TeleSheets Documentation

Release 1.0

Angel

May 21, 2020

CONTENTS:

CHAPTER

ONE

OVERVIEW

This app will allow you to connect your Google Sheet with a Telegram group using a set of commands to automatically configure and keep track of the changes you make in your sheet.

CHAPTER TWO

FEATURES

- Easy to configure.
- Rich set of commands for the bot.
- Supports multiple worksheets per sheet.
- Very easy to expand and modify functionality to suit your needs.
- Friendly and well structured documentation.
- Built on top of well documented libraries like pyrogram and pygsheets

2.1 Configuration

This section will guide you to configure the infraestructure for the app to run, including sensible data from some services that you need to obtain.

2.1.1 Getting the project

The first thing you need to do to start configuring the project is cloning it from GitHub using the following commands:

```
$ git clone https://github.com/angelhodar/TeleSheets
$ cd TeleSheets
```

Once you have done it, then you need to install all the dependencies for this project. If you use **pipenv** (which you should) you can just use this command to create a virtual environment and install all the dependencies:

```
$ pipenv install
```

If you dont use pipenv i have also provided a `requirements.txt` so using pip is just like this:

```
$ pip install -r requirements.txt
```

2.1.2 Environment Vars

Now that you have your project ready, you need to setup some environment vars listed below. It is always a good practice to keep sensitive data out of the code, thats why we are using them. For that, you need to create a `.env` file inside the `telesheets/config` directory. The vars will be loaded into python using the python-dotenv module in the `__init__.py` of that directory, so you can just use:

```
from telegram.config import <VAR_NAME>
```

The `.env` file should contain the following variables:

Telegram

- `TELEGRAM_BOT_TOKEN`: The token you get when creating your bot talking to **@BotFather** on telegram.
- `TELEGRAM_API_ID`: The API ID you obtain when creating your own telegram app.
- `TELEGRAM_API_HASH`: The API Hash you obtain when creating your own telegram app.

To create your telegram app just follow this link. The process is very straightforward, just remember to save the `api_id` and `api_hash`.

Google Cloud

- `CREDENTIALS_PATH` : The path of your credentials **json** file you get when creating your Google Service Account.

To create your service and get the credentials file you can see this video that shows all the process (just until minute 3:00)

Database

- `DB_USER` : The user of your db.
- `DB_PASSWORD` : The password for that particular user.
- `DB_NAME` : The name you want to give to the database when everything is going to be saved.
- `DB_HOST` : URI-style host of your db. For example if using mongodb and localhost it would be *mongodb://localhost/db_name*. If that URI contains **user**, **password** or the **db_name** keywords, they will be parsed with your `DB_USER`, `DB_PASSWORD` and `DB_NAME` values.

Before reading the following paragraphs, you should know that every database relation within the bot is collapsed in the `db.py` module, so the way you handle the functions there to interact with the database is completely up to you, so you could simply use the TinyDB python module, but i would recommend to only use it for playing with the system as it doesnt support important database properties. I also wanted to learn NoSQL so you would probably see that the modules used are too much for the simple use i have done with them. With that said, you can continue reading :)

The database system used in this project is MongoDB using mongoengine as ORM. In order to use mongo, you need to install it to run in your localhost with the following command:

```
$ sudo apt update
$ sudo apt install -y mongod
```

You can check if its running with the following command:

```
$ sudo systemctl status mongodb
```

MongoDB also has a friendly GUI client which i recommend you to install to have a nice view of your db. You can get it from here.

There is another option if you want to avoid installing anything on your computer, which is signing up on the cloud service MongoDB Atlas, which gives you a 512MB free cluster! (its something).

2.2 Custom Settings

Once you have configured the most important and sensible data of your app, now the last thing is to adjust the rest of the app settings to your needs. All the settings are stored in the `constants.py` located in the `telesheets\config` directory. These variables are grouped by usability so lets explain each group and its variables:

2.2.1 Commands

The values of this variables will define the text that the user needs to put in telegram with the / character to invoke each command (for example */start* or */commands*). These are:

- `START`: Shows simple start message.
- `CONFIG`: Tells the administrator how to connect and configure the group with Google Sheets.
- `COMMANDS`: Shows a message with all the available commands.
- `SHEET`: Configures the sheet url passed by parameter for the group where it is invoked.
- `CHECK`: Tells the administrator if there is any extra configuration needed.
- `EMAIL`: Shows the bot email used to share the group admin's Google Sheet.
- `CALENDAR`: Shows the calendar of nearby events.
- `ATTENDANCE`: Sends the attendance to the student.
- `GRADES`: Sends the grades to the student.

Note: In the case of `ATTENDANCE` and `GRADES` the info is sent by private message to the student instead of showing the data in a group message. The way the commands send the data varies depending on who invoked the command:

- If invoked by the **group admin**, each student will receive a private message with their grades.
- If invoked by a **student**, only that particular student will receive the grades.

2.2.2 Sheet

When working with your Google Sheet, the app needs to know some data about your worksheets:

- `GRADES_WKS_NAME`: The name of your grades worksheet.
- `ATTENDANCE_WKS_NAME`: The name of your attendance worksheet.
- `CALENDAR_WKS_NAME`: The name of your calendar worksheet.
- `IGNORE_HEADERS`: A dict containing a list of header names that will be ignored when sending data to students. For example, you need a header called `Telegram` in your grades and attendance worksheet for the bot to identify each student, but you dont need to send that info to the students.

2.2.3 Messages

There are some messages that will be sent to the group while configuring it or invoking some commands that always shows the same message (like the commands list for example).

Errors

- `COMMAND_ONLY_ADMINS`: Showed if a non-admin group member executes a command that requires admin privilege.
- `URL_ERROR`: If the passed sheet url to the `SHEET` command is not a Google Sheet.
- `INVALID_SHEET`: The url passed has the Google Sheet url format but the sheet is not valid.
- `NO_BOT_ADMIN`: Showed when invoking a command and the bot doesnt have admin privileges in the group.
- `NO_SHEET`: If the group doesnt have a sheet configured.

Confirmations

- `SHEET_UPDATED`: Showed when using the `SHEET` command succesfully.
- `CONFIG_SUCCESSFUL`: Showed when everything is well configured and there is nothing more to do.
- `GRADES_SENT`: Showed when an admin invokes the `GRADES` command.

Commands

- `START_PRIVATE`: Shoed when invoking `START` in a private conversation.
- `START_GROUP`: Showed when invoking `START` in a group.
- `COMMANDS_LIST`: Showed when invoking `COMMANDS`.
- `CONFIG_MESSAGE`: Showed when invoking `CONFIG`.

2.3 Running App

Once everything is configured to your needs, to run the app you just have to execute the `bot.py` module. If using pipenv:

```
$ pipenv shell
$ python bot.py
```

Or with just a single command:

```
$ pipenv run python bot.py
```

If you want to stop it just press Ctrl + C. Now its your turn to deploy the app in the cloud service of your choice to keep it running 24/7!