Product Requirements Document

TuTweetBiBot – Latest Tweet of your Best Streaming Website

Product Requirements Document

- 1. Objective
- 2. Release
- 3. Features
- 4. Implementation
- 5. Pre-requiste Instructions, Working and Test
- 6. Analytics
- 7. Future work

Key Stakeholders

Document Status	Draft
Engineering	Rahul Khandalkar
Design	Rahul K, Ryan H, Steven T
Product	Rahul K, Ryan H
Marketing / TME	Rahul K
Supportability / Telemetry	Rahul K

1. Objective: [Mock-up objective/Assumption]

Vision	Retrieve the 10 most recent tweets of Twitter account to show on Slack channel
Goals	Timeframe: 1 week – MVP
	Primary goal is simplifying the operations and management function that a website administrator has to perform for API integration of Twitter tool.
	Secondary goal of this API is for customer to adopt python framework solution and usage of library for Twitter API in an existing environment.
Initiatives	Incorporated organization stylings and wireframes
Persona(s)	IT Systems Engineer, Technical Marketing Engineer, Sales

2. Release

Release	tutweetbibot-0.0.1
Date	May 26, 2021
Initiative	Tu Tweet Bi Bot 1
Milestones	Minimum Valuable Product (MVP) Scalable solution
Features	 Live streaming of latest 10 tweet from designated account Deployable solution as web application and API interface
Dependencies	 Python 3.10 Libraries: dotenv, schedule, time, os, tweepy, slack_sdk Supports all OS. Preferred Windows.

3. Features

1. [Mock-up Story/Assumption]

Feature	Live streaming of latest 10 tweet from designated account	
Description	The feature will enable the IT Systems engineer and Tubi marketeer to provide product feature on tablet to show latest info of Tu Tweet Bi Bot. The live stream data will be updated on tablet from any location to show API data.	
Purpose	One-click application to show live stream of tweets	
User problem	The challenge to open browser for purpose of this feature is tedious	
User value	One-click application provide easy access and better satisfaction	
Assumptions	The end user have knowledge on Twitter knowledge, internet usage and Slack will provide required output	
Not doing	None at the moment	
Acceptance criteria	OS update version above x.x	

2.

Feature	Deployable solution as web application and API interface	
Description	The solution can be deployed as client-server architecture. The code works as deployable solution on server. The python framework capability to provide API interface can provide further development of MVP for frontend Angular.js to interface with middle layer Java application and H2 database	
Purpose	Scalable and readable solution of tutweetbibot-0.0.1	
User problem	Existing solution did not follow best practices of SLDC. The developer left the company and provided no documentation to cause frequent hotfixes. Thus, user did not receive reliable update on their earlier version of TuTweetBiBot .	
User value	A properly followed SLDC best practice will provide product reliability	
Assumptions	The end user has cleanly uninstalled the previous version of the application. The newly installed version is from TuTweetBiBot official repository.	
Not doing	The web application is not developed to have Angular.js framework. The MVP is limited to running application 24 times in a day because free version of Twitter and Slack API allows limited request per day.	
Acceptance criteria	Slack show the latest 10 tweets from designated Twitter account every hour Store count, start, running and end of each iteration.	

4. Implementation

Limitation:

- 1. MVP is built single-file/page application.
- 2. It does not follow Model, View, Controller framework.
- 3. All secrets will be stored in secret store.
- 4. Token needs to be refreshed every 12 hours manually.
- Service can be run daemon in a linux web server or as needed for future version
- 6. Logs and observability tools implementation in future version

The application is broken in four layers:

- Initialization and Authentication: Load environmental variable(stored insecurely). Authenticate the consumer key and secret, access token and secret from Twitter. OAuth authorization for Twitter and Slack, preparation for API calls using rate limit, tweepy and slack_sdk library.
- 8. Job Task: Retrieve 10 latest tweets from Twitter and display at Slack channel #general. Two function Job() and JobAnnouce() assist to perform this task. Job() does main task, JobAnnounce() is a helper function to keep count of iteration of calls. The code follows try, catch and finally to improve error handling.
- 9. Schedule handling: Task ensures to run once in a hour job() task
- 10. Model Slack has basic interface to show the incoming Tweets in every hour

Error handling:

Incorrect Twitter username showed correct handling of the 404 Not Found

```
def job():
     client.chat_postMessage(channel='#general', text="######")
     client.chat_postMessage(channel='#general', text="Hourly Update from Twitter")
     iobAnnounce()
     client.chat_postMessage(channel='#general', text="######")
     print("Code Starting")
         tweets = tweepy.Cursor(api.user_timeline, screen_name='livelyrahu', count=10).items()
     except tweepy.error.RateLimitError as e:
         print(e)
         print("Code Run " + str(count) + " Completed")
🌳 twitslackapi 🔀
  Code Running Count : 1
  Code Starting
  Code Run 2 Completed
  404 Not Found
  34 - Sorry, that page does not exist.
  Code Exiting
  Process finished with exit code 0
```

Rate Limit:

api = tweepy.API(auth, wait_on_rate_limit=True)

The wait_on_rate_limit parameter is used when creating an instance of the API class from the tweepy library. This parameter is a boolean that determines if the API client should wait when it hits the rate limit. If set to True, the client will wait until the rate limit resets before making further requests. If set to False, the client will raise a RateLimitError exception when the rate limit is hit. So here we pass True to wait_on_rate_limit which means the client will wait until the rate limit resets before making further requests.

tweets = tweepy.Cursor(api.user_timeline, screen_name='livelyrahul', count=10).items()

The tweepy. Cursor object is used to paginate through the results of a query made to the Twitter API. The ".items()" method is used to retrieve a specified number of tweets from the user's timeline. In this case, it is retrieving the most recent 10 tweets from the user with the screen name "livelyrahul". The api.user timeline method is used to retrieve tweets from a specific user's timeline.

11. Pre-requisite, Working and Test

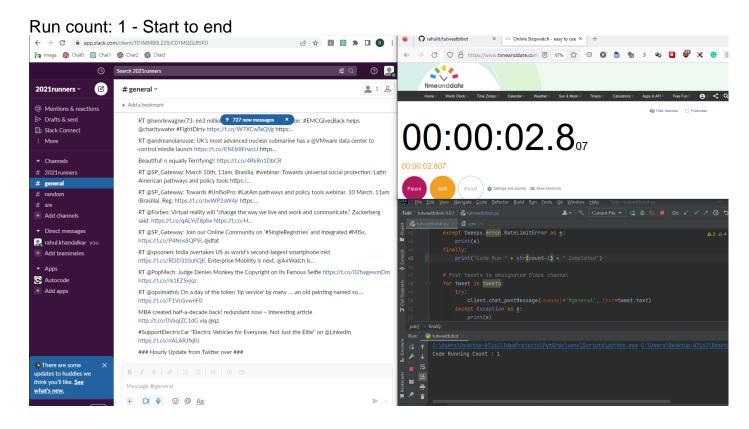
1. Install library using pip

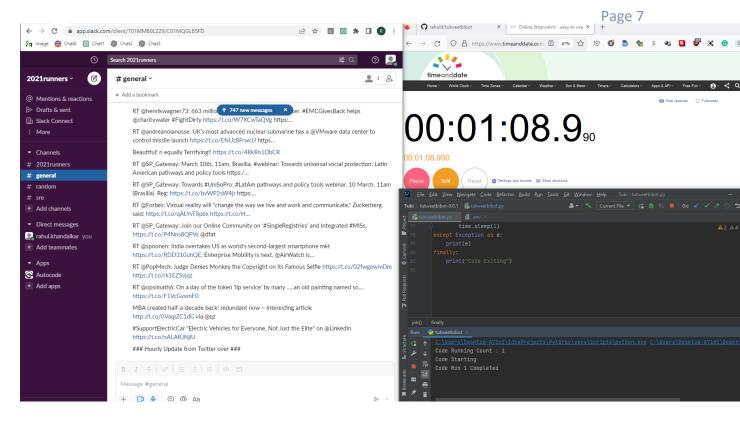
 Latest python version can be downloaded from https://www.python.org/downloads/

For maintenance:

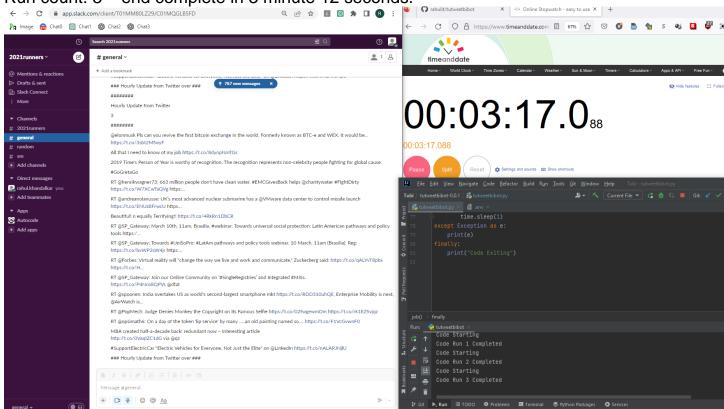
3. Refresh token https://api.slack.com/apps

Note: For test purpose, retrieve last 10 posts (at twitter) in 1 minute instead of usecase 1 hour is chosen to check the ratelimit and other performance

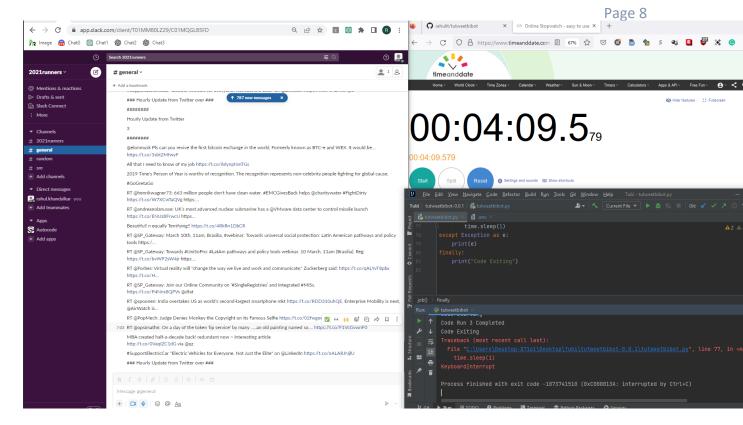




Run count: 3 – end complete in 3 minute 12 seconds.



KeyboardInterrupt to stop the program show proper error message



6. Analytics

Hypothesis: We believe web application will achieve to provide output in 1 minute 5 seconds for version tutweetbibot-0.0.1 and not limited to amount of TuTweetBiBot API Call

Note: For test purpose, retrieve last 10 posts (at twitter) in 1 minute instead of usecase 1 hour is chosen to check the ratelimit and other performance

Key performance indicator	Baseline	Target	Timeframe
Retrieval, and display Slack	1 minutes 10 seconds	1 minute 5 seconds	-

7. Future work

Future features	Purpose	Priority	Timeframe
Separate API call interface for each retrieval, provide better MVC model and library extension	Better management for future implementation to separate functionality	P3	1 week
Frontend angular.js solution	End-user comfort, lesser dependency on CLI functionality, one click interface	P1	3 weeks