**Bugzilla Data Mining Notes**

**Introduction**

Hello future interns, I am putting together as much of a comprehensive summary of the progress made into the Bugzilla Data Mining project as possible to make this transfer of knowledge as seamless and efficient as possible. However, I am an intern majoring in mechanical engineering, so my understanding of many of these topics is limited and may not be entirely accurate.

**Software**

1. Bugzilla

Bugzilla is a product of Mozilla (the company that made Firefox). It is used by TMEIC to log and track the status of bugs that arise on their various projects. When bugs are created, users can enter a lot of information on them to help others understand the problem/severity and chat about fixes.

Many different data entries are stored for each bug, making it possible to make very specific searches in Bugzilla’s database. The current project pulls data out of Bugzilla as a .json file. Bugzilla lacks the ability to easily view comprehensive bug data or common-verified solutions to prior bugs, which TMEIC project managers would find useful.

An API key for Bugzilla will allow you access to the database. From the home page of Bugzilla, click “User Preferences” -> “API Keys” -> click the checkbox “Generate a new API key with optional description” -> click on “Submit Changes” button. Make sure in any queries made to the database that you put an “&” between each filtering element.

1. Postman

Postman is an app that can be used to get and view information from online databases. We were able to get access to the database for Bugzilla and scroll through the exact .json files (data storing files) pulled by the searched links. It is a good tool for checking that the code is pulling the desired info in a clean formatting.

Make sure the request is set to “GET” and that the link you are querying begins with https://tools.tmeic.com/mh/rest/bug? (only if searching in mh Bugzilla). After that, different search criteria can be added into the link (check this link to Bugzilla [6.1.3. Bugs — Bugzilla 5.0.6 documentation](https://bugzilla.readthedocs.io/en/latest/api/core/v1/bug.html) for the type of data stored in the database and what keywords to use to search it). For example, to filter the search to Critical bugs, append “severity=Critical” to the link above.

Make sure that at the end of the link you append the API key. API keys give you access to the database. They act as the “middleman” between the database and the user who wants the data. The API key goes at the end of the “GET” link and will look like “api\_key=3IOYXyOoNDIM2E37BSQetdS9npaATJkakaGdCBE6”. However, you will need to generate your own API key from Bugzilla because mine will not work once my account is deactivated.

1. PyCharm

PyCharm is an IDE (integrated development environment) by JetBrains. It is what we used to code and run the python scripts we wrote for this project (free trial version lasted us long enough to get our work done). We were able to retrieve data directly from Bugzilla’s database and paste info directly into Excel spreadsheets.

We were able to set up GUIs (graphical user interfaces) in PyCharm to avoid needing to enter search parameters in the “run” window. We were then able to generate .exe (windows executable) files all within PyCharm. This effectively made our demo codes into executable programs with GUIs that could be used by anyone so that they do not need to run any python scripts in PyCharm.

All that is required to download to make the apps work is supporting Excel files for the two 2-week tracker demos.

1. Excel

Excel is Microsoft’s spreadsheet tool. I’m sure every demo we came up with could have been accomplished entirely within python, but we did not have the time nor experience to figure that out in this project. Excel has an internal code, Visual Basic, that can be used to automate certain processes. To use this, the generic (.xlsx) excel file needs to be saved instead as a macro-enabled spreadsheet (.xlsm).

Within the macro-enabled sheet was written a script in Visual Basic. This is accessed by turning on the “Developer” setting in the .xlsm spreadsheet. Only the demo that sends an automated email each day used the macros. Xlsx works for the searchable tracker.

1. ChatGPT

ChatGPT was great for asking very detailed and wordy questions and getting the exact in-depth answer desired. It helps explain and offer suggestions regarding any technical issues you might be experiencing trying to get things working as long as you give it enough information to work with. It can also be used to generate detailed breakdowns and review of any code (I frequently asked it to explain codes to me as if I had no prior understanding of what coding is).

**Demo Codes**

Working on this project, a few important demonstrative codes have been tested successfully as a starting point for continued efforts.

**General notes:**

* Ensure all file paths of excel spreadsheets match the path used in the codes so the demos run properly.
* All 3 demos run slow, so please be patient. The 2 .exe files may take 10-20 seconds to pull up the GUIs, and all 3 demos may take 15-45 seconds to run before results are pulled up. If it has been much longer than this with no results displaying or emails sending, then something is wrong.
* MAKE SURE TO GET A NEW API KEY
  + My key in the python scripts shouldn’t work after my account is deactivated.
  + The API keys are specific to the instance of Bugzilla you are getting it from
    - Metals API key only works for Metals Bugzilla; MH API key only works for MH Bugzilla

**2-Week Bug Tracker (Automated):**

* Uses python to retrieve data from materials handling Bugzilla at designated intervals and automatically pulls last 14 days of most severe (Blocker/Critical) bugs and high-moderate severity (Major/Normal) bugs created.
  + Can change the time interval to whatever you want for testing. We often tested at 1-minute intervals instead of 1 day to make sure that it would send multiple emails without errors.
* This data is pasted into an Excel spreadsheet (macro enabled - .xlsm), updating a table that is used to generate a line chart, hence updating the chart.
  + Make absolutely sure that you have your excel in the correct file path for the python script to find. You can change the excel file path to suit your needs in the python code file, but if you choose not to do this then make sure it is saved to the path "D:/OneDrive - TMEIC/Desktop/Bugzilla Tracker/TestEmaildoc.xlsm".
  + Using Visual Basic (Excel’s own code language), updating the table triggers Excel to send an automated email to designated recipients.
  + The sending and receiving email address will need to be changed as it currently uses the deactivated accounts of interns from the summer (2023).
* The whole process is automatic and works if the python script is running constantly in the background. The next step in this demo is to figure out how to have this process run automatically without needing to run python on some employee’s laptop 24/7. Another thing would be to make the potted data more specific to what certain project managers are interested in seeing.
  + Maybe 3rd party hosting services?

A graph on a black background

Description automatically generated

Figure 1: Example of Automated Email

**2-Week Bug Tracker (Searchable):**

* Uses python to retrieve data from Bugzilla when the “Run Bug Tracker” button is clicked in the GUI.
  + If running the code in PyCharm, it will open the GUI directly for use.
  + To run the exe, just double-click the “PlotSearch.exe” file and it will run the program from what looks like the Command Prompt windows app. The GUI takes longer to pull up on the screen than when running from PyCharm, so a bit of patience is required. The program should run without complication.
* Whatever data is entered into the GUI will filter down the API searches to Bugzilla’s database. Right now, entries need to be typed in with perfect spelling or else it will not work, so dropdown menus might help avoid mistakes. The last 14 days of Bug data is sorted to get a count of bugs created on each day that are still open at the time of search or have been closed at some point since creation. It is NOT a tracker of WHEN bugs were opened or closed. It tracks bugs CREATED ON EACH DAY, with two different datapoints on each day to show which of those bugs are STILL OPEN or have CLOSED AT SOME POINT since creation when the search is run. Sorry for the aggressive caps, but people have been getting this confused. I hope that clears up questions you may be having about the data displayed by this demo.
* This data is pasted into an Excel spreadsheet (just a normal one - .xlsx), updating a table that is used to generate a line chart, hence updating the chart.
  + Make absolutely sure that you have your excel in the correct file path for the python script to find. You can change the excel file path to suit your needs in the python code file, but if you choose not to do this then make sure it is saved to the path "D:/OneDrive - TMEIC/Desktop/Bugzilla Tracker/SearchedBugsPlot.xlsx".
  + No use of visual basic in this demo because it is not sending an email.
* After the data is entered to the Excel file and the table/chart have updated, the python script temporarily saves the chart as an image (.png) and pulls up that image in the Photos app on the desktop of whoever is running the search.

A screenshot of a computer screen

Description automatically generated

Figure 2: GUI and Image Generated Example

**Bug Counter:**

* Uses python to retrieve data from Bugzilla when the “Count Bugs” button is clicked in the GIU.
  + If running the code in PyCharm, it will open the GUI directly for use.
  + To run the exe, just double-click the “BugCounter.exe” file and it will run the program from what looks like the Command Prompt windows app. The GUI takes longer to pull up on the screen than when running from PyCharm, so a bit of patience is required. The program should run without complication.
* Whatever data is entered into the GUI will filter down the API searches to Bugzilla’s database. Right now, entries need to be typed in with perfect spelling or else it will not work, so dropdown menus might help avoid mistakes. This counter is very simple and goes through the material handling database and counts the number of bugs that are still open and the number of bugs that have been closed that fit the search criteria.
  + When an extremely broad range of bugs is being searched, some noticeable and unforeseen problems arise:
    - It has become apparent that the .json file in which the bug data is stored holds a max of 10,000 bugs from a search. Bugzilla has been used extensively by TMEIC since the early 2010s, so this becomes an issue when searching for all bugs created since before 2017. It is apparent that the counts become inaccurate above the 10,000 total bug threshold and I am not sure how to compensate for that (although it is unlikely that this information is something employees/manager will care about).

A screenshot of a computer

Description automatically generated

Figure 3: GUI with Example Response Generated