|  |
| --- |
| PyCQ Project scope |
| May 29, 2023 |

# Overview

## Project background and description

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | Specific project plan proposed as a final project python programming course (AI-college).  The main purpose of this project is displaying of abilities of student to develop high level of python windows desktop frontend application “PyCQ” (server-client style).  “PyCQ” applications allow the main central Windows PC station (further in the text ‘Caller’) to discover remote Windows PC terminals (further in the text ‘Station’) on Local Area Network (LAN), communicate with stations in unicast, multicast or broadcast modes (addressing specific station, selected group of stations or all available stations) and send pre-recorded short voice message and ~~text chat (optional feature)~~ to selected station/s.  The ‘Station' should be able to playback received voice message / ~~text~~ informing the station operator in visual and audible form.  Suggested use case: mid–large corporate segment with distributed workstation Windows PCs on LAN.  Example 1: ‘Caller’ app installed on manager PC. Manager initiates voice chat in broadcast: “All please come to main conference room for company meeting” (He will possibly inform on workforce reduction, but will not mention it in public… 😊)  Example 2: ‘Caller’ app installed on manager PC. Manager initiates voice chat in multicast: “Team leads, please come to my room for scrum meeting”  Example 3: ‘Caller’ app installed on company Admin (secretary) PC. Admin initiates voice chat in unicast: “Michael, your lunch has arrived – please take it from entrance counter…”  All these tasks can be performed with single click on Caller PC with no need in finding phone numbers, mail addresses, taking a look in phone book/WhatsApp/email app etc.  The project is exclusively performed by responsible developer (Michael Zhenin) with support by AI-college staff. |

## Project scope

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | The project includes development and implementation of server-client communication, voice processing, NW discovery, UI and auxiliary application modules for both ‘Caller’ and ‘Station’ applications. (Look specific module notes section to elaborate) The modules will be implemented in Python-3 language using standard Python libraries and open-source python GPL code.  Project budget is 1 developer (full time). Time margins of the project is 3-4 ww. Specific timelines – TBD  The project will include development cycle and developer unit tests. Formal QA test cycle is not a part of this scope.  The project will include the basic logging modules to simplify development and further debug.  The project **will include** UI/UX research and will implement basic UI applications.  The project **will include** applications code/memory performance optimizations (threads etc.) and will implement a working PoC of application.  The project will include partial applications error handling and recovery procedures and will implement a basic PoC |

## High-level requirements

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | The new system (sides listed in brackets [‘‘] below) must include (but not limited to) following: |

* Ability to record, distribute on LAN [‘Caller’] and playback of short voice (optional text) messages [‘Station’]
* Ability of [‘Caller’] performing periodic NW discovery and updating list of connected [‘Stations’]
* Ability of [‘Caller’] to address specific [‘Station’], group of [‘Stations’]
* Ability to maintain basic event log on both the sides [‘Caller’], [‘Station’] for debug purposes
* Ability of [‘Station’] to inform [‘Caller’] on successful message reception
* Ability of [‘Caller’] to maintain the local storage of sent messages
* TBD

## Deliverables

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | * Application code in repository (GitHub) * Release Notes * Application executables (compiled Windows .exe files) * Setup tools (InstallShield etc.) |

## Affected parties

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | None |

## Affected business processes or systems

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | Office Administration |

## Specific exclusions from scope

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | Text chat will not be implemented |

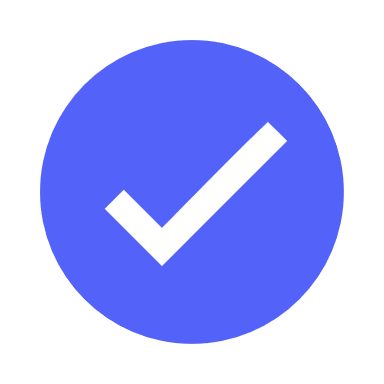
## Implementation plan

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | This single-cycle project is not including incremental development cycles. Any CI/CD procedures are out of this scope. No alpha, beta or development releases will be provided. Final release will include described in paragraph 4 of this scope. |

## High-level timeline/schedule

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | Planned for 3-4 work weeks |

## High-level system diagram



Discovery, REQ, Msg Data

Station\_1

Data ACK



TCP / UDP



**LAN**

Caller



|  |
| --- |
| Voice Recorder |
| NW Discovery |
| TCP/UDP sockets |
| Call initiation |
| Stations DB |
| Caller GUI |
| Debug Log |
| Chat (optional) |
| Message store (optional) |
| Call History (optional) |

Caller modules

Station\_n

|  |
| --- |
| Windows tray application |
| Voice Player |
| TCP/UDP sockets |
| Call Acknowledge |
| Station GUI |
| Debug Log |
| Chat (optional) |
| ~~Message store (optional)~~ |
| ~~Call History (optional)~~ |
| Replay received message |

Station modules

## Communication Protocol (Idle procedures)

**STATION**

**CALLER**

Listening for service

Announcement received

Announcing

Caller Announcements (UDP)

New station (update station list)

REGISTER (TCP)

Station Connected (wait for message)

REGISTER\_ACK (TCP)

Caller checks for Station presence

Periodic KEEP\_ALIVE\_REQ (TCP)

Station ‘Alive’ (update station list)

KEEP\_ALIVE\_ACK (TCP)

Station checks for Caller presence

Station Connected (wait for message)

REGISTER\_ACK (TCP)

Periodic REGISTER (TCP)

## Communication Protocol (Data Transfer)

**STATION**

**CALLER**

User sends new message

Station Connected (wait for message)

NEW\_MESSAGE\_IND (TCP)

HTTP\_CONNECT\_REQ

Station GET URL and

Download new message

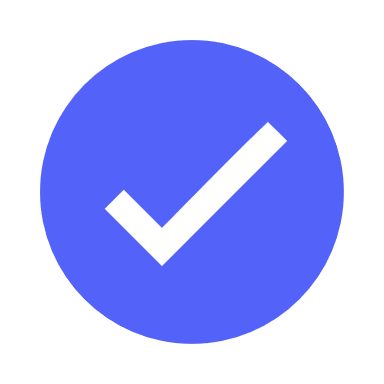
HTTP 200

Caller receive message read and updates local ‘Message History’

NEW\_MESSAGE\_ACK (TCP)

Station process new message and close message pop-up

## Specific module notes:



* Voice recorder module on ‘Caller’ implements following functionalities:
  1. Enumerates Windows audio devices and selects correct audio input (Mic-in) from enumerated HW list
  2. Configures audio device parameters (channel, sampling rate, duration) for recording
  3. Records raw audio data to ‘numpy’ array
  4. Converts raw audio data to audio file format and stores ‘.ogg’ file
* Voice player module on ‘Station’ implements following functionalities:
  1. Enumerates Windows audio devices and selects correct audio output (Speakers/Headphones) from enumerated HW list
  2. Configures audio device for playback
  3. Plays received ‘.ogg’ file
* TCP/UDP sockets implements following functionalities:
  1. Configures specific TCP/UDP ports for bidirectional/unidirectional communications
  2. Opens/closes TCP sockets for listen and HTTP for file transfer
* NW Discovery on ‘Caller’ implements following functionalities:
  1. Periodically checks stations “keep-alive” status and updates online stations list for display
  2. ~~In basic implementation will use predefined station list~~. In advanced implementation can search for new available stations and updates the list dynamically - ~~TBD~~ - **Implemented advanced station search**
* Call initiation on ‘Caller’ implements following functionalities:
  1. Initiates call to selected stations and sends recording ‘.wav’ file
  2. Receives call acknowledgement from ‘Station’ and updates status for display
* Call acknowledgement on ‘Station’ implements following functionalities:
  1. Upon call reception sends call acknowledgement to ‘Caller’
  2. When ‘Message Store’ / ‘Call History’ implemented updates specific – ~~TBD~~ **- Implemented**
* Stations DB – holds the list of known stations (in CSV or XML format)
* Debug Log – implements simple debug log (‘.log’ format) for both ‘Caller’ and ‘Station’ applications
* Caller GUI – implements entire ‘Caller’ UI procedures
* Station GUI – implements entire ‘Station’ UI
* Windows tray application – hidden resident service on ‘Station’ PC, calls main station UI – ~~TBD~~ **- Implemented**
* Message store (optional) – when implemented allows ’Caller’ user retrospective playback of sent messages (requires Message History implementation) – ~~TBD~~ - Implemented
* Chat (optional) – when implemented allows text chats between ‘Caller’ and ‘Station’ users – ~~TBD~~ **- Not implemented**

# Approval and Authority to Proceed

We approve the project as described above, and authorize the team to proceed.

|  |  |  |
| --- | --- | --- |
| Name | Title | Date |
| Michael Zhenin | Responsible Developer | 29.05.2023 |
|  |  |  |
|  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Approved By |  | Date | Approved By |  | Date |