IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 11, Issue 6, June 2022

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.118

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

Extracting of Customer-Support Chats Using Twitter-Bot

Mr.R. SRINIVAS

Sr. Assistant Professor, Dept. of CSE, MGIT, India

ABSTRACT: Many Apps include built-in options to report bugs or request features but also users still provide an increasing amount of feedback via social media, like Twitter, Facebook etc., for transparency. Compared to traditional issue trackers, the reporting process in social media is unstructured and the feedback often lacks basic context information, such as the app version or the device concerned when experiencing the issue. To make this feedback actionable to developers, support teams engage in recurring, effortful conversations with app users to clarify missing context items.

The main objective of this paper is about a simple approach that accurately extracts basic context information from unstructured, informal user feedback on mobile apps, including the platform, device, app version, and system version. For the extraction of the data, we will be using the tweets from twitter which contains little information about the problem. Combined with a chat-bot, that automatically requests missing context items from reporting users. The data which is being extracted from the user accounts are then analyzed which is used for developing bug reports and streaming live on issue tracker.

KEYWORDS: mobile apps, chat-bot, unstructured, bug reports.

I. INTRODUCTION

Modern apps include options to support users in providing relevant, complete, and correct context information when reporting bugs or feature requests. For example, Facebook attaches more than 30 context items to bug reports submitted via their apps, including the app version installed and the device in use . Despite the presence of such options, an increasing amount of users still report their issues via social media, such as Twitter. A possible reason might be to increase the pressure on software vendors through the public visibility of reported issues. Research has shown, that mining tweets allows additional features and bugs to be extracted, that are not reported in official channels as app stores . Mezouaretal. found that one third of the bugs reported in issue trackers can be discovered earlier by analysing tweets .

Many app vendors are aware of these benefits and have thus created Twitter support accounts as @Netflixhelps, @Snapchatsupport, or @SpotifyCares. Compared to structured reports in issue trackers that usually include context items , feedback on Twitter is primarily provided by non-technical users in a less structured way . Tweets that miss basic context items, such as the concerned platform, are likely to be non-actionable to developers. Hence, Fig. 1. Example of a Conversation between User and Support Team on Twitter to obtain Missing Basic Context Items (i.e., the Device and Platform).

Several support accounts prominently highlight the importance of this information in their Twitter bio. For instance, Spotify's profile includes "for tech queries, let us know your device/operating system", while Netflix states "for tech issues, please include device & error". However, tweets, such as "I can't open playlists shared via WhatsApp on my iPhone XR, iOS 12.1.4, Spotify 8.4.61" that include all basic context items, i.e., the concerned plat-form, device, app and system version, are rare. In contrast, support teams engage in recurring, effortful conversations with users to obtain missing information, as shown in Figure 1.1.

II. LITERATURE SURVEY

Developers organize their work using issue trackers. An issue usually corresponds to a unit of work to accomplish an improvement in a software system. Issues can be of different types, such as bug reports or feature

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

requests. When creating an issue of a specific type, issue trackers use structured templates that request specific context items to be provided by the reporter. Bug reports require the affected app version, while feature requests require a description of the desired feature. Traditionally, Research has shown that users include requirements-related information such as bug reports in about one third of their informal feedback. Recent studies specifically emphasized the benefits of mining tweets.

We aim to automatically extract basic context items from tweets. Our approach is intended to be used in combination with a feedback classification and a chatbot approach to auto-populate issue trackers with structured bug reports mined from user feedback.

1. **Extractingand Analyzing Context Information in User-Support Conversations on Twitter – arxiv (Daniel Martens, Walid Maalej) – 2019.**

This Paper proposes to support both users and developers in exchanging precise context information with the least possible effort. As a first step to this end, this paper discusses the automatic identification of context items in informal user-support conversions related to mobile apps. We introduce a simple unsupervised approach that uses predefined keyword lists, word vector representations, and text patterns to extract basic context items from tweets, including the platform, device, app version, and system version. The results allow to identify issues potentially actionable to developers or requiring further clarifications. Evaluated against a truth set of ~3,000 tweets, our approach achieved precisions from 81% to 99% and recalls from 86% to 98% for the different context item types.

2. **App store mining is not enough for app improvement – Springer (MaleknazNayebi, Henry Cho, Guenther Ruhe) - 2018.**

In this paper, we study how Twitter can provide complementary information to support mobile app development. By analyzing a total of 30,793 apps over a period of six weeks, we found strong correlations between the number of reviews and tweets for most apps. Moreover, through applying machine learning classifiers, topic modeling and subsequent crowd-sourcing,
They successfully mined 22.4% additional feature requests and 12.89% additional bug reports from Twitter. We also found that 52.1% of all feature requests and bug reports were discussed on both tweets and reviews.

3. **Are tweets useful in the bug fixing process? An empirical study on Firefox and Chrome - Springer (Mariam El Mezouar,FengZhang,Ying Zou) – 2018.**

The purpose of this research is to propose an approach to remove noisy tweets, and map the remaining tweets to bug reports. They conducted an empirical study to investigate the usefulness of Twitter in the bug fixing process. We choose two widely adopted browsers (i.e., Firefox and Chrome) that are also large and rapidly released software systems. We find that issue reports are not treated differently regardless whether users tweet about the issue or not, except that Firefox developers tend to label an issue as more severe if users tweet about it. The feedback from Firefox contributors confirms that the tweets are not currently leveraged in the bug fixing process, due to the challenges associated to discovering bugs through Twitter.

4. **What Makes a Good Bug Report? - IEEE (Sascha Just, Adrian Schroter, Nicolas Bettenburg) - 2017**

This study proposed a and conducted a survey among developers and users of APACHE, ECLIPSE, and MOZILLA to find out what makes a good bug report. The analysis of the 466 responses revealed an information mismatch between what developers need and what users supply. Most developers consider steps to reproduce, stack traces, and test cases as helpful, which are, at the same time, most difficult to provide for users. Such insight is helpful for designing new bug tracking tools that guide users at collecting and providing more helpful information. Our CUEZILLA prototype is such a tool and measures the quality of new bug reports; it also recommends which elements should be added to improve the quality. We trained CUEZILLA on a sample of 289 bug reports, rated by developers as part of the survey.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

III. METHODOLOGY

3.1 Architecture

The most important part of this system is to understand and reply to the users about the app and related queries. This may involve various different types of tweets. The below figure 3.1 describe about the flow of the system and provide better understanding of the system.

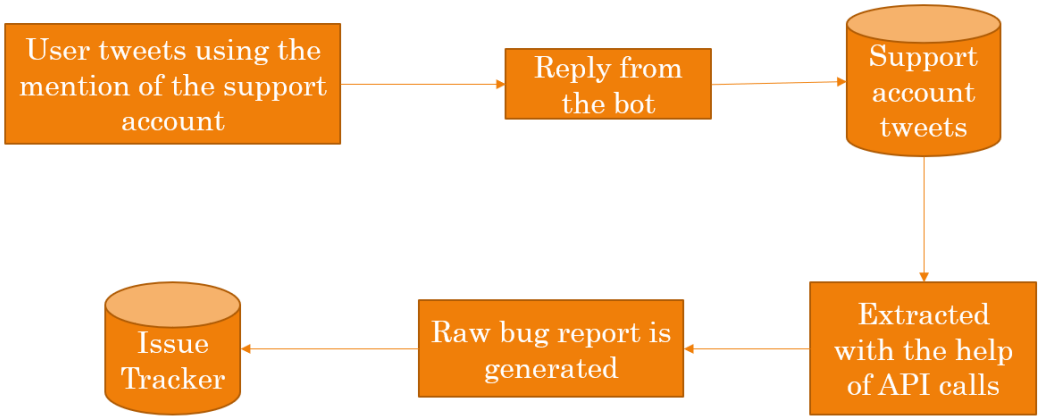

Fig. 3.1 System Architecture

The architecture diagram describes the flow of the data throughout the system. The messaging platform can be any social media platform. But, for this project we are concerned only with twitter platform. We are use twitter because on twitter people can only write limited number of words for a particular tweet. As a very first trail we feel that twitter would be a good platform for better understanding of the users taste and problems.

IV. IMPLEMENTATION

The most important part of this system is to understand and reply to the users about the app and related queries. This may involve various different types of tweets. The tweets will be classified based on the content or the tweet which is given by the user. The below figures describe about the flow of the system and provide better understanding of the system.

4.1 DATASET COLLECTION

4.1.1 Register Your App

In order to have access to Twitter data programmatically, we need to create an app that interacts with the Twitter API.

The first step is the registration of your app. In particular, you need to point your browser to http://apps.twitter.com, log-in to Twitter (if you're not already logged in) and register a new application. You can now choose a name and a description for your app .You will receive a consumer key and a consumer secret: these are application settings that should always be kept private. From the configuration page of your app, you can also require an access token and an access token secret. Similarly, to the consumer keys, these strings must also be kept private: they provide the application access to Twitter on behalf of your account. The default permissions are read-only, which is all we need in our case, but if you decide to change your permission to provide writing features in your app, you must negotiate a new access token. there are rate limits in the use of the Twitter API, see Fig 4.1,

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

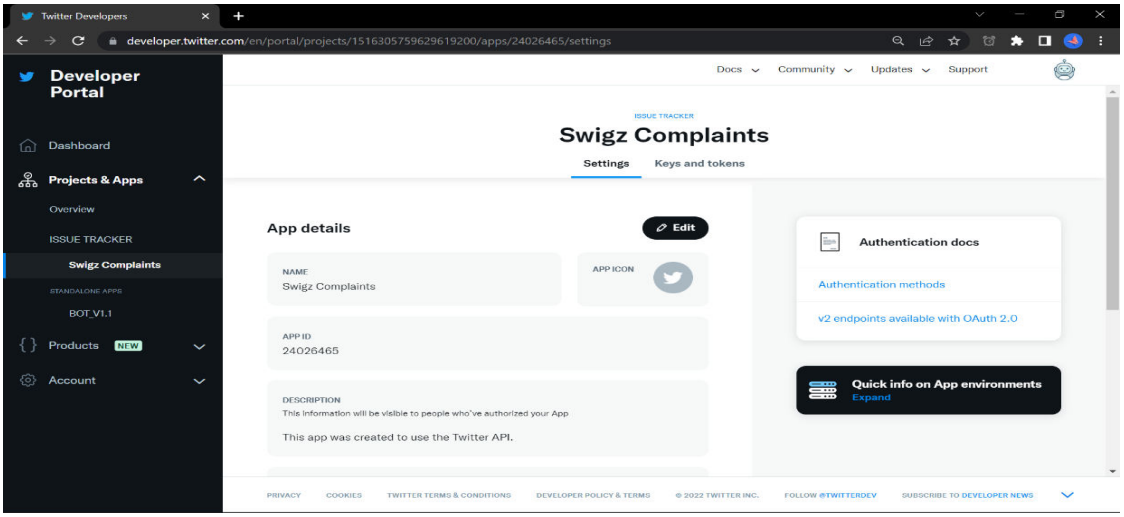

Fig.4.1 Twitter Developer Profile

4.1.2 Accessing the Data

Twitter provides REST APIs you can use to interact with their service. There is also a bunch of Python-based clients out there that we can use without re-inventing the wheel. In particular, Tweepy in one of the most interesting and straightforward to use, so let's install it:

Pip install tweepy

In order to authorise our app to access Twitter on our behalf, we need to use the OAuth interface :

```
import tweepy
from tweepy import OAuthHandler
consumer_key = 'YOUR-CONSUMER-KEY'
consumer_secret = 'YOUR-CONSUMER-SECRET'
access_token = 'YOUR-ACCESS-TOKEN'
access_secret = 'YOUR-ACCESS-SECRET'
access_secret = 'YOUR-ACCESS-SECRET'
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)
```

The api variable is now our entry point for most of the operations we can perform with Twitter.For example, we can read our own timeline (i.e. our Twitter homepage) with:

Tweepy provides the convenient Cursor interface to iterate through different types of objects. In the example above we're using 10 to limit the number of tweets we're reading, but we can of course access more. The status variable is an instance of the Status() class, a nice wrapper to access the data. The JSON response from the Twitter API is available in the attribute _json (with a leading underscore), which is not the raw JSON string, but a dictionary.

4.1.3 Streaming

In case we want to "keep the connection open", and gather all the upcoming tweets about a particular event, the streaming API is what we need. We need to extend the api.search() to customise the way we process the incoming data. A working example that gathers all the new tweets with the #python hashtag:

```
from tweepy import api
keywords = '#swigzissue'
```

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

```
limit = 300
#tweets = api.search_tweets(q=keywords)
tweets = tweepy.Cursor(api.search_tweets, q=keywords,
                       count=200, tweet_mode='extended').items(limit)
# create dataframe
columns = ['Time', 'User', 'Tweet']
data = []
for tweet in tweets:
data.append([tweet.created_at, tweet.user.screen_name, tweet.full_text])

df = pd.DataFrame(data, columns=columns)
print(df)
df.to_json('tweets.json')
```

Depending on the search term, we can gather tons of tweets within a few minutes. This is especially true for live events with a world-wide coverage (World Cups, Super Bowls, Academy Awards, you name it), so keep an eye on the JSON file to understand how fast it grows and consider how many tweets you might need for your tests.

4.2 ANATOMY OF THE TWEET

It is very simple and easy step to proceed. In this phase, we generally understand the structure of the tweet, which is nothing but it tells us what the tweet is about. In our dataset we will be having different tweets addressing different issues of users.

All the tweets may not be the problem or an issue mentioned by the user. It may be about different issue. Our main concentration in this phase is to identify the tweets which are about the problems or issues facing by the user and classifying the tweets which we require for our process.

Figure 4.2 Display of a tweet

The key attributes are the following:

• text: the text of the tweet itself
• created_at: the date of creation
• favorite_count, retweet_count: the number of favourites and retweets
• favorited, retweeted: boolean stating whether the authenticated user (you) have favourited or retweeted this tweet
• lang: acronym for the language (e.g. “en” for english)
• id: the tweet identifier
• place, coordinates, geo: geo-location information if available
• user: the author’s full profile

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

• entities: list of entities like URLs, @-mentions, hashtags and symbols
• in_reply_to_user_id: user identifier if the tweet is a reply to a specific user
• in_reply_to_status_id: status identifier id the tweet is a reply to a specific status

As you can see there's a lot of information we can play with. All the *_id fields also have a *_id_str counterpart, where the same information is stored as a string rather than a big int (to avoid overflow problems). We can imagine how these data already allow for some interesting analysis: we can check who is most favourited/retweeted, who's discussing with who, what are the most popular hashtags and so on. Most of the goodness we're looking for, i.e. the content of a tweet, is anyway embedded in the text, and that's where we're starting our analysis.This has a vital role to play in the building of the bug report which is used to auto-populate the issue trackers.

4.3 BOT CREATION

When the user raises a problem or an issue facing with the respective app, he generally addresses the issue in the twitter. So, when this happens there should be some process to reply about the problem to the user from the twitter so that the user knows that there will be some action taken regarding the problem. To make this happen we have to create a twitter bot which replies to the tweet in which the tweet is mentioned. The main libraries which are used for this bot are twit and chatterbot which are used to build the bot. With the help of the intents in json format and Lancaster stemmer module in the NLTK library they will try to understand the data. Fig 4.3 shows a official bot account created for a client.

Figure 4.3 : Bot Account

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

4.4 FINAL GENERATION OF THE REPORT

This is phase we will be extracting the tweets for the twitter with the help of the API from the respective support account. These tweets will then be classified into different categories like rating, feature recommendation and bug reports. The tweets which are extracted will be considered as the raw bug report. With the help of the spacy library we will tokenize the user feedback and retrieve the context items from the text and build the final report with context items and the text from the tweets to the issue tracker.
The report can also have various other representation of the tweets in the form of the graph. For example, many users are finding it difficult to understand and facing issues with one particular update. With the help of graph representation, developers can understand how many numbers of people or users having the same issue and what is the spike of the tweets in the support account and many other different aspects .

4.5 REAL-TIME ISSUE TRACKER

In this phase, a web-application using python and flask has been developed and using Twitter API v2, we can use 'client class functions to reply to a tweet .
The 'Reply' section of the issue tracker remains connected with the conversation on Twitter, so that developers can directly communicate with the reporting user to ask for further clarification or inform the user once the issue is fixed. Developer can directly 'view' tweet.see Fig 4.4,below

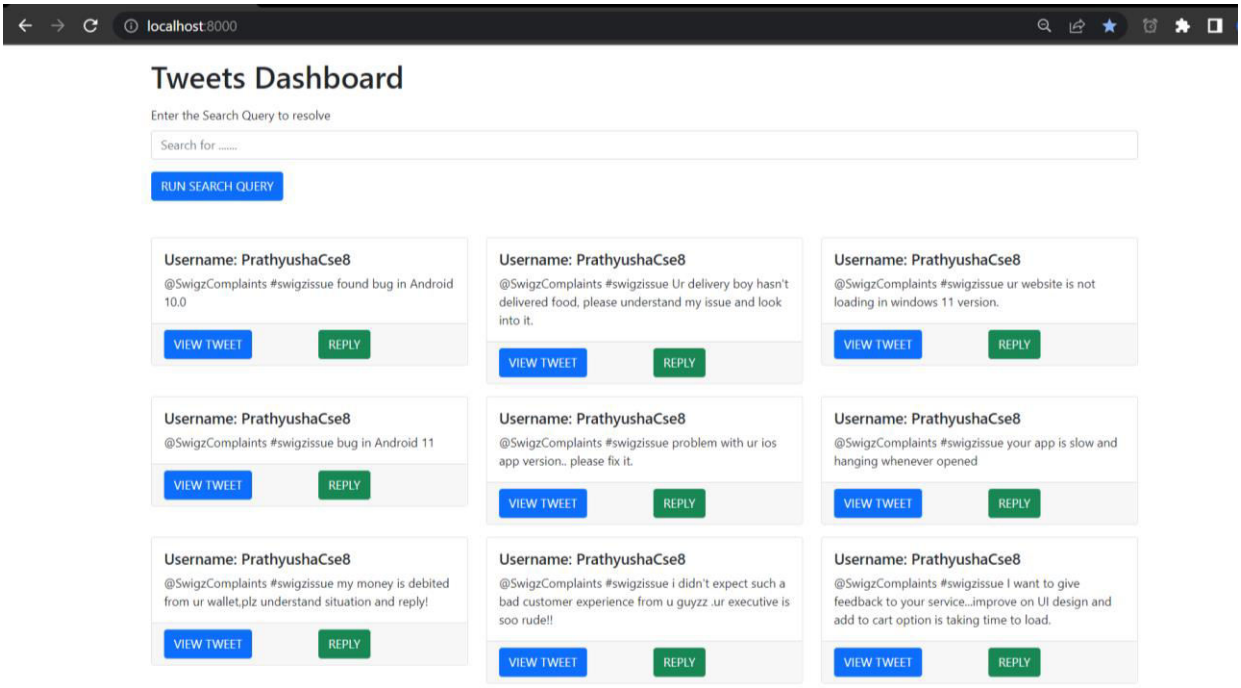

Figure 4.4. Dashboard of Tweets in Issue-Tracker

V. RESULTS

- **Customer Support Chat-bot :**

The final output is the interactive bot on twitter. If anyone wants to access it then the respective person must tweet by using of the tag @SwigzComplaints and #swigzissue.

The entire code is pushed to a cloud – Python anywhere which helps us to keep the bot always running. The major reason of pushing the code to the cloud is that the even when the code crashes there is a minimal chance of bot not

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

running in the background.

Figure 5.1 and 5.2 : @SwigzComplaintsis the chat-bot replying to the user

- **Final Bug/Issue Report (JSON):**

Below Fig.5.3 displays, Bug/Issue Report generated from user's tweets in JSON Format. This Report Contains Timestamp, User Id, Tweet by Customer.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106224 |

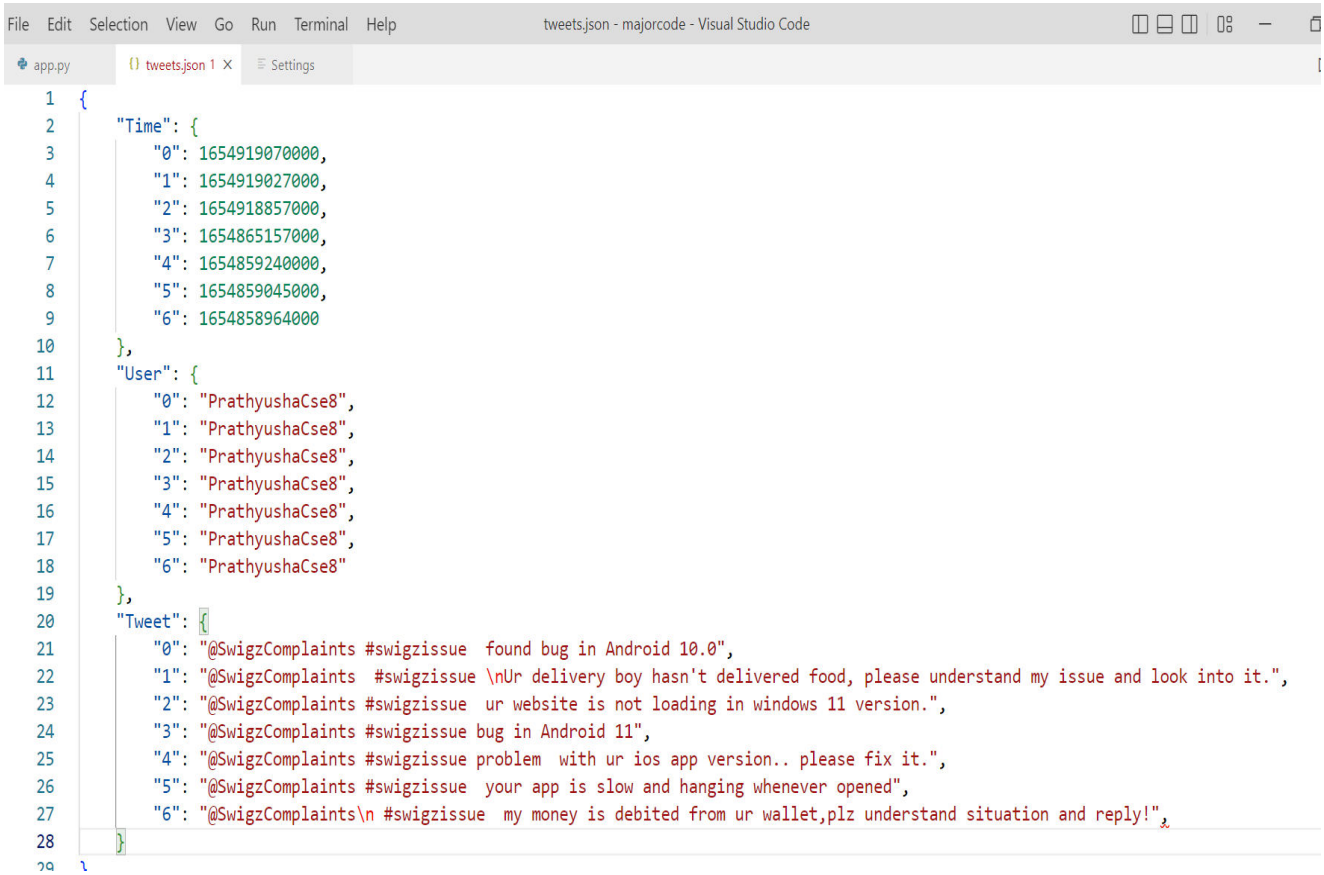

Figure 5.3: Raw Bug/Issue JSON Report Generated.

REFERENCES

[1] "Extracting and Analyzing Context Information in User-Support Conversations on Twitter", Daniel Martens, Walid Maalej ,2019

[2] M. Nayebi, H. Cho, and G. Ruhe, "App store mining is not enough for app improvement," Empirical Software Engineering, vol. 23, 02 2018.

[3] M. E. Mezouar, F. Zhang, and Y. Zou, "Are tweets useful in the bug fixing process? an empirical study on firefox and chrome," Empirical Softw. Engg., vol. 23, no. 3, pp. 1704–1742, Jun. 2018.

[4] N. Bettenburg, S. Just, A. Schrter, C. Weiss, R. Premraj, and T. Zim-mermann, "What makes a good bug report?" in Proceedings of the 16th International Symposium on Foundations of Software Engineering, November 2008.

[5] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Extracting structural information from bug reports," in Proceedings of the Fifth International Working Conference on Mining Software Repositories, May 2008.

[6] Twitter, Twitter Search API, April 2019,https://twitter.com/i/search/timeline.

[7] Netflix, Official Netflix Support Account on Twitter, April 2019, https://twitter.com/netflixhelps.

[8] Snapchat, Official Snapchat Support Account on Twitter, April 2019,https://twitter.com/snapchatsupport.

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 8.118

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

Scan to save the contact details