
Better Deal: Documentation

Updated as of: 2020-12-04

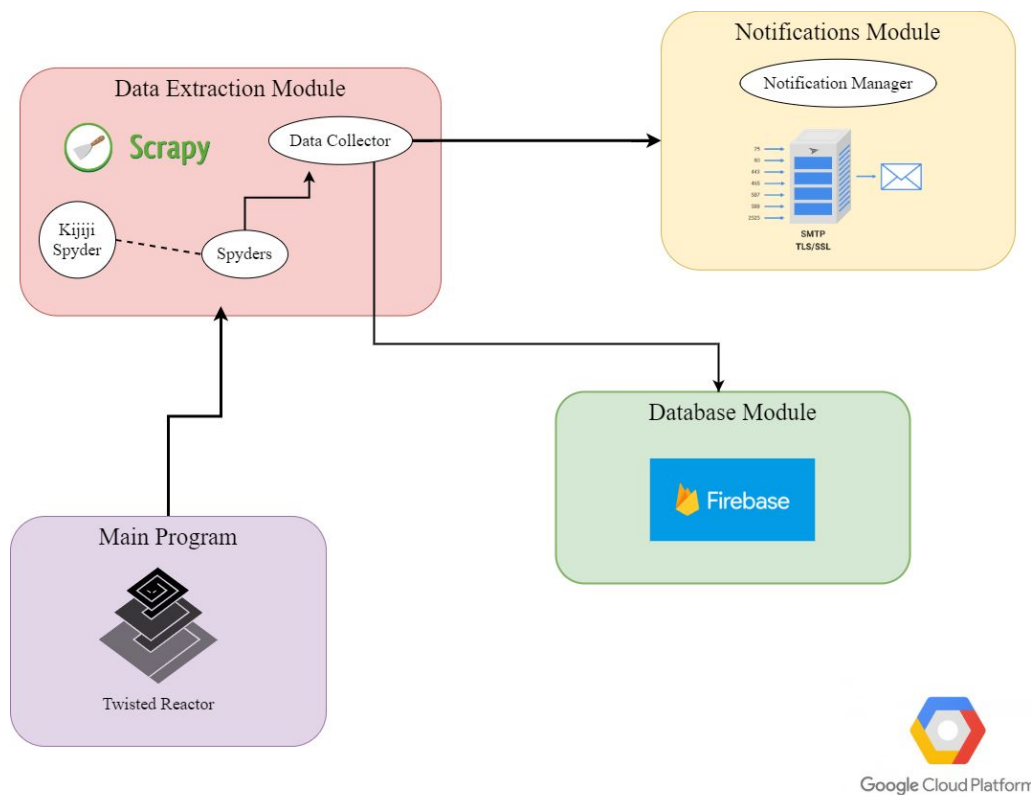
Server side

Code located at <https://github.com/uritrejo/BetterDeal.git>

This part of the documentation will refer to the service covering the following features:

- Data Extraction
- Data Collection
- Email Notification

Design



Please refer to the code comments for more details on implementation.

Running the better deal service

The server side is meant to run perpetually on a linux based cloud server.

Downloading packages and code

Assuming a clean VM, the following steps will need to be followed to get the process running. The commands can be run from any directory in the VM unless otherwise specified:

- General update
 - `sudo apt-get update`
- Install nano (Not required but will come in handy later)
 - `sudo apt install nano`
- Install python
 - `sudo apt-get install python3.6`
- Install pip3
 - `sudo apt-get install python3-pip`
- Install python packages
 - `pip3 install pycryptodome`
 - `pip3 install python-jwt`
 - `pip3 install jwt`
 - `pip3 install twisted`
 - `pip3 install Scrappy`
 - `pip3 install scrapy-user-agents`
 - `pip3 install pyrebase`
- Install git
 - `sudo apt install git-all`
- Clone repository (download program from Github):
 - `git clone https://github.com/uritrejo/BetterDeal.git`

Running the service

- Go into directory *BetterDeal* (downloaded from git)
 - `cd BetterDeal`
- Run the program
 - `nohup python3 main.py &`

- nohup is for "no hangups", so the process keeps running until manually stopped
- using '&' makes the process run in the background

At this point, the process should be running and retrieving data from the searches added in the database. To corroborate this, you can run the command `ps -ef` and look for 'python3 main.py' in the displayed list. Another way is to run, from the *BetterDeal* directory, `nano LogServer.Log`, this will display the logs of the server process in a fairly user friendly manner.

Once the process is running, no further action is needed, you can close your window of access to the VM and test using the Better Deal UI.

Useful commands to know

- To show all of the processes running (including the Process ID, PID of better deal):
 - `ps -ef`
- To kill the program in the VM (most likely case in need of resetting)
 - `kill PID`
 - PID is the number obtained from the previous command.

Current Deployment Information

At the time of deployment, the virtual machine is running at Google Cloud Platform.


The virtual machine is located at the Google Cloud Platform Console website (<https://console.cloud.google.com/>), under the account better.deal.notifier@gmail.com. In particular, it can be found under Compute Engine / My First Project / VM-Instances / better-dealer.

Note that this is a command line only ubuntu based VM, for any information regarding its usage you could refer to

<https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners#:~:text=Its%20distros%20come%20in%20GUI,%2Dterminal%2C%20and%20press%20enter.>

Additionally, google may be the perfect resource to find this information.

Configuration of Server



In the root directory, a file named *config.py* can be found. In this file, configuration information is specified. Fields such as source and destination email address are included here. In case any change is needed on this file, please restart the service after the modifications are made.

Logging

The application logs will be written to *logsServer.log* found in the root directory. This text file will contain information about the actions of the program, including date and time (UTC). In the same manner, almost any potential error (connection, memory, bad access, etc) will be logged in this file with all available details. In case of any unexpected behavior, please see this file.

A way to view the logs is: from the *BetterDeal* directory, run *nano logsServer.Log*, this will display the logs of the server process in a fairly user friendly manner.

Potential Troubleshooting Cases

If you run into any problem during any timespan, the problem could be on the client side or on the server side. The first step is to look into the logs of both programs (see *Logging* section). This will provide great insight into the problem (for example internet connection is down, gmail or firebase services are unavailable).

- If the server is failing at sending emails, it's most likely related to the permissions on the better dealer gmail account, make sure that less secure apps are enable, please follow the steps shown at: <https://hotter.io/docs/email-accounts/secure-app-gmail/>
- If the process seems to be running yet there is no updates from the cars logs show no error, a potential solution can be found at: <https://superuser.blog/debugging-stuck-process-linux/>
- Other potential problems could appear following a change in Kijijis website, in such case, code changes from someone with technical knowledge will be needed (code is well commented).

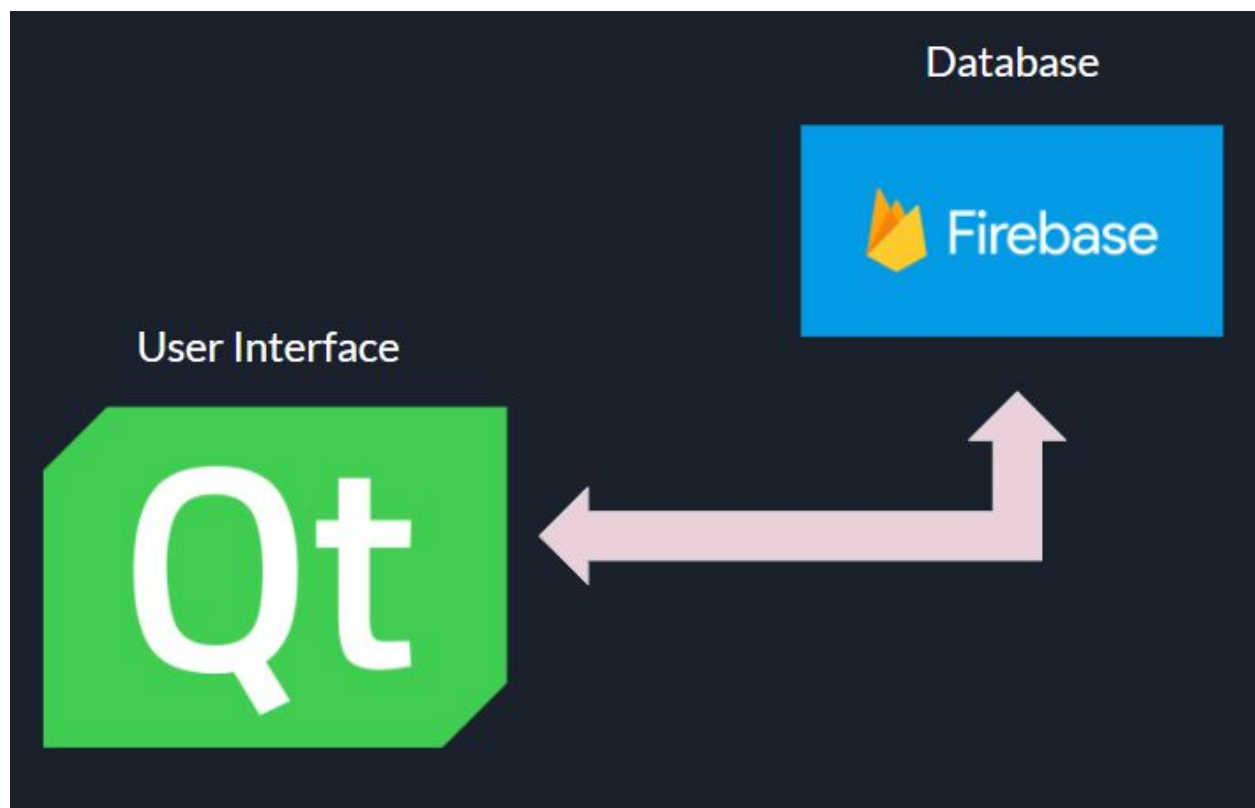
Client side

Code located at: <https://github.com/uritrejo/BetterDealUI>

This part of the documentation will refer to the User Interface that provides the following functionalities.

- Adding cars / links to the search list.
- Data display.
- Downloading data from the database.
- Other database functionalities.

Design



Please refer to the code comments for more details on implementation.

Running the User Interface

The user interface, in contrast to the server side, is a desktop application that can be opened and closed any number of times.

It is written in Python, yet only tested for compatibility with Windows. Please notice that a virtual environment (*venv* at root directory) containing all the software libraries and packages is already in place and is compatible with Windows. To run the UI, please run the file: '*betterDealUI.bat*' found in the root directory (Either by running the command on the command window or simply by double clicking on the file).

Logging

The application logs will be written to *logsClient.log* found in the root directory. This text file will contain information about the actions of the program, including date and time (UTC). In the same manner, almost any potential error (connection, memory, bad access, etc) will be logged in this file with all available details. In case of any unexpected behavior, please see this file.

Database

The database was implemented using Firebase . It can be accessed at <https://console.firebase.google.com/> at the Firebase Project *BetterDeals*, under the account better.deal.notifier@gmail.com . Please refer to the we

The tables containing the data can be found under *Real-Time Database* in the aforementioned link. The structure is straight-forward: A table called *Cars* contains the cars extracted from the searches, and a table called *Searches* contains the searches that the server will look into.

For more information on Firease, please refer to their website.