverTool is used to provide more detailed information when moving the mouse pointer over each line or bar of the charts. The dashboard is rendered each time the page is loaded or refreshed.

In order to implement the interactive aspect of the dashboard, it is necessary to use Bokeh Server rather than just the Boken Python libraries imported within the code. Bokeh Server is running on a Ubuntu Linux virtual machine aka "Droplet" utilizing DigitalOcean's cloud platform with a shared CPU, 1 GB of memory, and 10 GB SSD drive. This is one of Digital Ocean's least expensive at $6 per month or $0.009 per hour.

Since Bokeh Server alone is not intended for a production environment; Gunicorn HTTP server, NGINX reverse proxy, and Flask framework were used for enhanced security and robustness. Although the dashboard could have easily been implemented using one of more well-known and comprehensive cloud providers such as AWS or Azure, Digital Ocean was selected for their 60 day introductory offer with $100 credit and the opportunity to learn how to use another cloud platform.

The dashboard has been optimized for an HDMI monitor with 1920x1080 resolution, but can be easily viewed on laptops with resolution of 1366x768 by selecting full screen mode in the internet browser.