//DBFA Billing Framework

Python Project Report

TOPIC: Billing system

Pranav Balaji

XII – A

A C K N O W L E D G M E N T:

This is to certify that Pranav Balaji of Apeejay School, NOIDA (Class: 12-A) has created a project on the topic "Billing system", called the "DBFA Billing Framework".

He has generated this report after a lot of hard work. The report has been created under the guidance of the teacher Mrs. Sujata Bhardwaj and qualifies the benchmarks for the Python Project.

//About:

About the Project:

The project is focussed on being an all-rounder solution for services/ shops to manage their billing, customers, and record-keeping needs.

The project asks the user to select from nine options, namely: -

- INVOICING
- CUSTOMER REGISTRATION
- CUSTOMER REGISTRY
- CUSTOMER RECORDS
- OVERALL SALES AND ACCESS REGISTRY
- TO REVIEW THE STORE LISTING
- TO VIEW THE LICENSING
- REVIEW/ UPDATE STORE STOCK
- EXIT PROGRAM OPTION

Alongside a tenth option for internal testing, displayed upon entering a special testing code.

The same has been discussed in brief-detail below:

In the **invoicing mode**, the (optional) customer I.D. is asked for, and then the number of items being purchased. Then the user is required to enter each product's code, followed by the discount amount (can be zero) to be applied. The total amount is subsequently printed, logged in the registry file, and simultaneously synced to the Telethon web API, via requests.

In the **customer registration mode**, the user is asked to enter the customer's new I.D., full-name, and email address. The same is inserted into the SQL database.

In the **customer registry mode**, the program contacts the SQL database and displays the complete table of registered customers.

In the **customer records mode**, the program contacts a secondary included SQL database and displays a table with customer records including the customer I.D., number of purchases made and the amount those total to.

In the **view purchase and access registry mode**, the program first asks the user to authorize the action by asking for the password (default = "root"). Upon successful authentication, the program opens the sales log externally, with the default notepad application (system default .txt app).

The fifth option **view store listing** displays the complete store listing as entered in the programming.

The sixth option – **view licensing information** displays the brief license text and provides an option to display the whole externally.

The fifth option **review/ update stock** mode is quite a comprehensive one. An option is provided to either enforce stock universally, update individual stock, or to view all stock information, tabulated.

(NOTE: A negative number can be specified in either to decrease the stock directly)

The program also handles stock subtraction upon purchase, autonomously. Friendly alerts are also generated during invoicing if the product quantity (post invoicing) goes below 5.

Upon selecting the seventh – **exit option**, the program exits.

//Working:

Working Explained:

When the user **starts the script**, a GUI-based authentication script is launched. This accepts the password required to log-into DBFA. If and when an invalid password is provided, the script re-asks for the valid credentials.

Once authenticated, the python shell is opened and a UWP-native prompt appears, instructing the user to read the documentation supplied along with the program.

Then the program displays a sponsor-advert (optional feature), alongside the menu of options.

In the "**Billing**" option, the customer ID (if registered) is asked for, alongside the number of items being billed for, followed by their respective product codes. These are then matched with their prices and names as stored in two dictionaries (one provides cost; the other, the name). The cost is subsequently totalled and upon reaching the number of items purchased, the user (store cashier) is asked about the discount to be applied. Tax bars are auto applied, and the net payable amount is shown.

Simultaneously in the background, this data is sent via the Telethon web API to Telegram servers. The product details and total are also written to an external file.

The "**Customer Registration**" mode contacts the SQL database with the entered details and inserts them into a table by employing the mysql.connector or the derivative sqlite3 library.

The "**View Registered Customers**" mode refreshes the SQL connection and loads all the registration data from the data table.

The "**Customer Records**" mode refreshes the SQL connection and loads all the registration data from a secondary data table.

The "**View All Sales Records**" mode displays the externally written log in Notepad. A password is required to view the same. The user is provided with two chances for a valid authentication post which the option exits and the main screen reappears. Accessing this option, alongside the attempts required to complete authentication is sent via Telethon too, thereby ensuring maximum security.

The sixth, "**View Store Listing**" option displays the store listing fetched from the two dictionaries, each holding the commodity names and prices.

The seventh, "**View Licensing**" option displays a brief-up of the license with an option to view the whole. On selecting the same, the license file is opened externally.

The eighth, "**View/ Update Stock**" option, a choice is given to the user. This option presents another three internal options.

- **ENFORCE STOCK UNIVERSALLY:** A number is accepted and set as the stock for all product enforcing-ly.
- **UPDATE INDUVIDUAL STOCK:** This lets the stock be adjusted for individual products, on a additional schema, un-enforcing-ly.
- **VIEW ALL STOCK INFORMATION:** Displays all stock information in a tabular manner.

On selecting the ninth, “**EXIT**” option, the program simply exits.

On entering the internal testing code, “101”, the program opens a testing mode which can run an external script to REMOVE ALL CUSTOMER RECORDS, including, but not limited to registration data.

//Modules:

Functions Used:

from datetime import datetime()

- In-built function to display the system's date and time.

import time.sleep()

- In-built function used to specify delays in the program.

import os(), import time(), import sys()

- In-built modules to use system-wide functions.

import getpass()

- In-built function to get echo-less inputs, used especially for passwords.

import pathlib()

- In-built function to fetch the path of a file.

from PyWinGUI import PyWinGUI()

- Library to create GUI-based scripts. Use for the login process.

from win10toast import ToastNotifier()

- Module to display Windows 10 UWP toast notifications.

import mysql.connector() / sqlite3()

- Module(s) to connect and interact with MySQL databases.

import coloroma()

- A function used to colour/highlight text. Used for the optional advert in the main menu.

import shutil()

- Module used to change file paths (move files).

import reportlab()

- Module used to generate PDF files.

import requests()

- Module used to send/ get HTTP requests. Used for the Telegram Web API.

def mainmenu()

- Self-defined function to display the main-menu of the menu-driven program.

def inserter()

- Self-defined function to take values from the user and insert them into the SQL database selected.

def custcc()

- Self-defined function used to update the customer records registry.

def ssxstockmaster()

- Self-defined function to fetch individual product stock, used for billing-time low-stock alerts and to show the whole stock information.

def ssxstockmaintainer()

- Self-defined function used to adjust the stock difference during billing.

def ssxsuperfetch()

- Self-defined function used to fetch data from the stock registry in bulk.

def ssxupdatescript()

- Self-defined function used to add stock in option eight.

def massmaintainer()

- Self-defined function to re-build database structure when the database is found missing.

//A Few Things:

A Few Things:

Telegram-Integration:

DBFA, being focused on security, has a Telegram “bot” account. This medium is used to send all billing/ internal options access alerts to the store owner.

This option has been designed with a real-life store in mind. As the owner is not the cashier in most cases, this system lets the store owner know of the REAL sales information, so that the cashier cannot show a falsified record and pocket the remnant.

Moreover, access alerts are also sent when something which is not meant to be accessed by is accessed.

This process happens in the background using simple HTTP(S) requests, thereby using minimal bandwidth and being literally unobservable to the normal user.

Automated Database Structure Rebuilding:

In the modern world, there can be issues everywhere, and just anywhere.

There can be disk failures resulting in data loses, accidental file deletion, malwares or ransomwares, or even internal errors which can lead to corrupted databases. DBFA overcomes this by searching for the database files each time the software is booted.

If the required files are found to be missing, the database structure is rebuilt automatically, albeit there may be some data loss.

This feature is also useful in the internal-CIT testing option where the user can delete all customer records.

PDF Invoicing:

Whenever an invoice is issued, a PDF invoice is generated with the time of billing as it’s file name and moved to a folder names “Invoices”, in DBFA’s installation directory. All of this happens silently, in the background while using minimal system resources because of its dependence on system-wide functions.

Hardware Used:

- **CPU**

  For computation.

- **SSD or HDD**

  For program data storage.

- **RAM**

  For program functioning.

- **NETWORK**

  Used to communicate with the Telegram Web API.

Software Used:

- Windows 10
- Microsoft Visual Studio (for code-writing)
- Windows Terminal (for testing)
- Office Word (for documentation)

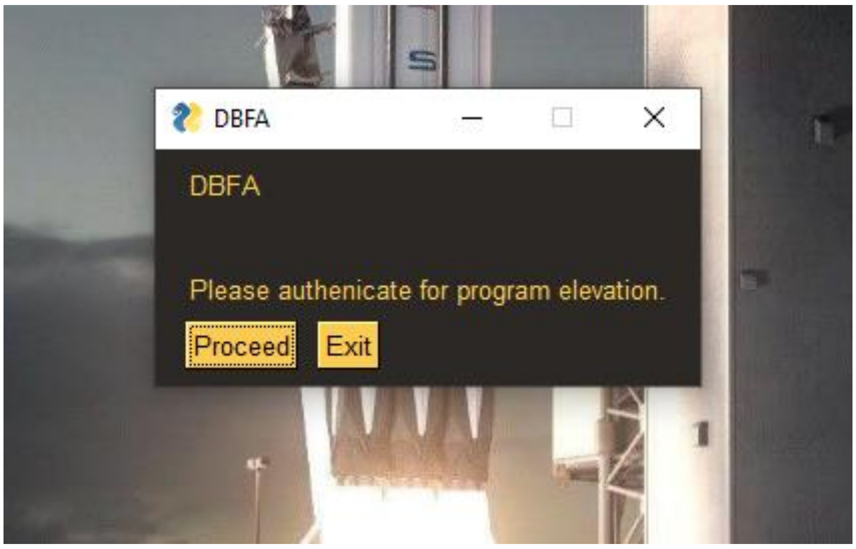
//Code:

//Code:

//Code:

//Runtime:

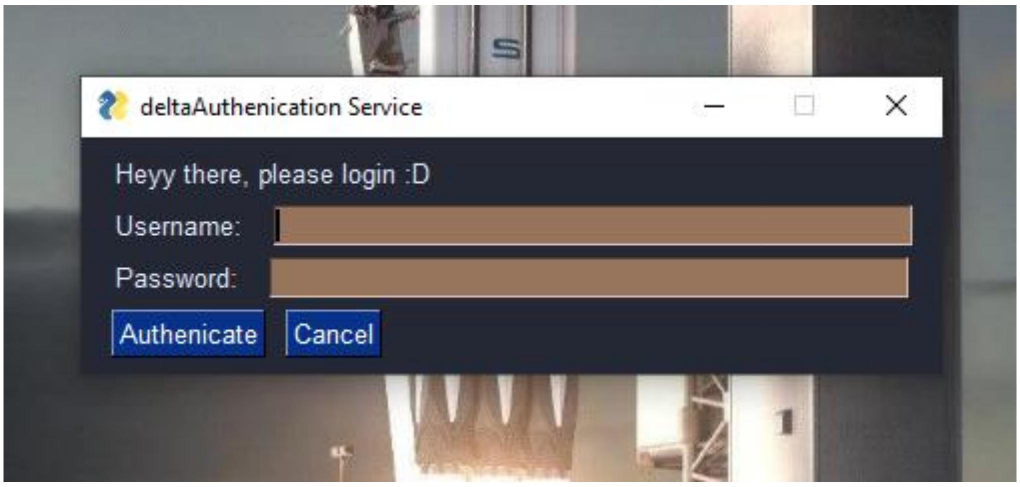
Start of the login script:



DBFA

Please authenicate for program elevation.

Authentication script:

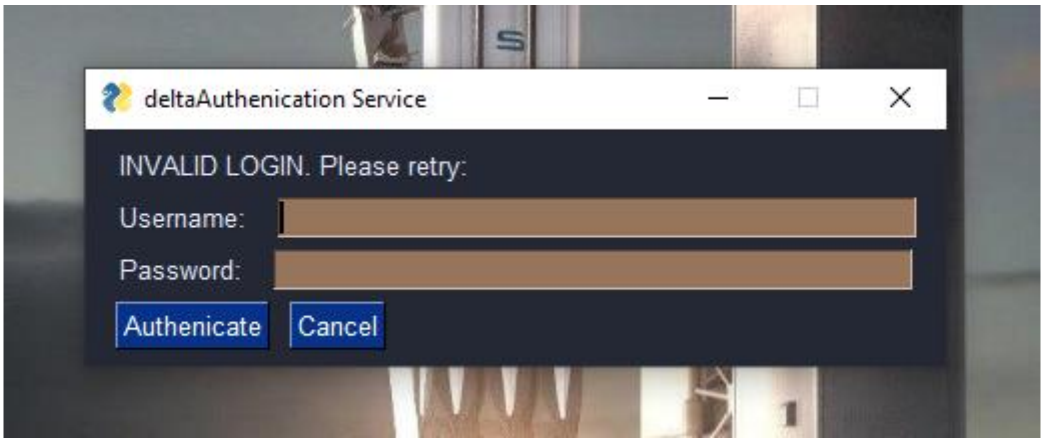


Heyy there, please login :D

Username:

Password:

Re-authentication upon invalid password entry:

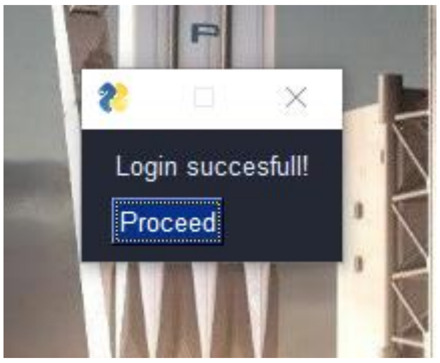


INVALID LOGIN. Please retry:

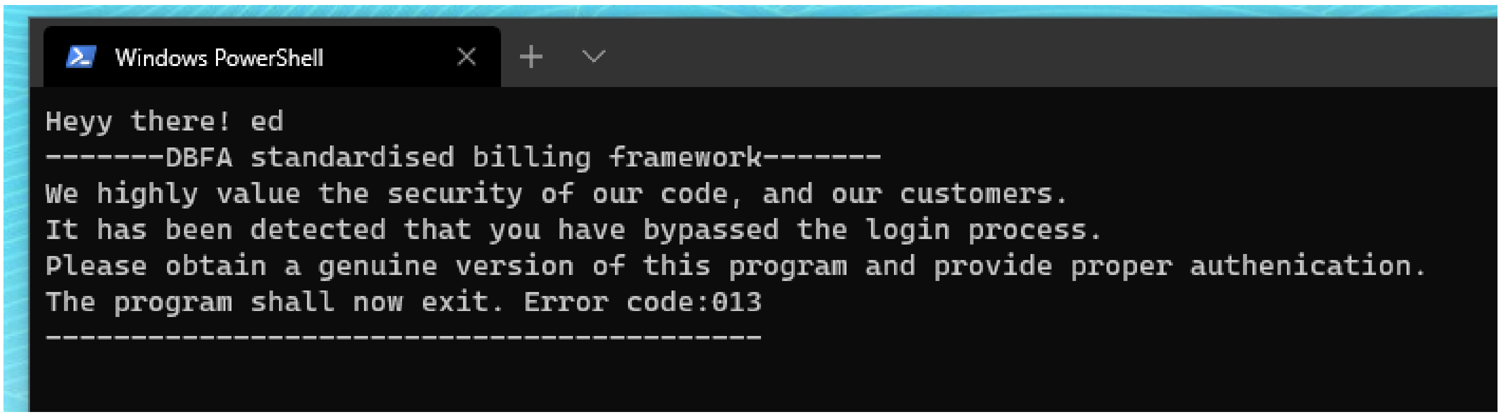
Username:

Password:

Successful authentication page:

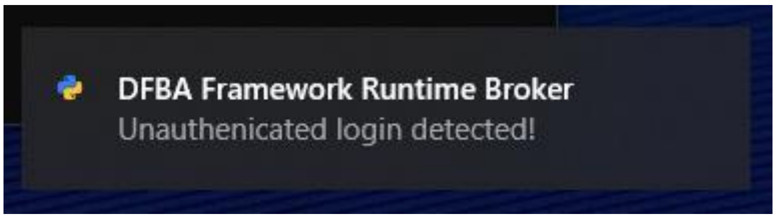


Page on bypassing the login script in the main program:



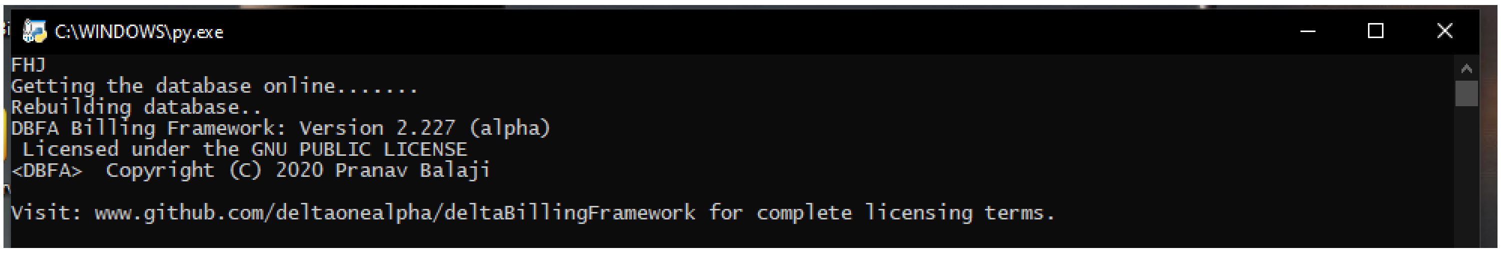
```
Heyy there! ed
-------DBFA standardised billing framework-------
We highly value the security of our code, and our customers.
It has been detected that you have bypassed the login process.
Please obtain a genuine version of this program and provide proper authenication.
The program shall now exit. Error code:013
------------------------------------------
```

Subsequently displayed UWP-toast:

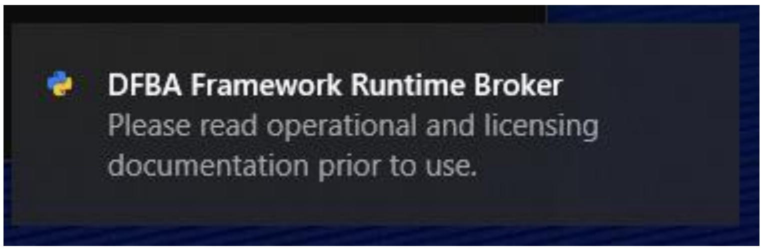


DFBA Framework Runtime Broker
Unauthenicated login detected!

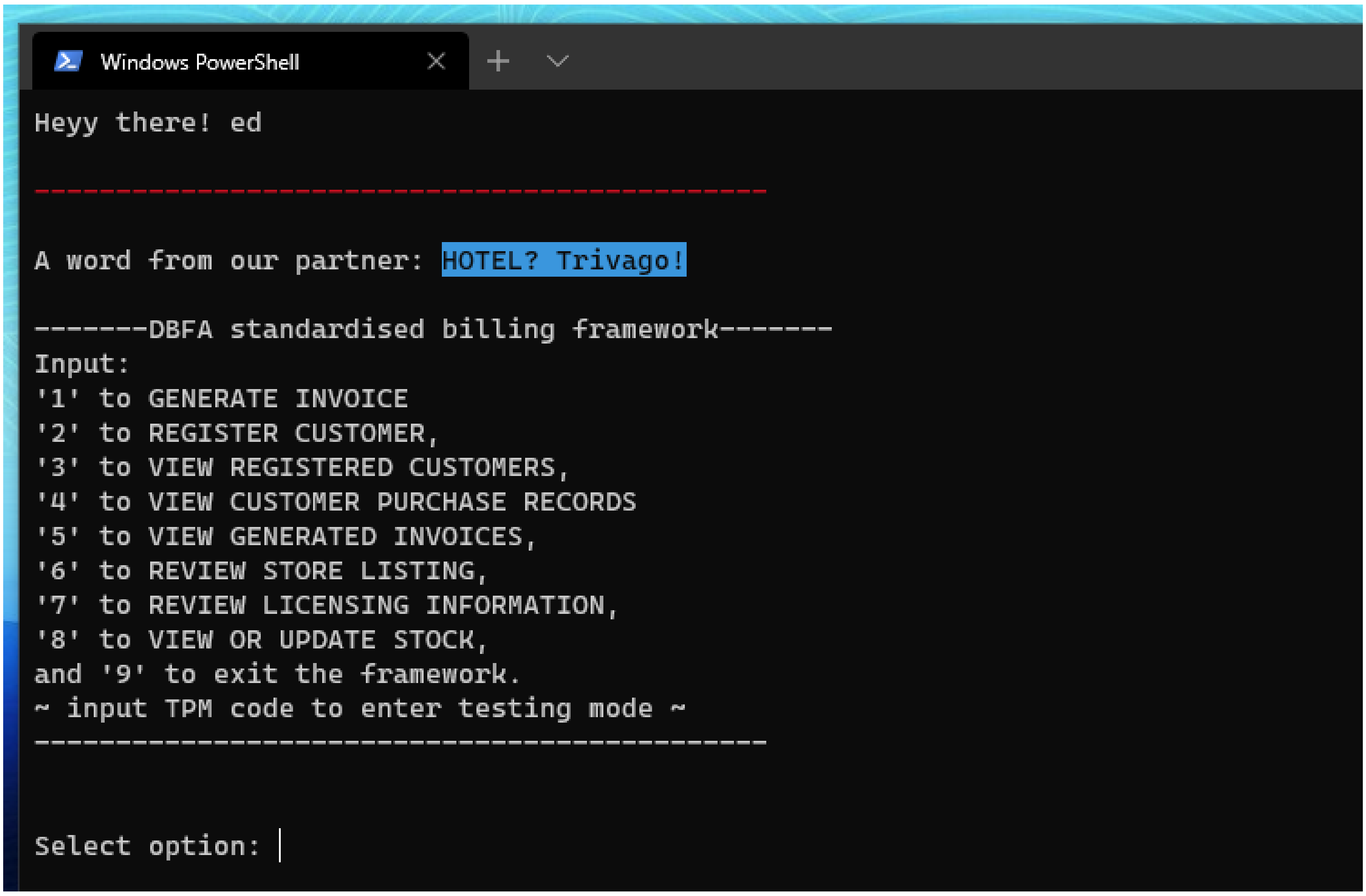
Main program with successful authentication:



```
FHJ
Getting the database online.......
Rebuilding database..
DBFA Billing Framework: Version 2.227 (alpha)
 Licensed under the GNU PUBLIC LICENSE
<DBFA>  Copyright (C) 2020 Pranav Balaji

Visit: www.github.com/deltaonealpha/deltaBillingFramework for complete licensing terms.
```

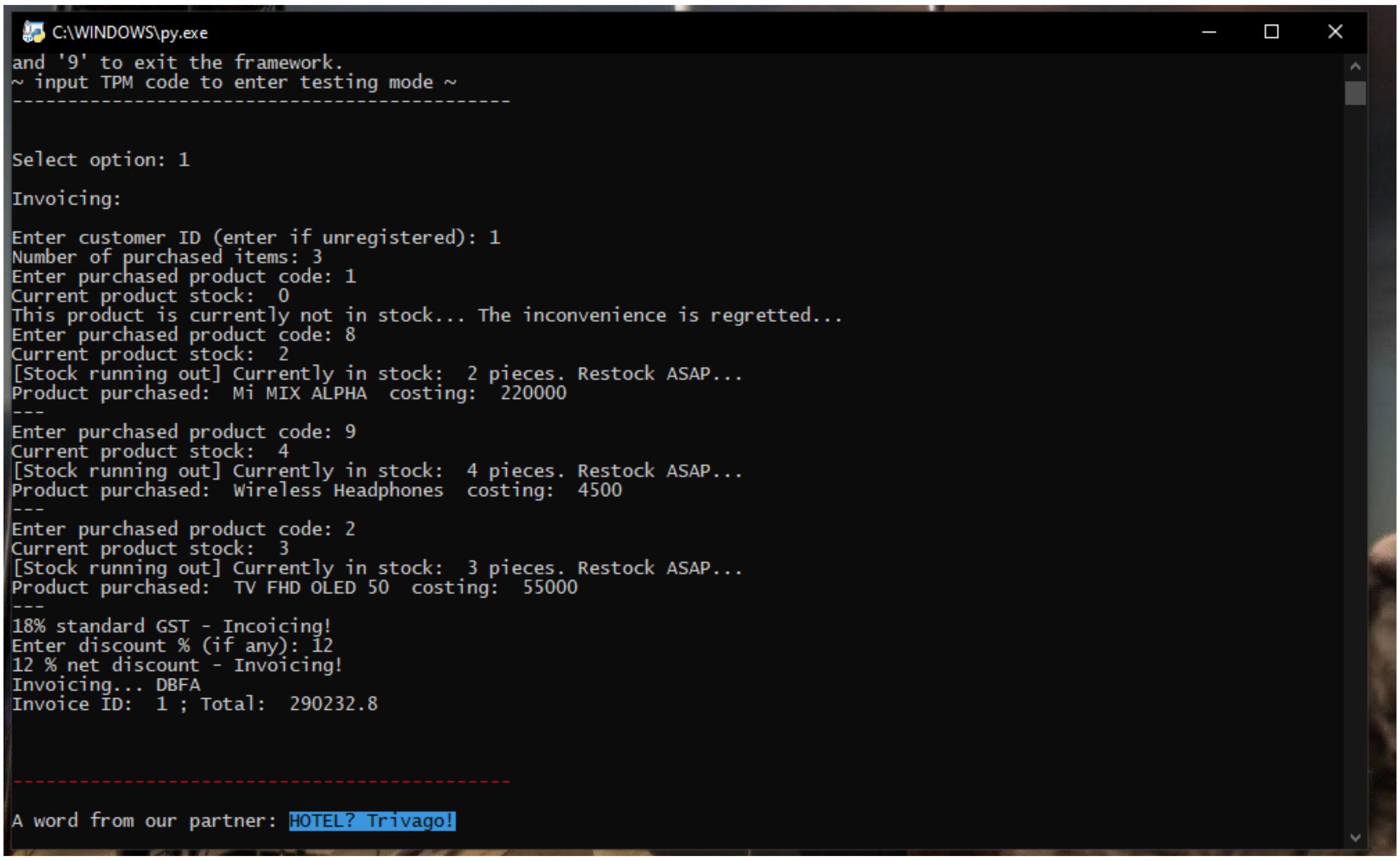


DFBA Framework Runtime Broker
Please read operational and licensing documentation prior to use.


Windows PowerShell

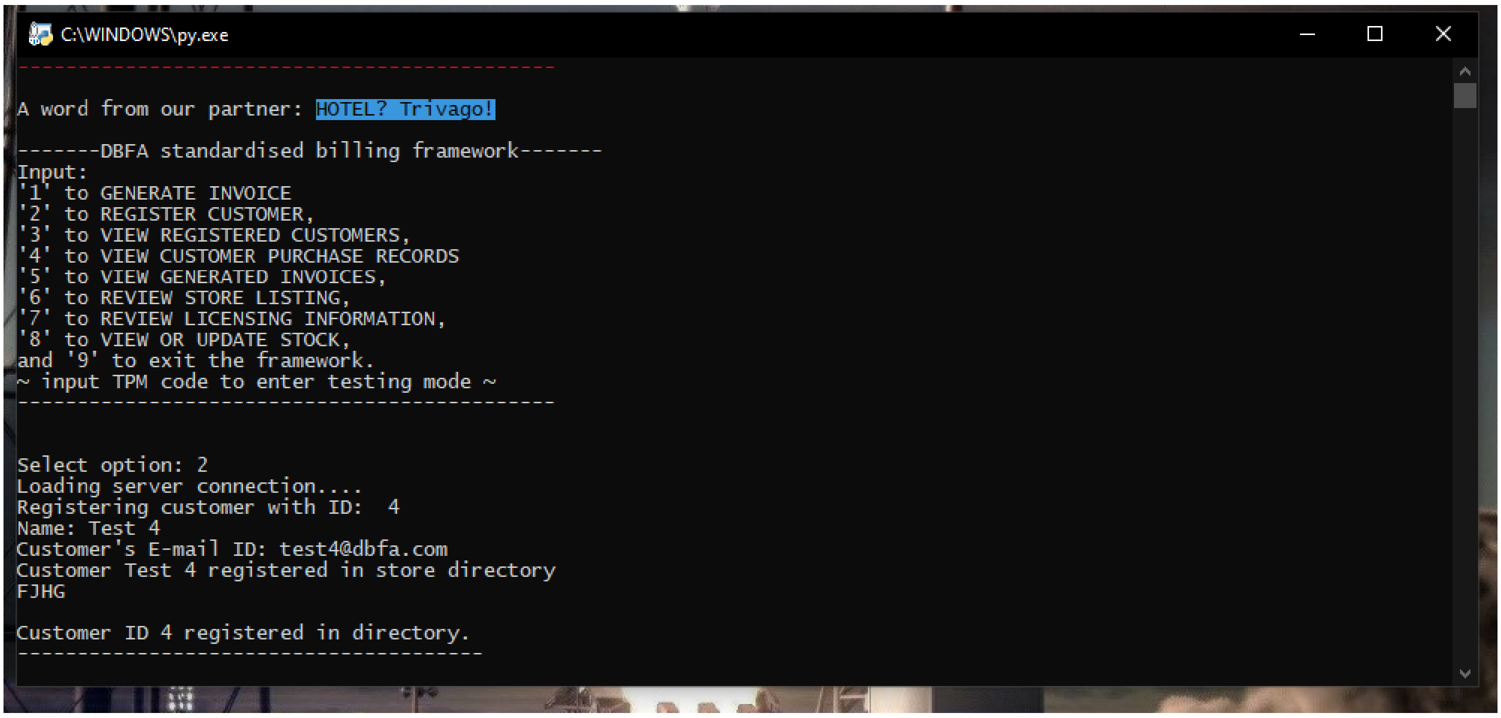
```
Heyy there! ed

-------------------------------------------------------------

A word from our partner: HOTEL? Trivago!

-------DBFA standardised billing framework-------
Input:
'1' to GENERATE INVOICE
'2' to REGISTER CUSTOMER,
'3' to VIEW REGISTERED CUSTOMERS,
'4' to VIEW CUSTOMER PURCHASE RECORDS
'5' to VIEW GENERATED INVOICES,
'6' to REVIEW STORE LISTING,
'7' to REVIEW LICENSING INFORMATION,
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
-------------------------------------------------------------


Select option:
```

Invoicing Mode:



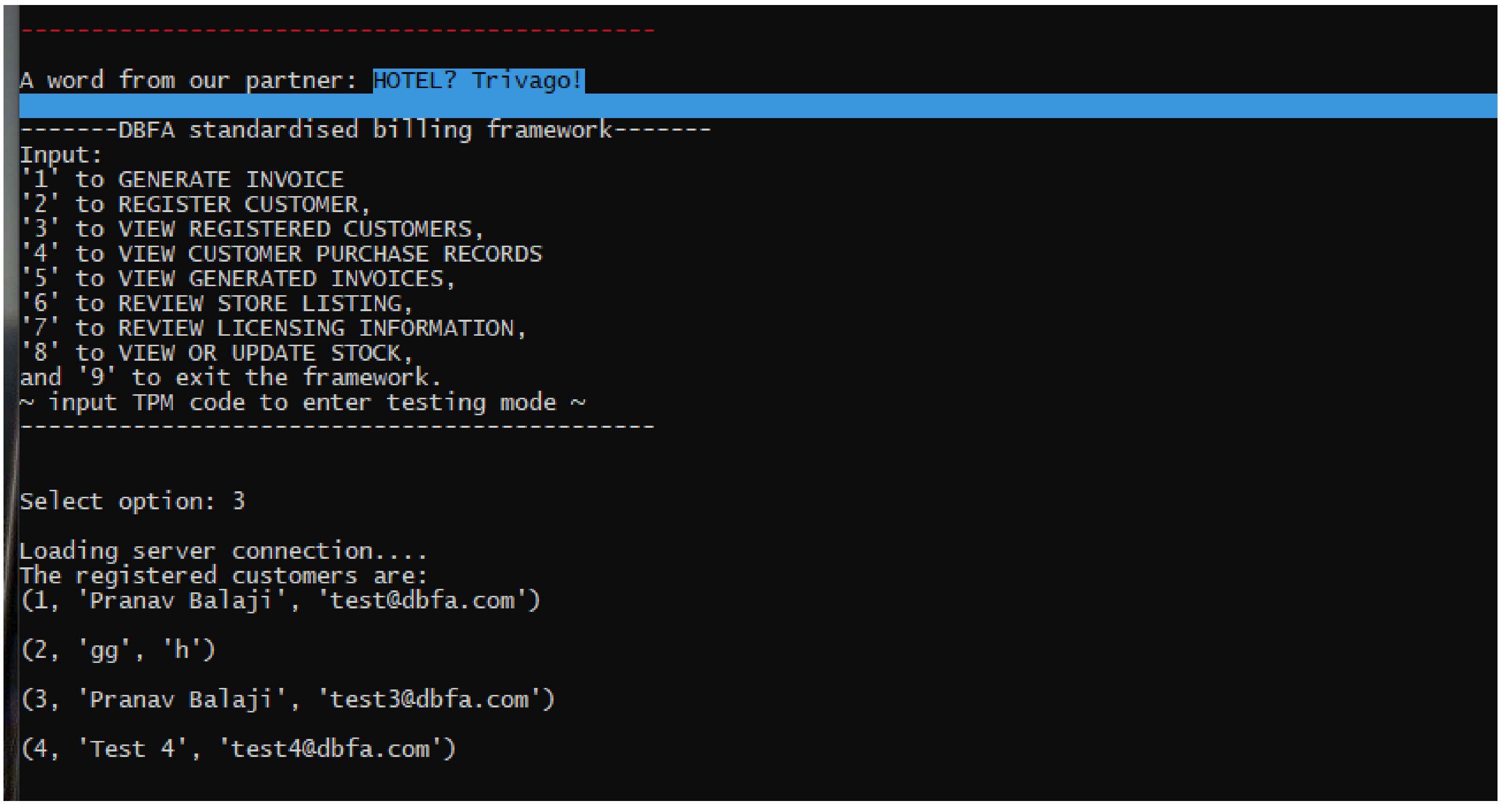
```
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 1

Invoicing:

Enter customer ID (enter if unregistered): 1
Number of purchased items: 3
Enter purchased product code: 1
Current product stock:  0
This product is currently not in stock... The inconvenience is regretted...
Enter purchased product code: 8
Current product stock:  2
[Stock running out] Currently in stock:  2 pieces. Restock ASAP...
Product purchased:  Mi MIX ALPHA  costing:  220000
---
Enter purchased product code: 9
Current product stock:  4
[Stock running out] Currently in stock:  4 pieces. Restock ASAP...
Product purchased:  Wireless Headphones  costing:  4500
---
Enter purchased product code: 2
Current product stock:  3
[Stock running out] Currently in stock:  3 pieces. Restock ASAP...
Product purchased:  TV FHD OLED 50  costing:  55000
---
18% standard GST - Incoicing!
Enter discount % (if any): 12
12 % net discount - Invoicing!
Invoicing... DBFA
Invoice ID:  1 ; Total:  290232.8



---------------------------------------------

A word from our partner: HOTEL? Trivago!
```

Customer Registration Mode:



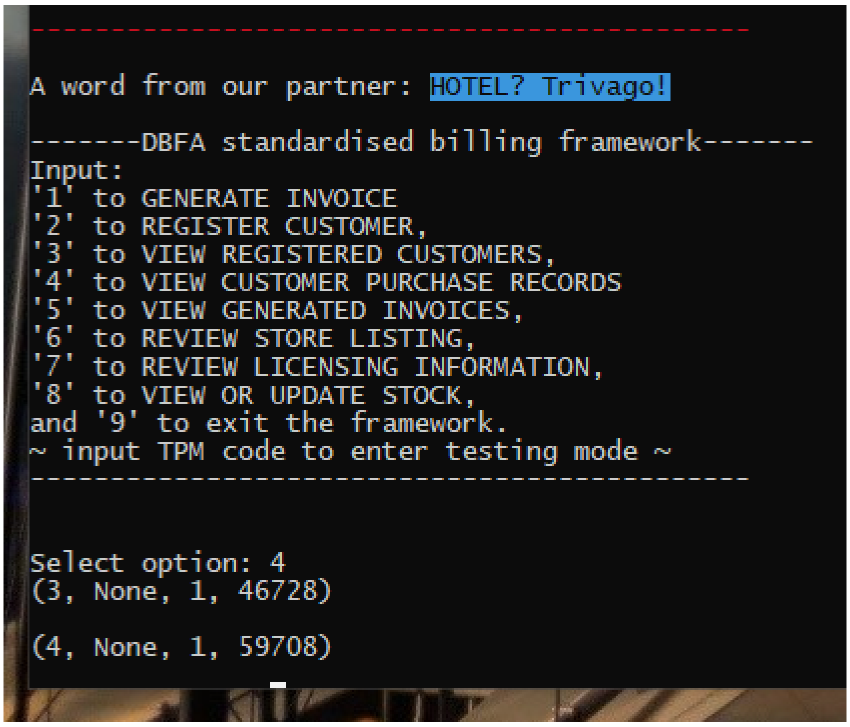
```
---------------------------------------------

A word from our partner: HOTEL? Trivago!

-------DBFA standardised billing framework-------
Input:
'1' to GENERATE INVOICE
'2' to REGISTER CUSTOMER,
'3' to VIEW REGISTERED CUSTOMERS,
'4' to VIEW CUSTOMER PURCHASE RECORDS
'5' to VIEW GENERATED INVOICES,
'6' to REVIEW STORE LISTING,
'7' to REVIEW LICENSING INFORMATION,
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 2
Loading server connection....
Registering customer with ID:  4
Name: Test 4
Customer's E-mail ID: test4@dbfa.com
Customer Test 4 registered in store directory
FJHG

Customer ID 4 registered in directory.
---------------------------------------
```

Registered Customers Viewing Mode:



```
--------------------------------------------

A word from our partner: HOTEL? Trivago!

-------DBFA standardised billing framework-------
Input:
'1' to GENERATE INVOICE
'2' to REGISTER CUSTOMER,
'3' to VIEW REGISTERED CUSTOMERS,
'4' to VIEW CUSTOMER PURCHASE RECORDS
'5' to VIEW GENERATED INVOICES,
'6' to REVIEW STORE LISTING,
'7' to REVIEW LICENSING INFORMATION,
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
--------------------------------------------


Select option: 3

Loading server connection....
The registered customers are:
(1, 'Pranav Balaji', 'test@dbfa.com')

(2, 'gg', 'h')

(3, 'Pranav Balaji', 'test3@dbfa.com')

(4, 'Test 4', 'test4@dbfa.com')
```

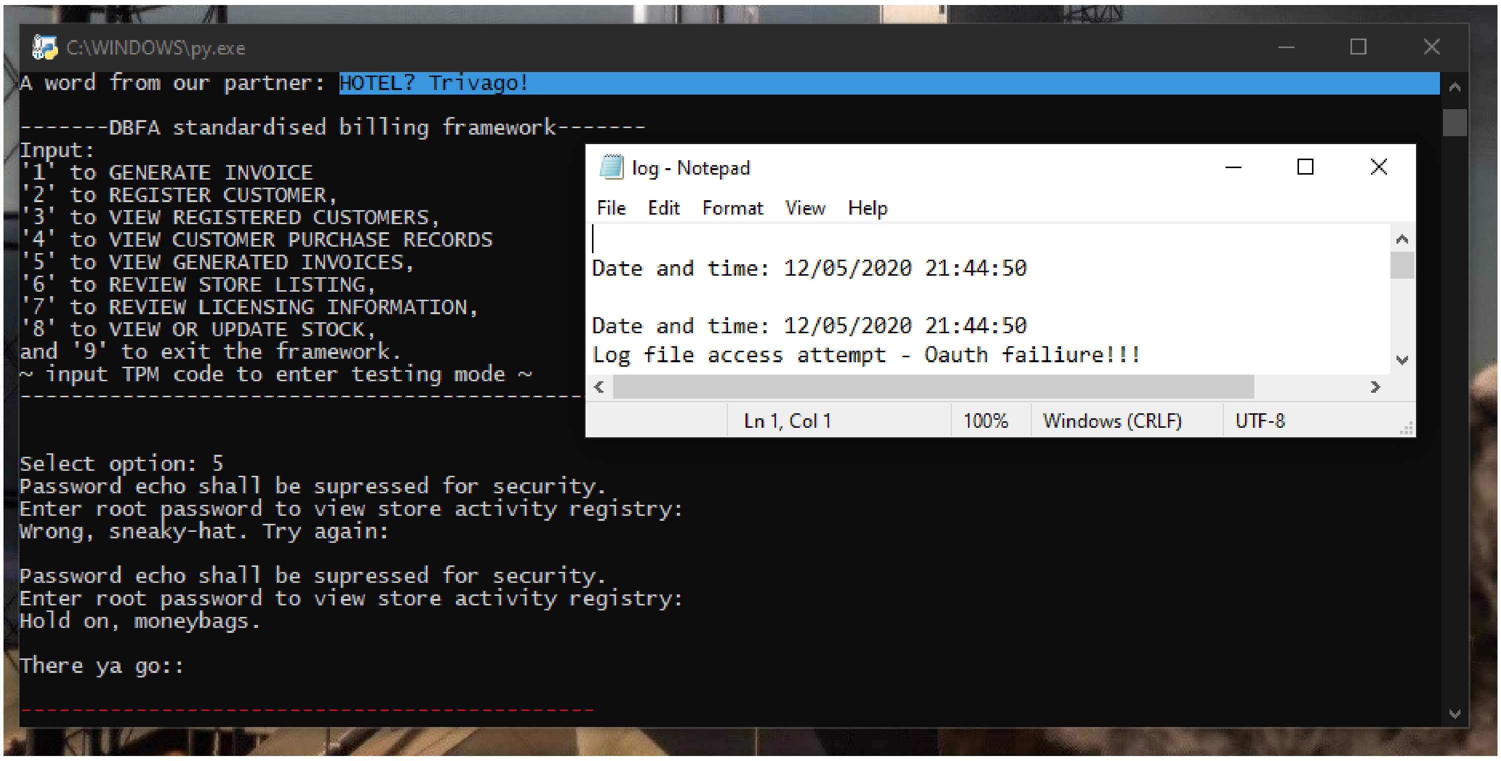
View Customer Purchase Records Mode:



```
--------------------------------------------

A word from our partner: HOTEL? Trivago!

-------DBFA standardised billing framework-------
Input:
'1' to GENERATE INVOICE
'2' to REGISTER CUSTOMER,
'3' to VIEW REGISTERED CUSTOMERS,
'4' to VIEW CUSTOMER PURCHASE RECORDS
'5' to VIEW GENERATED INVOICES,
'6' to REVIEW STORE LISTING,
'7' to REVIEW LICENSING INFORMATION,
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
--------------------------------------------


Select option: 4
(3, None, 1, 46728)

(4, None, 1, 59708)
```

//DBFA Billing Framework

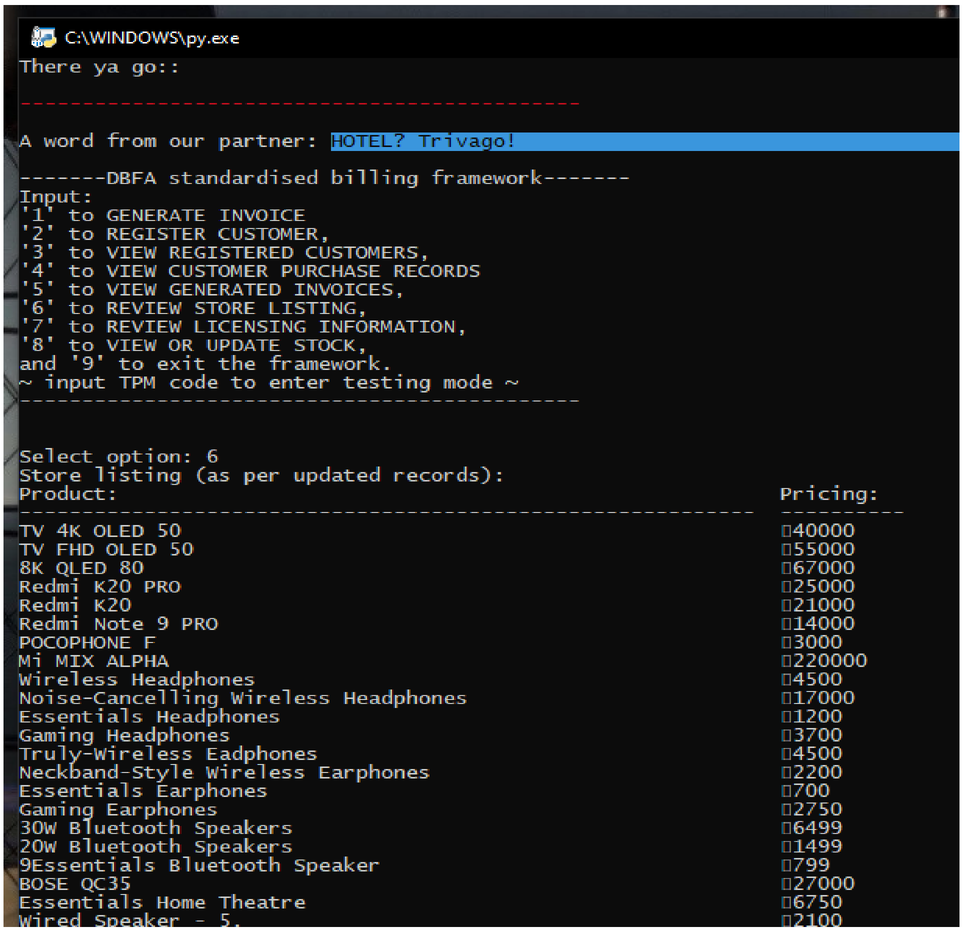
View Generated Invoiced Mode:



```
A word from our partner: HOTEL? Trivago!

-------DBFA standardised billing framework-------
Input:
'1' to GENERATE INVOICE
'2' to REGISTER CUSTOMER,
'3' to VIEW REGISTERED CUSTOMERS,
'4' to VIEW CUSTOMER PURCHASE RECORDS
'5' to VIEW GENERATED INVOICES,
'6' to REVIEW STORE LISTING,
'7' to REVIEW LICENSING INFORMATION,
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
--------------------------------------------


Select option: 5
Password echo shall be supressed for security.
Enter root password to view store activity registry:
Wrong, sneaky-hat. Try again:

Password echo shall be supressed for security.
Enter root password to view store activity registry:
Hold on, moneybags.

There ya go::

---------------------------------------------
```

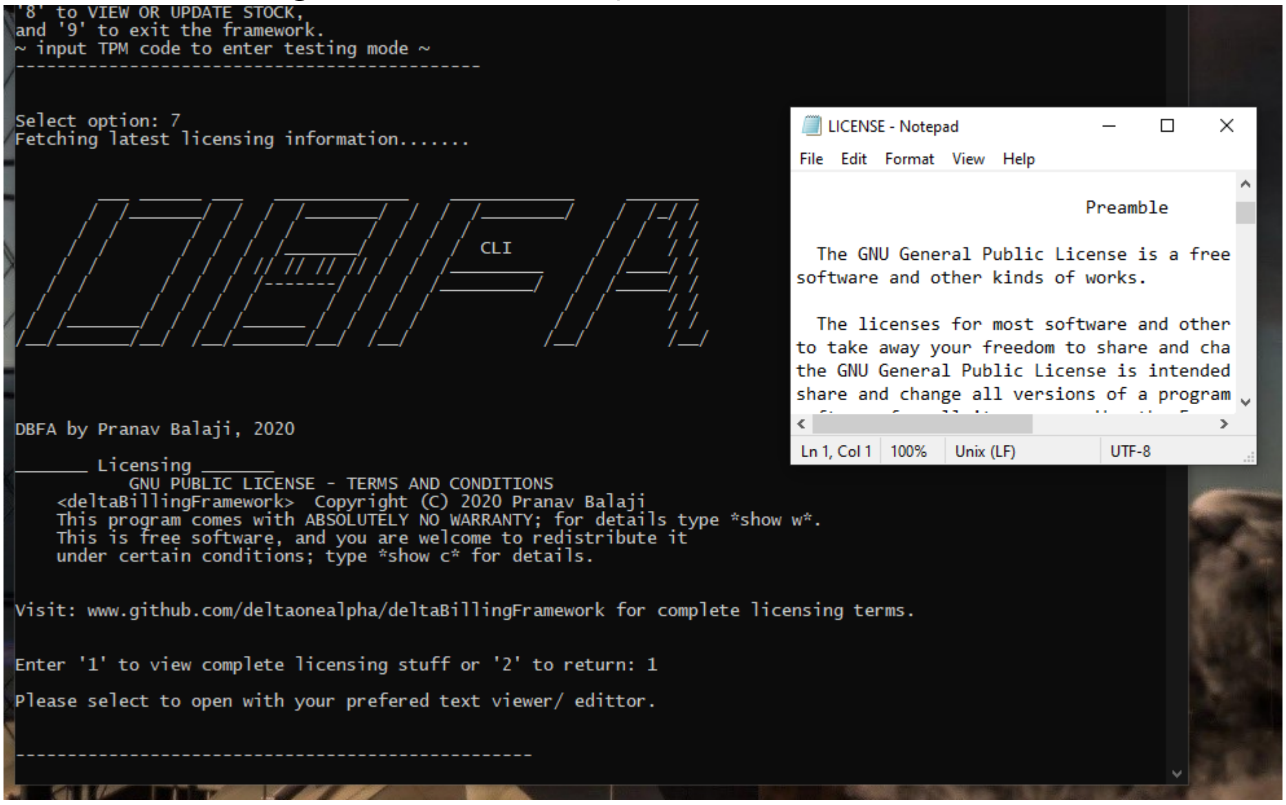
```

Date and time: 12/05/2020 21:44:50

Date and time: 12/05/2020 21:44:50
Log file access attempt - Oauth failiure!!!
```

View Store Listing Mode:



```
C:\WINDOWS\py.exe
There ya go::

---------------------------------------------

A word from our partner: HOTEL? Trivago!

-------DBFA standardised billing framework-------
Input:
'1' to GENERATE INVOICE
'2' to REGISTER CUSTOMER,
'3' to VIEW REGISTERED CUSTOMERS,
'4' to VIEW CUSTOMER PURCHASE RECORDS
'5' to VIEW GENERATED INVOICES,
'6' to REVIEW STORE LISTING,
'7' to REVIEW LICENSING INFORMATION,
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 6
Store listing (as per updated records):
Product:                                                      Pricing:
-----------------------------------------------------------   ----------
TV 4K OLED 50                                                 □40000
TV FHD OLED 50                                                □55000
8K QLED 80                                                    □67000
Redmi K20 PRO                                                 □25000
Redmi K20                                                     □21000
Redmi Note 9 PRO                                              □14000
POCOPHONE F                                                   □3000
Mi MIX ALPHA                                                  □220000
Wireless Headphones                                           □4500
Noise-Cancelling Wireless Headphones                          □17000
Essentials Headphones                                         □1200
Gaming Headphones                                             □3700
Truly-Wireless Eadphones                                      □4500
Neckband-Style Wireless Earphones                             □2200
Essentials Earphones                                          □700
Gaming Earphones                                              □2750
30W Bluetooth Speakers                                        □6499
20W Bluetooth Speakers                                        □1499
9Essentials Bluetooth Speaker                                 □799
BOSE QC35                                                     □27000
Essentials Home Theatre                                       □6750
Wired Speaker - 5.                                            □2100
```

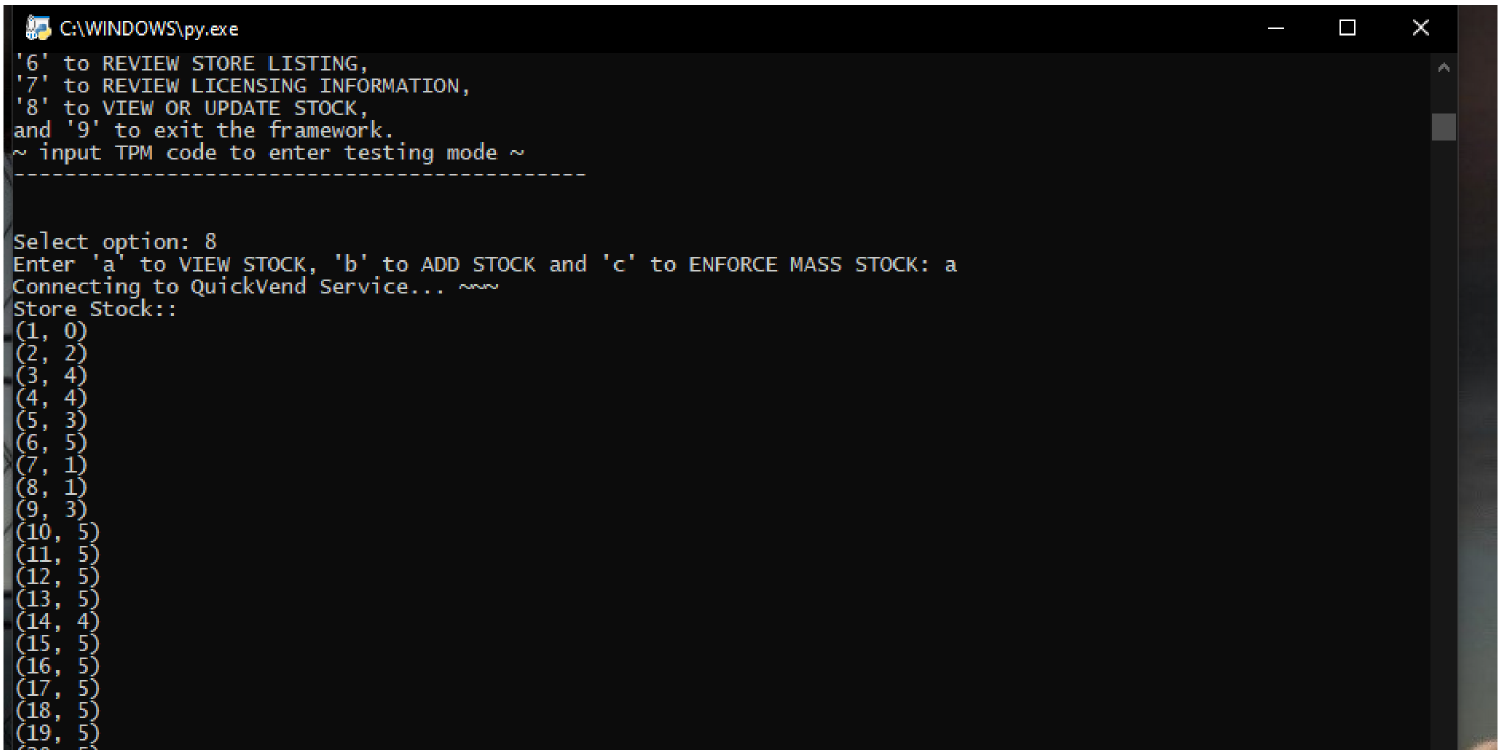
View Licensing Information Option:



```
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 7
Fetching latest licensing information.......



                                    CLI




DBFA by Pranav Balaji, 2020

______ Licensing
         GNU PUBLIC LICENSE - TERMS AND CONDITIONS
    <deltaBillingFramework>  Copyright (C) 2020 Pranav Balaji
    This program comes with ABSOLUTELY NO WARRANTY; for details type *show w*.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type *show c* for details.


Visit: www.github.com/deltaonealpha/deltaBillingFramework for complete licensing terms.


Enter '1' to view complete licensing stuff or '2' to return: 1

Please select to open with your prefered text viewer/ edittor.


---------------------------------------------
```

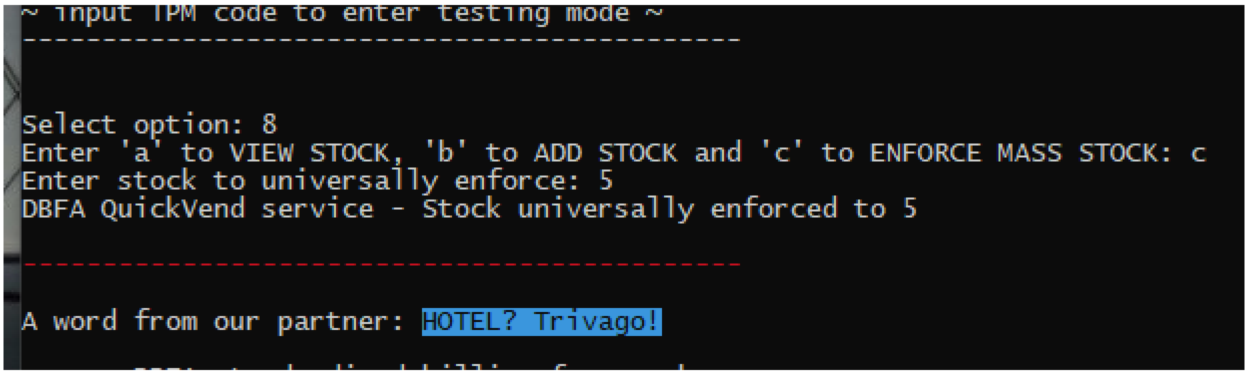
```
LICENSE - Notepad
File  Edit  Format  View  Help

                          Preamble

  The GNU General Public License is a free
software and other kinds of works.

  The licenses for most software and other
to take away your freedom to share and cha
the GNU General Public License is intended
share and change all versions of a program

Ln 1, Col 1   100%   Unix (LF)   UTF-8
```

View Store Stock Option:



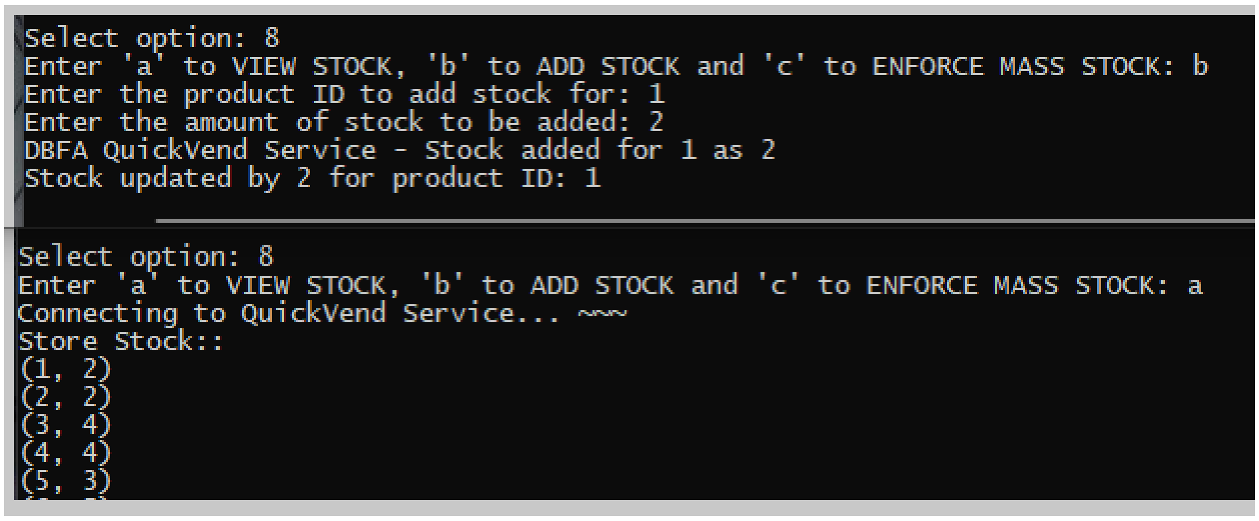
```
C:\WINDOWS\py.exe
'6' to REVIEW STORE LISTING,
'7' to REVIEW LICENSING INFORMATION,
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 8
Enter 'a' to VIEW STOCK, 'b' to ADD STOCK and 'c' to ENFORCE MASS STOCK: a
Connecting to QuickVend Service... ~~~
Store Stock::
(1, 0)
(2, 2)
(3, 4)
(4, 4)
(5, 3)
(6, 5)
(7, 1)
(8, 1)
(9, 3)
(10, 5)
(11, 5)
(12, 5)
(13, 5)
(14, 4)
(15, 5)
(16, 5)
(17, 5)
(18, 5)
(19, 5)
```

Enforce Store Stock Option:



```
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 8
Enter 'a' to VIEW STOCK, 'b' to ADD STOCK and 'c' to ENFORCE MASS STOCK: c
Enter stock to universally enforce: 5
DBFA QuickVend service - Stock universally enforced to 5

---------------------------------------------

A word from our partner: HOTEL? Trivago!
```

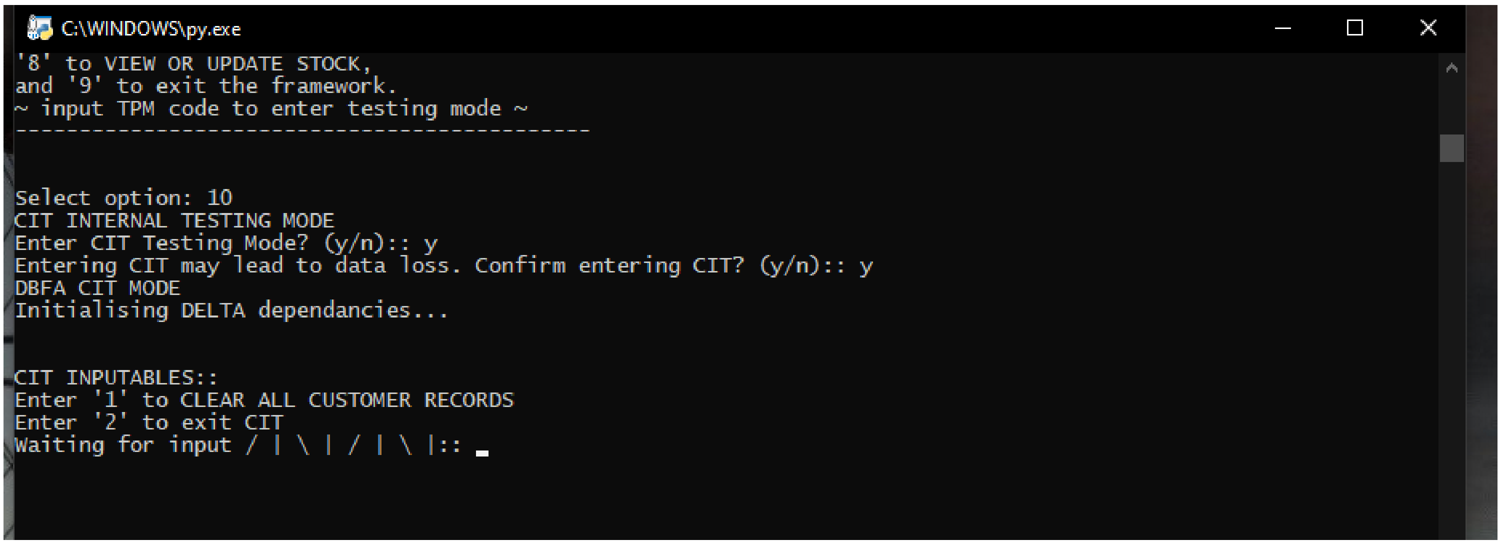
Set Individual Stock Option:



```
Select option: 8
Enter 'a' to VIEW STOCK, 'b' to ADD STOCK and 'c' to ENFORCE MASS STOCK: b
Enter the product ID to add stock for: 1
Enter the amount of stock to be added: 2
DBFA QuickVend Service - Stock added for 1 as 2
Stock updated by 2 for product ID: 1
```

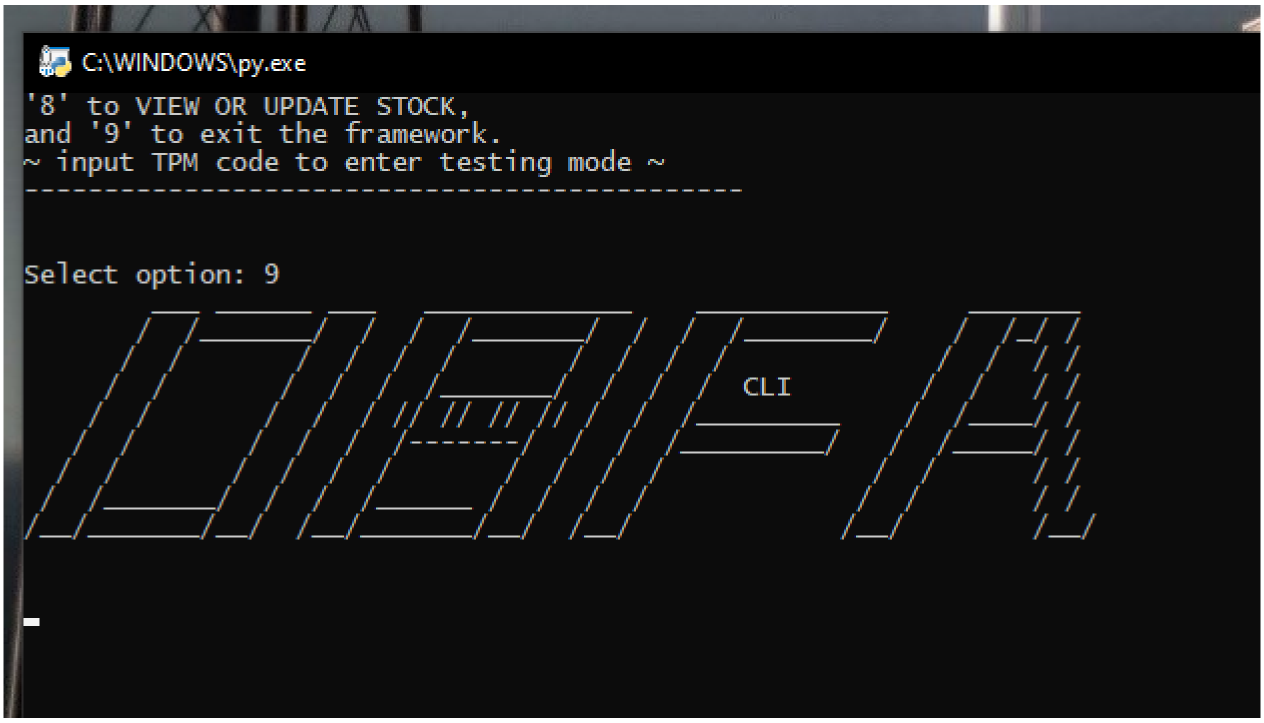
```
Select option: 8
Enter 'a' to VIEW STOCK, 'b' to ADD STOCK and 'c' to ENFORCE MASS STOCK: a
Connecting to QuickVend Service... ~~~
Store Stock::
(1, 2)
(2, 2)
(3, 4)
(4, 4)
(5, 3)
```

Internal Testing Mode (CIT):



```
C:\WINDOWS\py.exe
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 10
CIT INTERNAL TESTING MODE
Enter CIT Testing Mode? (y/n):: y
Entering CIT may lead to data loss. Confirm entering CIT? (y/n):: y
DBFA CIT MODE
Initialising DELTA dependancies...


CIT INPUTABLES::
Enter '1' to CLEAR ALL CUSTOMER RECORDS
Enter '2' to exit CIT
Waiting for input / | \ | / | \ |:: _
```

Exit Program Option:



```
C:\WINDOWS\py.exe
'8' to VIEW OR UPDATE STOCK,
and '9' to exit the framework.
~ input TPM code to enter testing mode ~
---------------------------------------------


Select option: 9

                                              CLI
```

//Telegram

Integration:

//Bibliography:

Bibliography:

Works sighted from:

- Core Python Programming by R. Nageshwar Rao
- Debugging assistance from StackOverflow – stackoverflow.com
- Google – google.com
- Python for Class – XII by Preeti Arora

Thank You

-- DBFA Billing Framework --

Developed by Pranav Balaji

Python Board Project