WhatsApp based Chatbot using Selenium and Python

By

Saurabh Ashok Karalkar

ABSTRACT

As the digital age progresses, interacting with each other becomes easier. With so many interactions and texting going on, many messages and data often becomes lost in transactions. Currently most of the personal as well as professional interactions happen on the platform WhatsApp. WhatsApp was launched in November of 2009 has seen exponential growth in downloads and active users per month. Seeing its popularity for professional as well personal use I have developed a personal assistant that will store information and deliver the same upon specific request. The app developed functions exactly like a personal assistant which stores important information, reminders, meeting agendas, and it can get screenshot of a webpage and store them so that we can easily remember the cited sources.

Literature review

There have been numerous works in the field of chatbots and userassistance tools. Some recent examples of such works are Amazon Alexa (https://en.wikipedia.org/wiki/Amazon_Alexa), Echo (https://en.wikipedia.org/wiki/Amazon_Echo), Siri (https://en.wikipedia.org/wiki/Siri), etc. While working on this project I have drawn inspiration from such of these tools which have proven to be very helpful to the users.

Modules used

Selenium

Selenium is a portable framework for testing web applications. Selenium provides a playback tool for authoring functional tests without the need to learn a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala. The tests can then run against most modern web browsers. Selenium runs on Windows, Linux, and macOS. It is open-source software released under the Apache License 2.0.

With the help of selenium we can interact with the WhatsApp clickable components and do the necessary interactions like saving some text into the MySQL database and retrieve the same.

With this tool we can scrape messages for commands that are being made to the personal assistant and the use the Python backend to store what data the user was specifying to be stored and in what fashion (Notes, Birthdates, Meeting agendas). Then the python backend would store the data in the my SQL Server. Which would use the same backend to retrieve the user stored data upon a specific request.

MySQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation).In 2010, when Oracle acquired Sun, Widenius forked the opensource MySQL project to create MariaDB.

MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress. MySQL is also used by many popular websites, including Facebook, Flickr, MediaWiki,Twitter,and YouTube.

Proposed System

The system consists of 1 interface, which the main user needs to access only once. Using Selenium, we open a WhatsApp Web window on the main user’s desktop. The user then has to login to their own WhatsApp using this window. Once logged in, the bot will monitor the groups that are pre defined in the code, for new messages. On receiving a new message, the bot will analyze the message, check if it belongs to any of the syntax described in the bot’s “!help” function, and then respond accordingly, depending on the analysis.

Code used

main.py

```
import time
import re
import os
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as
EC from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.chrome.options import Options
#import redditScrapper as red
import databaseHelper as db
import datetime as dt
import traceback

target = '"HCI Project Testing"'  # Group/person's name
x_arg = '//span[contains(@title,' + target + ')]'
std_pat = re.compile("![A-Za-z]+.*$")  # For finding out if message is a
command
command_pat = re.compile("![A-Za-z]+")  # For isolating the command
title_pat = re.compile("\".+\"")
date_pat = re.compile("[0-9]{1,2}-[0-9]{1,2}-[0-9]{4}")
date_sans_year_pat = re.compile("[0-9]{1,2}-[0-9]{1,2}")

help_dict = {
    #Notes
    "!addNote": "\"title\" note_content",
    "!getNote": "\"title\"",
    "!getAllNotes": "",
```

```
        "!deleteNote": "\"title\"",

        #Birthday
        "!addBirthday": "\"name\" dd-mm-yyyy",
        "!getBirthday": "\"name\"",
        "!getAllBirthdays": "",
        "!deleteBirthday": "\"name\"",

        #Reminders
        "!reminder": "\"title\" reminder_content",
        "!getAllReminders": "",
        "!deleteReminder": "\"title\"",

        #Misc
        "!help": "",
        "!screenshot": "link"
    }

    def screenshot(file_name, url):     chrome_options = Options()
    chrome_options.add_argument('--headless')
    chrome_options.add_argument('--start-maximized')
    temp_driver = webdriver.Chrome(options=chrome_options)
    temp_driver.get(url)     time.sleep(2)
        # original_size = temp_driver.get_window_size()
    required_width = temp_driver.execute_script('return
    document.body.parentNode.scrollWidth')
        required_height = temp_driver.execute_script('return
    document.body.parentNode.scrollHeight')
    temp_driver.set_window_size(required_width, required_height)     #
driver.save_screenshot(path)  # has scrollbar
    temp_driver.find_element_by_tag_name('body').screenshot(file_name)
# avoids scrollbar
```

```
    temp_driver.quit()

def sendMessage(msg):
    finalMsg = msg

    message =
driver.find_elements_by_xpath('//*[@id="main"]/footer/div[1]/div[2]/div/div[2]')[0]
    message.send_keys(finalMsg)
    sendbutton =
driver.find_elements_by_xpath('//*[@id="main"]/footer/div[1]/div[3]/button')[0]
    sendbutton.click()

def sendPhoto(file_name):
    button =
driver.find_element_by_xpath('//*[@id="main"]/header/div[3]/div/div[2]/d iv/span')     button.click()     image =
driver.find_element_by_xpath('//*[@id="main"]/header/div[3]/div/div[2]/span/div/div/ul/li[1]/button/input')
    image.send_keys(os.path.join(source_dir, file_name))
time.sleep(2)

    sendButton = driver.find_element_by_xpath('//*[ @ id =
"app"]/div/div/div[2]/div[2]/span/div/span/div/div/div[2]/span/div')  #
'/html/body/div[1]/div/div/div[2]/div[2]/span/div/span/div/div/div[2]/span[2
]/div/div')
    sendButton.click()



source_dir = os.path.dirname(os.path.realpath(__file__))
driver = webdriver.Chrome(os.path.join(source_dir, "chromedriver.exe"))
```

```
driver.get("https://web.whatsapp.com/")
wait = WebDriverWait(driver, 600)
print("Connected to WhatsApp")

group_title = wait.until(EC.presence_of_element_located((
By.XPATH, x_arg)))
print("Found group")

time.sleep(2) #Compensating for delay in fetching messages

group_title.click()
print("Group selected")
print("Chatbot started")
prev_msg = ""

while True:

   #elements = driver.find_elements_by_class_name('_1ays2')[-
1].find_elements_by_class_name("FTBzM")
   elements = driver.find_elements_by_class_name('selectable-text')[-2]
print(elements)

try:
      #data = elements[-
1].find_elements_by_class_name('selectabletext')[0].text
      data = elements.text
      if data == "!close":  # Exit condition
break

      if data != prev_msg or data == "!help":
         print(data)          if
re.match(std_pat, data):
            command =
command_pat.findall(data)[0]              if
```

```
command in help_dict.keys():                    if
command == "!addNote":
                    title = title_pat.findall(data)[0].replace("\"", "")
                    content = data.replace(command + " \"" + title + "\" ", "", 1)
print(title, content)
                    db.addRow("notes", [title, content])
sendMessage("Note added : " + title)

                elif command == "!getAllNotes":
                    result = db.searchTable("notes", ["title", ])
for res in result:
                        sendMessage(res[0])

                elif command == "!getNote":
                    title = title_pat.findall(data)[0].replace("\"", "")
                    result = db.searchTable("notes", ["title", "content"], "title",
title)
                    sendMessage("*" + result[0][0] + "*")
sendMessage(result[0][1])

                elif command == "!deleteNote":
                    title = title_pat.findall(data)[0].replace("\"", "")
db.deleteRow("notes", "title", title)
sendMessage("Note " + title + " deleted")

                elif command == "!addBirthday":
                    name = title_pat.findall(data)[0].replace("\"", "")
                    date = dt.datetime.strptime(date_pat.findall(data)[0], "%d-
%m-%Y").date().strftime("%Y-%m-%d")
                    db.addRow("birthdays", [name, date])
                    sendMessage("Birthdate added : " + name + " " + date)

                elif command == "!getAllBirthdays":
```

```
                result = db.searchTable("birthdays", ["name", "birth_date"])
for res in result:
                    sendMessage("Name : " + res[0] + " -> Date : " + res[1])

            elif command =="!getBirthday":
                name = title_pat.findall(data)[0].replace("\"", "")
                result = db.searchTable("birthdays", ["name", "birth_date"],
"name", name)
                sendMessage("*" + result[0][0] + ":*" + result[0][1])

            elif command == "!deleteBirthday":
                name = title_pat.findall(data)[0].replace("\"", "")
db.deleteRow("birthdays", "name", name)
sendMessage("Birthday " + name + " deleted")

            elif command == "!screenshot":
                link = data.replace(command + " ", "")
                print("Link acquired")
file_name = "screenshot.jpeg"
screenshot(file_name, link)
print("Screenshot taken")
sendPhoto(file_name)
print("Screenshot uploaded")

            elif command == "!reminder":
                name = title_pat.findall(data)[0].replace("\"", "")
                date =
dt.datetime.strptime(date_sans_year_pat.findall(data)[0], "%d-
%m").date().strftime("%m-%d")
                db.addRow("reminder", [name, date])
                sendMessage("Reminder added : " + name + " " + date)

            elif command == "!getAllReminders":
                result = db.searchTable("reminders", ["title",
```

```
"remind_date"])
                    for res in result:
                        sendMessage("Title : " + res[0] + " -> Date : " + res[1])

                elif command == "!deleteReminder":
                    title = title_pat.findall(data)[0].replace("\"", "")
db.deleteRow("reminder", "title", title)
sendMessage("Reminder " + title + " deleted")

                elif command == "!help":
for key, value in help_dict.items():
sendMessage(key + " " + value)

                    prev_msg = data
                else:                    sendMessage("You seem to have entered a wrong command. Please use !help to get more help on the correct command formats")

    except Exception as e:
        print(traceback.format_exc())
print("Last message deleted")

    time.sleep(2) #To reduce request spam


driver.close()
```

database.py

```
import sqlite3 from
sqlite3 import Error
import os

source_directory = os.path.dirname(os.path.realpath(__file__))
database = os.path.join(source_directory, "Databases", "database.db")
print(database)

def appendTuple(tup, append_value):
    '''Helper function for dealing with
tuples'''     tup = list(tup)
tup.append(append_value)
    tup = tuple(tup)

    return tup



def createConnection(db_file):
try:
        conn = sqlite3.connect(db_file)
        return conn
except Error as e:
        print(e)
    return None



def createTable(table_name, cols):
    base_str = "create table if not exists " + table_name + "("
for col in cols:
        base_str += col + ", "

    base_str = base_str[:len(base_str) - 2] + ");"  # Done to remove the last
comma
```

```
try:
        conn = createConnection(database)
c = conn.cursor()
        c.execute(base_str)
except Error as e:
        print(e)



def updateTable(conn, table_name, where_col, where_value, update_col,
update_value):
    base_str = "update ? set ? = ? where ? = ?"
    params = (table_name, update_col, update_value, where_col,
where_value)

try:
        conn = createConnection(conn)
        cursor = conn.cursor()
cursor.execute(base_str, params)
conn.commit()     except Error as e:



def deleteRow(table_name, where_col, where_value):
    base_str = "delete from " + table_name + " where " + where_col + " = ?"
params = (where_value, )

try:
        conn = createConnection(database)
cursor = conn.cursor()
cursor.execute(base_str, params)
conn.commit()     except Error as e:
        print(e)
```

```
print(e)

        def clearTable(table_name):
        base_str = "delete from ?"
           params = (table_name)

        try:
              conn = createConnection(database)
        cursor = conn.cursor()
              cursor.execute(base_str, params)
        conn.commit()     except Error as e:
              print(e)



        def addRow(table_name, values):
           base_str = "insert into " + table_name + " values(NULL, '"

           for value in values:
              base_str += value + "','"

           base_str = base_str[:len(base_str) - 3] + "');"
        print(base_str)

        try:
              conn = createConnection(database)
              cursor = conn.cursor()
        cursor.execute(base_str)
        conn.commit()     except
        Error as e:         print(e)

def searchTable(table_name, ret_cols = None, where_col = None,
where_value = None):     base_str = "select "
```

```
    if ret_cols is not None:
        returnCols = ""         for
col in ret_cols:
returnCols += col + ","
print(returnCols)
        returnCols = returnCols[:len(returnCols) - 1]
base_str += returnCols + " from " + table_name     else:
        base_str += "* from" + table_name
    '''
    if where_col is not None:
        params = appendTuple(params, where_col)
        params = appendTuple(params, where_value)
    '''
    if where_col is not None:
        base_str += " where " + where_col + " = \"" + where_value + "\";"
try:
        print(base_str)
        conn = createConnection(database)
        cursor = conn.cursor()
cursor.execute(base_str)
        selection = cursor.fetchall()


        return selection
except Error as e:
        print(e)


    return None



if __name__ == "__main__":
```

```
    createTable("notes", ["id integer primary key autoincrement", "title text NOT NULL", "content text NOT NULL"])
    createTable("birthdays", ["id integer primary key autoincrement", "name text NOT NULL", "birth_date date"])
    createTable("reminder", ["id integer primary key autoincrement", "title text NOT NULL", "remind_date date"])
```

Results and Discussion

I was able to successfully create a utility based personal assistant using WhatsApp and Selenium. The following set of commands were used to operate the utility-based tool:

!addNote "title" note_content

!getNote "title"

!getAllNotes

!deleteNote "title"

!addBirthday "name" dd-mm-yyyy

!getBirthDay "name"

!getAllBirthdays

!deleteBirthday "name"

!reminder "title" reminder_content

!getAllReminders

!deleteReminder "title"

!help

!screenshot link

Above are the commands which when used in the chat with the above mentioned syntax can be used to avail the services offered by the chatbot. The syntax is very simple, comprising of basic English Language commands preceded with an exclamation mark. Following is the explanation of each of the above mentioned command:

!addNote "title" note_content

This command is used to add a note in the database of the tool to be invoked later by the user when needed. The command uses a title to store the note followed by the content that is to be stored in the node.

!getNote "title"

This command is used to get the content of a note stored in the database. It is a very user-friendly command that uses the keyword getNote

preceded with an exclamation mark and followed by the title of the note to be extracted. The tool uses the title to get the content of the node stored in the SQL database.

!getAllNotes

This command returns the title of all the nodes stored in the database. A user can easily use this command to obtain the names of all the notes, if the user forgets any and thus, it prevents any strain on the user's memory.

!deleteNote "title"

This is also a very easy to grasp command which is utilized to delete a note stored in the tool's database.

!addBirthday "name" dd-mm-yyyy

This command is used to enter the birthday of an individual to the database. The data is to be entered in the specific database and then it can be accessed anytime via the name of the individual.

!getBirthDay "name"

This command is used to get the birthday of an individual using his/her name from the SQL database.

!getAllBirthdays

This command is used to get the names of all the individuals whose birthdays are stored in the database. This command again is very helpful to let the user recall what data is present in the database.

!deleteBirthday "name"

This command is used to delete the birthday information of an individual already stored in the database. It takes the name of the individual as an input as that is the primary key for this data in the database.

!reminder "title" reminder_content

This command is used to set a reminder in the database. This command sets a reminder using the title to distinguish between the various other reminders stored in the database.

!getAllReminders

This command returns the titles of all the reminders stored in the database. This is a very user-friendly command and can be very easily invoked.

!deleteReminder "title"

This command is used to delete a reminder stored in the database using the title of the reminder as an input by the user.

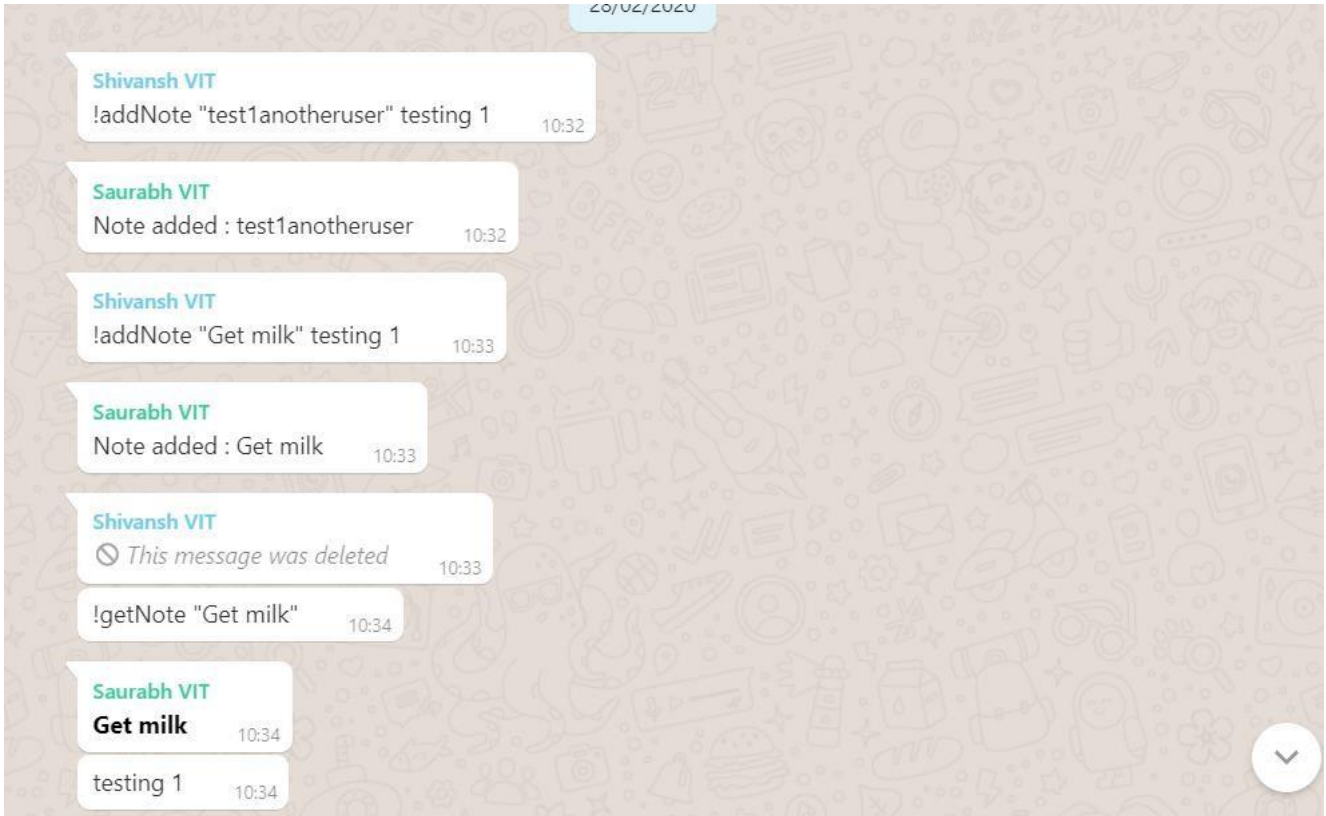
!help

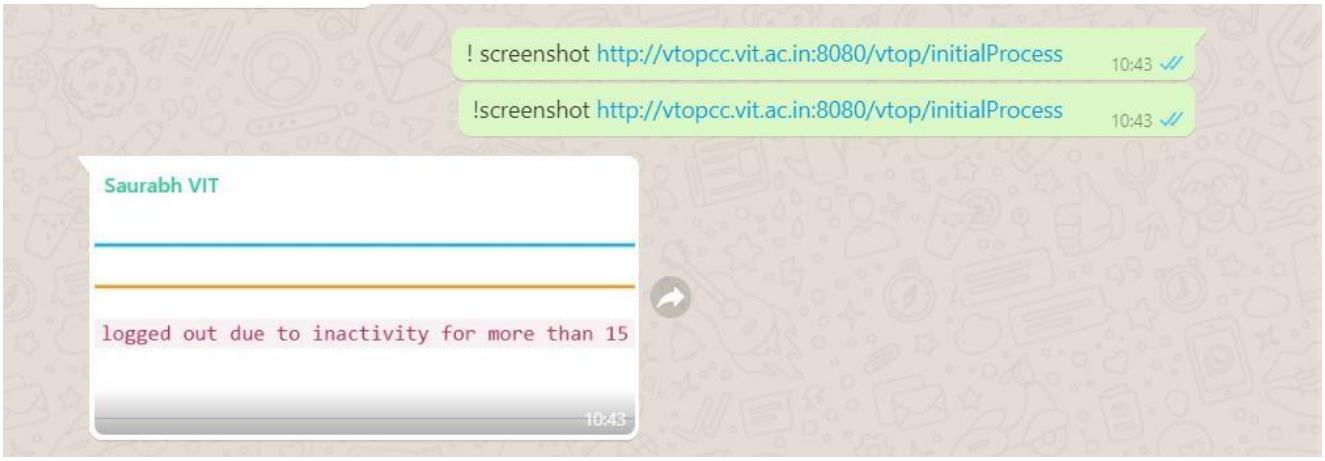
This is the command used for error and exception handling. This command can be invoked by the user to get the list of all the commands required to invoke all the facilities provided by the utility based tool, along with their proper syntax. The rudimentary syntax allows the user to use an exclamation mark to notify the tool that the data entered by the user is actually a command. So if a user enters gibberish or erroneous data preceded by an exclamation mark, then the system itself notifies the user to use the !help command to get the correct syntax of commands used.

!screenshot link

This command is used to get the Screenshot of any link provided. This command takes a URL as an input and then uses Selenium to scrape data from that URL and present it as a screenshot. This allows the user to go through the data from a website without the actual need to visit the website.

Output Screenshots




! screenshot http://vtopcc.vit.ac.in:8080/vtop/initialProcess 10:43
!screenshot http://vtopcc.vit.ac.in:8080/vtop/initialProcess 10:43
Saurabh VIT
logged out due to inactivity for more than 15
10:43


!help 14:09
Saurabh VIT
!addNote "title" note_content 14:09
!getNote "title" 14:09
!getAllNotes 14:09
!deleteNote "title" 14:09
!addBirthday "name" dd-mm-yyyy 14:09
!getBirthDay "name"
!getAllBirthdays 14:09
!deleteBirthday "name" 14:09
!reminder "title" reminder_content 14:09
!getAllReminders 14:09
!deleteReminder "title" 14:09
!help 14:09
!screenshot link 14:09

Summary

I was successfully able to create a utility based tool, using WhatsApp and Selenium. This tool has proved to be a very user-friendly tool, which very interactive and usable, even for the most naïve of users. This tool can be very helpful in storing data on the go, simply using WhatsApp as a medium for storing information. This tool can free the user from the unnecessary hassle to manage various different apps to store data likewise a calendar application to store birthdays, a notepad application to store notes, and a clock application to store reminders. Using this utility based tool, we can store all this data pretty easily using one application and we can access it pretty easily too. There is no need to scroll down a vast list of notes to get the desired data as we can just use one command to obtain the required result. Therefore, this is a very useful application which makes the life of any user pretty easy and it is very easy to operate as well.