GPU In-stock Bot User Docs

A high-level description:

The GPU In-Stock Bot notifies users when GPUs come in stock on various online retailers and resellers. Users can opt into this notification service via Email or phone number (or both) and select a GPU(s) from our assorted list to be notified for. The service currently only offers stock alerts from Best Buy. Users can unsubscribe from this service (ending ALL notifications for ALL graphics cards) by inputting their information, like they would when they subscribe. The service will let the user know via their preferred notification methods that they have unsubscribed.

How to Install the Software:

Requirements:

1. **OS Version:** Mac version 10.0+ or Windows 10
2. **Python:** Have Python 3.6+ https://www.python.org/downloads/
3. **Pip:** Have the latest version of the pip package installer: https://pip.pypa.io/en/stable/installing/
4. **Virtualenv:** Have virtualenv installed using pip NOT brew (I actually do not know if this matters, but for continuity, we installed virtualenv using pip3 and on a Mac): https://gist.github.com/pandafulmanda/730a9355e088a9970b18275cb9eadef3

Pre-OS Specific Instructions:

5. **Clone the repository:** Clone it from GitHub using this link: https://github.com/argonaeut/GPU_Instock_Bot

Mac:

1. **PostgreSQL:** If you don't have it already, our system used PostgreSQL for the database so make sure to install it from here: https://www.postgresql.org/download/macosx/
    a. **INSTALL THE POSTGRES.APP**: There are several types of downloads for Mac OS PostgreSQL download section, choose the postgres.app download for the easiest experience.
    b. Open the app and click the start button
2. **Navigate to Directory of Local Repo:** Open the Mac terminal and change into the GPU_Instock_Bot directory
3. **Create a virtual environment:** Within the GPU_Instock_Bot root directory, create a virtual environment by running '*virtualenv {desired_name}*' using terminal
4. **Activate the virtual environment:** Within the GPU_Instock_Bot root directory, activate the virtual environment by running '*source {desired_name}/bin/activate*' using terminal
5. **Install dependencies:** Within the GPU_Instock_Bot root directory, install dependencies by running '*pip install -r requirements.txt*' using terminal
6. **Set up the database:** Using terminal & starting from the GPU_Instock_Bot root directory

   a. Run command: '*cd GPU_Instock_Bot_V2_src*'
   b. Run command: '*chmod u+x setupdb.sh*' (**NOTE: different if doing peer review, run: *chmod u+x peer_review_setupdb.sh***)
   c. Run command: '*./setupdb.sh*' (**NOTE: different if doing peer review, run: *./peer_review_setupdb.sh***)

Windows: (NOTE: Windows OS build is untested/unfinished)

1. **PostgreSQL:** If you don't have it already, our system used PostgreSQL for the database so make sure to install it from here: https://www.postgresql.org/download/windows/
   a. Choose all default values and be sure to remember the password you specified.
   b. Once installation has completed be sure to uncheck the option to run the package installer as it is not required
   c. You can verify that a PostgreSQL database server is now running on port 5432 by running the command or '*service postgresql-x64-13*' in PowerShell.
   d. You can also start/stop the service by running the command: '*start/stop-service postgresql-x64-13*' in PowerShell
2. **Get to directory:** Open the PowerShell and navigate to the GPU_Instock_Bot directory
3. **Create a virtual environment:** Within the GPU_Instock_Bot root directory, create a virtual environment by running '*virtualenv {desired_name}*'
4. **Activate the virtual environment:** Within the GPU_Instock_Bot root directory, activate the virtual environment by running: '*.\{desired_name}\Scripts\activate*'
5. **Install dependencies:** Within the GPU_Instock_Bot root directory, install dependencies by running command: '*pip install -r requirements.txt*' using terminal
6. **Set up the database:** Using command prompt & starting from the GPU_Instock_Bot root directory
   a. Run command: '*cd GPU_Instock_Bot_V2_src*'
   b. Run command: '*pip install psycopg2*'
   c. Run command: '*psql -U postgres -f psqlcreator.sql*'
   d. Run command: '*psql -U postgres -d gpuinstockbotdb -f beta_datadump*' (**Note:** For peer review builds, use the **peerreview_datadump** to populate the database!)

How to Run the Software:

1. **Start Django Web Server:** Navigate to the GPU_Instock_Bot_V2_src directory run command: '*python manage.py runserver*'
   a. This will run the Django server on the default: localhost:8000 (if this port is being used, go to the port you personally specify)
   b. To view the database, go to localhost:8000/admin (user = admin, pass=superuser)

2. **Before starting the backend service:**

   a. First, the authentication token for Twilio (for our text notification feature) will last roughly a half day. Before starting the script this authentication token needs to be inputted again. The documentation said it'd be live for anywhere from an hour to 24 hours - also noting that it may be shorter because twilio keeps finding out that our api keys are public. Because our service isn't meant to be hosted on different machines over and over (as is the case in this peer review assignment), please contact any of us (Andy Lee, Derek Carlson, Leah Tran, David Cueva, Milan Crone, or Elizabeth Lin) and we will send you a fresh authentication token for you to use.
   b. The file you'd want to edit for inserting the new authentication token is in ScrapeQueryNotifierMicroservice/scrape_query_notify.py. in line 388, you will want to edit the second field of Client(.., ...)

3. **Start the Backend Service**
   a. You may want to hold off on this step until you have successfully subscribed to some GPU supported by our service via the GUI. For more information see “How to Use the Software”
   b. Navigate to the ScrapeQueryNotifyMicroservice folder and run command: ‘*python scrape_query_notify.py’.* This will start the backend service to scrape GPU Stock from Best Buy and notify subscribers of when their preferential GPU is in stock to purchase.
   c. After starting the backend service, the Gmail authentication page might pop up requesting access to send emails from our service account, so you can log in using these credentials once that occurs. (gmail account: user = gpuinstockbot@gmail.com pass=cse403group).

<u>How to Use the Software</u>:

Subscribing: We have one test GPU up that can be subscribed to (Nvidia GT 710). Multiple GPUs can be subscribed to if the PostgreSQL database is initialized with the beta_daadump file. However, none of them are in-stock and probably won't be for a while.

1. **Phone Number:** Enter in the phone number box a valid US Phone number in this format: +1 and then your 10 digit phone number i.e. +12061234567.
   a. DISCLAIMER: IF TESTING PHONE NUMBERS, WE MUST MANUALLY ADD YOUR PHONE NUMBER TO THE VERIFIED RECIPIENTS ON OUR TWILIO CONSOLE DUE TO BEING ON A FREE VERSION. IF YOU'D LIKE TO TEST THIS CONTACT US (contact information at step 3 via slack).
2. **Email:** Enter a valid email to subscribe to in the email box.
3. **Select GPU:** Check the box on the (Nvidia GT 710)
4. **Click Subscribe:** Click the red subscribe button to subscribe and watch the magic happen!
5. **Unsubscribe: DO NOT USE UNSUBSCRIBE, IT IS UNFINISHED**

<u>How to report a bug:</u>

Bugs can be appended to this github file:
https://github.com/argonaeut/GPU_Instock_Bot/blob/main/BUG_REPORT.md

If you do not have access to append a bug to the above file, you can email me the bug (derekcar@cs.washington.edu) and I will append it to the file.

The new and improved format for reporting bugs will be as follows:

1. **Name:** Write down the name of the feature contain the bug, for example "The Subscribe Button"
2. **Description:** Give a short description of the bug
3. **Environment:** Give the details about the environment the bug occurred in. Which browser, IDE, Computer OS, etc.
4. **Console Logs:** Copy and paste the output of the lines relevant within the console log when the bug occurs.
5. **Visual Proof:** Capture a screenshot of the bug when it occurs
6. **Steps to reproduce:** Write a detailed list of the steps needed to reproduce the bug.
7. **Expected vs. Actual Results:** Explain the original intended results/correct functionality that was supposed to happen vs. what actually happened.
8. **Severity:** Rank the the severity of the bug (critical, major, minor, trivial)

Known bugs:

Existing bugs are documented at :
https://github.com/argonaeut/GPU_Instock_Bot/blob/main/BUG_REPORT.md

- Unsubscribe button, work in progress.
- Test environment credentials don't work, live credentials used.
- Both email and phone number are required

GPU Instock Bot Developer Docs

How to obtain the source code:

Source code can be obtained from: https://github.com/argonaeut/GPU_Instock_Bot.

Directory Structure:

- **.github/workflows:** Contains the 'main.yml' file for CI configuration with github actions.
- **GPU_Instock_Bot_V2_src:** Contains all files associated with the Django web server and its configuration with the PostgreSQL database server.
    - **dashboard:** Contains general configuration files for the Django web server
    - **gpus:** Contains all files associated with gpu database objects
    - **staticfiles:** Contains all files associated with the frontend user interface
    - **subscribers:** Contains all files associated with subscriber database objects
    - **subscriptions:** Contains all files associated with gpu database objects
- **GPU_Scrapers:** Contains all files associated with scraping GPU Stock information from the internet
- **ScrapeQueryNotifierMicroservice:** Contains all files associated with running the scrapers, querying the database for subscriptions, and notifying subscribers via text and email if a tracked GPU is in stock.
    - **drivers:** Contains some chrome drivers for scraping services
- **Weekly_Reports:** All markdown files associated with our weekly reports
- **tests:** Contains all files associated with testing the backend modules
- **BUG_REPORT.md:** Bug reports for the application are compiled here
- **README.md:** General system info such as motivation and installation is compiled here
- **requirements.txt:** All python requirements for running the Django, PostgreSQL, and the backend modules are compiled here

How to Test the Software:

Django Tests:

- Generalized: '*python GPU_Instock_Bot_V2_src/manage.py test [directory/app_name]*'
- '*python GPU_Instock_Bot_V2_src/manage.py test gpus*'
- '*python GPU_Instock_Bot_V2_src/manage.py test subscribers*'
- '*python GPU_Instock_Bot_V2_src/manage.py test subscriptions*'

Backend Module Tests:

- Generalized: '*python -m unittest tests/{test_name.py}*'
- '*python -m unittest tests/test_best_buy_scraper.py*'
- '*python -m unittest tests/test_notifiers.py*'
- '*python -m unittest tests/test_scraper.py*'
- Or to run all module tests at once: '*python -m unittest discover -s tests -t tests*'

These commands are also found in the CI file `workflows/main.yml`. To run a new test file in addition to the ones already implemented, run the corresponding command with the right formatting for the type of test. If adding new automated tests, add the command for it to the corresponding section in the CI file `workflows/main.yml`.

Making New Module Tests:

Tests are written in the ‘tests’ directory and under certain directories in the `GPU_Instock_Bot_V2_src` folder.

The `tests` directory contains tests for backend functionality, such as the scrapers and notification modules. These use unittest, which is supported by Python out of the box.
Tests can be added to the files that are titled `test_*.py`, such as `test_notifiers.py`, or a new test file can be created in the directory titled in the same `test_*.py` format.

When adding new tests based on different web layouts (item in stock/ out of stock), save a snapshot of the website by getting its raw html (Ctrl + U on windows) and saving it as an html file in the tests directory. In order to access and run the unittests off of the html that has been saved follow these directions in any python test file:

- Get file info like so: *html_page = os.path.join(os.path.dirname('tests/your_file'), 'your_file.html')*
- In the setUp method open that file and begin reading like so
  - *file = open(html_page, encoding= “utf8”)* Note:utf8 may not be needed depending on the html extracted
  - *data = file.open()*
- In the tearDown method close the file after you are done reading like so
  - *file.close()*

Making New Django Tests:

The Django tests are in the directory of `GPU_Instock_Bot_V2_src` under the following folder that correspond to each component: subscriptions, subscribers, gpus, dashboard. The tests are checking different scenarios of user authentication, identification, and component functionality. We add tests inside each subfolder, to test the specific component we are viewing. Internal tests allow us to check and validate product verification and functionality. Follow the steps below to guide you through how to create tests for each Django Component.

How to Add and Modify Tests for Components:

1. Create a tests.py file if test.py does not exist in the component directory. If the test.py file already exists, then you can go to the next step.
2. Add `from django.test import TestCase` to top of the file
3. Create a class called Test{NameOfComponent} and subsequent functions that represent different use cases. You will be able to import other components in your tests (For Example, you can have a test that uses subscription and subscriber objects by creating them in your function).

4. Modify functions to include multiple tests in each function within your Test class. It is best to organize your functions to represent a type of test/scenario you want to check. Adding multiple tests per function is possible.

More documentation about writing Django tests can be found here.

How to Build the Software:

Reference how to install and how to run in the user documentation.