**1. Introduction**

A Telegram bot, which can download video and/or audio from Youtube.

* 1. **Purpose**

Often people want to download songs or videos on their local machine, so they would be able to have the access to the media without internet.

* 1. **Scope**

Bot will download files only from Youtube and no other site.

**1.3 Definitions, Acronyms and Abbreviations**

Not needed.

**1.4 References**

Not needed.

**1.5 Technologies to be used**

Three Python packages:

python-telegram-bot - a library that helps you to interact with the Telegram Bot API.  
pytube - a library to download videos from YouTube.  
ffmpeg - a library to convert files.  
eyed3 – a library that gives access to mp3-files metadata.

**1.6 Overview 2. Overall Description**

The concept is simple:

Bot will receive links to videos, user will be able to choose from 2 options:

A) Download audio from one video;  
B) Download multiple audio files from playlist;

That’s it.

User interface:

There are NUM commands:

/start – tells user to check commands /help and /info, to learn how to use bot.  
/help – gives user a description of what this bot is and how some parts of it work.  
/commands – tells user how to use the bot.

If user would use command that was not implemented, bot would send a message, explaining that this command is not supported.

1. To download a file user must send a message that looks like:

“трек, LINK\_TO\_SONG”

Then user would receive mp3-file with an audio from that Youtube video, or a message, explaining what went wrong.

1. To download files from a playlist user must send a message that looks like:

“лист, STARTING\_POINT-END\_PONT, LINK\_TO\_PLAYLIST”.

Then user would receive mp3-files with an audio from that Youtube video, or a messages, explaining what went wrong.

2.1 UseCase Model Survey

Not needed.  
There are not many things that could go wrong:

1. User could provide a broken link or a link to another site.

In this case program would tell user to provide a correct link.

1. The video user wants to download is too big.

In this case user would be asked to wait till download is complete.

1. User may interrupt a download willingly or unwillingly.

All progress would be lost. Such a pity.

1. Youtube may be unavailable.

User would be notified to try again later.

2.2 Architecture diagram & database design

No database required.  
  
Four modules:

1. Main – telegram api lies in that module. All modules are being run from it. Responsible for deleting an mp3-file after it’s was sent to user.
2. Check\_availability – checks if url is correct and response code from server.
3. Get\_mp3 – downloads webm-file, converts it into mp-3, then deletes webm.
4. Prepare\_text – checks if message from user meets the standards to download songs from playlist.
5. TEXTS – contains all texts that would go to user, and other string-type constants.

2.3 Assumptions and Dependencies

3. Specific Requirements

Program should work.

3.1 UseCase Reports

3.2 Supplementary Requirements

None.

4. Supporting Information

None

5. Concerns / Queries / Doubts if any:

None.