TARTU UNIVERSITY
FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE
STUDY PROGRAM „INFORMATION TECHNOLOGY SYSTEMS DEVELOPMENT “

Andrei Smirnov

“P.C.O.”: Telegram Bot for Controlling Computer Programs and Volume

Diploma Thesis

Supervisor: Assistant Andre Sääsk

NARVA 2024

Olen koostanud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....

Töö autori allkiri ja kuupäev

Non-exclusive license to reproduce thesis.

I, Andrei Smirnov (date of birth: 11.09.2001),

1. herewith grant the University of Tartu a free permit (non-exclusive license) to reproduce, for the purpose of preservation, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright.

““P.C.O.”: Telegram Bot for Controlling Computer Programs and Volume” supervised by assistant Andre Sääsk

2. I am aware of the fact that the author retains the right referred to in point 1.

3. This is to certify that granting the non-exclusive license does not infringe the intellectual property rights or rights arising from the Personal Data Act.

Narva, 01.01.2024

"PCO”: Telegram Bot for Managing Computer Programs and Volume

Abstract: This thesis explores the creation of a project called "PCO", designed to control computer programs and adjust sound volume. The bot aims to simplify the management of computer applications and audio settings by offering functions such as switching applications and adjusting volume through a user-friendly Telegram interface. The research explores the development process and functionality of "PCO" to improve user interaction with computer systems.

Keywords: PCO, Telegram bot, application management, volume control, python

Table of Contents

Introduction

The Problem

The problem is that, as an author, I need to be able to remotely turn on or off applications on my personal computer. It would also be nice to be able to adjust the sound on the device. In general, I wanted to have a simple access to my personal computer through the phone.

The Solution

Addressing the complexities of existing remote control and monitoring solutions, the proposed solution is the development of the project "PCO." This solution aims to empower users with the ability to remotely control their computers and applications seamlessly, including volume level management. The focus is on simplicity, affordability, and user-friendly functionalities that is both efficient and accessible.

The Objectives

The objectives are to create an application and a remote control for that application, so that performance is at an acceptable level, user-friendly and intuitive to operate. It will be necessary to have a stable connection between the application and the control panel.

The Motivation

This project is motivated by the author's desire to have an easy way to remotely manage applications. Also, the use of such a project will simplify the daily routine and free time of an ordinary user.

Outline

The project structure will include an introduction, development details, and a conclusion. The development section will cover the technical aspects of creating the PCO Telegram bot, its features, functionality, and potential applications. The conclusion will summarize the impact of the solution and possible future developments.

1 Implemented technologies and available solutions

1.1 Available solutions

1.1.1 Microsoft Remote Desktop

Microsoft Remote Desktop offers a solution for users seeking to control their computers remotely from mobile devices. It provides a seamless connection to Windows-based computers, enhancing accessibility. While praised for its remote desktop capabilities, users may experience variations in accessibility depending on the specific platforms and device compatibility.

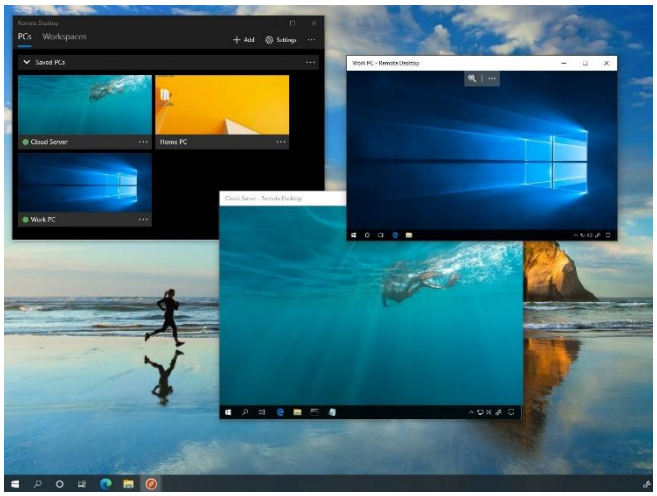

Figure 1. Microsoft remote desktop usage

1.1.2 Unified Remote

Unified Remote stands out as a versatile application that transforms a mobile device into a multifunctional remote control for computers. With a user-friendly approach to control, Unified Remote offers a variety of functions. Users appreciate its diverse features, though its compatibility and performance may vary based on the specific devices involved.

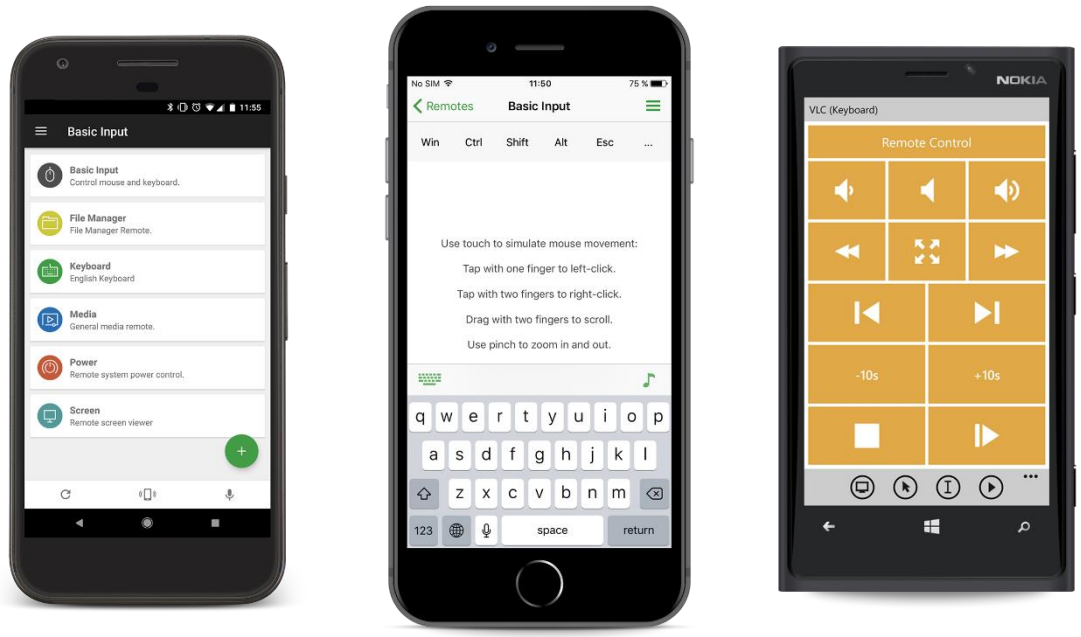

Figure 2. Unified Remote interface

1.1.3 TeamViewer

TeamViewer is a widely utilized application for remote access and control, presenting an alternative for users seeking remote management solutions. Enabling users to access their computers remotely, TeamViewer offers robust features, including file transfer and real-time collaboration. However, its full functionality might be subject to subscription plans, and accessibility could depend on the user's location and network conditions.

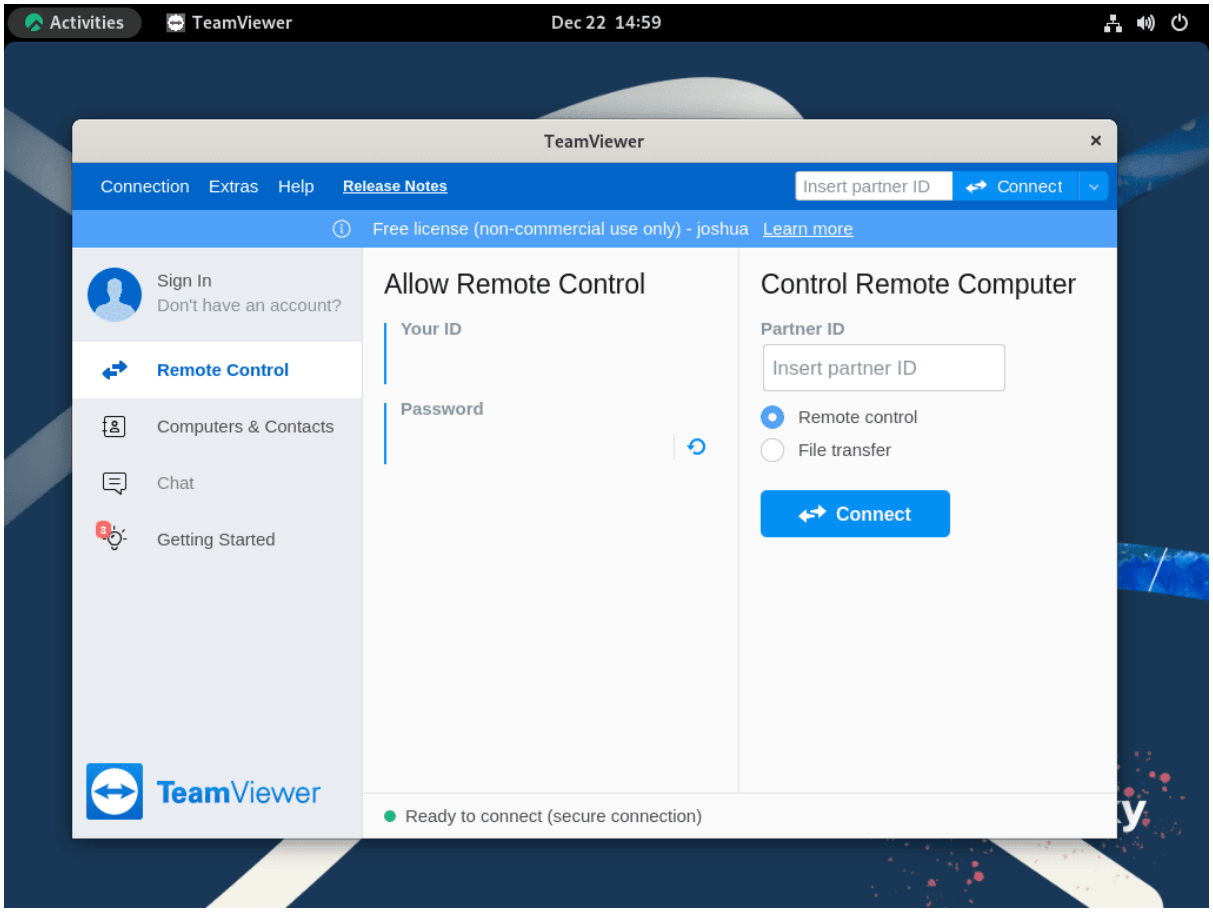

Figure 3. TeamViewer interface

1.1.4 AnyDesk

AnyDesk is a remote desktop application distributed by AnyDesk Software GmbH. The patented software provides platform-independent remote access to personal computers and other devices running the host application.

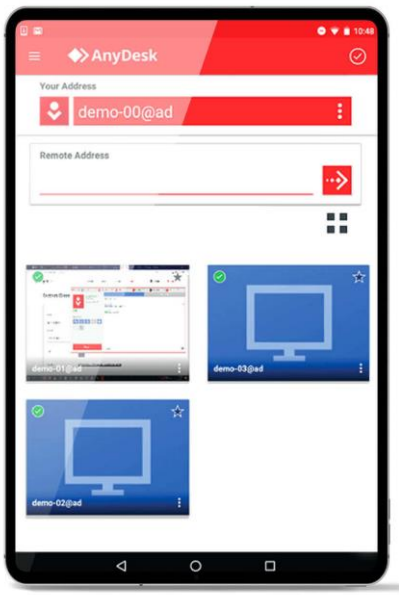

Figure 4. AnyDesk interface

1.1.5 Conclusion

In the process of searching for possible solutions, none of them were suitable for the author. Some of them required the use of one computer to connect to another, while the author wanted to use a mobile device. Others offered cell phone options but required a surcharge for more convenient use and the entire application had to be downloaded to the computer and smartphone. All options gave too much control, requiring either downloading and using a premium app or using a second computer. The most suitable option of all is AnyDesk, but it requires downloading an application to the phone. In the end, it was decided to create its own app and remote control to it, to be used by the author, for his own purposes.

1.2 Possible technical solutions

It was decided to start researching technologies and select the most suitable for the project.

1.2.1 Client application

It was decided to choose among those programming languages in which or for which graphical modules had already been written, allowing me in the future to create an interface from ready-made assets. As a result of some thought, the programming language “python” was chosen. Firstly, this is a language in which a large number of libraries allow you to draw an interface, for example, “tkinter”; we were introduced to many of them in college. More modern and graphically higher-level engines or programming languages were not needed for the main purpose. Secondly, the author is most familiar with this programming language.

1.2.2 Remote control

To implement the idea, the user on the phone must be able to control the application on the computer. Since the author did not want the application to be downloaded separately. It was decided to use the Bot in a popular messenger for convenient and quick access. As a result, it was chosen to write a bot for the Telegram messenger, because most young people use or have heard about this messenger. To write a bot, the Python programming language was chosen. Since the author is most familiar with this programming language. Among other things, the author had experience using the Python "pyrogram" library to configure the bot.

1.2.3 Data base

To store data about future users and be able to link telegram accounts with personal computers while maintaining this connection. It was decided to use a data base; the simplest SQLite database, which did not require a separate server, was chosen to store information. To work with this database, the Python programming language was chosen because the entire project was written in this programming language. For convenient database management, the peewee library was chosen. The choice fell on this library because of the recommendations of a programmer friend.

1.2.4 Communication between application and bot

Initially, in the draft version of the project, a not entirely correct decision was made. It consisted in the fact that the application would constantly check a certain plate in the database for changes. After implementing such a mechanism, it was decided to abandon it due to possible problems with application and database performance. It was decided to write a server that has access to the database and can receive, process, redirect, and send information. Python was chosen as the programming language because the user knows this programming language best and because all other code is written in Python. The JSON format was chosen to transmit information, since it is the most popular format for transmitting information on servers at the moment.

2 The design and architecture of “P.C.O”

“PCO” is a project that will consist of an application, a telegram bot, a server, and a database. The user will be provided with an external control interface in the form of a visual part of the application. A Telegram bot is a kind of control panel through which the user can send a signal to an application on his computer through the server. The server processes, receives and sends information.

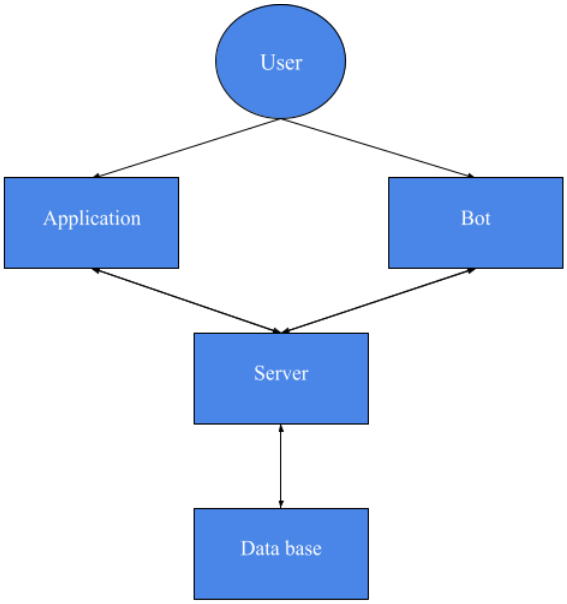

Figure 5. View connections in PCO

2.1 Functional requirements

Functional requirements describe the services represented by the application, its behavior in certain situations, the response to certain input data and actions that the system will allow users to perform.

1 The PCO app must have a login view.
2 The PCO application must have a main view.
3 The server must be stable.
4 Only the server needs to be connected to the database.
5 The server must be able to receive, process, send or redirect information.
6 The user can create an account. Registration should be easy and possible with the telegram bot.
7 The user can log in to the "PCO" application if previously registered.
8 In the "PCO" application, it is possible to add applications.
9 In the "PCO" application, previously added applications can be viewed.
10 Through the telegram bot, applications added can be turned on and off.
11 In the "PCO" application, applications can be deleted by the user.
12 The volume on the personal computer can be adjusted through the telegram bot.
13 Login details can be saved in the "PCO" application by the user.
14 In the "PCO" application, applications can be filtered by the user.

15 The option to add an application to favorites is available in the "PCO" application.

16 In the "PCO" application, the path to the selected application can be opened.

2.2 Non-functional requirements

Non-functional requirements describe the characteristics of the system and its environment, rather than the behavior of the system. Also, a list of restrictions imposed on the actions and functions performed by the system. They include time constraints, restrictions on the system development process, standards.

1 The application can run on all Windows computers starting with version 7.0 and higher.

2 The application can run on most Linux computers.

3 The user must have access to the internet for full use of the application.

4 The user should not have difficulty using the interface.

5 The user must have Telegram account for registration and for full use of the application.

2.3 Client application

Initially, the tkinter library was used. But the author did not like the visual appearance of the application, so it was decided to switch to the "customtkinter" library, which complements and expands the capabilities of the tkinter library. The part responsible for communication with the server is made with the help of "asyncio". The ability to use the application on both Linux and Windows, implemented by the usual method, defining the platform using the library "platform", for each operating system. In a client application, the backend consists of connecting to a remote, primary server and receiving and sending requests to it.

The application consists of the following views:

1 User login

2 Base applications view

3 Favorite applications view

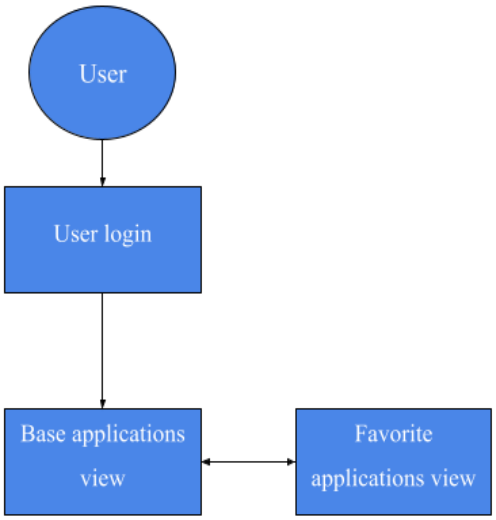

Figure 6. View connections in PCO client application

2.3.1 Login window

The application login window is seen by everyone who runs the application. For new users who have not clicked the "Remember me" button, the data entry field will be empty, for those users who have already logged in, the data field will be automatically filled in. To log in as an already registered user, users need to enter their Telegram ID in the field designed for this purpose, so that he doesn’t have to enter it every time, there is a special "Remember me" checkbox. When entering any text in the field, the application sends this text to the server, which in turn searches for such a user in the database. Whatever response the server receives from the database, it returns it to the application, most of important errors or actions will be shown on top of the interface.

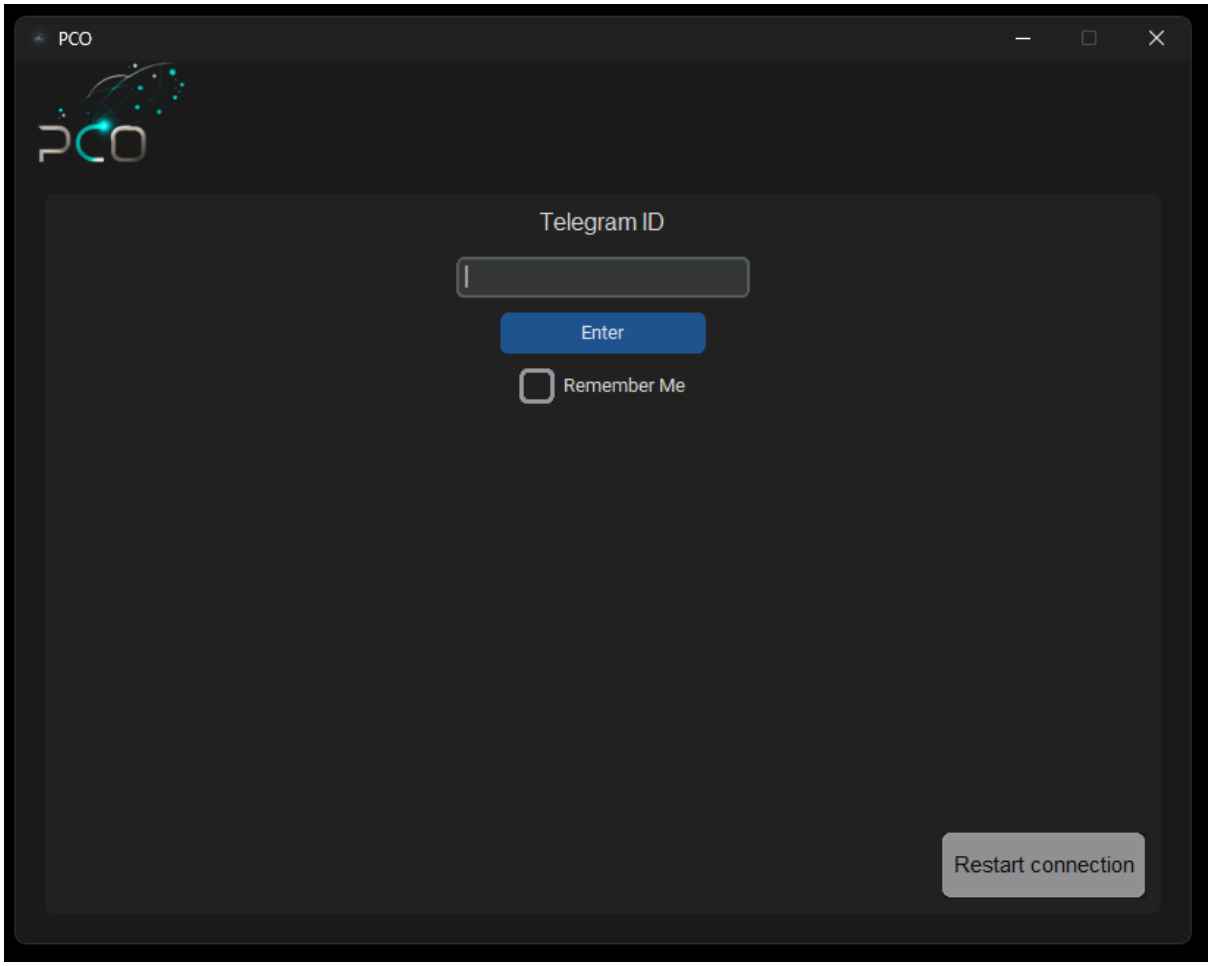

Figure 7. Login window view

2.3.2 Base applications menu

"Base applications menu" is the main application screen, here user can add an application by inserting its path in the defined field. The user can delete applications, add it to his favorite, open the path to it. There is a filter for convenient search. As well as on all windows there is a button for reconnection.

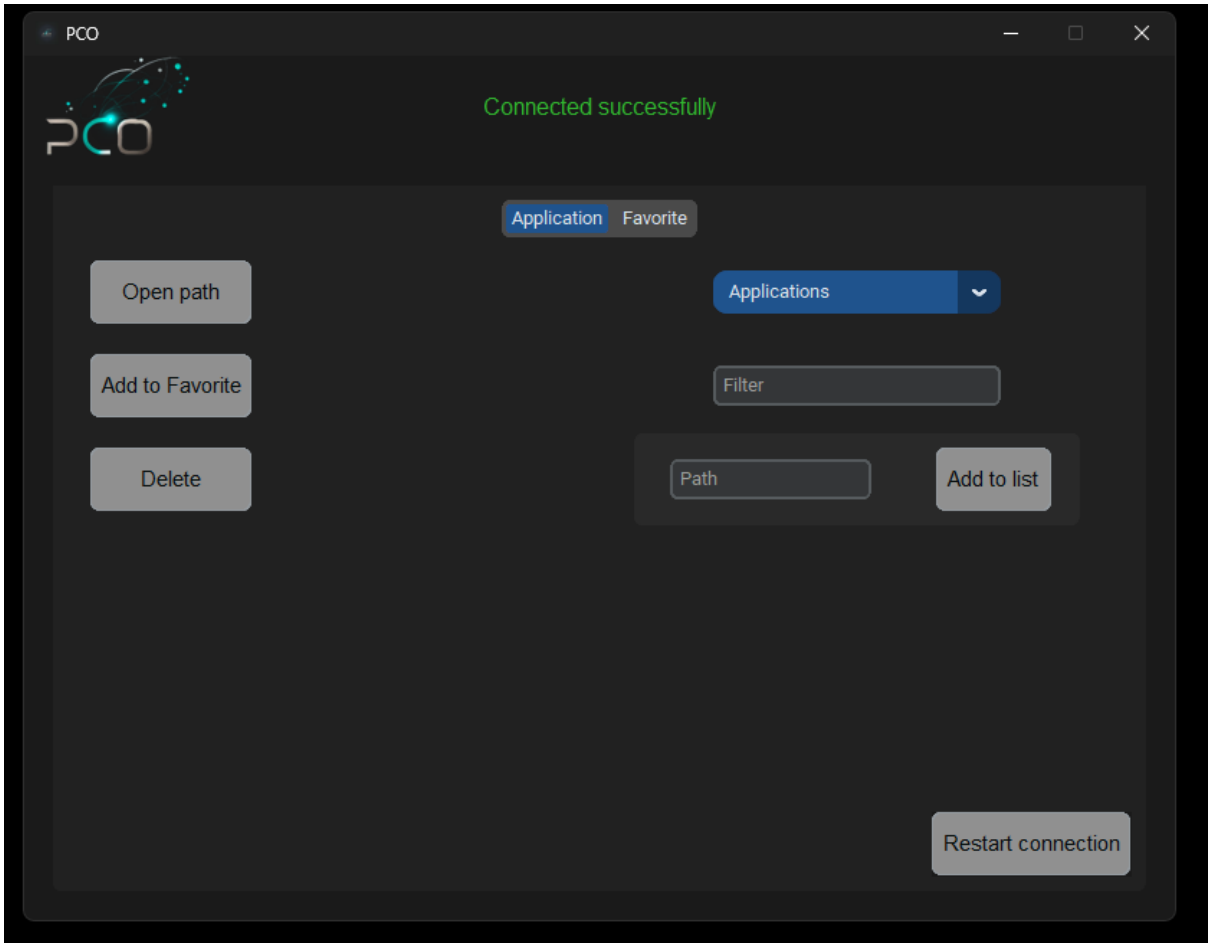

Figure 8. Base application menu view

2.3.3 Favorite applications menu

When creating the work, there were some problems with adding a separate list for favorite applications. All the versions revised and reworked by the author either looked ugly and not aesthetically pleasing, or had built-in errors and bugs, the correction of which would require intervention in the library code and would be too energy consuming. It was decided to create an additional view separately for applications added to favorites, called "Favorite application menu". "Favorite applications menu" - as is clear from the name, a view for displaying and working with applications added to the favorite. the add button was removed because it is not needed as well as the field for entering the path. Also, the delete button was changed to delete from favorite button.

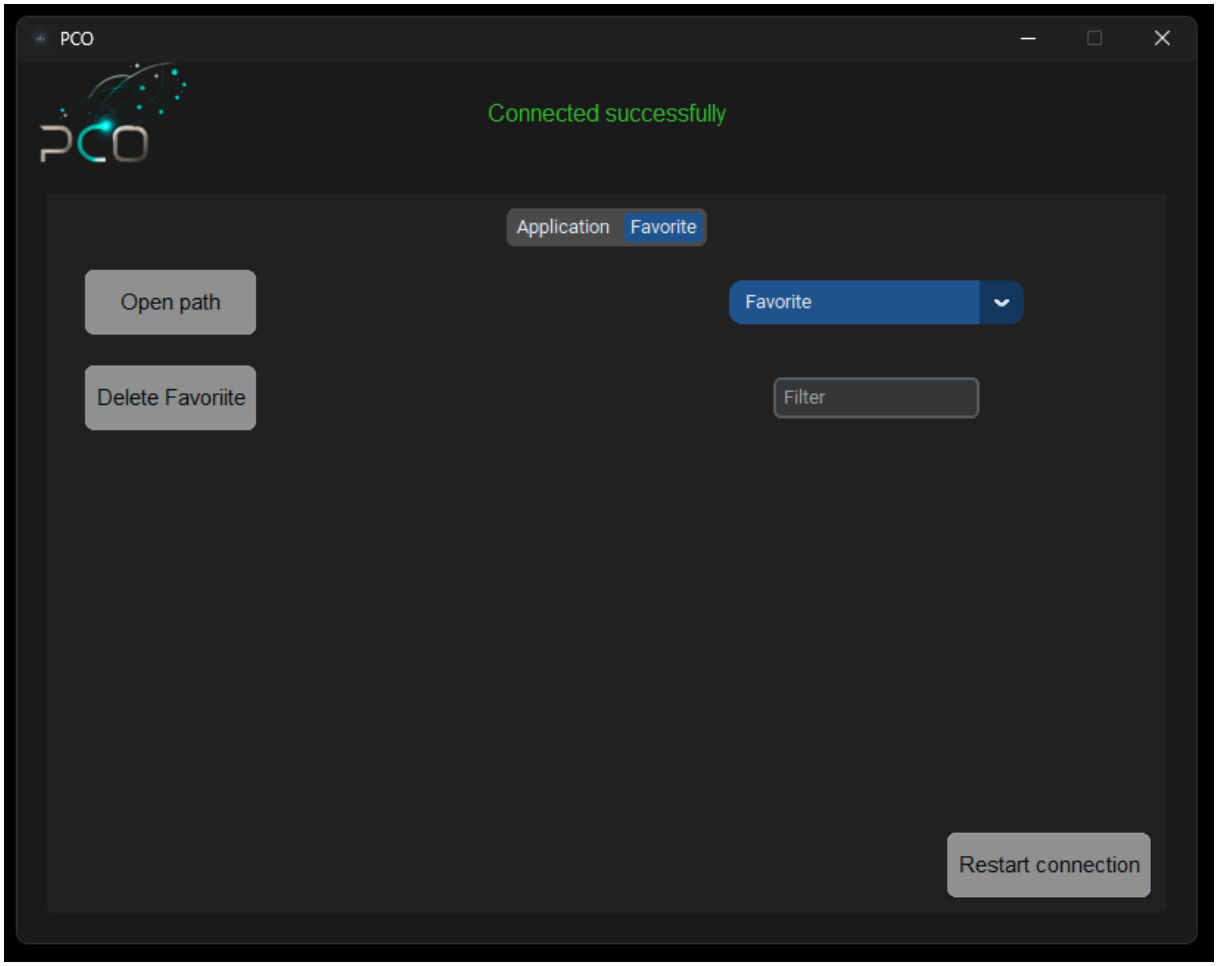

Figure 9. Favorite application menu view

2.4 Bot

In Telegram, creating a bot is easy and accessible to everyone. It is enough to write to a bot called “BotFather” - a bot created by the Telegram team. After a little customization it will give you a bot token. But for the “pyrogram” library that I use, also need the API of Telegram, which it provides free of charge on its site after the bot is registered on it.
More than 60% of the bot functionality is based on sending and processing requests to the server. The rest is the bot interface and translate system. The telegram uses pre-made assemblies. This means that it is impossible to create custom buttons or any interface elements in Telegram chat itself. Everything visual in the bot is made with this in mind.

2.4.1 Bot content

The content in the bot represents a button press and the actual information that this button will provide. Next, a list of content in the bot is presented:

1 “/help” information

2 “/start” information

3 “Create Token” button.

4 “Create Token” information.

5 “Connect to PC” button.

6 “Connect to PC” information

7 “Set up sound” button

8 “Set up sound” information

9 “Volume up” button

10 “Volume down” button

11 Applications information

12 Application “on” button

13 Application “off” button

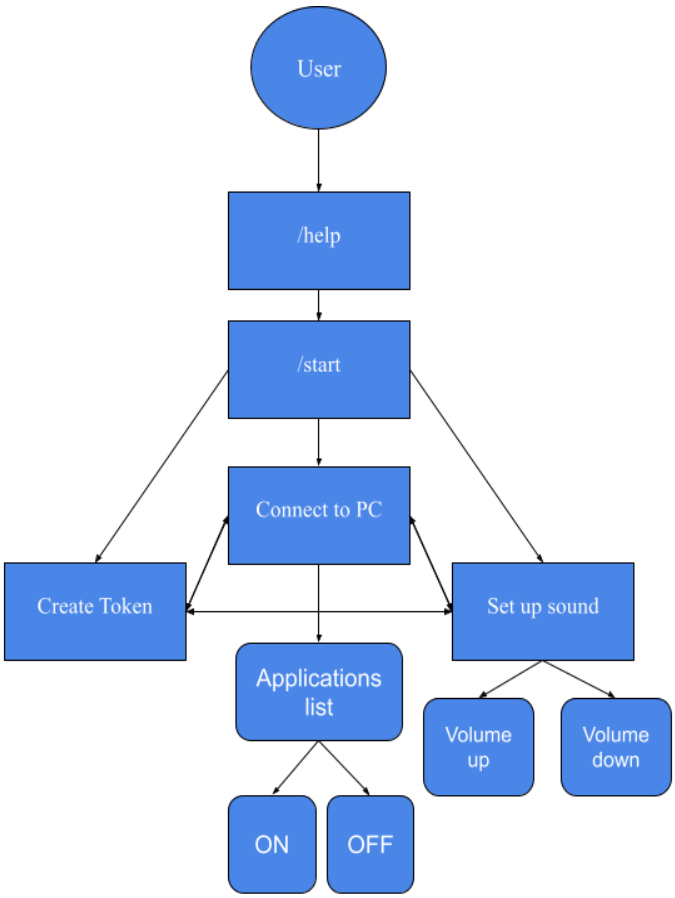

Figure 10. View connection in PCO bot

2.4.2 Bot functional

The bot's functionality includes several key steps. First, it is capable of converting data into JSON format. Then, the bot sends this data to the server and waits for a response, accepting the necessary information. After receiving the data from the server, the bot performs the conversion back from the JSON format. It also performs the important function of translating the valuable information into a readable format for the user. The bot then sends the received information to the end user. Finally, as part of its functionality, the bot provides the ability to tag favorite applications, making the interaction process more convenient and personalized.

2.5 Data base

The default SQLite file is used as the database. It is created in the server part. There are 3 tables in the database, first default that is “sqlite_master”. The second table is “customer” table which contains information about users. Third table is “application” which contains information about user applications. The “sqlite_master” table is unique to SQLite and does not exist in other database management systems. To work with the database, author used “peewee” library.

2.5.1 Customer

The “customer” table is needed to store important data for the user. This table in data base contains such information as: id – primary key, telegram id, user token (generated on the server using uuid4), pc token (generated on the server using uuid4), pc status.

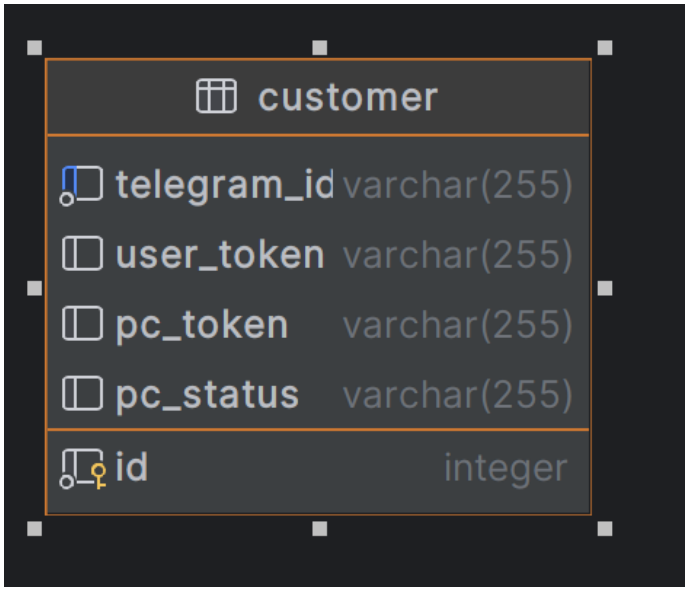

Figure 11. “Customer” table in PCO data base

2.5.2 Application

The “application” table is needed to store information about the user applications. This table in data base contains such information as: id – primary key, user id - foreign key connected with telegram id, app path, app name, app size, app status (for future updates), app favorite.

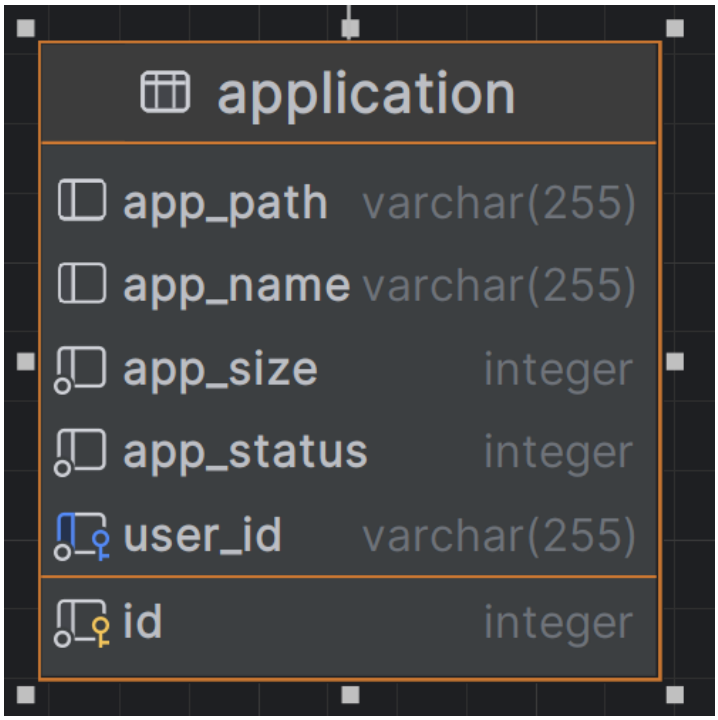

Figure 12. “Application” table in PCO data base

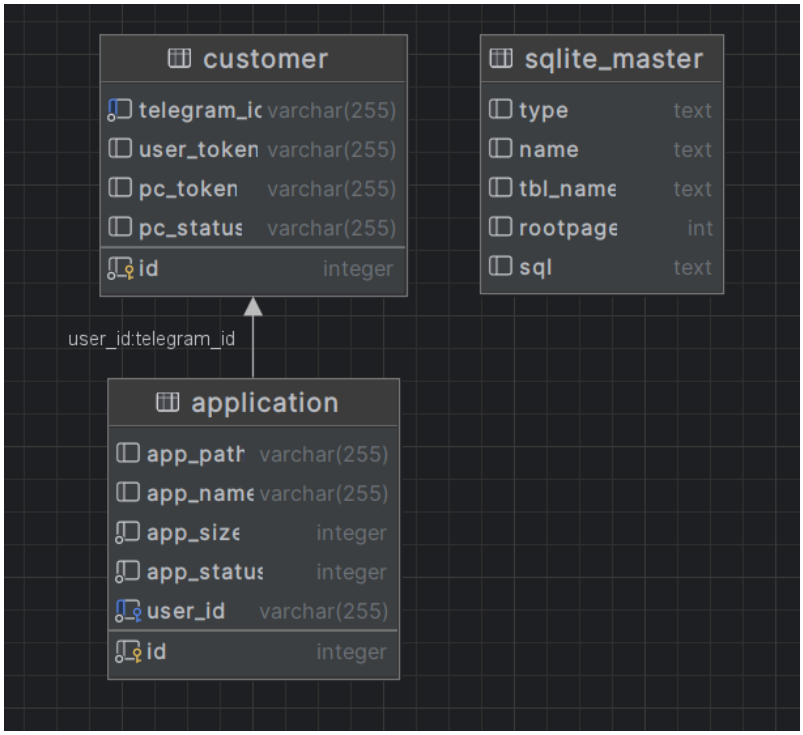

Figure 13. Full data base view

2.6 Server

The server in the "P.C.O." is the primary curator of information. It receives, processes, and sends about 80% of all information to the “P.C.O.”. The JSON format is used to transfer information. Connection to the server takes place both in the bot and in the application. the "command: info" view is used for communication. Server logic is made through receiving commands and information related to it in JSON format. Also, all operations with the database take place on the server.

2.6.1 Server and client application

To implementation the connection and communication with the server, a class "Client" was created, which handles requests to/from the server. The server receives from the "Client" is information about online and different action with the added applications.
To send signals to the application to turn on and off certain applications or to adjust the sound, a separate command "send_command" was created on the server, the logic of which is not processed on the server, instead it is forwarded to the specified user in the application, where all the logic is executed. When an application connects to the server, server adds data about it to a list stored in memory. When the server needs to send information to the client

application, it checks the received data with the data in the list and if there is data in the list, it sends it.

2.6.2 Server and Bot

Server to bot communication is a very important part of the project. It is the bot that sends the server data about the user for further registration. The bot also sends "send_command" command to the server, which is sent to the application. Since the class "Bot" created by the author is a subclass of "Client" - the main class in the library "pyrogram", it was possible to add its own variables, thus implementation the sending of messages to the server. Since now every created bot class has a built-in connection to the server.

2.6.3 Server and Data base

All actions related to the database go exclusively through the server. The server creates tables, fills tables, takes information from them. To work with databases on the server, the class "AIOModel" from the library "peewee_aio.model" is used. All created tables are subclasses.

2.6.4 Server tests

As the server is a crucial part of the project, and the number of commands on the server is quite extensive, manually checking it each time would be challenging. Therefore, the decision was made to write tests. For it using the "pytest" library. The tests cover a significant portion of functions related to server connection and data processing. To execute the tests, another object of the "server" class and a test database are created, and the test database is deleted after the tests are completed.

2.7 Future plans

The production of this project will not stand still, in the future it is planned to add the following:
Enable the application to initiate automatically upon system startup. Integrate the application into the Microsoft Store, subjecting it to security evaluations and making it accessible to a wider audience. Implement a feature to store information regarding the status (enabled or disabled) of applications. Segment the primary functionality into two sections: one for regular

use and another for video playback. In the latter, users will have the capability to rewind and adjust the sound of the currently playing video on their computers.
Add Estonian and English languages.

Conclusion

As a result of a personal issue, which involved difficulties in easily turning applications on and off on the computer and adjusting its sound from a phone, the decision was made to create the PCO application. The PCO project was developed, consisting of a bot, an application, a server, and a database. User authentication and registration are implemented. To successfully log in or register, simply press a single button. For a Telegram account, only one registration is allowed, and in case of an error, the application or bot will display an error as a regular message.

Registered users can add applications, view applications, and delete their applications. Users can add applications to favorites, view their path, launch or close applications through the Telegram bot, and also adjust the computer's sound through it. All application buttons can be pressed, and after pressing, the corresponding request to the server will usually be processed; if the request does not require a database, the application itself will handle it. If a user wants to add an application to favorites, they only need to press the corresponding button. Favorite applications have their list and are highlighted in the bot with a star.

If a user wants to turn on or off an application, the user must add the application to the "PCO" application; it is added by registering the main path to the application in the system. After adding at least one application, the basic functionality of the bot will be unlocked, allowing you to press just a couple of buttons on your phone, turn on any file or application, and adjust the computer's sound.

All the essential functions required in this application are present and ready to use. The application operates stably; as of January 1, 2024, it has been tested on three people, including the author and his computer. At the end of the dissertation, the author describes future work necessary for the application's implementation in production.

During the development, there were problems: The initial, first version of PCO did not have a server and had an unattractive interface for the main application; the interaction with the database was poorly implemented. The application checked the database for changes every 2 seconds; in case of changes, it performed the corresponding action. As a result, there was a rethinking, and the project underwent a complete rewrite; everything was rewritten: the database, the library working with it, the bot and its functionality. The interface in the application was changed, and the logic was completely rewritten; all the code was rewritten in a more readable and high-quality format. A server was created from scratch, which was challenging because the author had no experience in this area.

During the work on the official version of PCO, there were also problems. Starting from trivial bugs, fixing some of which took 3-4 days, to incorrectly written tests and the logic of connecting to sockets in the application. In general, the project was complex and massive.

In the future, I plan to add the following:

Allow the application to start after system launch.
Add the application to the Microsoft Store, where it will undergo security checks and become available to everyone. Add the ability to save information about the state of applications, whether they are enabled or disabled.
Divide the main functionality into two parts: for regular use and for watching videos, in the latter case, there will be the ability to rewind and adjust the sound of the current video on the user's computer.
Currently, both the bot and the "PCO" application exist only in English. In the future, the author would like to add Russian and Estonian languages. In Estonia, most people communicate in Estonian, but there are also Russian-speaking users. Adding languages should help expand the application's audience.

Resümee

Tulemuseks oli isiklik probleem, mis seisnes arvutis rakenduste lihtsal sisse- ja väljalülitamisel ning selle helitugevuse muutmisel telefonist. Otsustati luua PCO rakendus.

PCO projekt töötati välja, hõlmates bote, rakendust, serverit ja andmebaasi. Kasutaja autentimine ja registreerimine on realiseeritud. Edukaks sisselogimiseks või registreerimiseks piisab vaid ühest nupulevajutusest. Telegrami konto puhul on lubatud ainult üks registreerimine ja vea korral kuvab rakendus või bot vea tavalise sõnumina.

Registreeritud kasutajad saavad lisada rakendusi, vaadata rakendusi ja kustutada oma rakendusi. Kasutajad saavad lisada rakendusi lemmikutesse, vaadata nende teekonda, käivitada või sulgeda rakendusi läbi Telegrami bote ja reguleerida ka arvuti helitugevust. Kõiki rakenduse nuppe saab vajutada, ja pärast nupulevajutust töödeldakse tavaliselt vastavat päringut serverile; kui päring ei vaja andmebaasi, siis tegeleb sellega rakendus ise. Kui kasutaja soovib rakendust lemmikutesse lisada, piisab vastava nupu vajutamisest. Lemmikrakendustel on oma loend ja botis on need tähistatud tähega.

Kui kasutaja soovib rakendust sisse või välja lülitada, peab kasutaja lisama rakenduse "PCO" rakendusse; seda lisatakse registreerides põhilise tee rakendusele süsteemis. Pärast vähemalt ühe rakenduse lisamist avaneb bote põhifunktsionaalsus, võimaldades teil telefoni mõne nupuvajutusega käivitada faili või rakendust ja reguleerida arvuti helitugevust.

Kõik selles rakenduses vajalikud põhifunktsioonid on olemas ja valmis kasutamiseks. Rakendus toimib stabiilselt; seisuga 1.
jaanuari 2024 on see testitud kolmel inimesel, sealhulgas autoril ja tema arvutil. Doktoritöö lõpus kirjeldab autor tulevast tööd, mis on vajalik rakenduse rakendamiseks tootmises.

Arendamise käigus tekkisid probleemid: PCO esialgsel, esimesel versioonil ei olnud serverit ja sellel oli peamine rakenduse ebatraktiline liides; andmebaasiga suhtlus oli halvasti realiseeritud. Rakendus kontrollis andmebaasi muudatuste suhtes iga 2 sekundi tagant; muudatuste korral teostati vastav toiming. Selle tulemusel toimus uuesti mõtlemine, ja projekt läbis täieliku ümbertegemise; kõik sai ümber kirjutatud: andmebaas, sellega töötav raamatukogu, bot ja selle funktsionaalsus. Rakenduses muudeti liidest ja loogikat täielikult; kogu kood kirjutati ümber loetavamasse ja kvaliteetsemasse vormingusse. Loodi server nullist, mis oli keeruline, kuna autoril polnud selles valdkonnas kogemusi.

Töötamise ajal ametliku PCO versiooni kallal olid ka probleemid. Alates triviaalsetest vigadest, mõne neist parandamine võttis 3-4 päeva, kuni valesti kirjutatud testide ja sokettidega ühenduse loogika rakenduses. Üldiselt oli projekt keeruline ja mahukas.

Tulevikus kavatsen lisada järgmist:

Võimaldada rakendusel käivituda pärast süsteemi käivitamist.
Lisada rakendus Microsofti poodi, kus see läbib automaatse turvakontrolli ja on kõigile kättesaadav. Lisage võimalus salvestada teavet rakenduste oleku kohta, kas need on sisse või välja lülitatud.
Jagage põhifunktsionaalsus kaheks osaks: tavaliseks kasutamiseks ja videote vaatamiseks, viimase puhul saate videos kerida ja reguleerida kasutaja arvuti hetkevideo helitugevust.
Praegu on nii bot kui ka "PCO" rakendus saadaval ainult inglise keeles. Tulevikus tahaks autor lisada vene ja eesti keeled. Eestis suhtleb enamik inimesi eesti keeles, kuid on ka venekeelseid kasutajaid. Keele lisamine peaks aitama rakenduse sihtrühma laiendada.

References

Coleifer, C. (n.d.). Peewee: a small, expressive ORM. https://docs.peewee-orm.com/en/latest/

Python Software Foundation. (2020, October 5). asyncio - Asynchronous I/O, event loop, coroutines and tasks. https://docs.python.org/3/library/asyncio.html

Pyrogram. (n.d.). Pyrogram - Telegram MTProto API Client Library for Python. https://docs.pyrogram.org

pytest contributors. (2021, September 28). pytest documentation. https://docs.pytest.org/en/7.1.x/contents.html

Kumar, S. (2020, October 9). customtkinter 0.3. PyPI. https://pypi.org/project/customtkinter/0.3/

Python Software Foundation. (2020, October 5). Tkinter — Python interface to Tcl/Tk. https://docs.python.org/3/library/tk.html

Appendices

Git of project:

https://github.com/bratxemss/PCO