
def package_dbfadonager():
/**
* The program DBFAdonager implements
* an application that houses a damn nice
* store management software.
*
* @author deltaonealpha & sushimuncher
*/
from DBFA import DBFAdocs
if DBFAdocs.ExecutionStatus() == False:
print(DBFAdocs.HelloWorld())
DBFAdocs.runExecution()
package_dbfadonager()
DBFA
Store Manager
by deltaonealpha & sushimuncher

//DBFA Billing Framework

Python Project Report

TOPIC: Billing system

Pranav Balaji and Sushant Gupta

XII – A

ACKNOWLEDGMENT:

This is to certify that Pranav Balaji and Sushant Gupta of Apeejay School, NOIDA (Class: 12-A) have created a project on the topic 'Store Management', called the 'DBFA Store Manager'.

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

(Please Turn Over)

(Please Turn Over)

//About DBFA

About the Project:

DBFA is focussed on being an all-rounder solution for shops to manage their billing, customers, inventory and employee management, and record-keeping needs.

We've tried to integrate all that a store would ever need into DBFA.

DBFA's home menu offers 14 broad options with sub-options, namely: -

- **INVOICING**
  - IN-STORE PURCHASES OR DELIVERIES
- **MANAGE CUSTOMERS**
  - REGISTER A CUSTOMER
  - VIEW THE CUSTOMER REGISTRY
  - VIEW CUSTOMER-SPECIFIC PURCHASE RECORDS
  - FIND A CUSTOMER
  - EXPORT CUSTOMER DATA AS CSV
- **STORE OPTIONS**
  - MANAGE STOCK
    - VIEW STOCK, ADD STOCK (INDUVIDUALLY AND FOR THE MASS)
  - DBFA STOCK MASTER
    - ORDER STOCK, UPDATE OR FETCH STATUS, VIEW, EDIT AND CONTACT VENDOR (PRODUCT SPECIFIC), MODIFY LOW-STOCK WARNING BAR
  - MANAGE VOUCHERS
    - GENERATE AND VIEW GENERATED VOUCHERS
  - SEE PRODUCT LISTINGS
  - VIEW THE SALES LOG
  - EXPORT STORE AND SALES DATA AS CSV
- **GENERATE STORE REPORT**
- **MANAGE DELIVERIES**
  - VIEW PENDING/ MARK AS DELIVERED
- **DBFA OPTIONS (PERSISTENT SETTINGS)**
- **DBFA BACKUP & SWITCH**
- **ANALYSE SALES (SALES PLOTTER)**
- **DBFA EMPLOYEE MANAGER (SPECIAL TRIGGER)**
- **MARK EMPLOYEE ATTENDANCE (SPECIAL TRIGGER)**
- **VIEW SOFTWARE LICENSE**
- **ABOUT DBFA 8.52**
- **CHECK FOR UPDATES**
- **QUIT**

The home menu also contains special keyword triggers for certain functions and also shortcut keywords for each sub-function:

- **DBFA Employee Manager**
  ("emp", "EMP", "EMPLOYEE", "employee", "manager", "MANAGER", "empm", "EMPM")
- **Mark Attendance**
  ('attendance', 'ATTENDANCE', 'mark', 'MARK', 'mArK', 'MaRk', 'maRK', 'MArk', 'm a r k', 'M A R K', 'M A r k', 'm a R K')

(Please Turn Over)

//Libraries

Libraries used by DBFA:

from datetime import datetime

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

import os, time, sys, shutil, pathlib, logging, socket

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

import urllib, json, math, random

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

import telegram_send

- Library which leverages the Telegram BOT API v2

from telegram import InlineKeyboardButton, InlineKeyboardMarkup

- Library which leverages the Telegram BOT API v2

from telegram.ext import Updater, CommandHandler, CallbackQueryHandler

- Library which leverages the Telegram BOT API v2

import getpass

- In-built function to get echo-less inputs, used for password inputs/

from PySimpleGUI import PySimpleGUI

- Library to create GUI-based elements. DBFA makes use of this in places which require authentication.

from win10toast import ToastNotifier

- Module to display Windows 10 UWP toast notifications.

import sqlite3

- Module to connect and interact with SQLite3 databases. Comes built-in with Python 3.

import cv2

- Open Computer Vision module used to open images in-shell.

import colorama

- A function used to colour/highlight text.
  DBFA makes use of this library to artfully present the home menu without straining the eye.

import reportlab

- Module used to generate PDF files. Used for invoicing and DBFA Report Generator

import requests

- Module used to send and receive HTTP(S) requests.

import pyqrcode, png

- Used for creating QR codes and to export it in .png

from pyqrcode import QRCode

- Used for creating QR codes and to export it in .png

from PIL import Image, ImageDraw, ImageFont

- Module used to interface with image files.

from tabularprint import table

- Special module used to print SQLite3 tables in a specialised way.

from tqdm import tqdm

- Module used to display neat progress bars.

import webbrowser

- Module used to open and interface with the device's default web browser

import spotilib

- A wonderful module by XanderMJ used to interface with Spotify.
  CREDITS: (https://github.com/XanderMJ/spotilib)

from SwSpotify import spotify

- Another module used to interface with Spotify.

import email

- Module used to send and receive emails.

import pandas

- A very advanced library used for data analysis

import csv

- An inbuilt library used to interface with CSV files.

from pynput.keyboard import Key, Controller

- A wonderful library used to emulate keyboard inputs from within Python

import platform

- A module used to extract system information.

import matplotlib.pyplot

- A very advanced library used for data graphing (plotting).

import oschmod

- A module used to elevate/ decline access permissions for certain files.

from DBFADeepArchivalEngine import alphadecoder, dttdecoder, encoder_deeparchival, decoder_deeparchival, deepfetch_deeparchival

- Custom-developed module which provides functions for various services of the deep archival engine, such as:
  - **alphadecoder** : Decodes encoded alphabets to normal ones.
  - **dttdecoder** : Decodes dates from encoded strings to normal ones.
  - **encoder_deeparchival** : Encodes data and archives keys in a database.
  - **decoder_deeparchival** : Decodes key and passes back the original data.
  - **deepfetch_deeparchival** : Used to fetch original data. (Calls decoder functions by itself)

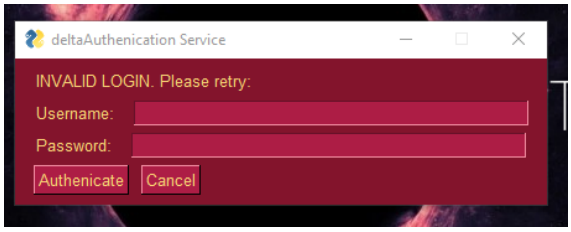
(Please Turn Over)

//Working

Working Explained:

A shortcut has been provided with every installation of DBFA. This shortcut redirects the user to a file called 'run_DBFA.pyw'. This is a console-less Python script that launches a GUI-based authentication dialogue. This accepts a username and password.

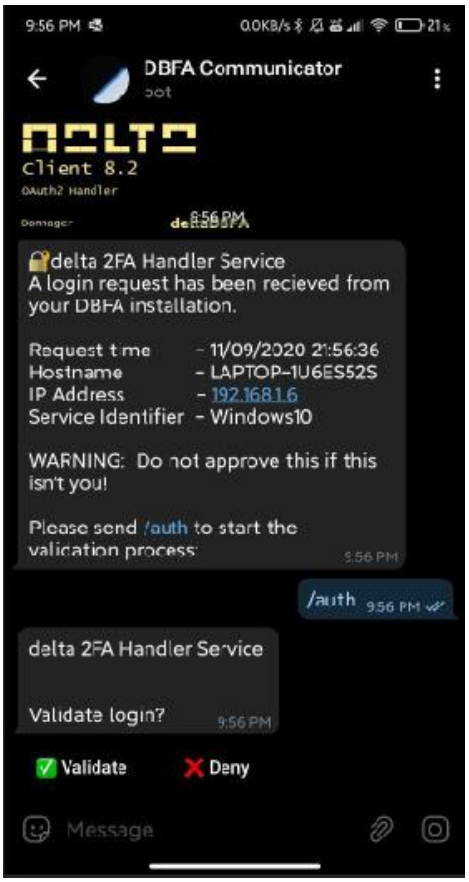
Upon successful login, a confirmation is provided and DBFA is subsequently launched. However, if the credentials entered are found to be incorrect, the same login page is relaunched, this time with a red background and text indicative of invalid creds.



With DBFA 8, we have established synchronous scanning of settings as the code runs. This ensures that settings-controlled operations are executed smoothly.

Once the login is cleared, the main DBFA script is auto-launched, which displays a boot-image using OpenCV, if and only if **'SHOW BOOT IMAGE'** is enabled in DBFA options.

Now, DBFA initiates two-factor-authentication if enabled in settings. This makes used of two very capable libraries, namely, requests and Telegram's BOT API v2. A message is sent to the store owner's Telegram account, requesting authorization to access their installation of DBFA. Leveraging BOT API v2, we are able to provide inline buttons to do so. This is a very safe process and cannot be network-injected easily as all communication is over HTTPS and is processed locally.

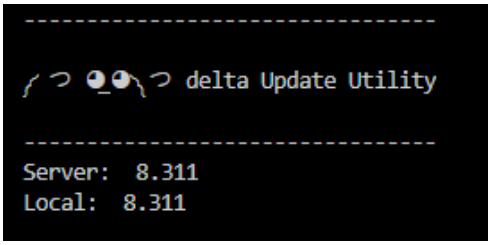


As this part makes use of an indefinite loop to be able to process each input from Telegram (including incorrect message replies to the request), DBFA uses SIGINT (Signal Interrupt) to break out of this loop. If the authentication is denied, DBFA exists gracefully.

NOTE: 2FA and boot screen can be disabled from DBFA Options.

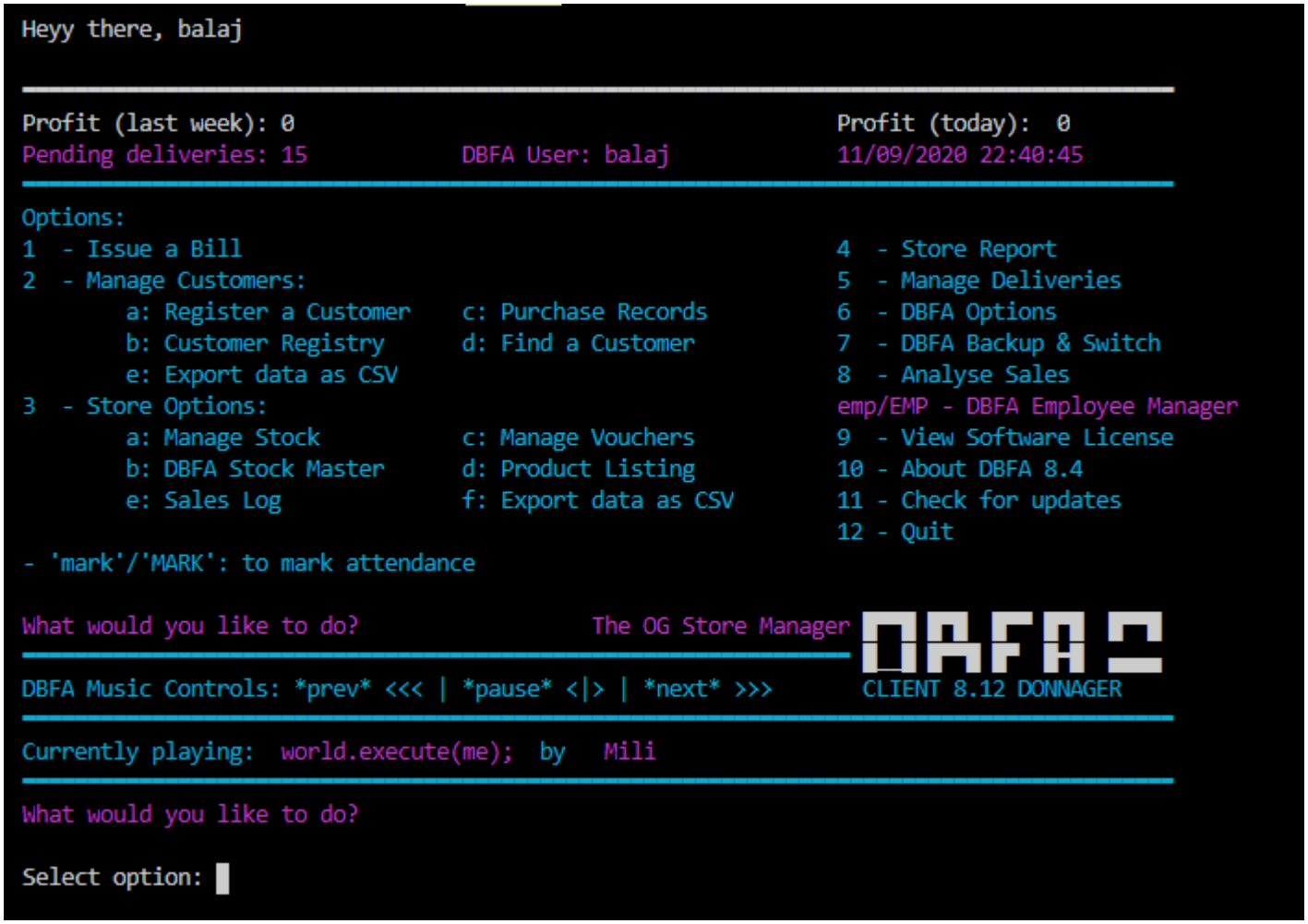
```
- telegram.ext.updater - INFO - Received signal 2 (SIGINT), stopping...
```

Now DBFA Intellisense fetches the latest update ID from DBFA's repository on GitHub's servers and compares it to that of the local installation. If a new update is available, the same is displayed. DBFA can also detect if the local installation is actually the master development copy, and advice the developer to commit the build.



Subsequently the new home menu is displayed, colour coded to facilitate easy readability (colorama) alongside various 'quick glance' snippets of information. The new home menu also shows the currently playing track along-with universal media controls, if enabled from DBFA options. DBFA also fetches user details from the OS itself for the access logs.

The new home menu also houses a beautiful DBFA logo made using ASCII block characters.



NOTE: DBFA Music Controls Service can be disabled from DBFA Options.

Now we'll be discussing each menu option separately.

INVOICING

DBFA asks the user for their customer ID (if registered). If not, the purchase is redirected to customer ID 0 (for unregistered customers). Then DBFA let's the user enter product codes for their purchases. This lasts indefinitely till '0' is entered to proceed. Every product's stock is compared and a 'Not in stock' or 'Low on stock' message is accordingly generated.

An optional option is presented to enter a voucher code. Voucher details are fetched from DBFA's database and accordingly acted upon.

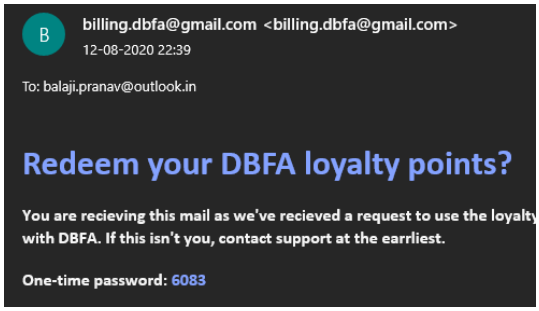
At this point, the user (presumably a store cashier) is given an option to either process the invoice as an in-store purchase, or as a delivery. In case of a delivery, the customer's name, and address (multi-line input) are enquired for and the same is stored in a file.

NOTE: Deliveries do not support loyalty point redemption.

The customer is then redirected to the payments page where they get an option to redeem their loyalty points (only for registered customers).

------- **Loyalty points redemption** ----------

As loyalty points can essentially discount purchases massively, DBFA takes them seriously. Loyalty points can only be applied when an OTP is provided by the customer, as sent on their email ID. The email is sent to the ID registered by the customer during registration with DBFA.



The user is then redirected to the payments page. DBFA offers various payment options, including but not limited to:

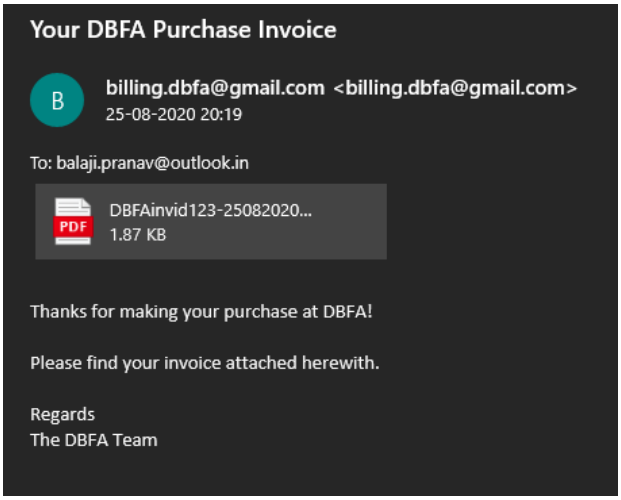
- Debit/ Credit Cards
- UPI
- Digital Wallets
- Cash

NOTE: Deliveries do not support any form of payment except pay-on-delivery.

PAYMENT QR CODES (UPI-only for now): DBFA auto-generated QR codes for UPI payments with the beneficiary's UPI ID, and the amount embedded. It's just a matter of scanning and entering your UPI pin to pay. As simple as that!

We at DBFA take security very seriously. That is why every DBFA purchase is logged in the owner's Telegram account via DBFA's communicator bot. This also ensures that cashiers do not cheat owners by pocketing cash and showing reduced sales.

After every successful purchase, DBFA creates a PDF invoice which is automatically moved to the 'Generated Invoices' directory in its installation location. This data is also passed to the Deep Archival Engine which encodes the data and stores it in a secure database.



Every registered customer instantly receives an e-mail from DBFA with his/ her invoice attached along with from billing.dbfa@gmail.com. This process happens over a secure HTTPS TSL-encrypted connection to Google's servers.

NOTE: Sending customers their invoices can be disabled from DBFA Options.

CUSTOMER MANAGEMENT

This option houses five sub-options:

- REGISTER A CUSTOMER
- VIEW THE CUSTOMER REGISTRY
- VIEW CUSTOMER-SPECIFIC PURCHASE RECORDS
- FIND A CUSTOMER
- EXPORT CUSTOMER DATA AS CSV

REGISTER A CUSTOMER

DBFA asks for the customer's name and e-mail address. Customer IDs are automatically allotted. This is stored across two separate sources with automatic structure-rebuilding capabilities.

```
Select option: 2a
Connecting to server..
Registering customer with ID:  13
Customer Name: delta
Customer's E-mail ID: contactdelta@delta.com
Customer delta registered in store directory
FJHG

Customer ID 13 registered in directory.
---------------------------------------
```

VIEW THE CUSTOMER REGISTRY

This option simply displays customer records as stored with DBFA.

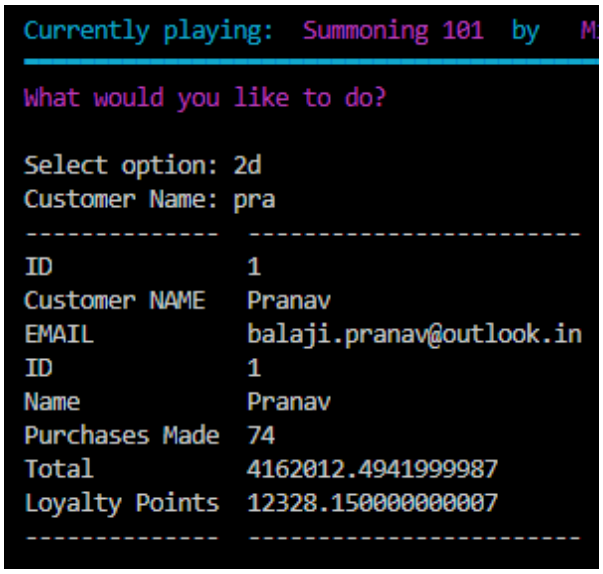
VIEW CUSTOMER-SPECIFIC PURCHASE RECORDS

This option displays all records for a customer. This includes purchase information (total purchases, total amount) as well as the customer ID and e-mail.

```
Customer Purchase Records:
------------------------------------------------------------
   Customer ID         Name  Purchases Made         Total  Loyalty Points
0            0  Unregistered              11  2.830903e+05         2803.87
1            1       Pranav              74  4.162012e+06        12328.15
2            2       Sushant              4  8.555300e+04         1878.25
3            3       Atharv               0  0.000000e+00            0.00
```

FIND A CUSTOMER

DBFA uses advanced self-developed search algorithms with up to 6 different else...if clauses to filter out every attribute in its storage from across sources. DBFA also scans for random wildcard characters in both, the string to b e searched for, and the source.



EXPORT CUSTOMER DATA AS CSV

Customer data from across sources is collected, published into a CSV, saved, and opened in the device's default CSV-viewer, using this option.

STORE MANAGEMENT

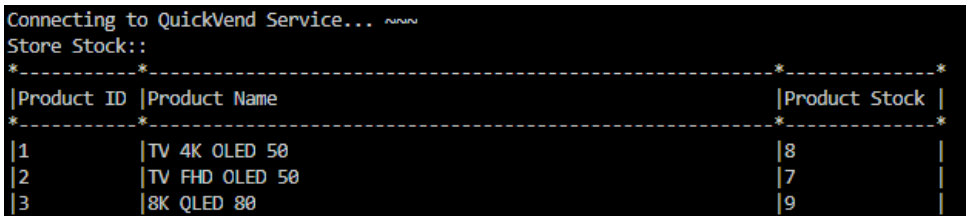
This option houses six sub-options:

- MANAGE STOCK
  - VIEW STOCK, ADD STOCK (INDUVIDUALLY AND FOR THE MASS)
- DBFA STOCK MASTER
  - ORDER STOCK, UPDATE OR FETCH STATUS, VIEW, EDIT AND CONTACT VENDOR (PRODUCT SPECIFIC), MODIFY LOW-STOCK WARNING BAR
- MANAGE VOUCHERS
  - GENERATE AND VIEW GENERATED VOUCHERS
- SEE PRODUCT LISTINGS
- VIEW THE SALES LOG
- EXPORT STORE AND SALES DATA AS CSV

MANAGE STOCK (CURRENT)

This lets the user perform three actions as mentioned below:

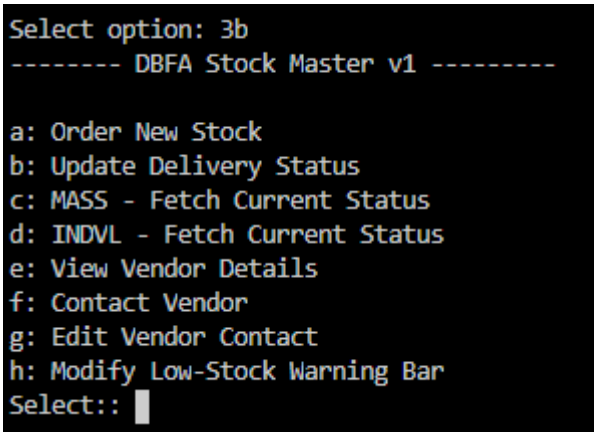
1. VIEW STOCK
   This prints out all stock information.



2. ADD INDUVIDUAL STOCK
   This can be used to add stocks for a single product.
   Negative values can also be provided to reduce stock manually.
3. ENFORCE MASS STOCK
   NOTE: This is to be used for initial store setup purposes .et cetera
   This overwrites all existing stock information and enforces a single value for all stock.

DBFA STOCK MANAGER (ADVANCED)

This brand new option provides a way for stores to order stock, see their delivery status, change the status upon delivery from vendors, manage vendors, change vendors for a product, contact vendors via auto-fetched emails and even alter the 'low stock' warning limit.



MANAGE VOUCHERS

This lets one create DBFA vouchers with limited number of uses and view all generated vouchers.

SEE PRODUCT LISTINGS

Using this option one can view all products a store sells, as available with DBFA.

VIEW THE SALES LOG

Using this option one can view all sales data, including but not limited to the username used for billing, the date and time, products purchased, vouchers and discounts offered, along with the total.

As this is sensitive information, DBFA requires the administrator password to be entered to allow access to the same.

EXPORT STORE AND SALES DATA AS CSV

Store and sales data from across sources is collected, published into a CSV, saved, and opened in the device's default CSV-viewer, using this option.

GENERATE STORE REPORT

This is one of the flagship offerings of DBFA.

This option automatically picks data from multiple sources, analyses it and segregates useful data from the rest, and creates an artfully arranged PDF file.

DBFA's report generator uses data from sources like:

- STOCK REGISTRY
- STOCK ORDERS REGISTRY
- SALES AND PROFITS REGISTRY
- CUSTOMER REGISTRY

We also make use of advanced data-analysis libraries like MatPlotLib and append a neat sales graph into the report as per the latest data streams available.

DBFA Report contents:

- MOST SOLD LISTING(S)
- MOST PROFIT-MAKING LISTING(S)
- TOTAL PROFIT PER LISTING (WITH SALES COUNT)
- CUSTOMER PURCHASES AND LOYALTY POINTS ACCUMULATED
- PRODUCT STOCK (THAT IS YET TO BE RECEIVED)
- DBFA SALES ANALYSIS GRAPH

NOTE: A sample of DBFA's Stock Reporter's work has been attached after this ('//Working') section ends.

DELVERY MANAGER

VIEW PENDING DELIVERIES

A list of pending delivery tickets is displayed.

VIEW PENDING DELIVERIES COUNT

The number of pending deliveries is counted and displayed.

MARK A DELIVERY TICKET AS DELIVERED

All pending deliveries are displayed with their delivery ID distinctly shown.

The user is supposed to enter the delivery ID to mark as 'delivered' here.

DBFA OPTIONS

DBFA Options is an attempt to give the user complete control over how their DBFA installation performs. DBFA keeps its way of operation updated in accordance to the settings set in DBFA Options.

DBFA's options page is beautifully displayed with the status for every setting like this:

1: Display boot image : | ON |
2: Email invoice to registered customers : | ON |
3: Enable DBFA Music Controls (beta): : | ON |
4: Open CSV when exported : | ON |
5: Enable database encryption (under development) : | OFF|
6: Enable DBFA Secure Two-Factor-Authentication : | OFF|
7: Use new DBFA Menu style : | ON |
8: Create DBFA Desktop Shortcut : | Proceed > |
9: Delete customer records : | Proceed > |
10: Delete store records : | Proceed > |
11: Check for updates : | Proceed > |
12: Return to Main Menu : | Proceed > |

DISPLAY BOOT IMAGE

Boot time image can be disabled/ enabled (recommended) here.

EMAIL INVOICE TO REGISTERED CUSTOMERS

This can disable/ enable (recommended) the sending of invoices to registered customers post purchase.

ENABLE DBFA MUSIC CONTROLS (BETA)

This can disable/ enable (recommended) DBFA's music control service.

OPEN CSVs WHEN EXPORTED

This can disable/ enable (recommended) the opening of CSV files once exported.

ENABLE DATABASE ENCRYPTION

DBFA is currently working on database encryption using SQLCipher. This is currently in testing and has not been rolled-out. The option for this is just a temporary placeholder.

ENABLE DBFA SECURE TWO-FACTOR AUTHENTICATION

This lets you enable / disable DBFA's secure 2FA service.

(NOT RECOMMENDED TO DISABLE)

NOTE: Modifying this security setting requires OTP-based authorization from Telegram.

USE NEW DBFA MENU STYLE

This lets you switch between the new DBFA home menu and the older style.

We recommend using the newer style as it puts vertical screen space to better use, whereas the older style had horizontal over-spanning issues.

CREATE DBFA DESKTOP SHORTCUT

This option automatically gets the device's desktop directory's path using OS-level functions and creates a shortcut there.

DELETE ALL CUSTOMER RECORDS/ DELETE ALL STORE RECORDS

This opens an external script which DELETES ALL CUSTOMER/ SALES-STORE RECORDS.

CHECK FOR UPDATES

This option redirects to the already existing updater in the home menu.

(THIS WOULD BE REMOVED IN FUTURE BUILDS OF DBFA, AS THIS OPTION ALREADY EXISTS IN THE HOME MENU'S UPDATER)

These options are persistent across boot-cycles. Settings data is retained even across installations (when DBFA Backup & Switch is used)

DBFA BACKUP & SWITCH

This leverages system-level libraries (like OS and SHUTIL) to copy all the necessary data from your DBFA installation if you want to switch devices or just create a personal backup.

All databases, settings, authentication keys and records are copied into a .zip file and placed in a folder inside DBFA's installation directory.

As this is a sensitive operation, authorization via normal login & 2FA is required for the same.

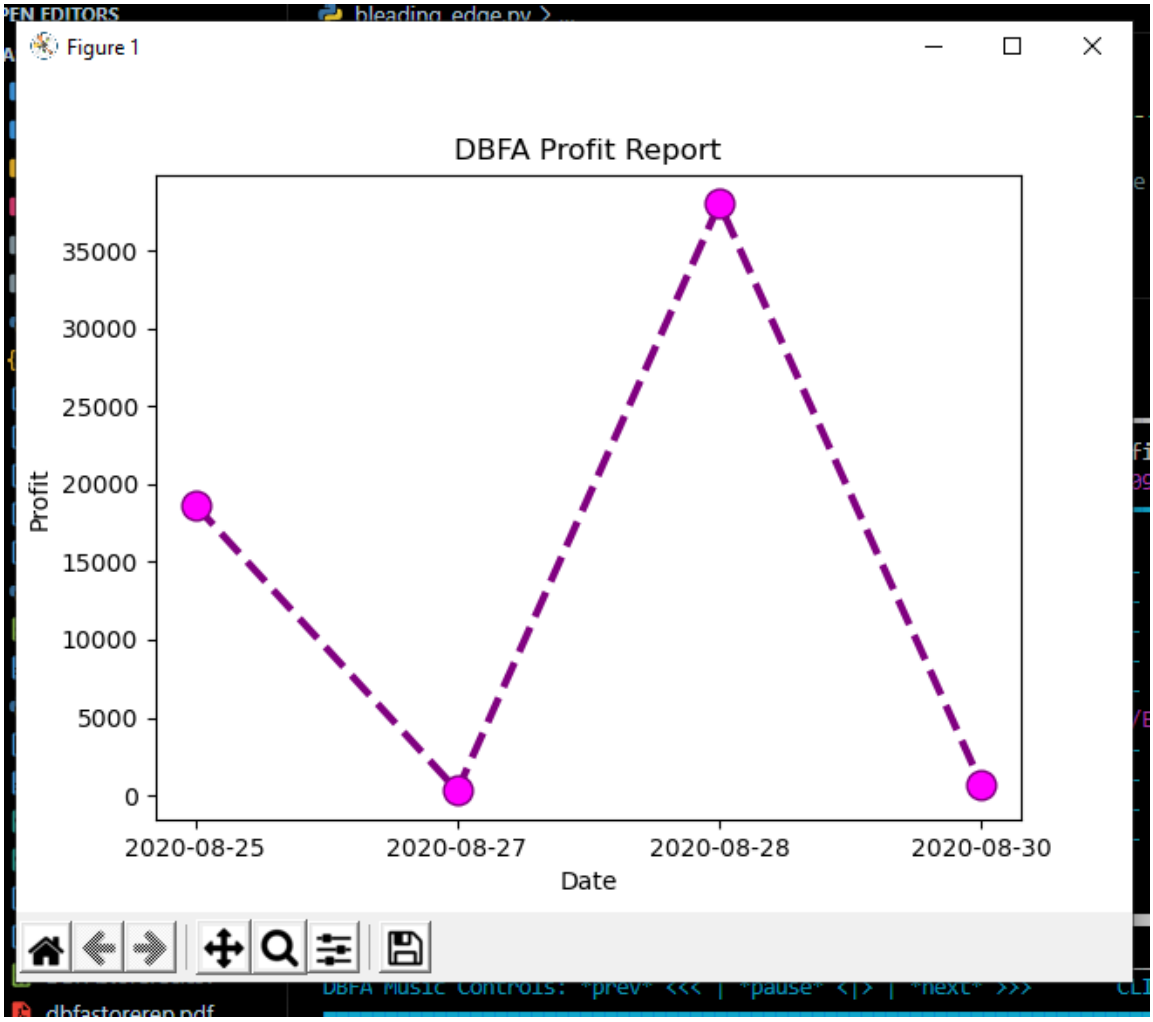
NOTE: 2FA CANNOT BE DISABLED FOR THIS IN ANY CASE.

ANALYSE SALES (SALES PLOTTER)

We make use of advanced data-analysis libraries like Matplotlib to create a neat and artful graphical analysis of all sales till date.

This is opened in a native-to-matplotlib viewer which allows the user to zoom into specific parts of the graph, to adjust the overall dimensions and to save the analysis as an image file.

SAMPLE:



DBFA EMPLOYEE MANAGER

This is a special new feature that just got integrated into DBFA.

DBFA Employee Manager can be triggered on typing “emp”, “EMP”, or other random case variations of the same from the home menu itself.

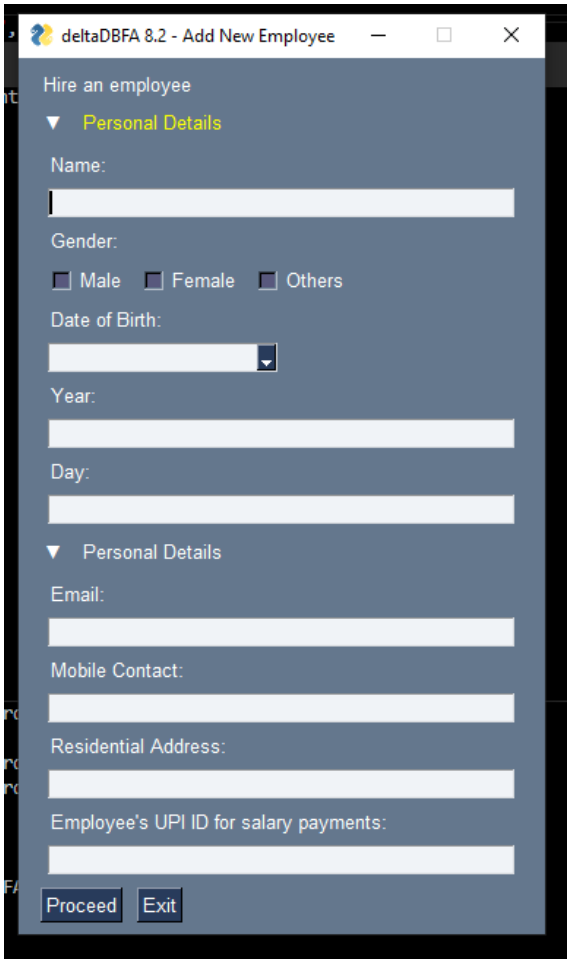
This is a high-level employee manager which provides a variety of functions, such as:

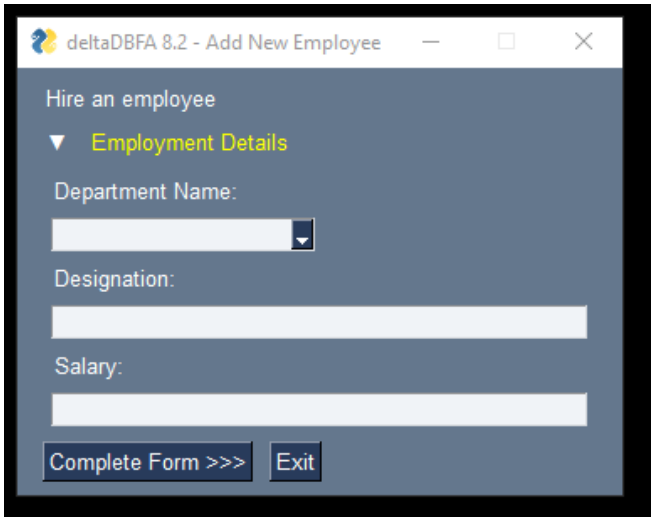
- HIRE AN EMPLOYEE
- VIEW EMPLOYEE RECORDS
- CHANGE EMPLOYEE DETAILS
- FIRE AN EMPLOYEE ʃ•'ᴗ'•ʖ⊃

- MARK ATTENDANCE
- ATTENDANCE RECORDS – ALL
- ATTENDANCE RECORDS - OID-SPECIFIC
- ATTENDANCE RECORDS - ALL (THIS MONTH)
- ATTENDANCE RECORDS - OID-SPECIFIC (THIS MONTH)
- PAY SALARY

HIRE AN EMPLOYEE

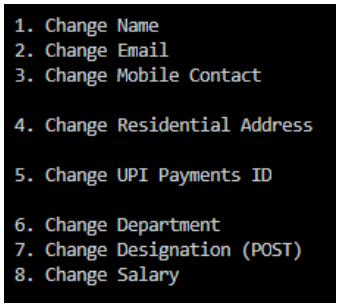
This option presents the user with a GUI-based form to enter details easily. These details are categorized automatically and sent to their respective storage spots.





VIEW EMPLOYEE RECORDS

This displays all employee details as recorded by DBFA.



CHANGE EMPLOYEE DETAILS

This option allows one to change every aspect of an employee's details.

FIRE AN EMPLOYEE ʃ•'ᴗ'•ʖ⊃

This allows one to fire an employee. This is logged in Telegram with high priority.

MARK ATTENDANCE

Employees can mark daily entry/ exit attendance with this option. This can also be called from the main menu by typing "mark" or "MARK" or random case variations of the same.

Attendance can only be recorded twice a day per OiD (employee ID). Any attempts to mark more than 2 per-day attendance would be errored-out gracefully.

ATTENDANCE RECORDS – ALL

This option displays all attendance records for all employees.

ATTENDANCE RECORDS – OID-SPECIFIC

This option displays all attendance records for a specific employee.

ATTENDANCE RECORDS – ALL (THIS MONTH)

This option displays all attendance records for all employees for the past 31 days.

ATTENDANCE RECORDS – OID-SPECIFIC (THIS MONTH)

This option displays all attendance records for a specific employee for the past 31 days.

PAY SALARY

This option fetches the UPI ID of a specific employee ID (OiD; as entered) and creates a UPI payment QR. After payment of salary to the employee, a confirmation is raised to log whether the salary was paid or not.

MARK EMPLOYEE ATTENDANCE

Employees can mark daily entry/ exit attendance with this option. This can be called from the main menu by typing “mark” or “MARK” or random case variations of the same.

Attendance can only be recorded twice a day per OiD (employee ID). Any attempts to mark more than 2 per-day attendance would be errored-out gracefully.

VIEW SOFTWARE LICENSE

DBFA’s licensing details are opened in the device’s default browser via a telegra.ph page.

ABOUT DBFA *(8.52)*

Certain details about the program are displayed and the main changelog for the current build of DBFA is automatically opened in the device’s default browser via a telegra.ph page.

DBFA UPDATER

This is one of the flagship features that DBFA offers. DBFA updater is an OTA-based system, meaning it only downloads the changed parts of an update package, thereby reducing update times and data usage heavily.

DBFA’s updater uses an advanced logic to determine the version ID of the local installation and compares is to the version ID fetched from DBFA’s master repository aboard GitHub’s servers using a HTTPS-secured request.

If a new update is found, DBFA gives the user an option to update then and there itself or update later.

If the user choses to update, DBFA uses system-level commands to run a **git clone** operation inside itself. Only newly changed files are downloaded. Hence this is an OTA update system and only downloads the changed parts of the package.

The user is then instructed to manually copy-replace the update files from a specific folder in DBFA’s installation directory, to the main installation directory.

The user is also directed to an instructions file inside the same directory.

QUIT

This option simply raises **os._exit(0)** by leveraging OS-level functions in the shell to exit DBFA.

(Please Turn Over)

//Quality of life Improvements

AUTOMATED CODE INTEGRITY SCANS ON CRASH

With build 8.52, DBFA now uses advanced methods to scan the code's integrity. The whole code is now encompassed in a *try... except* block which starts the Integrity Scanner when an exception is raised. This tedious process has been simplified greatly to avoid heavy execution time by using *MD5* hashing to detect every single change.

MD5 is a widely used hashing algorithm used in most software distributions to verify whether the downloaded copy is the same as the hosted copy of the code. The benefit of using MD5 hashing is:

- It generates a simple and small key which is easy to store anywhere.
- The hash key changes when there is a difference of even a single character (incl. whitespaces), which results in ultimate accuracy!

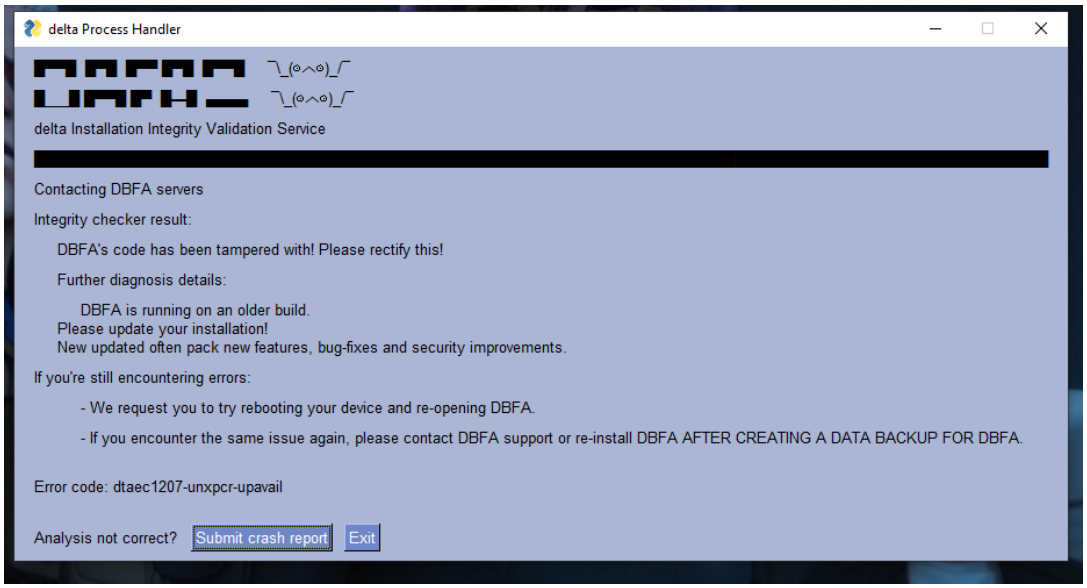
This scan proceeds in multiple steps, as detailed below:

- Crash detection:
  Upon crash detection DBFA uses the *OS module* to fetch the exception message and prints the same, followed by a three second delay to aid easy error readability.

- Live Deployment and Server Hashing:
  DBFA now uses hashing algorithms to generate *MD5* hashes for each installation file critical to the stable running of DBFA. The *requests* module is now used to fetch the *MD5* of the most recent build from DBFA's servers.

- Hash Comparison and Mismatch Detection
  Two sets of hashes, one from the server and one from the live deployment are now compared for each file. Whenever a mismatch is detected, a custom GUI-alert is raised which details on the issue and possible steps to fix it. DBFA now exits, leaving the GUI-alert prompt open.

This way we are able to improve the quality of our code by providing automatic crash detection services.



DBFA INVOICE DEEP ARCHIVAL ENGINE

As you know, DBFA generated PDF invoices upon each purchase. These can be immediately printed and are also automatically mailed to registered customers.

However, maintaining invoice records by storing these PDF invoices can be gruesome when dealing with large sale volumes.

To deal with this, DBFA houses a ***custom-developed deep archival algorithm.*** With this, DBFA can encode the complete data of an invoice into a string, randomize its characters for security and store this in a secure database in a matter of milliseconds!

Whenever a back-dated invoice is required, DBFA can instantly fetch the invoice's encoded key, decode it, and pass the details to the same trusty PDF engine that typically generated its invoices. As always, this invoice is auto moved to a separate directory within DBFA's installation.

This not only helps stores to maintain decades and centuries of invoice information, but also helps them save space compared to storing full-blown PDF invoices!

NOTE: *The deep archival engine can be triggered via the "Invoice Manager" option.*

AUTOMATED DATABASE STRUCTURE REBUILDING

In the modern world, data is everything, everywhere, but is never permanent.

Disk failures, accidental deletions, malwares, viruses, ransomwares, or even internal errors can lead to corrupted databases. DBFA overcomes this by searching for its databases each time the software is booted.

If the required files are found to be missing, the database structure is automatically rebuilt from scratch.

BYPASS PREVENTION SERVICE v2

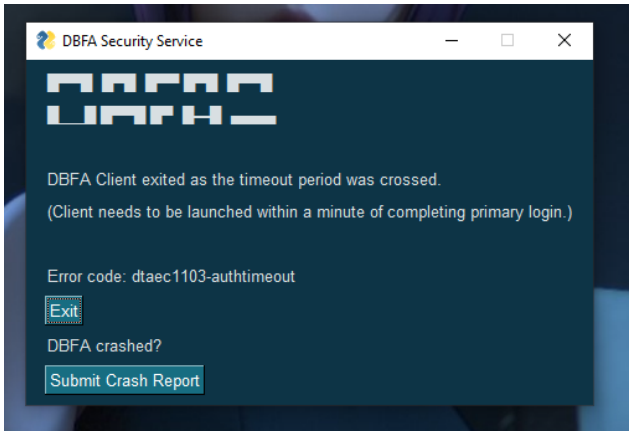
As DBFA is based on Python and uses multiple scripts, one can bypass the main login script by opening multiple scripts in succession directly.

This is taken care of as each successful login generates a login token which is stores in a secure database, without which the main program won't boot. Moreover, a timeout has been implemented in recent builds of the DBFA Client.

This does not apply to DBFA's 2FA system as that uses a completely different system with even more security.

LOGIN-CLIENT-RUN TIMEOUT

To further delay the process in-case a hacker/ cracker gets his/ her ha0nds on a DBFA installation, DBFA uses a 60-second timeout. If the main client is not booted by then, the program exits and displays a GUI-based prompt.



TELEGRAM LOGGING

Every sensitive access, attendance and sale is logged in the store owner's Telegram account via DBFA's communicator bot. These communications are executed over TLS-encrypted HTTPS thereby eliminating any chances of easy external network-based injections.

TWO-FACTOR-AUTHENTICATION

DBFA puts to use its connections with Telegram to provide a completely secure 2FA experience.

The best part about the **updated authentication system aboard DBFA 8.52** is that it uses inline buttons instead of relying on text inputs that can be manipulated.

OTP-BASED 2FA MODIFICATION PROCESS

As DBFA allows the user to disable two-factor-authentication, this is a sensitive act.

DBFA uses OTP-based validation via its secure Telegram bot to process such requests. This variation in methods used to get authorization increases security even more.

DYNAMIC MODULE IMPORTS

DBFA uses more than 50 modules to execute each feature it offers. So, only the 'core-modules' are imported. These 'core-modules' are more than enough for most functions to work. However, when certain features demand specialized modules, they are imported there and then, ensuring that unused imports don't clog up program memory as not every feature is used at once.

INVOICE ID INDEXING

Each DBFA invoice is numbered uniquely, to ease invoice searching at any point in time.

GRACEFUL ERRORS

Instead of erroring-out and exiting the shell, DBFA shows graceful error messages and returns to the home menu.

CROSS-CHECKING OF INPUTS

Features which modify values for a particular asset in databases cross-verify the presence of that particular asset before making changes.

DYNAMIC REFRESHING OF DATA SETS

To avoid data inconsistencies, DBFA refreshes the data sets it operates on with each iteration/ loop.

COMPLETE CONTROL OVER OPERATIONS

With DBFA Options, one can enable/ disable most 'non-core' features.

While disabling these is not recommended, we still give the option. DBFA uses synchronous scanning of its settings files to ensure disabled features do not show up.

PDF INVOICING

Whenever a billing cycle is completed, a PDF invoice is generated with a great many number of details. Each PDF invoice is given an invoice ID and is moved to a folder named 'Generated Invoices', in DBFA's installation directory.

All of this happens silently, in the background while using minimal system resources.

COMPLETE REGISTRY LOGGING

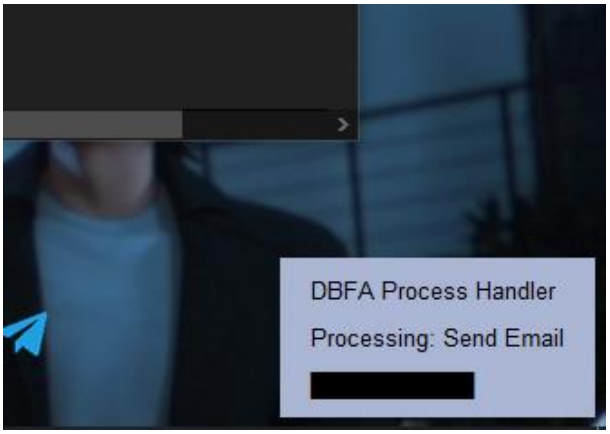
DBFA ensures complete security by logging every action, albeit how small or large.

Logging logs everything including but not limited to sales activities, access logs, feature uses, reading databases .et cetera

GUI PROGRESS BAR FOR VARIOUS PROCESSES

DBFA uses small progress bars during certain process, which is elegantly placed in the rightmost bottom of the user's screen.

DBFA uses specialised functions to automatically fetch the device's screen dimensions to scale the progress bar appropriately.



//Information

HARDWARE USED

- **CPU**

  AMD RYZEN 7 4700U

- **SOLID STATE DRIVE**

  SAMSUNG 950 Evo

- **RAM**

  SK Hynix 8GB DDR4-2600mhz

- **NETWORK**

  TSL-encrypted HTTPS over WLAN

SOFTWARE USED

- Python 3.7.4 > programming language
- Microsoft Windows 10 (x64)
- Microsoft Visual Studio Code > for programming
- Microsoft Windows Terminal > for testing
- Microsoft Office Word 365 > for documentation
- Microsoft GitHub > for code version tracking, hosting and backups
- Microsoft OneDrive > for code backups

<Main Program – DBFA Client>

// The Code 1/10

```
'''
```


```
by deltaonealpha and sushimuncher

package dbfafartingspider
* The program FartingSpider implements an application that
* houses a solution for complete store management.
*
* @author deltaonealpha
'''
#vs

import traceback

try:
    import getpass, time, pathlib, sqlite3, sys, os #sys, os for system-level ops
    from tabularprint import table
    from tqdm import tqdm
    import webbrowser

    # Credits to XanderMJ (https://github.com/XanderMJ/spotilib) for Spotify controls
    import spotilib

    from SwSpotify import spotify


    filedel = open('./DBFAdeliveries.txt', 'a+')
    filedel.close()

    from email.mime.text import MIMEText
    from email.mime.multipart import MIMEMultipart
    import math, random
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders

    from reportlab.pdfbase import pdfmetrics
    from reportlab.pdfbase.ttfonts import TTFont
    pdfmetrics.registerFont(TTFont('MiLanProVF', r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\MiLanProVF.ttf'))
    from reportlab.pdfgen import canvas
    from reportlab.lib.pagesizes import A4
    from reportlab.platypus import SimpleDocTemplate, Paragraph
    from reportlab.lib.styles import getSampleStyleSheet
    from reportlab.lib.units import cm
    from reportlab.lib.enums import TA_JUSTIFY
    from reportlab.lib.pagesizes import letter
    from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
    from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
    from reportlab.lib.units import inch
    from tabulate import tabulate
    from reportlab.lib import colors
    from reportlab.lib.pagesizes import letter
```

```
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
    import requests
    import cv2


    def settingscommonfetch(SettingsType):
        import sqlite3
        settings = sqlite3.connect(r'dbfasettings.db')
        settingsx = settings.cursor()
        settingsx.execute(("SELECT Value from settings WHERE SettingsType = ?"),
(SettingsType,))
        settingsfetch = (settingsx.fetchall()[0][0])
        return settingsfetch

    def settingsmodifier(SettingsType, NewValue):
        import sqlite3
        settings = sqlite3.connect(r'dbfasettings.db')
        settingsx = settings.cursor()
        settingsx.execute(("UPDATE settings SET Value = ? WHERE SettingsType = ?"),
(NewValue, SettingsType))
        settings.commit()



    logox = ('''
    _____   ____    ____  ____  _____
    / /  // / /  \\ /___  /__||  |--//
    / /  // / /===| ///// ////||    //
    /_/__// /_/__ / /     /    ||   //
        ''')

    global cccheck

    #auth verification
    if os.path.exists(r'userblock.zconf'):
        print("Verifying login...")
        print(" ")
    if os.path.exists(r'userblock.txt'):
        userblock = open(r"userblock.txt","r") #Opening / creating (if it doesn't
exist already) the .txt record file
        redflag = 1
    else:
        redflag = 0
    if os.path.exists(r'DBFA.zconf'):
        pass
    else:
        print("Getting the database online.......")
        time.sleep(0.2)
        con = sqlite3.connect(r'DBFA.db')
        print("Rebuilding database..")
        c = con.cursor()
        #c.execute("DROP TABLE cust;")
        c.execute("""CREATE TABLE IF NOT EXISTS cust
            (custt INTEGER PRIMARY KEY,
            custname TEXT,
            email TEXT);""")
    try:
        if os.path.exists(r'userblock.txt'):
            os.remove(r'userblock.txt')
        if os.path.exists(r'userblock.zconf'):
```

```
            os.remove(r'userblock.zconf')
    except PermissionError:
        pass



    #Stock Order Manager
    xon = sqlite3.connect(r'DBFA_vend.db')
    xbr7 = xon.cursor()
    xbr7.execute("""CREATE TABLE IF NOT EXISTS stock
        (prodid INTEGER PRIMARY KEY,
        prodname CHAR,
        ordqty VARCHAR(500),
        delivered INT,
        delstat CHAR,
        vendor CHAR,
        vendcont CHAR,
        lowstock INT);""")
    xon = sqlite3.connect(r'DBFA_vend.db')
    xbr7 = xon.cursor()
    if os.path.exists(r'DBFA_vend.db'):
        pass
    else:
        xbr7.execute("""CREATE TABLE IF NOT EXISTS stock
        (prodid INTEGER PRIMARY KEY,
        prodname CHAR,
        ordqty VARCHAR(500),
        delivered INT,
        delstat CHAR,
        vendor CHAR,
        vendcont CHAR
        lowstock INT);""")

    st = sqlite3.connect(r'DBFA_vend.db')
    stx = st.cursor()
    stockq = """UPDATE stock SET delstat = ? WHERE ordqty = 0"""
    qans = ("DELIVERED", )
    stx.execute(stockq, qans)
    stockq = """UPDATE stock SET delstat = ? WHERE ordqty != 0"""
    qans = ("TBD", )
    stx.execute(stockq, qans)
    st.commit()

    #NEW Sales Report v2 DB Logger
    sales = sqlite3.connect(r'dbfasales.db')
    salesx = sales.cursor()
    if os.path.exists(r'dbfasales.db'):
        pass
    else:
        salesx.execute("""CREATE TABLE IF NOT EXISTS sales
        (sno INT PRIMARY KEY,
        custid INT,
        prodid INT,
        net INT,
        prof INT,
        date DATE);""")
        sales.commit()
        print("Table restructured! ")


```

```
176
177     class HiddenPrints:
178                 def __enter__(self):
179                     self._original_stdout = sys.stdout
180                     sys.stdout = open(os.devnull, 'w')
181                 def __exit__(self, exc_type, exc_val, exc_tb):
182                     sys.stdout.close()
183                     sys.stdout = self._original_stdout
184                     print()
185
186     # TG Communicator
187     def telegram_bot_sendtext(bot_message):
188         with HiddenPrints():
189             bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
190             bot_chatID = '680917769'
191             send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?
chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
192             response = requests.get(send_text)
193             return response.json()
194
195     def getOTP():
196         global otp
197         digits = "0123456789"
198         otp = ""
199         otp += (digits[math.floor(random.random() * 10)])
200         otp += (digits[math.floor(random.random() * 10)])
201         otp += (digits[math.floor(random.random() * 10)])
202         otp += (digits[math.floor(random.random() * 10)])
203
204
205     # DBFA Logo Printer
206     def logoprintxrt():
207                 print("        ___ ______ ___   _____________    _____________
_______")
208                 time.sleep(0.05)
209                 print("       /  /_______/  /  /  /_______/  /  /  /________/    /
/_/ /")
210                 time.sleep(0.05)
211                 print("      /  /       /  /  /  /       /  /  /  /            /  /
/ /")
212                 time.sleep(0.05)
213                 print("     /  /       /  /  /  /_______/  /  /  /  CLI        /  /
  / /")
214                 time.sleep(0.05)
215                 print("    /  /       /  /  / // // // // /  /  /_________    /
/____/ /")
216                 time.sleep(0.05)
217                 print("   /  /       /  /  /  /-------/  /  /  /_________/   /
/_____/ /")
218                 time.sleep(0.05)
219                 print("  /  /       /  /  /  /       /  /  /  /            /  /
/ /")
220                 time.sleep(0.05)
221                 print(" /  /_______/  /  /  /______ /  /  /  /            /  /
  / /")
222                 time.sleep(0.05)
223                 print("/__/_______/__/  /__/_______/__/  /__/            /__/
/__/")
224                 print(" ")
225                 print(" ")
```

```
226
227
228
229     # Database section
230     # Stock Records Master DB
231     ssh = sqlite3.connect(r'DBFA_handler.db')
232     ssh7 = ssh.cursor()
233     #c.execute("DROP TABLE cust;")
234     ssh7.execute("""CREATE TABLE IF NOT EXISTS sshandler
235         (prodid INTEGER,
236         prodname CHAR,
237         ssstock INTEGER);""")
238     ssh = sqlite3.connect('DBFA_handler.db')
239     ssh7 = ssh.cursor()
240     if os.path.exists(r'DBFA_handler.db'):
241         pass
242     else:
243         ssh7.execute("""CREATE TABLE IF NOT EXISTS sshandler
244             (prodid INTEGER,
245             prodname CHAR,
246             ssstock INTEGER);""")
247
248
249     # Invoice Master DB
250     inmas = sqlite3.connect(r'invoicemaster.db')
251     inmascur = inmas.cursor()
252     #c.execute("DROP TABLE cust;")
253     inmascur.execute("""CREATE TABLE IF NOT EXISTS inmas
254         (indid INTEGER);""")
255     inmas = sqlite3.connect('invoicemaster.db')
256     inmascur = inmas.cursor()
257     if os.path.exists(r'invoicemaster.db'):
258         pass
259     else:
260         inmascur.execute("""CREATE TABLE IF NOT EXISTS inmas
261         (indid INTEGER);""")
262
263
264     # Voucher Records Master DB
265     isol = sqlite3.connect(r'cponmgmtsys.db')
266     isolx = isol.cursor()
267     isolx.execute("""CREATE TABLE IF NOT EXISTS cponmaster
268         (cponid CHAR PRIMARY KEY,
269         cponlim INTEGER,
270         cponvalue INTEGER,
271         cpondtb DATE);""")
272     isol = sqlite3.connect('cponmgmtsys.db')
273     isolx = isol.cursor()
274     if os.path.exists(r'cponmgmtsys.db'):
275         pass
276     else:
277         isolx.execute("""CREATE TABLE IF NOT EXISTS cponmaster
278             (cponid CHAR PRIMARY KEY,
279             cponlim INTEGER,
280             cponvalue INTEGER,
281             cpondtb DATE);""")
282
283
284     # Customer Records Master DB
285     xon = sqlite3.connect(r'DBFA_CUSTCC.db')
```

```
    xbr7 = xon.cursor()
    xbr7.execute("""CREATE TABLE IF NOT EXISTS custcc
        (custid INTEGER PRIMARY KEY,
        custname VARCHAR(500),
        purchasecount INTEGER,
        ptotalx INTEGER,
        points INTEGER);""")
    xon = sqlite3.connect('DBFA_CUSTCC.db')
    xbr7 = xon.cursor()
    if os.path.exists(r'DBFA_CUSTCC.db'):
        pass
    else:
        xbr7.execute("""CREATE TABLE IF NOT EXISTS custcc
        (custid INTEGER PRIMARY KEY,
        custname VARCHAR(500),
        purchasecount INTEGER,
        ptotalx INTEGER,
        points INTEGER);""")



    conn = sqlite3.connect('DBFA.db')
    if os.path.exists(r'DBFA.db'):
        pass
    else:
        xbr7.execute("""CREATE TABLE IF NOT EXISTS custcc
            (custt INTEGER PRIMARY KEY,
            custname VARCHAR(500),
            purchasecount INTEGER,
            ptotalx INTEGER);""")



    # Report Record Master
    rec = sqlite3.connect(r'recmaster.db')
    recx = rec.cursor()
    recx.execute("""CREATE TABLE IF NOT EXISTS recmasterx
        (prodid INTEGER PRIMARY KEY,
        prodname CHAR,
        prodprofit INTEGER,
        prodsales INTEGER,
        netprof INTEGER);""")
    rec = sqlite3.connect('recmaster.db')
    recx = rec.cursor()
    if os.path.exists(r'recmaster.db'):
        pass
    else:
        recx.execute("""CREATE TABLE IF NOT EXISTS recmasterx
        (prodid INTEGER PRIMARY KEY,
        prodname CHAR,
        prodprofit INTEGER,
        prodsales INTEGER,
        netprof INTEGER);""")
        namiex = ["TV 4K OLED 50", "TV FHD OLED 50", "8K QLED 80", "Redmi K20 PRO",
"Redmi K20", "Redmi Note 9 PRO", "POCOPHONE F1", "Mi MIX ALPHA", "Wireless
Headphones", "Noise-Cancelling Wireless Headphones", "Essentials Headphones",
"Gaming Headphones", "Truly-Wireless Eadphones", "Neckband-Style Wireless
Earphones", "Essentials Earphones", "Gaming Earphones", "30W Bluetooth Speakers",
"20W Bluetooth Speakers", "Essentials Bluetooth Speaker", "BOSE QC35", "Essentials
```

```
Home Theatre", "Wired Speaker - 5.1", "Essentials Wired Speaker - STEREO", "Tactical Series Power Bank 30000mah", "Essentials Power Bank 10000mah", "Essentials Mouse", "Logitech G604 LightSpeed Wireless", "Tactical Essentials Keyboard", "DROP GS21k RGB Gaming Keyboard", "Polowski Tactical Flashlight", "OneFiber Wi-Fi Router AX7", "Mijia Mesh Wi-Fi Router", "lapcare 45W Laptop Adapter", "lapcare 60W Laptop Adapter","Spigen Phone Case(s)", "Essentials Phone Charger 15W", "HyperPower Type-C Gallium-Nitride Charger 120W", "ASUS Zephyrus G4 Gaming Laptop", "DELL XPS 5 Content Creator's Laptop", "Hewlett-Packard Essential's Student's Laptop (Chromebook)"]
341         profitx = [2000, 4500, 5700, 2000, 2100, 1470, 300, 11000, 400, 2000, 100, 370, 450, 120, 50, 275, 649, 140, 50, 1050, 978, 150, 100, 320, 98, 75, 170, 60, 275, 90, 210, 780, 50, 35, 50, 30, 100, 8000, 9000, 1790]
342         profitmarker = 0
343         for crrt in namiex:
344             profvalue = profitx[profitmarker]
345             gg = (namiex.index(crrt)) + 1
346             strx = "insert into recmasterx(prodid, prodname, prodprofit, prodsales, netprof) values(?, ?, ?, ?, ?)"
347             io = (gg, crrt, profvalue, 0, 0)
348             profitmarker += 1
349             recx.execute(strx, io)
350             rec.commit()
351             print("Added record: ", crrt)
352
353
354
355
356
357
358
359
360
361     # Functions::
362
363     # DBFA Stock Master v1
364
365
366     def orderstock(prodid, qty):
367         st = sqlite3.connect(r'DBFA_vend.db')
368         stx = st.cursor()
369         stockq = """UPDATE stock SET ordqty = ordqty + ? WHERE prodid = ?"""
370         qans = (qty, prodid)
371         stx.execute(stockq, qans)
372         st.commit()
373         print("Recieved order for product", prodid, "; qty", qty)
374         st = sqlite3.connect(r'DBFA_vend.db')
375         stx = st.cursor()
376         stockq = """UPDATE stock SET delstat = ? WHERE ordqty = 0"""
377         qans = ("DELIVERED", )
378         stx.execute(stockq, qans)
379         stockq = """UPDATE stock SET delstat = ? WHERE ordqty != 0"""
380         qans = ("TBD", )
381         stx.execute(stockq, qans)
382         st.commit()
383
384
385     def delivered(qty, prodid):
386         st = sqlite3.connect(r'DBFA_vend.db')
387         stx = st.cursor()
388         stockq = """SELECT ordqty FROM stock WHERE prodid = ?"""
389         qans = (prodid, )
```

```
390         stx.execute(stockq, qans)
391         st.commit()
392         aaa = (int(stx.fetchall()[0][0]))
393         if aaa >= qty:
394             st = sqlite3.connect(r'DBFA_vend.db')
395             stx = st.cursor()
396             xstockq = """UPDATE stock SET ordqty = ordqty - ? WHERE prodid = ?"""
397             xqans = (qty, prodid, )
398             stx.execute(xstockq, xqans)
399             xstockq = """UPDATE stock SET delivered = delivered + ? WHERE prodid =
?"""
400             xqans = (qty, prodid, )
401             stx.execute(xstockq, xqans)
402             st.commit()
403             print("Quantity ", qty, "recieved.")
404         else:
405             print("Cannot set status for qty:", qty, "for prod:", prodid, "as the
ordered qty is lower than the delivered qty being set.")
406         st = sqlite3.connect(r'DBFA_vend.db')
407         stx = st.cursor()
408         stockq = """UPDATE stock SET delstat = ? WHERE ordqty = 0"""
409         qans = ("DELIVERED", )
410         stx.execute(stockq, qans)
411         stockq = """UPDATE stock SET delstat = ? WHERE ordqty != 0"""
412         qans = ("TBD", )
413         stx.execute(stockq, qans)
414         st.commit()
415
416
417     def delstatmass():
418         isol = sqlite3.connect(r'DBFA_vend.db')
419         isolx = isol.cursor()
420         print("\n\nProduct stock yet to be recieved: \n")
421         isolx = isol.cursor()
422         isolx.execute(("SELECT * from stock WHERE delstat = ?"), ("TBD", ))
423         rows = isolx.fetchall()
424         col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor",
"Vendor", "Vendor Contact", "Lowstock Bar")
425         table(col_labels, rows)
426         print("\n\nProduct stock recieved: \n")
427         isolx.execute(("SELECT * from stock WHERE delstat = ?"), ("DELIVERED", ))
428         rows = isolx.fetchall()
429         col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor",
"Vendor", "Vendor Contact", "Lowstock Bar")
430         table(col_labels, rows)
431
432     def delstatindvl(prodid):
433         isol = sqlite3.connect(r'DBFA_vend.db')
434         isolx = isol.cursor()
435         print("\n\nDetails for product ID", prodid, ": \n")
436         isolx.execute(("SELECT * from stock WHERE prodid = ?"), (prodid, ))
437         rows = isolx.fetchall()
438         if rows in ([], (), "", " ", None):
439             print("-------------------------------------------------------------")
440             print("| Entered product ID could not be located in DBFA's records |")
441             print("-------------------------------------------------------------
\n\n")
442         else:
443             col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor",
"Vendor", "Vendor Contact", "Lowstock Bar")
```

```
            table(col_labels, rows)


    def vendorfetch(prodid):
        isol = sqlite3.connect(r'DBFA_vend.db')
        isolx = isol.cursor()
        print("\n\nVendor details for product ID", prodid, ":")
        isolx.execute(("SELECT vendor from stock WHERE prodid = ?"), (prodid, ))
        rows = isolx.fetchall()
        if rows in ([], (), "", " ", None):
            print("-------------------------------------------------------------")
            print("| Entered product ID could not be located in DBFA's records |")
            print("-------------------------------------------------------------\n")
        else:
            print("Vendor:", rows[0][0])


    def vendorcontact(prodid):
        isol = sqlite3.connect(r'DBFA_vend.db')
        isolx = isol.cursor()
        print("\n\nVendor contact for product ID", prodid, ":")
        isolx.execute(("SELECT vendcont from stock WHERE prodid = ?"), (prodid, ))
        rows = isolx.fetchall()
        if rows in ([], (), "", " ", None):
            print("-------------------------------------------------------------")
            print("| Entered product ID could not be located in DBFA's records |")
            print("-------------------------------------------------------------\n")
        else:
            print("Vendor contact:", rows[0][0])
            confac = input("Contact vendor? (y/n): ")
            if confac == "y":
                import webbrowser
                webbrowser.open(('mailto:'+'%s'%rows[0][0]), new=1)
            elif confac == "n":
                pass
            else:
                print("Invalid option entered. ")
                vendorcontact(prodid)


    def lowbarmodif(prodid):
        isol = sqlite3.connect(r'DBFA_vend.db')
        isolx = isol.cursor()
        print("\n\nLow-stock bar for product ID", prodid, ":")
        isolx.execute(("SELECT lowstock from stock WHERE prodid = ?"), (prodid, ))
        rows = isolx.fetchall()
        if rows in ([], (), "", " ", None):
            print("-------------------------------------------------------------")
            print("| Entered product ID could not be located in DBFA's records |")
            print("-------------------------------------------------------------\n")
        else:
            print("Current low-stock bar:", rows[0][0])
            modifier = int(input("Enter the new limit: "))
            if modifier <=0:
                print("Cannot set a negative/ null value as low-stock warning limit!
")
                print("Try again: ")
                time.sleep(1)
                lowbarmodif(prodid)
                print("Modified. New limit:", rows[0][0])
```

```
            isol = sqlite3.connect(r'DBFA_vend.db')
            isolx = isol.cursor()
            isolx.execute(("UPDATE stock SET lowstock = ? WHERE prodid = ?"), (modifier, prodid, ))
            isol.commit()
            rows = isolx.fetchall()
            isolx.execute(("SELECT lowstock from stock WHERE prodid = ?"), (prodid, ))
            rows = isolx.fetchall()
            print("New limit set: ", rows[0][0])
            time.sleep(1)


    def lowbar(prodid):
        isol = sqlite3.connect(r'DBFA_vend.db')
        isolx = isol.cursor()
        isolx.execute(("SELECT lowstock from stock WHERE prodid = ?"), (prodid, ))
        rows = isolx.fetchall()
        print(rows[0][0])


    # Report Stock Fetcher
    namiex = ["TV 4K OLED 50", "TV FHD OLED 50", "8K QLED 80", "Redmi K20 PRO", "Redmi K20", "Redmi Note 9 PRO", "POCOPHONE F1", "Mi MIX ALPHA", "Wireless Headphones", "Noise-Cancelling Wireless Headphones", "Essentials Headphones", "Gaming Headphones", "Truly-Wireless Eadphones", "Neckband-Style Wireless Earphones", "Essentials Earphones", "Gaming Earphones", "30W Bluetooth Speakers", "20W Bluetooth Speakers", "Essentials Bluetooth Speaker", "BOSE QC35", "Essentials Home Theatre", "Wired Speaker - 5.1", "Essentials Wired Speaker - STEREO", "Tactical Series Power Bank 30000mah", "Essentials Power Bank 10000mah", "Essentials Mouse", "Logitech G604 LightSpeed Wireless", "Tactical Essentials Keyboard", "DROP GS21k RGB Gaming Keyboard", "Polowski Tactical Flashlight", "OneFiber Wi-Fi Router AX7", "Mijia Mesh Wi-Fi Router", "lapcare 45W Laptop Adapter", "lapcare 60W Laptop Adapter","Spigen Phone Case(s)", "Essentials Phone Charger 15W", "HyperPower Type-C Gallium-Nitride Charger 120W", "ASUS Zephyrus G4 Gaming Laptop", "DELL XPS 5 Content Creator's Laptop", "Hewlett-Packard Essential's Student's Laptop (Chromebook)"]
    def repstockfetch():
        global tabarter
        ssh = sqlite3.connect('DBFA_handler.db')
        ssh.row_factory = lambda cursor, row: row[0]
        ssh7 = ssh.cursor()
        ssh7.execute("SELECT DISTINCT prodid FROM sshandler WHERE ssstock < 5;")
        axrows = ssh7.fetchall()
        tabarter = []
        for i in axrows:
            ssh7.execute("SELECT DISTINCT ssstock FROM sshandler WHERE prodid = ?;", (i,))
            a = [('%s'%(i)), namiex[i], "Stock Remaining: ", '%s'%(ssh7.fetchall()[0])]
            tabarter.append(a)
        if tabarter == []:
            tabarter.append("--")

    def repdatafetch():
        global charter, rows
        charter = ""
        charter += "DBFA STORE REPORT\n"
        rec = sqlite3.connect(r'recmaster.db')
        recx = rec.cursor()
        charter += "\nSales data:: \n\n"
```

```
546         time.sleep(1)
547         recx.execute("SELECT DISTINCT prodid, prodname, prodprofit, prodsales,
netprof FROM recmasterx")
548         rows = recx.fetchall()
549         '''
550         for row in rows:
551             print(row)
552         '''
553         ll = [("P.ID","Prod. Name","Profit P.U.","Qty. Sold","Net Profit")]
554         rows = ll + rows
555         time.sleep(0.1)
556         #print(" ")
557 
558 
559     def repupdate(prodid):
560         rec = sqlite3.connect(r'recmaster.db')
561         recx = rec.cursor()
562         # hidden prints here ig
563         updatexr = ("UPDATE recmasterx SET prodsales = prodsales + 1 WHERE prodid =
?")
564         updatexxr = ("UPDATE recmasterx SET netprof = prodsales*prodprofit WHERE
prodid = ?")
565         indicator = (prodid, )
566         recx.execute(updatexr, indicator)
567         recx.execute(updatexxr, indicator)
568         rec.commit()
569         recx.close()
570 
571 
572     # Feature not released
573     # Invoice Master Record Maintainer
574     def inmaintainer():
575         inmas = sqlite3.connect('invoicemaster.db')
576         inmascur = inmas.cursor()
577         updatetrtt = """UPDATE inmas SET indid = indid + 1"""
578         inmascur.execute(updatetrtt)
579         inmas.commit()
580         inmascur.close()
581         #time.sleep(1)
582         #toaster.show_toast+("DBFA QuickVend Service - Background Sync", duration =
0.4)
583 
584     def infetch():
585         global inval
586         inmas = sqlite3.connect('invoicemaster.db')
587         inmascur = inmas.cursor()
588         inmascur.execute("SELECT DISTINCT indid FROM inmas")
589         rows = inmascur.fetchall()
590         inval = int(rows[0][0])
591 
592 
593     # Stock System
594     # Mass Stock Allocator
595     def massmaintainer(inxstock):  #defining a function to input data into the SQL
database's table
596         try:
597             idList =
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
,32,33,34,35,36,37,38,39,40]
598             ssxconn = sqlite3.connect(r"DBFA_handler.db")
```

```
            ssxsql = 'DELETE FROM sshandler'
            ssxcur = ssxconn.cursor()
            ssxcur.execute(ssxsql)
            ssxconn.commit()
        except sqlite3.Error as error:
            print("Failed to flush multiple records from sqlite table", error)

        ssh = sqlite3.connect(r'DBFA_handler.db')
        ssh7 = ssh.cursor()
        namiex = ["TV 4K OLED 50", "TV FHD OLED 50", "8K QLED 80", "Redmi K20 PRO",
"Redmi K20", "Redmi Note 9 PRO", "POCOPHONE F1", "Mi MIX ALPHA", "Wireless
Headphones", "Noise-Cancelling Wireless Headphones", "Essentials Headphones",
"Gaming Headphones", "Truly-Wireless Eadphones", "Neckband-Style Wireless
Earphones", "Essentials Earphones", "Gaming Earphones", "30W Bluetooth Speakers",
"20W Bluetooth Speakers", "Essentials Bluetooth Speaker", "BOSE QC35", "Essentials
Home Theatre", "Wired Speaker - 5.1", "Essentials Wired Speaker - STEREO", "Tactical
Series Power Bank 30000mah", "Essentials Power Bank 10000mah", "Essentials Mouse",
"Logitech G604 LightSpeed Wireless", "Tactical Essentials Keyboard", "DROP GS21k RGB
Gaming Keyboard", "Polowski Tactical Flashlight", "OneFiber Wi-Fi Router AX7",
"Mijia Mesh Wi-Fi Router", "lapcare 45W Laptop Adapter", "lapcare 60W Laptop
Adapter","Spigen Phone Case(s)", "Essentials Phone Charger 15W", "HyperPower Type-C
Gallium-Nitride Charger 120W", "ASUS Zephyrus G4 Gaming Laptop", "DELL XPS 5 Content
Creator's Laptop", "Hewlett-Packard Essential's Student's Laptop (Chromebook)"]
        for crrt in namiex:
            gg = (namiex.index(crrt)) + 1
            str = "insert into sshandler(prodid, prodname, ssstock) values(?, ?, ?)"
            strxx = (gg, crrt, inxstock,)
            ssh7.execute(str, strxx)
            ssh.commit()
        ssh7.close()
        time.sleep(1)
        print("DBFA QuickVend service - Stock universally enforced to", inxstock)
        time.sleep(1)


    # Induvidual Stock Allocator
    def ssxupdatescript(inxssincremental, prodid):
        ssh = sqlite3.connect('DBFA_handler.db')
        ssh7 = ssh.cursor()
        updatetr = """UPDATE sshandler SET ssstock = ssstock + ? WHERE prodid = ?"""
        xrindicator = (inxssincremental, prodid)
        ssh7.execute(updatetr, xrindicator)
        ssh.commit()
        ssh7.close()
        time.sleep(1)
        print("DBFA QuickVend Service - Stock added for", prodid, "as",
inxssincremental)

    # Stock Data Fetcher
    def ssxsuperfetch():
        ssh = sqlite3.connect('DBFA_handler.db')
        ssh7 = ssh.cursor()
        print("Connecting to QuickVend Service... ~~~")  #SQL connection prompt
        print("Store Stock:: ")
        time.sleep(1.5)
        #Re-writing to refresh connection
        ssh7 = ssh.cursor()
        ssh7.execute("SELECT DISTINCT prodid, prodname, ssstock FROM sshandler")
        rows = ssh7.fetchall()
        col_labels = ("Product ID", "Product Name", "Product Stock")
```

```
        table(col_labels, rows)


    # Purchase-time Stock Handler
    def ssxstockmaintainer(prodid):
        ssh = sqlite3.connect('DBFA_handler.db')
        ssh7 = ssh.cursor()
        updatetrtt = """UPDATE sshandler SET ssstock = ssstock - 1 WHERE prodid =
?"""
        xrindicatortt = (prodid,)
        ssh7.execute(updatetrtt, xrindicatortt)
        ssh.commit()
        ssh7.close()
        #time.sleep(1)
        #toaster.show_toast("DBFA QuickVend Service - Background Sync", duration =
0.4)

    # Induvidual Stock Fetcher
    def ssxstockmaster(prodid):
        global ssxvarscheck
        ssxvarscheck = 0
        ssh = sqlite3.connect('DBFA_handler.db')
        ssh.row_factory = lambda cursor, row: row[0]
        ssh7 = ssh.cursor()
        csrr = ("SELECT ssstock FROM sshandler WHERE prodid = (?);")
        csrrxt = (prodid,)
        ssh7.execute(csrr, csrrxt)
        rows = ssh7.fetchall()
        # print(rows) #debug point
        values = ','.join(str(v) for v in rows)
        ssxdsccheck = "1 2 3 4"
        isolx = sqlite3.connect(r'DBFA_vend.db')
        isolxx = isolx.cursor()
        isolxx.execute(("SELECT lowstock from stock WHERE prodid = ?"), (prodid, ))
        rowsr = isolxx.fetchall()
        limiterx = []
        for ix in range (1, int((rowsr[0][0]))+1):
            limiterx.append(ix)
        if int(values) in limiterx:
            print("[Stock running out] Currently in stock: ", values, "pieces.
Restock ASAP...")
            ssxvarscheck = 1
        elif int(values) == 0:
            print("Current product stock: ", values)
            ssxvarscheck = 2
        elif int(values) < 1:
            print("Current product stock: ", values)
            ssxvarscheck = 2
        else:
            print("Current product stock: ", values)
            ssxvarscheck = 1
        time.sleep(0.2)
        #toaster.show_toast("DBFA QuickVend Service - Background Sync", duration =
0.3427)



    # Voucher System
    # Voucher User
```

```
    def cponuse(cponid):
        isol = sqlite3.connect('cponmgmtsys.db')
        isolx = isol.cursor()
        mod = """UPDATE cponmaster SET cponlim = cponlim - 1 WHERE cponid = ?"""
        idler = (cponid, )
        isolx.execute(mod, idler)
        isol.commit()

    # Single Voucher Data Fetcher
    def cpon_singlefetch(cponid):
        isol = sqlite3.connect('cponmgmtsys.db')
        isolx = isol.cursor()
        isol.row_factory = lambda cursor, row: row[0]
        csrr = ("SELECT cponid, cponlim, cponvalue FROM cponmaster WHERE cponid =
(?);")
        csrrxt = (cponid, )
        isolx.execute(csrr, csrrxt)
        rows = isolx.fetchall()
        values = ','.join(str(v) for v in rows)
        print("DNSS Coupon ", values)


    # Single Voucher Data Fetcher For Billing
    def cpon_ssinglefetch(cponid):
        isol = sqlite3.connect('cponmgmtsys.db')
        isolx = isol.cursor()
        isol.row_factory = lambda cursor, row: row[0]
        csrr = ("SELECT cponid FROM cponmaster WHERE cponid = (?);")
        csrrxt = (cponid, )
        isolx.execute(csrr, csrrxt)
        rows = isolx.fetchall()
        values = ','.join(str(v) for v in rows)
        global sfetch_values
        sfetch_values = values[2:-3]

    # Voucher Issuer
    def cponissuer(cponid, cponlim, cponvalue):
        isol = sqlite3.connect('cponmgmtsys.db')
        isolx = isol.cursor()
        #try:
        str = "insert into cponmaster(cponid, cponlim, cponvalue) values('%s', '%s',
'%s')"
        iox = (cponid, cponlim, cponvalue)
        isolx.execute(str % iox)
        isol.commit()
        cpon_singlefetch(cponid)
        print("DSNN: Coupon", cponid, "having discount %", cponvalue, "created for",
cponlim, "times of usage.")
        #except sqlite3.IntegrityError:
            #print("DNSS voucher already exists")
            #cpon_singlefetch(cponid)

    # Single Voucher Value Fetcher
    def cpon_valfetch(cponid):
        global valdock
        isol = sqlite3.connect('cponmgmtsys.db')
        isolx = isol.cursor()
        isol.row_factory = lambda cursor, row: row[0]
        csrr = ("SELECT cponvalue FROM cponmaster WHERE cponid = (?);")
        csrrxt = (cponid, )
```

```
        isolx.execute(csrr, csrrxt)
        rows = isolx.fetchall()
        values = ''.join(str(v) for v in rows)
        #print(values[1:-2])
        valdock = values[1:-2]
        print(valdock)

    #Voucher Limit Data Fetcher
    def cpon_limfetch(cponid):
        isol = sqlite3.connect('cponmgmtsys.db')
        isolx = isol.cursor()
        isol.row_factory = lambda cursor, row: row[0]
        csrr = ("SELECT cponlim FROM cponmaster WHERE cponid = (?);")
        csrrxt = (cponid, )
        isolx.execute(csrr, csrrxt)
        rows = isolx.fetchall()
        values = ''.join(str(v) for v in rows)
        limx = values[1:-2]
        if limx == 0:
            print("DNSSexemption: Coupon no longer valid. ")
        else:
            pass

    # Mass Voucher Listing Fetcher
    def cpon_masterfetch():
        isol = sqlite3.connect(r'cponmgmtsys.db')
        isolx = isol.cursor()
        isolx.execute("SELECT DISTINCT cponid, cponlim FROM cponmaster")
        rows = isolx.fetchall()
        col_labels = ("Coupon ID: ", "Usage Limit Left")
        table(col_labels, rows)




    # Customer System
    # Customer Record Creator
    def inserter(custt, custname, email):  #defining a function to input data into
the SQL database's table
        con = sqlite3.connect(r'DBFA.db')
        conn = con.cursor()
        str = "insert into cust(custt, custname, email) values('%s', '%s', '%s')"
        io = (custt, custname, email)
        conn.execute(str % io)
        con.commit()
        print("Customer", custname, "registered in store directory")

    # Customer Purchase Updater
    def custcc(custid, custname, purchasecount, ptotalx):  #defining a function to
input data into the SQL database's table
        global xon
        xon = sqlite3.connect(r'DBFA_CUSTCC.db')
        xbr7 = xon.cursor()
        str = "insert into custcc(custid, custname, purchasecount, ptotalx, points)
values(?, ?, ?, ?, 0)"
        io = (custid, custname, purchasecount, ptotalx)
        xbr7.execute(str, io)
        xon.commit()
        xbr7.close()
        print("FJHG")
```

```
815
816     # Customer Purchase Updater
817     def updatescript(custid, pincrement, billiemaster):
818         try:
819             xon = sqlite3.connect('DBFA_CUSTCC.db')
820             xbr7 = xon.cursor()
821             # hidden prints here ig
822             points = (billiemaster/100)*1
823             updatexr = """UPDATE custcc SET purchasecount = purchasecount + 1 WHERE
custid = ?"""
824             updatexxr = """UPDATE custcc SET ptotalx = ptotalx + ? WHERE custid =
?"""
825             updatexxxr = """UPDATE custcc SET points = points + ? WHERE custid =
?"""
826             indicator = (custid, )
827             xrindicator = (pincrement, custid)
828             pindicator = (points, custid)
829             xbr7.execute(updatexr, indicator)
830             xbr7.execute(updatexxr, xrindicator)
831             xbr7.execute(updatexxxr, pindicator)
832             xon.commit()
833             xbr7.close()
834         except sqlite3.Error as error:
835             pass
836
837
838     def pointfetch(custid):
839         global lylpoints
840         lylpoints = 0
841         xon = sqlite3.connect('DBFA_CUSTCC.db')
842         xbr7 = xon.cursor()
843         findinx = "select points from custcc WHERE custid = ?"
844         findinxx = (custid, )
845         xbr7.execute(findinx, findinxx)
846         arterxout = xbr7.fetchall()
847         lylpoints = int((arterxout[0])[0])
848
849
850     def pointmassfetch():
851         xon = sqlite3.connect('DBFA_CUSTCC.db')
852         xbr7 = xon.cursor()
853         findinx = "select DISTINCT points from custcc"
854         xbr7.execute(findinx)
855         rows = xbr7.fetchall()
856         for row in rows:
857             print(row[0])
858
859     def pointsuse(custid, deduct):
860         xon = sqlite3.connect('DBFA_CUSTCC.db')
861         xbr7 = xon.cursor()
862         updatexxxr = """UPDATE custcc SET points = points - ? WHERE custid = ?"""
863         pindicator = (deduct, custid)
864         xbr7.execute(updatexxxr, pindicator)
865         xon.commit()
866         xbr7.close()
867
868     def emailfetch(custid):
869         global custmail
870         con = sqlite3.connect(r'DBFA.db')
871         conn = con.cursor()
```

```
872         findinx = "select DISTINCT email from cust WHERE custt = ?"
873         findinxx = (custid, )
874         conn.execute(findinx, findinxx)
875         rows = conn.fetchall()
876         custmail = (rows[0][0])
877 
878     global custcheckindic
879 
880     global custt
881 
882     def custcheck(custt):
883         global cccheck, lylpoints
884         if custt in ("", " ", 0, None , "0"):
885             lylpoints = 0
886 
887         cccheck = 0
888         custcheckindic = 0
889         con = sqlite3.connect(r'DBFA.db')
890         conn = con.cursor()
891         conn.execute("SELECT custt FROM cust WHERE custt = ?", (custt,))
892         data = conn.fetchall()
893         if len(data)==0:
894             custcheckindic = 0
895             print("Customer", custt, "NOT found. ")
896             print("- No customer selected -")
897             custt = 0
898             print("Using unregistered customer account")
899             cccheck = 0
900         else:
901             ccustcheckindic = 1
902             cccheck = 0
903             pass
904 
905 
906     # Customer Validity Checker
907     def cust_listfetch(custid):
908         clfetch = sqlite3.connect(r'DBFA_CUSTCC.db')
909         clfetchx = clfetch.cursor()
910         clfetchx.execute("SELECT custid FROM custcc")
911         rows = clfetchx.fetchall()
912         custyes = 1
913         custno = 2
914         custcount = 0
915         for row in rows:
916             row = row[0]
917             if custid == row:
918                 custcount += 1
919             else:
920                 pass
921         if custcount == 1:
922             return custyes
923         else:
924             return custno
925 
926     def saleslogger(custid, prodid, netpay):  #defining a function to input data
    into the SQL database's table
927         sales = sqlite3.connect(r'dbfasales.db')
928         salesx = sales.cursor()
929 
930         netprof = 0
```

```
931         from datetime import date
932         datex = date.today()
933 
934         profer = [2000, 4500, 5700, 2000, 2100, 1470, 300, 11000, 400, 2000, 100,
    370, 450, 120, 50, 275, 649, 140, 50, 1050, 978, 150, 100, 320, 98, 75, 170, 60,
    275, 90, 210, 780, 50, 35, 50, 30, 100, 8000, 9000, 1790]
935         for i in prodid:
936             netprof += profer[int(i)]
937 
938         salesx.execute("SELECT MAX(sno) FROM SALES")
939         sno = (int(salesx.fetchall()[0][0]) + 1)
940 
941         prodidxs = ""
942         for i in prodid:
943             prodidxs += '%s'%i + ", "
944 
945         str = "insert into sales(sno, custid, prodid, net, prof, date) values(?, ?,
    ?, ?, ?, ?)"
946         io = (sno, custid, prodidxs, netpay, netprof, datex)
947         salesx.execute(str, io)
948         sales.commit()
949         print("Sales activity logged. ")
950 
951 
952     def salesdatefetch():  #defining a function to input data into the SQL
    database's table
953         from datetime import date
954         import datetime
955         sales = sqlite3.connect(r'dbfasales.db')
956         salesx = sales.cursor()
957         salesx.execute("SELECT prof FROM sales WHERE date BETWEEN datetime('now',
    '-6 days') AND datetime('now', 'localtime')")
958         sumer = 0
959         for i in salesx.fetchall():
960             sumer += int((i[0]))
961         return sumer
962 
963     def salestodayfetch():  #defining a function to input data into the SQL
    database's table
964         from datetime import date
965         import datetime
966         sales = sqlite3.connect(r'dbfasales.db')
967         salesx = sales.cursor()
968         salesx.execute("SELECT prof FROM sales WHERE date = ?", (date.today(), ))
969         sumerx = 0
970         for i in salesx.fetchall():
971             sumerx += int((i[0]))
972         return sumerx
973 
974 
975     os.system('cls')
976 
977 
978 
979     def floodscreen():
980         image = cv2.imread("imagepx.png")
981         cv2.imshow("Loading.... ", image)
982         cv2.waitKey(2000)
983         cv2.destroyAllWindows()
984 
```

```
985
986     def floodpay():
987         import cv2
988         image = cv2.imread("qr-code.png")
989         cv2.imshow("Pay With UPI", image)
990         cv2.waitKey(150000)
991         cv2.destroyAllWindows()
992
993
994     # Main Menu
995     # New Main Menu
996     def mainmenu(): #defining a function for the main menu
997         from colorama import init, Fore, Back, Style #color-settings for the
    partner/sponsor adverts
998         init(convert = True)
999         url = "https://raw.github.com/deltaonealpha/DBFA/master/updates.txt"
1000        r = requests.get(url)
1001        dbfaver = ((str(r.content.decode('utf-8'))))[4:]
1002        xdbfaver = ((str(r.content.decode('utf-8'))))
1003        with open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\updates.txt',
    'r+') as upread:
1004            upread = (str(upread.read())).strip()
1005        #print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
1006        time.sleep(1)
1007        spass1 = []
1008        spass2 = []
1009        for i in dbfaver:
1010            spass1.append(i)
1011        for j in upread[4: ]:
1012            spass2.append(j)
1013        try:
1014            if float(upread[4: ]) > float(dbfaver):
1015                pass
1016        except:
1017            print("Error with updater. Error code : *bckshln12* ╭( ◕_◕ )╮ ╭( ◕_◕ )╮")
1018        else:
1019            if spass1 != spass2:
1020                print("A new DBFA update is available: DBFA", dbfaver)
1021            #elif spass1 == spass2:
1022                #print("DBFA is up-to-date")
1023        # Count pending deliveries
1024        delcount = 0
1025        netprof = sqlite3.connect('recmaster.db')
1026        netprofx = netprof.cursor()
1027        netprofx.execute("SELECT netprof FROM recmasterx")
1028        rows = netprofx.fetchall()
1029        netproffetchx = []
1030        for i in rows:
1031            netproffetchx.append(i[0])
1032        print(" ")
1033        filedel = open('./DBFAdeliveries.txt', 'r+')
1034        for line in filedel:
1035            delcount+=1
1036        filedel.close()
1037        pro7d = salesdatefetch()
1038        protd = salestodayfetch()
1039        time.sleep(1)
1040        from colorama import init, Fore, Back, Style #color-settings for the
    partner/sponsor adverts
```

```
        logoxold = (Fore.CYAN+'''
                          Options:
█▀▀█ █▀▀█ █▀▀ █▀▀█ █▀▀█     1  - Issue a Bill                4  - Store Report
█__█ █▀▀█ █▀  █▀█  ▄▄▄▄     2  - Manage Customers:           5  - Manage Deliveries
CLIENT 8.12 DONNAGER                   a: Register a Customer    c: Purchase Records       6  - DBFA Options
'''+Fore.MAGENTA+'''  The OG Store Manager'''+Fore.CYAN+'''              b: Customer Registry      d: Find a Customer        7  - Start DBFA Backup & Switch
                                       e: Export data as CSV                               8  - Analyse Sales
                          3  - Store Options:                '''+Fore.MAGENTA+'''emp/EMP - DBFA Employee Manager'''+Fore.CYAN+'''
                                       a: Manage Stock           c: Manage Vouchers        9  - View Software License
                                       b: DBFA Stock Master      d: Product Listing        10 - About DBFA 8.4
                                       e: Sales Log              f: Export data as CSV     11 - Check for updates
                                       g: Invoice Deep Archive                             12 - Quit
- 'mark'/'MARK': to mark attendance

'''+Fore.MAGENTA+'''
DBFA Music Controls:: *prev* - << previous | *pause* - <|> pause/play | *next* - >> next  '''+Fore.CYAN+'''
-------------------------------------------------------------------------------------------------------------------''')


        logoxnew = (Fore.CYAN+'''Options:
1  - Issue a Bill                                              4  - Store Report
2  - Manage Customers:                                         5  - Manage Deliveries
        a: Register a Customer    c: Purchase Records          6  - DBFA Options
        b: Customer Registry      d: Find a Customer           7  - DBFA Backup & Switch
        e: Export data as CSV                                  8  - Analyse Sales
3  - Store Options:             '''+Fore.MAGENTA+'''emp/EMP - DBFA Employee Manager'''+Fore.CYAN+'''
        a: Manage Stock           c: Manage Vouchers           9  - View Software License
        b: DBFA Stock Master      d: Product Listing           10 - About DBFA 8.4
        e: Sales Log              f: Export data as CSV        11 - Check for updates
        g: Invoice Deep Archive                                12 - Quit
- 'mark'/'MARK': to mark attendance
'''+Fore.MAGENTA+'''
What would you like to do?                        The OG Store Manager'''+Fore.WHITE+''' █▀▀█ █▀▀█ █▀▀ █▀▀█ █▀▀█'''+Fore.CYAN+'''
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━      '''+Fore.WHITE+'''█__█ █▀▀█ █▀  █▀█  ▄▄▄▄'''+Fore.CYAN+'''
DBFA Music Controls: *prev* <<< | *pause* <|> | *next* >>>            CLIENT 8.552 DONNAGER
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━''')

        # To underline What would you like to do?::
```

```
        if settingscommonfetch(7) == 1:
            if delcount != 0:

 print("==========================================================================
==================")
                lener1 = "Profit (last week): " + '%s'%pro7d
                print(lener1 + (62-len(lener1))*" ", "Profit (today): ", protd)
                #pro7d, (56-len(str(pro7d)))*" ", "DONNAGER 8.01 RC-2 Test Beta")
                print(Back.BLACK + Fore.MAGENTA+ "Pending deliveries: " +
str(delcount) + " "  + "           DBFA User: " + os.getlogin() + "             "+
dt_string + Fore.CYAN)

 print("==========================================================================
==================")
            else:
                print("DONNAGER 8.01 RC-2 Test Beta")
                print(Fore.BLACK + Back.CYAN + "No deliveries pending! " +
Back.BLACK + Fore.CYAN)
            print(logoxnew)
        else:
            if delcount != 0:
                print("----------------------------------------------------------------
------------------------------------------------------")
                lener1 = "Profit (last week): " + '%s'%pro7d
                print(lener1 + (95-len(lener1))*" ", "   Profit (today): ", protd)
                #pro7d, (56-len(str(pro7d)))*" ", "DONNAGER 8.01 RC-2 Test Beta")
                print(Back.BLACK + Fore.MAGENTA+ "Pending deliveries: " +
str(delcount) + " "  + "       DBFA User: " + os.getlogin() + "              "+
dt_string + Fore.CYAN)
                print("----------------------------------------------------------------
------------------------------------------------------")
            else:
                print("DONNAGER 8.01 RC-2 Test Beta")
                print(Fore.BLACK + Back.CYAN + "No deliveries pending! " +
Back.BLACK + Fore.CYAN)
            print(logoxold)
        #Settings Checker
        if settingscommonfetch(3) == 1:
            time.sleep(0.2)
            try:
                if spotify.current() not in ("", " ", [], (), None):
                    print("Currently playing:", Fore.MAGENTA , spotify.current()[0],
Fore.CYAN, "by ", Fore.MAGENTA, spotify.current()[1], Fore.CYAN)
                else:
                    print(Fore.MAGENTA, "No music playing. Use Spotify to play your
favourite music and control it via DBFA", Fore.CYAN)
            except Exception as e:
                print(Fore.MAGENTA, "No music playing. Play your favourite music and
control it via DBFA", Fore.CYAN)
            if settingscommonfetch(7) == 1:

 print("==========================================================================
==================", Fore.MAGENTA)
            else:
                print("----------------------------------------------------------------
------------------------------------------------------", Fore.MAGENTA)
        else:

 print("==========================================================================
==================")
```

```
            print("Re-enable DBFA Music Controls Service from the settings to be able to control your music ")

 print("=============================================================================================================")

        #underline_byte = b'\xcc\xb2'
        #underline = str(underline_byte,'utf-8')
        #for x in ("What would you like to do?"):
        #    if x.isspace() == False:
        #        print(x+underline,end='')
        #    else:
        #        print(x,end='')
        print("What would you like to do?", Fore.WHITE)
        print()

    global netpay

    def xpayboxie():
        command = "cls"
        os.system(command)
        global xrt, payindic
        xrt = 0
        from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
        print("Amount to pay: ", netpay)
        print("Payment methods available: ")
        print("1. Credit/ Debit Card")
        print("2. Digital Wallet")
        print("3. UPI")
        print("4. Cash")
        print("*exit* to cancel this billing cycle")
        time.sleep(0.5)
        paycheck = input("Pay with: ")
        print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
        if paycheck == "1":
            payindic = "Paid with credit/ debit Card"
        elif paycheck == "2":
            payindic = "Paid with a digital wallet"
        elif paycheck == "3":
            payindic = "Paid with UPI"
            floodpay()
        elif paycheck == "4":
            payindic = "Paid with cash"
        elif paycheck == "exit":
            print("Cancelling this billing cycle")
            xrt = 1
        else:
            xpayboxie()

    global netpay

    # Payments Handler
    def payboxie(custid, total):
        global custcheckindic
        if custt not in (0, "0", "", " ", None) and cccheck == 0:
            command = "cls"
            os.system(command)
            global payindic, netpay, redeemindic, lylpoints
            xrt = 0
```

```
            redeemindic = 0
            payindic = 0
            from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
            print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
            init(convert = True)
            print("Amount to be paid: ₹","%.2f" % total)
            print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
            pointfetch(custid)
            if lylpoints != 0:
                print("You have loyalty points: ", lylpoints, "worth: ", lylpoints)
                time.sleep(0.5)
                pointscheck = input("Use points (y/n)? ")
                print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
                if pointscheck == "y":
                    if total > lylpoints:
                        redeemam = lylpoints
                    else:
                        redeemam = total
                    getOTP()
                    emailfetch(custid)
                    print("Please wait..")
                    email = 'billing.dbfa@gmail.com'
                    password = 'dbfaidlepass'
                    send_to_email = custmail
                    subject = 'Redeem your DBFA loyalty points'
                    messageHTML = ('''
                    <h1><span style="color: #496dd0">Redeem your DBFA loyalty points?</span></h1>
                    <h6> </h6>
                    <h4>You are recieving this mail as we've recieved a request to use the loyalty points on your account for a purchase with DBFA. If this isn't you, contact support at the earrliest.</h4>
                    <h6> </h6>
                    <h4>One-time password: <span style="color: #496dd0">''' + "%s"%otp + '''</span></h4>
                    <h6> </h6>
                    <h6>Points in your account: ''' + "%s"%lylpoints + '''</h6>
                    <h6>Points that will be redeemed: ''' + "%s"%redeemam + '''</h6>
                    <h6>Each DBFA point is worth ₹1</h6>
                    <h6>This purchase will give you: ''' + "%s"%((billiemaster/100)*1) + ''' points</h6>
                    <h6></h6>
                    <h6></h6>
                    <h6> </h6>
                    <h6> </h6>
                    <h6> </h6>
                    <h5>When you share this OTP with a DBFA-store, you authorize us to use your loyalty points to discount your purchase.</h5>
                    <h6>Do not share this OTP with any third party.</h6>
                    <h6>DBFA or any of its affiliates will not be resposible in anyway for any implications this OTP or any part of it might have on anyone in any way.</h6>
                    <h6>Offering these reward points does not benefit DBFA in any immediate manner.</h6>
                    ''')
                    messagePlain = 'DBFA Security'
                    msg = MIMEMultipart('alternative')
                    msg['From'] = email
                    msg['To'] = send_to_email
```

```
                    msg['Subject'] = subject
                    # Attach both plain and HTML versions
                    msg.attach(MIMEText(messagePlain, 'plain'))
                    msg.attach(MIMEText(messageHTML, 'html'))
                    os.startfile(r'Process_Handlers\\emailprocesswindow.pyw')
                    server = smtplib.SMTP('smtp.gmail.com', 587)
                    server.starttls()
                    server.login(email, password)
                    text = msg.as_string()
                    server.sendmail(email, send_to_email, text)
                    server.quit()
                    otpverif = input("Enter the OTP recieved on " + "%s"%custmail +
": ")
                    if otpverif == otp:
                        if total > lylpoints:
                            total = total - lylpoints
                            pointsuse(custid, lylpoints)
                            print("New total: ", total)
                            time.sleep(2)
                            redeemindic = 1
                            netpay = total
                            print("Points redeemed worth: ", lylpoints)

                        else:
                            redeemindic = 1
                            netpay = total
                            pointsuse(custid, total)
                            time.sleep(0.3)
                            total = 0
                            print("Points redeemed worth: ", lylpoints)
                        time.sleep(0.3)

                    else:
                        print("Wrong OTP. (1) attempt(s) remaining")
                        time.sleep(0.2)
                        otpverif = input("Enter the OTP recieved on " +
"%s"%custmail + ": ")
                        if otpverif == otp:
                            if total > lylpoints:
                                total = total - lylpoints
                                pointsuse(custid, lylpoints)
                                print("New total: ", total)
                                time.sleep(2)
                                redeemindic = 1
                                netpay = total
                                print("Points redeemed worth: ", lylpoints)
                            else:
                                netpay = 0
                                pointsuse(custid, total)
                                time.sleep(0.3)
                                total = 0
                                print("Points redeemed worth: ", lylpoints)
                        else:
                            print("Wrong OTP. (0) attempt(s) remaining")
                            time.sleep(0.2)
                elif pointscheck == "n":
                    redeemindic = 0
                    netpay = total
                else:
                    pass
```

```
1282                time.sleep(1)
1283                os.system(command)
1284            else:
1285                netpay = total
1286
1287        else:
1288            redeemindic = 0
1289            netpay = total
1290            redeemindic = 0
1291            lylpoints = 0
1292            netpay = total
1293
1294
1295    global custname, email, idd
1296
1297    def del2a():
1298        try:
1299            if os.path.exists(r'userblock.txt'):
1300                os.remove(r'userblock.txt')
1301            if os.path.exists(r'userblock.zconf'):
1302                os.remove(r'userblock.zconf')
1303        except PermissionError:
1304                pass
1305        print("Connecting to server..")  #SQL connection prompt
1306        time.sleep(0.4)  #for a seamless experience
1307        #conn.execute("select * from cust")
1308        #takes values from the SQL database
1309        conn = sqlite3.connect('DBFA.db')
1310        cursor = conn.cursor()
1311        cursor.execute("select * from cust")
1312        results = cursor.fetchall()
1313        idd = len(results)+1
1314        print("Registering customer with ID: ", idd)
1315        custname = input("Customer Name: ")
1316        email = input("Customer's E-mail ID: ")
1317        inserter(idd, custname, email) #argumental function to insert values into
    the SQL database
1318        nullvalue = 0
1319        custcc(idd, custname, nullvalue, nullvalue)
1320        print(" ")
1321
1322        print("Customer ID", idd, "registered in directory.")
1323        print("---------------------------------------")
1324        print(" ")
1325        print(" ")
1326        #conn.close()
1327        time.sleep(1) #for a seamless experience
1328
1329
1330    def del2b():
1331        try:
1332            if os.path.exists(r'userblock.txt'):
1333                os.remove(r'userblock.txt')
1334            if os.path.exists(r'userblock.zconf'):
1335                os.remove(r'userblock.zconf')
1336        except PermissionError:
1337                pass
1338        print()
1339        print("Connecting to server..")  #SQL connection prompt
1340        time.sleep(0.7) #for a seamless experience
```

```
1341        print("Registered customers are: ")
1342        #Re-writing to refresh connection
1343        conn = sqlite3.connect('DBFA.db')
1344        cur = conn.cursor()
1345        cur.execute("SELECT * FROM cust")
1346        rows = cur.fetchall()
1347        col_labels = ("ID", "Customer NAME", "EMAIL")
1348        table(col_labels, rows)
1349        toaster.show_toast("DNSS QuickSync", "Database acessed", duration = 2)
1350        #takes values from the SQL database
1351        '''
1352        while row is not None:
1353            print(row)
1354            #row = conn.fetchone()
1355        '''
1356
1357        conn.close()
1358        conn.close()
1359        print()
1360        print()
1361        time.sleep(2) #delay for easy-table viewing
1362
1363
1364    def del2c():
1365        try:
1366            if os.path.exists(r'userblock.txt'):
1367                os.remove(r'userblock.txt')
1368            if os.path.exists(r'userblock.zconf'):
1369                os.remove(r'userblock.zconf')
1370        except PermissionError:
1371                pass
1372        xon = sqlite3.connect(r'DBFA_CUSTCC.db')
1373        xbr7 = xon.cursor()
1374        xbr7.execute("SELECT * FROM custcc")
1375        l = xbr7.fetchall()
1376        print("\nCustomer Purchase Records: ")
1377
1378        import pandas as pd
1379
1380        flat_list = []
1381        print("-------------------------------------------------------------")
1382        for sublist in l:
1383            flat_list.append(sublist)
1384        mydf = pd.DataFrame(flat_list, columns=['Customer ID', 'Name', 'Purchases
    Made', 'Total', 'Loyalty Points'])
1385        mydf.pivot(index='Customer ID', columns='Purchases Made',
    values='Total').fillna(value='-')
1386        print(mydf)
1387        print("-------------------------------------------------------------")
1388        time.sleep(2)
1389
1390        '''
1391        for row in rows:
1392            print(row)
1393            print(" ")
1394        '''
1395        xbr7.close()
1396        toaster.show_toast("DFBA QuickSync", "Database acessed", duration = 0.5)
1397
1398
```

```
1399
1400    def del2d():
1401        try:
1402            con = sqlite3.connect(r'DBFA.db')
1403            conn = con.cursor()
1404
1405            conx = sqlite3.connect(r'DBFA_CUSTCC.db')
1406            connx = conx.cursor()
1407
1408            searchcon = str(input("Customer Name: "))
1409            if " " in searchcon:
1410                for i in searchcon:
1411                    consplit = searchcon.split(" ")
1412                    for j in consplit:
1413                        conn.execute("SELECT custt FROM cust WHERE custname LIKE ?",
    (("%"+searchcon+"%"), ))
1414                        searchdata = conn.fetchall()
1415                    else:
1416                        searchcon = searchcon.replace(" ", "")
1417                        conn.execute("SELECT custt FROM cust WHERE custname LIKE ?",
    (("%"+searchcon+"%"), ))
1418                        searchdata = conn.fetchall()
1419                        if len(searchdata) != 0:
1420                            pass
1421                        else:
1422                            searchdata = "No such customer found."
1423            else:
1424                conn.execute("SELECT custt FROM cust WHERE custname LIKE ?",
    (("%"+searchcon+"%"), ))
1425                searchdata = conn.fetchall()
1426
1427            if len(searchdata) != 0:
1428                if len(searchdata) > 1:
1429                    for i in searchdata:
1430                        conn.execute("SELECT * FROM cust WHERE custt = ?", (i[0], ))
1431                        custdata = conn.fetchall()
1432                        #col_labels = ("ID", "Customer NAME", "EMAIL")
1433                        #table(col_labels, custdata)
1434
1435                        connx.execute("SELECT * FROM custcc WHERE custid = ?",
    (i[0], ))
1436                        custdatax = connx.fetchall()
1437                        ccrt = []
1438                        for jk in custdata:
1439                            for jkx in jk:
1440                                ccrt.append(str(jkx))
1441                        for jk in custdatax:
1442                            for jkx in jk:
1443                                ccrt.append(str(jkx))
1444
1445                        col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name',
    'Purchases Made', 'Total', 'Loyalty Points')
1446                        print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f"))
1447
1448                        print(" ")
1449                else:
1450                    conn.execute("SELECT * FROM cust WHERE custt = ?",
    (searchdata[0][0], ))
1451                    custdata = conn.fetchall()
1452
```

```
1453                     connx.execute("SELECT * FROM custcc WHERE custid = ?",
(searchdata[0][0], ))
1454                     custdatax = connx.fetchall()
1455                     ccrt = []
1456                     for jk in custdata:
1457                         for jkx in jk:
1458                             ccrt.append(str(jkx))
1459                     for jk in custdatax:
1460                         for jkx in jk:
1461                             ccrt.append(str(jkx))
1462 
1463                     col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name',
'Purchases Made', 'Total', 'Loyalty Points')
1464                     print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f"))
1465             else:
1466                 srtx = "%"
1467                 for i in searchcon:
1468                     srtx += '%s'%i+"%"
1469                 conn.execute("SELECT custt FROM cust WHERE custname LIKE ?",
((srtx), ))
1470                 searchdata = conn.fetchall()
1471                 if len(searchdata) != 0:
1472                     conn.execute("SELECT * FROM cust WHERE custt = ?", (i[0], ))
1473                     custdata = conn.fetchall()
1474                     #col_labels = ("ID", "Customer NAME", "EMAIL")
1475                     #table(col_labels, custdata)
1476 
1477                     connx.execute("SELECT * FROM custcc WHERE custid = ?", (i[0], ))
1478                     custdatax = connx.fetchall()
1479                     ccrt = []
1480                     for jk in custdata:
1481                         for jkx in jk:
1482                             ccrt.append(str(jkx))
1483                     for jk in custdatax:
1484                         for jkx in jk:
1485                             ccrt.append(str(jkx))
1486 
1487                     col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name',
'Purchases Made', 'Total', 'Loyalty Points')
1488                     print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f"))
1489                 else:
1490                     print("Customer not found.")
1491         except:
1492             print("Error encountered.  ")
1493 
1494 
1495 
1496 
1497     def del2e():
1498         import sqlite3 as sql
1499         import os
1500         import csv
1501         from sqlite3 import Error
1502 
1503         try:
1504 
 csvex=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA.db')
1505             cursor = csvex.cursor()
1506             cursor.execute("select * from cust")
1507             print("Fetching data from database - I...")
```

```
            with open("DBFAcustrec.csv", "w") as csv_file:
                csv_writer = csv.writer(csv_file, delimiter="\t")
                csv_writer.writerow([i[0] for i in cursor.description])
                csv_writer.writerows(cursor)

 csvexx=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_CUSTCC.db')
                print("Fetching data from database - II...")
                cursorx = csvexx.cursor()
                cursorx.execute("select * from custcc")
                csv_writer = csv.writer(csv_file, delimiter="\t")
                csv_writer.writerow([i[0] for i in cursorx.description])
                csv_writer.writerows(cursorx)
            dirpath = os.getcwd() + "/DBFAcustrec.csv"
            print("Data exported Successfully into {}".format(dirpath))
            time.sleep(2)

            #Settings Checker
            if settingscommonfetch(4) == 1:
                os.startfile(r"DBFAcustrec.csv")
            else:
                pass

        except Error as e:
            print(e)

        finally:
            csvex.close()



    def del3a():
        decsfacter = str(input("Enter 'a' to VIEW STOCK, 'b' to ADD STOCK and 'c' to ENFORCE MASS STOCK: "))
        if decsfacter == "a":
            ssxsuperfetch()
        elif decsfacter == "b":
            objid = int(input("Enter the product ID to add stock for: "))
            stockincrement = int(input("Enter the amount of stock to be added: "))
            ssxupdatescript(stockincrement, objid)
            print("Stock updated by", stockincrement, "for product ID:", objid)
        elif decsfacter == "c":
            ggrtrr = int(input("Enter stock to universally enforce: "))
            massmaintainer(ggrtrr)


    def del3b():
        print("-------- DBFA Stock Master v1 ---------")
        print(" ")
        time.sleep(1)
        print("a: Order New Stock")
        print("b: Update Delivery Status")
        print("c: MASS - Fetch Current Status")
        print("d: INDVL - Fetch Current Status")
        print("e: View Vendor Details")
        print("f: Contact Vendor")
        print("g: Edit Vendor Contact")
        print("h: Modify Low-Stock Warning Bar")
        stkmaster = input("Select:: ")
        if stkmaster in ("a", "A"):
```

```
1565            idquery = int(input("Enter the product ID: "))
1566            qtyquery = int(input("Enter the amount to order: "))
1567            orderstock(idquery, qtyquery)
1568            print("\n----------------------------------------------------------")
1569        elif stkmaster in ("b", "B"):
1570            idquery = int(input("Product ID of the product recieved: "))
1571            qtyquery = int(input("Quantity recieved: "))
1572            delivered(qtyquery, idquery)
1573            ssxupdatescript(qtyquery, idquery)
1574            print("\nChanges made in all related databases. \n")
1575            print("\n----------------------------------------------------------")
1576        elif stkmaster in ("c", "C"):
1577            delstatmass()
1578            print("\n----------------------------------------------------------")
1579        elif stkmaster in ("d", "D"):
1580            idquery = int(input("Product ID to locate: "))
1581            delstatindvl(idquery)
1582            print("\n----------------------------------------------------------")
1583        elif stkmaster in ("e", "E"):
1584            idquery = int(input("Product ID to get vendor details for: "))
1585            vendorfetch(idquery)
1586            print("\n----------------------------------------------------------")
1587        elif stkmaster in ("f", "F"):
1588            idquery = int(input("Product ID to contact vendor for: "))
1589            vendorcontact(idquery)
1590            print("\n----------------------------------------------------------")
1591        elif stkmaster in ("g", "G"):
1592            print("-- Change Vendor Contact --")
1593            time.sleep(0.5)
1594            try:
1595                prodidvendc = int(input("Enter product ID to change vendor contact
    for: "))
1596                if prodidvendc > 40:
1597                    print("Please enter a valid product ID")
1598                    prodidvendc = input("Enter product ID to change vendor contact
    for: ")
1599
1600            except:
1601                print("Please enter a valid product ID")
1602                time.sleep(0.3)
1603                prodidvendc = input("Enter product ID to change vendor contact for:
    ")
1604                if prodidvendc > 40:
1605                    print("Please enter a valid product ID")
1606                    prodidvendc = input("Enter product ID to change vendor contact
    for: ")
1607            while(1):
1608                vendorchange = input("Enter the new E-Mail ID: ")
1609                if "@" in vendorchange:
1610                    if "." in vendorchange:
1611                        st = sqlite3.connect(r'DBFA_vend.db')
1612                        stx = st.cursor()
1613                        stx.execute("""UPDATE stock SET vendcont = ? WHERE prodid =
    ?""", (vendorchange, prodidvendc,))
1614                        st.commit()
1615                        print("Vendor contact changed for product ID: ",
    prodidvendc)
1616                        time.sleep(1.5)
1617                        break
1618                        mainmenu()
```

```
                    else:
                        print("Please enter a valid E-Mail format! ")
                else:
                    print("Please enter a valid E-Mail format! ")

        elif stkmaster in ("h", "H"):
            idquery = int(input("Product ID to alter bar for: "))
            lowbarmodif(idquery)
            print("\n-----------------------------------------------------------")
        else:
            print("Please select a valid option! ")
            print("\n-----------------------------------------------------------")
            time.sleep(1)
            mainmenu()



    def del3c():
        print("Enter '1' to generate a voucher; ")
        descx = int(input("'2' to view generated vouchers: "))
        if descx == 1:
            print("DNSS CouponMaster: Issuer")
            print("NOTE: Reusing existing coupon codes may result in overwritten
data.")
            cponid = input("Coupon ID: ")
            cponlim = input("Number of times to allow coupon usage: ")
            cponvalue = input("Coupon discount percentage: ")
            cponissuer(cponid, cponlim, cponvalue)
            time.sleep(0.4)
        elif descx == 2:
            print("DNSS CouponMaster: Viewer")
            cpon_masterfetch()
            time.sleep(0.4)



    def del3d():
        try:
            if os.path.exists(r'userblock.txt'):
                os.remove(r'userblock.txt')
            if os.path.exists(r'userblock.zconf'):
                os.remove(r'userblock.zconf')
        except PermissionError:
                pass
        print("Store listing (as per updated records): ")
        print(tabulate(tablx, headers = titlex, floatfmt = ".4f"))



    def del3e():
        try:
            if os.path.exists(r'userblock.txt'):
                os.remove(r'userblock.txt')
            if os.path.exists(r'userblock.zconf'):
                os.remove(r'userblock.zconf')
        except PermissionError:
                pass
        #password verification as sales record is not to be shown to all;
        print(" - Echo-supressed input - ")
```

```
1677        passw = getpass.getpass(prompt='Enter root password to view store activity
registry: ', stream=None)
1678        if passw == "root":
1679                time.sleep(1) #for a seamless experience
1680                print("Hold on, moneybags.")
1681                with HiddenPrints():
1682                    try:
1683                        sender = telegram_bot_sendtext(dt_string + "\n" + "Registry
files and sales DB records accessed - DBFA SECURITY")
1684                        print(sender)
1685                    except Exception:
1686                        pass
1687                time.sleep(0.2) #for a seamless experience
1688                print()
1689                sales = sqlite3.connect(r'dbfasales.db')
1690                salesx = sales.cursor()
1691                salesx.execute("SELECT * FROM sales")
1692                salesrows = salesx.fetchall()
1693                col_labels = ("SalesID", "CustomerID", "Product Codes Purchased",
"Total", "Profit Earned", "Date of Purchase")
1694                table(col_labels, salesrows)
1695                toaster.show_toast("DNSS QuickSync", "Database acessed", duration =
2)
1696                time.sleep(1.4) #for a seamless experience
1697                os.startfile('registry.txt') #to open the external notepad
application
1698        else:
1699            print("Ehh that'd be wrong, sneaky-hat. Try again: ")
1700            print(" ")
1701            print(" - Echo-supressed input - ")
1702            passw = getpass.getpass(prompt='Enter root password to view store
activity registry: ', stream=None)
1703            if passw == "root":
1704                    time.sleep(1) #for a seamless experience
1705                    print("Hold on, moneybags.")
1706                    with HiddenPrints():
1707                        try:
1708                            sender = telegram_bot_sendtext(dt_string + "\n" +
"Registry files and sales DB records accessed accessed - DBFA SECURITY: ATTEMPT 02")
1709                            print(sender)
1710                        except Exception:
1711                            pass
1712                    print("There ya go:: ")
1713                    print()
1714                    sales = sqlite3.connect(r'dbfasales.db')
1715                    salesx = sales.cursor()
1716                    salesx.execute("SELECT * FROM sales")
1717                    salesrows = salesx.fetchall()
1718                    col_labels = ("SalesID", "CustomerID", "Product Codes
Purchased", "Total", "Profit Earned", "Date of Purchase")
1719                    table(col_labels, salesrows)
1720                    toaster.show_toast("DNSS QuickSync", "Database acessed",
duration = 2)
1721                    time.sleep(0.6) #for a seamless experience
1722                    # print(logger.read())
1723                    # print()
1724                    # print("Opening sales log externally now. ")
1725                    time.sleep(1.4) #for a seamless experience
1726                    os.startfile('registry.txt')
1727            else:
```

```
1728                with HiddenPrints():
1729                    try:
1730                        sender = telegram_bot_sendtext(dt_string + "\n" + "[ACCESS
DENIED!!] - Registry file  - DBFA SECURITY [ACCESS DENIED!!]")
1731                        print(sender)
1732                    except Exception:
1733                        pass
1734                print("Multiple Unsuccesfull Attempts Detected. Re-run the program
to login now. ")
1735                logger.write("(MULTIPLE ATTEMPTS!): Log file access attempt -
AUTHORIZATION FAILED!!! \n")
1736                time.sleep(1.4) #for a seamless experience
1737                print()
1738                print()
1739
1740
1741
1742
1743    def del3f():
1744        import sqlite3 as sql
1745        import os
1746        import csv
1747        from sqlite3 import Error
1748
1749        try:
1750
 csvex=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\recmaster.
db')
1751            print("Exporting sales data to CSV....")
1752            cursor = csvex.cursor()
1753            cursor.execute("select * from recmasterx")
1754            with open("DBFAstorerec.csv", "w") as csv_file:
1755                csv_writer = csv.writer(csv_file, delimiter="\t")
1756                csv_writer.writerow([i[0] for i in cursor.description])
1757                csv_writer.writerows(cursor)
1758
1759            dirpath = os.getcwd() + "/DBFAcustrec.csv"
1760            print("Data exported Successfully into {}".format(dirpath))
1761            time.sleep(2)
1762
1763            #Settings Checker
1764            if settingscommonfetch(4) == 1:
1765                os.startfile(r"DBFAcustrec.csv")
1766            else:
1767                pass
1768
1769        except Error as e:
1770            print(e)
1771
1772        finally:
1773            pass
1774
1775    def del3g(): #delta DEEP ARCHIVAL VAULT ENGINE
1776        print("DBFA uses advanced algorithms to store invoice data in easily
indecipherable strings.")
1777        print("These can be used at any point to generate an invoice from any date,
given its key is stored with the deep archival VAULT.")
1778        print("This ensures data protection and enables us to practically store
infinity invoices in a relatively non-existent amount of space\n\n")
1779        print("This option can be used to generate such back-dated invoices.")
```

```
1780        time.sleep(2)
1781        from DBFADeepArchivalEngine import deepfetch_deeparchival,
    encoder_deeparchival
1782        from DBFADeepArchivalEngine import alphadecoder, dttdecoder,
    decoder_deeparchival
1783        deepfetch_deeparchival()
1784
1785    print("----------------------------------------------------------------------
    ---------------------------")
1786
1787    # Store listing::
1788    data = {"1":40000, "2":55000, "3":67000, "4":25000, "5":21000, "6":14000,
    "7":13000, "8":220000, "9":4500, "10":17000, "11":1200, "12":3700, "13":4500,
    "14":2200, "15":700, "16":2750, "17":6499, "18":1499, "19":799, "20":27000,
    "21":6750, "22":2100, "23":1199, "24":3210, "25":989, "26":750, "27":1700, "28":600,
    "29":2175, "30":890, "31":2100, "32":7158, "33":597, "34":347, "35":500, "36":300,
    "37":1097, "38":80000, "39":87900, "40":23790}
1789    namie = {"1":"TV 4K OLED 50", "2":"TV FHD OLED 50", "3":"8K QLED 80", "4":"Redmi
    K20 PRO", "5":"Redmi K20", "6":"Redmi Note 8 PRO", "7":"POCOPHONE F1", "8":"Mi MIX
    ALPHA", "9":"Wireless Headphones", "10":"Noise-Cancelling Wireless Headphones",
    "11":"Essentials Headphones", "12":"Gaming Headphones", "13":"Truly-Wireless
    Eadphones", "14":"Neckband-Style Wireless Earphones", "15":"Essentials Earphones",
    "16":"Gaming Earphones", "17":"30W Bluetooth Speakers", "18":"10W Bluetooth
    Speakers", "19":"Essentials Bluetooth Speaker", "20":"ULTRA Home Theatre",
    "21":"Essentials Home Theatre", "22":"  Wired Speaker - 5.1", "23":"  Essentials
    Wired Speaker - STEREO", "24":"Tactical Power Bank 30000mah", "25":"Essentials Power
    Bank 10000mah", "26":"Essentials Mouse", "27":"Logitech G604 LightSpeed Wireless",
    "28":"Tactical Essentials Keyboard", "29":"DROP GS21k RGB Gaming Keyboard",
    "30":"Polowski Tactical Flashlight", "31":"OneFiber Wi-Fi Router AX17", "32":"Mijia
    Mesh Wi-Fi Router", "33":"lapcare 120W Laptop Adapter", "34":"lapcare 60W Laptop
    Adapter", "35":"Spigen Phone Case(s)", "36":"Essentials Phone Charger 10W",
    "37":"HyperPower Type-C Gallium-Nitride Charger 100W", "38":"ASUS Zephyrus G14
    Gaming Laptop", "39":"L XPS 15 Content Creator's Laptop", "40":"Hewlett-Packard
    Essential's Student's Laptop (Chromebook)"}
1790    namiex = ["TV 4K OLED 50", "TV FHD OLED 50", "8K QLED 80", "Redmi K20 PRO",
    "Redmi K20", "Redmi Note 9 PRO", "POCOPHONE F1", "Mi MIX ALPHA", "Wireless
    Headphones", "Noise-Cancelling Wireless Headphones", "Essentials Headphones",
    "Gaming Headphones", "Truly-Wireless Eadphones", "Neckband-Style Wireless
    Earphones", "Essentials Earphones", "Gaming Earphones", "30W Bluetooth Speakers",
    "20W Bluetooth Speakers", "Essentials Bluetooth Speaker", "BOSE QC35", "Essentials
    Home Theatre", "Wired Speaker - 5.1", "Essentials Wired Speaker - STEREO", "Tactical
    Series Power Bank 30000mah", "Essentials Power Bank 10000mah", "Essentials Mouse",
    "Logitech G604 LightSpeed Wireless", "Tactical Essentials Keyboard", "DROP GS21k RGB
    Gaming Keyboard", "Polowski Tactical Flashlight", "OneFiber Wi-Fi Router AX7",
    "Mijia Mesh Wi-Fi Router", "lapcare 45W Laptop Adapter", "lapcare 60W Laptop
    Adapter","Spigen Phone Case(s)", "Essentials Phone Charger 15W", "HyperPower Type-C
    Gallium-Nitride Charger 120W", "ASUS Zephyrus G4 Gaming Laptop", "DELL XPS 5 Content
    Creator's Laptop", "Hewlett-Packard Essential's Student's Laptop (Chromebook)"]
1791    datax = [40000, 55000, 67000, 25000, 21000, 14000, 3000, 220000, 4500, 17000,
    1200, 3700, 4500, 2200, 700, 2750, 6499, 1499, 799, 27000, 6750, 2100, 1199, 3210,
    989, 750, 1700, 600, 2175, 890, 2100, 7158, 597, 347, 500, 300, 1097, 80000, 87900,
    23790]
1792
1793    # dataxr is currently redundant
1794    dataxr = []
1795    for i in datax:
1796        i = "₹" + '%d' % i
1797        dataxr.append(i)
1798    tablx = zip(namiex, dataxr)
1799    titlex = ["Product:", "Pricing:"]
```

```
1800
1801    print('''
1802    DBFA Billing Framework 7 [Bellaire] (stable)
1803    <GNU Public License> Copyright (C) 2020 Pranav Balaji and Sushant Gupta
1804    View the license file in the installation dir for more info.
1805    \n\n
1806    Fetching Windows login details..
1807    Fetching Windows login details....
1808    \n''')
1809
1810    logoprintxrt()
1811    time.sleep(0.3)
1812    command = "cls"
1813    os.system(command)
1814
1815
1816    #vs2
1817    from datetime import datetime  #for reporting the billing time and date
1818    now = datetime.now()
1819    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object containing
current date and time
1820    logger = open(r"registry.txt", "a+")  #Opening / creating (if it doesn't exist
already) the .txt record file
1821    logger.write("---------------------------------------- \n")
1822    logger.write('ed')
1823    logger.write("\n")
1824    logger.write("Automated Store Registry:\n")
1825
1826    #Settings Checker
1827    if settingscommonfetch(1) == 1:
1828        floodscreen() #comment to disable boot-flash screen
1829    else:
1830        pass
1831
1832
1833    import requests, time, json, urllib, os, math, random, sqlite3
1834    from tqdm import tqdm
1835    import logging, os, time, requests, socket
1836    import telegram_send
1837    from pynput.keyboard import Key, Controller
1838    # Telegram BOT API 2 (FULL)
1839    from telegram import InlineKeyboardButton, InlineKeyboardMarkup
1840    from telegram.ext import Updater, CommandHandler, CallbackQueryHandler
1841
1842    logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %
(message)s',
1843                        level=logging.INFO)
1844    xlogger = logging.getLogger(__name__)
1845
1846    def telegram_bot_sendtext(bot_message):
1847        bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
1848        bot_chatID = '680917769'
1849        send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?
chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
1850        response = requests.get(send_text)
1851        return response.json()
1852
1853    def start(update, context):
1854        keyboard = [[InlineKeyboardButton("✅ Validate", callback_data='1'),
1855                     InlineKeyboardButton("❌ Deny", callback_data='2')]]
```

```
1856
1857        reply_markup = InlineKeyboardMarkup(keyboard)
1858
1859        update.message.reply_text("delta 2FA Handler Service\n\n\nValidate login?",
reply_markup=reply_markup)
1860
1861
1862    def button(update, context):
1863        query = update.callback_query
1864        # CallbackQueries need to be answered, even if no notification to the user
is needed
1865        # Some clients may have trouble otherwise. See
https://core.telegram.org/bots/api#callbackquery
1866        inlet = ("{}".format(query.data))
1867        if inlet in (1, "1"):
1868            query.edit_message_text(text="✅ delta 2FA approved! \n\nThis allows
your installation of the DBFA client, and its data to be accessed. \n\nIf this
wasn't you, contact support and revoke your bot login at the earliest.\n\ndelta
Security Service")
1869            with
open(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\deltatgstickerlogonew.we
bp", "rb") as f:
1870                telegram_send.send(stickers=[f])
1871                keyboard = Controller()
1872                print("\n\nValidation recieved! DBFA Client will start in a
moment\n\n")
1873                print("telegram_extended.updtr_pushreq(deltaonealpha, set.webhook:
(on, getUpdated.redir(servers.gokku.com/deltaonealpha/arterxt1, callback=False)))")
1874                print("\n\nif there's no print below for around 5 secs from now,
press ctrl+c to continue\n\n")
1875                keyboard.press(Key.ctrl)
1876                keyboard.press('c')
1877                keyboard.release('c')
1878                keyboard.release(Key.ctrl)
1879
1880        if inlet in (2, "2"):
1881            query.edit_message_text(text="❌🔐 Denied delta 2FA request.\n\ndelta
Security Service")
1882            with
open(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\deltatgstickerlogonew.we
bp", "rb") as f:
1883                telegram_send.send(stickers=[f])
1884            keyboard = Controller()
1885            print("\n\nThe login request for this session has been DENIED.\n\n")
1886            time.sleep(1)
1887            print("telegram_extended.updtr_pushreq(deltaonealpha, set.webhook: (on,
getUpdated.redir(servers.gokku.com/deltaonealpha/arterxt1, callback=False)))")
1888            print("\n\nif there's no print below for around 5 secs from now, press
ctrl+c to continue\n\n")
1889            keyboard.press(Key.ctrl)
1890            keyboard.press('c')
1891            keyboard.release('c')
1892            keyboard.release(Key.ctrl)
1893            time.sleep(1)
1894            time.sleep(1)
1895            print("DBFA Client will now exit! ")
1896            time.sleep(5)
1897            os._exit(0)
1898        if inlet in (3, "3"):
1899            query.edit_message_text(text="Use */help*")
```

```
1900
1901
1902    def OAuthvalidate():
1903        from datetime import datetime
1904        import sqlite3, time, os
1905        os.system('cls')
1906        now = datetime.now()
1907        try: #To avoid error when time is 00:00:00
1908            netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 +
    int(now.strftime("%S"))
1909        except:
1910            time.sleep(1)
1911            netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 +
    int(now.strftime("%S"))
1912        oauth = sqlite3.connect(r'dbfasettings.db')
1913        oauthx = oauth.cursor()
1914        oauthx.execute("SELECT Value FROM LoginHandler")
1915        check = oauthx.fetchall()[0][0]
1916        if check != 1:
1917            print("Login was bypassed! DBFA will now exit!")
1918            time.sleep(1)
1919            os._exit(0)
1920        oauthx.execute("SELECT count(*) FROM LoginHandler")
1921        rows = oauthx.fetchall()
1922        if (rows[0][0]) > 1:
1923            print("DBFA Authenticator has been tampered with! DBFA WILL EXIT NOW!")
1924            oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
1925            oauth.commit()
1926            oauth.close()
1927            time.sleep(2)
1928            os._exit(0)
1929        oauthx.execute("SELECT TimeMark FROM LoginHandler")
1930        stamp = oauthx.fetchall()[0][0]
1931        if int(netr)-int(stamp) > 60:
1932            print("netr")
1933            oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
1934            oauth.commit()
1935            oauth.close()
1936            os.startfile('authtimeout.pyw')
1937            time.sleep(1)
1938            os._exit(0)
1939        elif int(netr)-int(stamp) < 60:
1940            print("- - - Valid login detected - - -\n\n")
1941        else:
1942            oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
1943            oauth.commit()
1944            oauth.close()
1945            print("Login was bypassed! DBFA will now exit!")
1946            time.sleep(1)
1947            os._exit(0)
1948
1949        oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
1950        oauth.commit()
1951        oauth.close()
1952
1953    OAuthvalidate()
1954
1955
1956    def help_command(update, context):
```

```
        update.message.reply_text("Use /auth when prompted. This bot will only respond when a delta service raises a request. ")


    def main():
        # Create the Updater and pass it your bot's token.
        # Make sure to set use_context=True to use the new context based callbacks
        # Post version 12 this will no longer be necessary
        updater = Updater("1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis", use_context=True)

        updater.dispatcher.add_handler(CommandHandler('auth', start))
        updater.dispatcher.add_handler(CallbackQueryHandler(button))
        updater.dispatcher.add_handler(CommandHandler('help', help_command))

        # Start the Bot
        updater.start_polling()

        # Run the bot until the user presses Ctrl-C or the process receives SIGINT,
        # SIGTERM or SIGABRT
        updater.idle()


    def settingscommonfetch(SettingsType):
        import sqlite3
        settings = sqlite3.connect(r'dbfasettings.db')
        settingsx = settings.cursor()
        settingsx.execute(("SELECT Value from settings WHERE SettingsType = ?"),(SettingsType,))
        settingsfetch = (settingsx.fetchall()[0][0])
        return settingsfetch

    if settingscommonfetch(6) == 1:
        def telegram_bot_sendtext(bot_message):
            bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
            bot_chatID = '680917769'
            send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
            response = requests.get(send_text)
            return response.json()

        print("delta2 Authenication Service")
        print("=======================================")
        print("DBFA 2FA Service ")
        print("================ ")
        time.sleep(1)
        print("")


        print("You have DBFA 2FA activated. Please validate the login from your Telegram account. ")
        for i in tqdm (range (10), desc="Connecting.."):
            time.sleep(0.00001)
        if __name__ == '__main__':
            hostname = socket.gethostname()
            ip_address = socket.gethostbyname(hostname)
            import platform
            from datetime import datetime  #for reporting the billing time and date
            now = datetime.now()
```

```
            dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object containing current date and time
            with open(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\deltatgstickerlogonew.webp", "rb") as f:
                telegram_send.send(stickers=[f])
            telegram_bot_sendtext("🔐delta 2FA Handler Service\nA login request has been recieved from your DBFA installation.\n\nRequest time        - " + '%s'%dt_string + f"\nHostname             - {hostname}\n" + f"IP Address - {ip_address}\n" + "Service Identifier  - "+ platform.system() + platform.release() +"\n\nWARNING:  Do not approve this if this isn't you!\n\nPlease send */auth* to start the validation process: ")
            main()
        os.system('cls')

    else:
        print("DBFA 2FA is disabled. We recommend you to turn it on from the settings for a more secure experience with DBFA client.")


    print("---------------------------------\n\n┌( ◕_◕ )┐ delta Update Utility\n\n---------------------------------")
    time.sleep(0.5)
    print("DBFAIntellisense")
    print("Fetching update details from server : : : : ")
    url = "https://raw.github.com/deltaonealpha/DBFA/master/updates.txt"
    r = requests.get(url)
    dbfaver = ((str(r.content.decode('utf-8'))))[4:]
    xdbfaver = ((str(r.content.decode('utf-8'))))
    with open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\updates.txt', 'r+') as upread:
        upread = (str(upread.read())).strip()
    print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
    time.sleep(1)
    spass1 = []
    spass2 = []
    for i in dbfaver:
        spass1.append(i)
    for j in upread[4: ]:
        spass2.append(j)
    if float(upread[4: ]) > float(dbfaver):
        pass
    time.sleep(1)

    os.system('cls')


    from win10toast import ToastNotifier
    toaster = ToastNotifier()
    toaster.show_toast("DFBA System","Read documentation prior to use.", duration = 1)
    print("Heyy there,", os.getlogin()) #enable parts in the auth script to enable user detection
    time.sleep(0.5)
    if redflag == 0:
        logger.write("Auth bypass - registering for security")
        time.sleep(1)
        print("------- !!!!!!! -------")
        print("We highly value the security of our code and our customers.")
        toaster.show_toast("DBFA Security","Terminating Session!")
```

```
2058        print("It has been detected that you have bypassed the login process.")
2059        time.sleep(1)
2060        print("Please obtain a genuine version of this program and/ or provide
    proper authenication.")
2061        time.sleep(1)
2062        print("The program shall now exit. Error code:013")
2063        time.sleep(2)
2064        print("----------------------------------------")
2065        time.sleep(5)
2066        exit()
2067
2068
2069
2070    # Main Program Starts Here
2071    while(1): #while (always) true
2072        mainmenu() #mainmenu
2073        writer = ""
2074        telethon = ""
2075        time.sleep(0.037)  #for a seamless experience
2076        decfac = input("Select option: ")
2077
2078        #DBFA Music Controls v1.2
2079        #All possible case-combinations; found using recursion
2080        if decfac in ('prev', 'preV', 'prEv', 'prEV', 'pRev', 'pReV', 'pREv',
    'pREV', 'Prev', 'PreV', 'PrEv', 'PrEV', 'PRev', 'PReV', 'PREv', 'PREV'):
2081            try:
2082                spotilib.previous()
2083            except Exception as e:
2084                pass
2085        elif decfac in ('pause', 'pausE', 'pauSe', 'pauSE', 'paUse', 'paUsE',
    'paUSe', 'paUSE', 'pAuse', 'pAusE', 'pAuSe', 'pAuSE', 'pAUse', 'pAUsE', 'pAUSe',
    'pAUSE', 'Pause', 'PausE', 'PauSe', 'PauSE', 'PaUse', 'PaUsE', 'PaUSe', 'PaUSE',
    'PAuse', 'PAusE', 'PAuSe', 'PAuSE', 'PAUse', 'PAUsE', 'PAUSe', 'PAUSE'):
2086            try:
2087                spotilib.pause()
2088            except Exception as e:
2089                pass
2090
2091        elif decfac in ('next', 'nexT', 'neXt', 'neXT', 'nExt', 'nExT', 'nEXt',
    'nEXT', 'Next', 'NexT', 'NeXt', 'NeXT', 'NExt', 'NExT', 'NEXt', 'NEXT'):
2092            try:
2093                spotilib.next()
2094            except Exception as e:
2095                pass
2096
2097        #DBFA Mark Attendance
2098        if decfac in ('attendance', 'ATTENDANCE', 'mark', 'MARK', 'mArK', 'MaRk',
    'maRK', 'MArk', 'm a r k', 'M A R K', 'M A r k', 'm a R K'):
2099            import sqlite3, time, os, requests
2100            from datetime import datetime  #for reporting the billing time and date
2101            empmas = sqlite3.connect(r'dbfaempmaster.db')
2102            empmascur = empmas.cursor()
2103
2104            empmascur.execute("SELECT DISTINCT * FROM emp")
2105            dump = empmascur.fetchall()
2106            Oiddump = []
2107            for row in dump:
2108                Oiddump.append(row[0])
2109            print("OiDs registered: ", Oiddump)
2110            try:
```

```
2111                Oid = int(input("DBFA MARK ATTENDANCE- Enter your OiD: "))
2112                trypass = 1
2113            except:
2114                print("OiDs are integer-only. Please retry using valid credentials!
    \n")
2115                trypass = 0
2116
2117            if trypass == 1:
2118                if Oid in Oiddump:
2119                    print("OiD found ")
2120                    time.sleep(0.5)
2121                    now = datetime.now()
2122                    dt_string = now.strftime("%Y/%m/%d")  #datetime object
    containing current date and time
2123                    tm_string = now.strftime("%H:%M:%S")  #datetime object
    containing current date and time
2124                    empmascur.execute("SELECT count(*) FROM attendance WHERE Date =
    ? AND OiD = ?", (dt_string, Oid,))
2125                    data = empmascur.fetchone()[0]
2126                    if data==0:
2127                        #print('No record on %s'%dt_string+' for Employee %s'%Oid)
2128                        #print("insert into attendance(Date, OiD, Time, YN, IO)
    values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
2129                        empmascur.execute("insert into attendance(Date, OiD, Time,
    YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
2130                        #empmascur.execute("UPDATE attendance SET Oid = 'Y' WHERE
    DATE = ?", (dt_string))
2131                        empmas.commit()
2132                        print("\n----------------DBFA-----------------")
2133                        print("C1 ENTRY: Marked Attendance! OiD: ", Oid)
2134                        print("-------------------------------------\n")
2135                    elif data==1:
2136                        #print('Component %s found in %s row(s)'%(dt_string, data))
2137                        empmascur.execute("insert into attendance(Date, OiD, Time,
    YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'O'))
2138                        empmas.commit()
2139                        print("\n----------------DBFA-----------------")
2140                        print("C2 DAY END: Marked Attendance! OiD: ", Oid)
2141                        print("-------------------------------------\n")
2142                    elif data > 1:
2143                        print("You can only mark in/out attendance once a day! \n")
2144
2145                else:
2146                    print("OiD not found! \n")
2147
2148        #Billing Mode
2149        elif decfac == "1":
2150            print()
2151            try:
2152                if os.path.exists(r'userblock.txt'):
2153                    os.remove(r'userblock.txt')
2154                if os.path.exists(r'userblock.zconf'):
2155                    os.remove(r'userblock.zconf')
2156            except PermissionError:
2157                    pass
2158            print("--- BIlling ---")
2159            print()
2160            custt = input("Customer ID (optional): ")
2161            if custt in ("0", 0, "", " "):
2162                print("Unregistered Customer")
```

```
                custt = "0"
                custcheck(custt)
                cccheck = 0
            try:
                if (cust_listfetch(int(custt))) == 1:
                    print("Customer found")
                    custcheck(custt)
                else:
                    print("Unregistered Customer")
                    cccheck = 0
            except:
                custt = "0"
                print("Unregistered Customer")

            #print(cccheck)
            logger.write("-----------------  ") #writing to log file
            logger.write("Cust. ID: \n")
            logger.write(custt)
            logger.write("  \n")
            logger.write("Date and time: \n") #including the date and time of billing (as taken from the system)
            logger.write(dt_string)
            logger.write(" \n")
            abcd1 = 1
            infetch()
            purcheck = ""
            profer = []
            time.sleep(0.3) #for a seamless experience
            telethon = "DBFA Billing System" + "\n" + dt_string + "\n" + "Customer: " + custt + "\n" + "Invoice ID:" + '%s'%inval
            inmaintainer()
            writer = writer + "DBFA Billing Framework" + "\n" + "One-stop solution for all your billing needs!" + "\n" + "\n" + "Billing time: " + dt_string + "\n" + "Customer ID: " + custt + "\n" + "----------------------------" + "\n" + "Invoice ID: " + '%s'%inval + "\n \n"
            global billiemaster
            billiemaster = 0 #variable for totalling the price
            time.sleep(0.0247) #for a seamless experience
            afac = 0
            dde_productlist = ""
            while(1):
                item = input("Enter product code: ")
                if item == "0":
                    break
                elif item in data:
                    ssxstockmaster(item)
                    if ssxvarscheck == 1:
                        billiemaster+=data[item]
                        dde_productlist += str(item) + '00'
                        print("Purchased: ", namie[item], " for: ", data[item])
                        repupdate(item)
                        lenxr = len(namie[item])
                        costlenxr = len(str(data[item]))
                        cj = 10 - costlenxr
                        pi = 60 - lenxr
                        idlerxx = namie[item] + " "*pi + "₹"+'%d'%data[item] + "*cj + "1 qty. ~"
                        purcheck += idlerxx
                        print("---")
                        priceprod = "₹" + '%d' % data[item]
```

```
                        logger.write("Appending product to order: \n")  #writing to file
                        profer.append(item)
                        logger.write(namie[item])
                        ssxstockmaintainer(item)
                        logger.write(" \n")
                        writer = writer + "\n Purchased: " + "\n" + namie[item] + "\n" + priceprod + "\n"
                        afac+=1
                    else:
                        print("Product currently out-of-stock. The inconvenience is regretted..\n")
                        print("---")
                        continue
                else:
                    print("Product not found. Please retry ")
                    print("---")

            #tax = int(input("Enter the net tax %: "))  #comment and uncomment tkinter lines to use GUI-based input
            time.sleep(0.15)  #for a seamless experience
            try:
                cponid = str(input("Enter voucher code (if any): "))
            except (EOFError, ValueError):
                pass

            if cponid != "":
                isol = sqlite3.connect(r'cponmgmtsys.db')
                isolx = isol.cursor()
                isolx.execute(("SELECT DISTINCT cponid from cponmaster WHERE cponid = ?"), (cponid, ))
                if isolx.fetchall() in ("", " ", [], (), None):
                    print("Invalid voucher identifier! Not applying any!")
                    discount = int(input("Enter discount % (if any): "))
                else:
                    cpon_limfetch(cponid)
                    print("")
                    print("----")
                    cpon_valfetch(cponid)
                    discount = int(valdock)
                    #print(valdock)
                    idler = "\nUsed DNSS voucher:\n" + str(cponid) + "\n \n"
                    writer = writer + idler
                    telethon = telethon + idler
                    cponuse(cponid)
            else:
                try:
                    discount = int(input("Enter discount % (if any): "))
                except:
                    discount = 0
            print(discount, "% net discount")
            time.sleep(0.15)  #for a seamless experience
            print("-----------------")
            time.sleep(0.15)  #for a seamless experience
            tota = ((billiemaster)-(((discount)/100)*billiemaster))
            global total
            total = (tota + ((tota/100)*18))
            if total != 0:
                discountx = '%d' % discount
```

```
                telethon = telethon + "\n" + "Tax amount: 18%" + "\n"  + "Discount: " + discountx + "%" + "\n" + "\n"
                writer = writer + "\n" + "\n" + "----------------------------" + "\n" + "Tax amount: 18%"  + "\n"  + discountx + "\n"  + "\n"
                delxfac = input("Enter *d* for delivery; skip for in-store purchase: ")
                if delxfac != "d":
                    payboxie(custt, total)
                    xpayboxie()
                else:
                    # Add a delivery
                    print("---- DBFA Deliveries ----")
                    delname = input("Customer's Name: ")
                    time.sleep(0.1)
                    print("Customer's Address:")
                    def getaddress():
                        print("         Enter/paste the address. Press Ctrl-Z ( windows ) to save it.")
                        contents = []
                        while True:
                            try:
                                line = input()
                            except EOFError:
                                break
                            contents.append(line)
                        address = ""
                        for i in contents:
                            address += i+", "
                        return address
                    addressx = getaddress()
                    print("Address entered: ", addressx)
                    addressfac = input("Confirm address or change? (y/n): ")
                    if addressfac == "y":
                        # Count pending deliveries
                        delcount = 0
                        filedel = open('./DBFAdeliveries.txt', 'r')
                        for line in filedel:
                            delcount+=1
                        filedel = open('./DBFAdeliveries.txt', 'a+')
                        strprofer = ""
                        for i in profer:
                            strprofer+='%s'%i
                        filedel.write("\ndel"+str(delcount+1) + "  Purchased: " + strprofer +"    " + addressx)
                        filedel.close()
                    if addressfac == "n":
                        getaddress()
                    elif addressfac not in ("y", "n"):
                        print("Invalid option! ")
                        pass
                    time.sleep(0.5)
                    time.sleep(0.5)
                    sfetch_values = ""
                    redeemindic = 0
                    writer += "\n\nDBFA Delivery\n\n"
                    payindic = "DBFA Delivery: PAY ON DELIVERY"
                    print("-----DBFA Delivery Ticket-----")
                    print("Customer: ", delname)
                    print("Delivery Address: ", addressx)
                    print("----------------------------")
```

```
                    telethon += "\n\nDBFA Delivery\n\n"
                    time.sleep(0.5)
                    print("Deliveries only support pay-on-delivery.")
                    time.sleep(0.5)
                    xrt = 0
                    netpay = total
                    lylpoints = 0

                if xrt == 1:
                    writer = writer + "----------------- BILLING CYCLE CANCELLED ---
----------------"
                    break
                else:
                    rupeesymbol = "₹".encode("utf-8")
                    if delxfac != "d":
                        saleslogger(custt, profer, netpay)
                    else:
                        saleslogger(custt, profer, total)
                    inmaintainer()
                    #infetch()
                    print("\n\n-----------------------------------------------------
--------------------------")
                    print("Invoice ID: ", inval, "| Time: ",dt_string, "| No. of
items: ", afac)
                    print(payindic)
                    print("-----------------------------------------------------
----------------------")
                    printobj = purcheck.split("~")
                    for i in printobj:
                        print(i)
                    print("-----------------------------------------------------
----------------------")
                    print("Sub total        : ₹",billiemaster)
                    cpon_ssinglefetch(cponid)
                    if sfetch_values not in (None, " ", ""):
                        print("Voucher used     :",sfetch_values)
                    discountstr = "Discount "+"("+'%d'%discount+"%)    :"
                    print(discountstr, "₹","%.2f" % (((discount)/100)*billiemaster))
                    print("IGST             : ₹","%.2f" % ((9/100)*billiemaster))
                    print("CGST             : ₹","%.2f" % ((9/100)*billiemaster))
                    if redeemindic == 1:
                        print("Redeemed loyalty points worth: ₹",lylpoints)
                    print("-----------------------------------------------------
----------------------")
                    print("Amount to be paid: ₹","%.2f" % netpay)
                    print("-----------------------------------------------------
----------------------")
                    toaster.show_toast("DFBA Billing:  Total billed for-
",str(total), duration = 1)
                    logger.write("Total amount billed for: \n") #writing to file
                    if custt in ("", " ", 0, None) and cccheck == 0:
                        pass
                    else:
                        writer += "Used DBFA loyalty points worth: " +
'%s'%lylpoints + "\n"
                    #regin.write("NET TOTAL: \n") #writing to file
                    telethon = telethon + "NET TOTAL: \n" + "₹" + str(netpay) + "\n"
                    writer = writer + "NET TOTAL: \n" + str(netpay) + "\n"
                    logger.write(str(total))

```

```
2377                    # delta DDE - Deep Archival Engine
2378                    from DBFADeepArchivalEngine import encoder_deeparchival
2379                    #Data Format: <order ids>///<customer name>///<customer
    id>///<date and time string>///tax///discount///loyalty///net
2380                    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object
    containing current date and time
2381                    if redeemindic == 1:
2382                        data = [dde_productlist, 'dde', custt, dt_string, 18,
    discount, lylpoints, netpay]
2383                    else:
2384                        data = [dde_productlist[:-1], 'dde', custt, dt_string, 18,
    discount, 0, netpay]
2385                    encoder_deeparchival(data, inval)
2386
2387                    logger.write("\n")
2388                    #regin.write(str(total))
2389                    #regin.write("\n")
2390                    updatescript(custt, total, billiemaster) #adds billed amount to
    the customer's record
2391                    abcd1+=1
2392                    afac+=1
2393                    now = datetime.now()
2394                    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object
    containing current date and time
2395                    daterey = (dt_string.replace("/","")).replace(":", "")
2396                    namer = 'DBFAinvid'+'%s'%inval+"-"+daterey+'.pdf'
2397                    can = SimpleDocTemplate(namer, pagesize=A4,
2398                                            rightMargin=2*cm,leftMargin=2*cm,
2399                                            topMargin=2*cm,bottomMargin=2*cm)
2400                    #can.setFont("MiLanProVF", 24)
2401                    can.build([Paragraph(writer.replace("\n", "<br />"),
    getSampleStyleSheet()['Normal']),])
2402
2403                    import shutil
2404                    source = namer
2405                    destination =
    r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\Generated_invoices'
2406                    dest = shutil.move(source, destination)
2407                    time.sleep(1.5) #for a seamless experience
2408
2409
2410                    if custt not in ("", " ", 0, "0") and cccheck == 0:
2411                        #Settings Checker
2412                        if settingscommonfetch(2) == 1:
2413                            emailfetch(custt)
2414                            print("Please wait..")
2415                            fromaddr = "billing.dbfa@gmail.com"
2416                            toaddr = custmail
2417                            msg = MIMEMultipart()
2418                            msg['From'] = fromaddr
2419                            msg['To'] = toaddr
2420                            msg['Subject'] = "Your DBFA Purchase Invoice"
2421                            body = """Thanks for making your purchase at
    DBFA!\n\nPlease find your invoice attached herewith.\n\nRegards\nThe DBFA Team"""
2422                            msg.attach(MIMEText(body, 'plain'))
2423                            filename = namer
2424                            attachment =
    open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\\Generated_invoices\\'+'
    %s'%namer, "rb")
2425                            attac= MIMEBase('application', 'octet-stream')
```

```
                            attac.set_payload((attachment).read())
                            encoders.encode_base64(attac)
                            attac.add_header('Content-Disposition', "attachment;
filename= %s" % filename)
                            msg.attach(attac)

 os.startfile(r'Process_Handlers\\emailprocesswindow.pyw')
                            email = smtplib.SMTP('smtp.gmail.com', 587)
                            email.starttls()
                            email.login(fromaddr, "dbfaidlepass")
                            message = msg.as_string()
                            email.sendmail(fromaddr, toaddr, message)
                            print("Invoice mailed. ")
                            print("-------------------------------------------------
------------------------\n\n")
                        #Settings Checker
                        elif settingscommonfetch(1) == 2:
                            pass
                    #regin.close()
                    with HiddenPrints():
                        try:
                            sender = telegram_bot_sendtext(telethon)
                            print(sender)
                        except Exception:
                            pass

                    print()

            else:
                print("\n\n-------------------------------------------------------
----------------")
                print("No purchase made.")
                print("-------------------------------------------------------------
------------")
                print("Amount           : ₹",billiemaster)
                print("-------------------------------------------------------------
------------")
                print("Amount to be paid: ₹","%.2f" % total)
                print("-------------------------------------------------------------
------------")
                toaster.show_toast("DBFA: No purchase made  ",str(total), duration =
1)
                logger.write("Billing cancelled: Reason: Net amount null\n")
#writing to file
                #regin.write("NET TOTAL: \n") #writing to file
                telethon = telethon + "Billing cancelled. Net amount null." + "\n"
                writer = writer + "BILLING CALCELLED. No purchase made." + "\n"
                logger.write(str(total))
                logger.write("\n")
                abcd1+=1
                afac+=1
                now = datetime.now()

                #regin.close()
                with HiddenPrints():
                    try:
                        sender = telegram_bot_sendtext(telethon)
                        print(sender)
                    except Exception:
                        pass
```

```
2477
2478
2479        #Manage Customers
2480        elif decfac == "2":
2481            print("Select: ")
2482            print("    a: Register a Customer ")
2483            print("    b: Customer Registry ")
2484            print("    c: Customer Purchase Records ")
2485            print("    d: Find a Customer ")
2486            print("    e: Export Records as CSV \n")
2487            selected = input("What would you like to do? ")
2488            print("\n")
2489            if selected in ("a", "A"):
2490                del2a()
2491                logger.write("---------------------------------------- \n")
2492                logger.write("  \n")
2493                logger.write("Date and time: ") #including the date and time of
    billing (as taken from the system)
2494                logger.write(dt_string)
2495                logger.write(" \n")
2496                logger.write("New customer registered: ")
2497                x = " custname: " + custname + " custemail: " + email + "\n"
2498                logger.write(x)
2499                logger.write("---------------------------------------- \n")
2500
2501            if selected in ("b", "B"):
2502                del2b()
2503                logger.write("---------------------------------------- \n")
2504                logger.write("  \n")
2505                logger.write("Date and time: ") #including the date and time of
    billing (as taken from the system)
2506                logger.write(dt_string)
2507                logger.write(" \n")
2508                logger.write("Customer registry accessed! \n")
2509                logger.write("---------------------------------------- \n")
2510
2511            if selected in ("c", "C"):
2512                del2c()
2513                logger.write("---------------------------------------- \n")
2514                logger.write("\nDBFA Customer Purchase Records accessed! \n")
2515
2516            elif selected in ("d", "D"):
2517                del2d()
2518                logger.write("---------------------------------------- \n")
2519                logger.write("\nCustomer search used. \n")
2520
2521            elif selected in ("e", "E"):
2522                del2e()
2523                logger.write("\n\n---------------------------------------- \n")
2524                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2525                logger.write("Customer data exported to CSV! ")
2526                with HiddenPrints():
2527                    try:
2528                        sender = telegram_bot_sendtext(dt_string + "\n" + "Sales
    data exported to CSV- REDFLAG Urgent Security Notice!")
2529                        print(sender)
2530                    except Exception:
2531                        pass
2532                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2533                logger.write("---------------------------------------- \n\n\n")
```

```
2534
2535            elif selected not in ("a", "b", "c", "d", "e", "A", "B", "C", "D", "E"):
2536                print("Please chose a valid option!")
2537                time.sleep(1)
2538
2539        #Store Options:
2540        elif decfac == "3":
2541            print("Select: ")
2542            print("    a: Manage Stock ")
2543            print("    b: DBFA Stock Master ")
2544            print("    c: Manage Vouchers ")
2545            print("    d: Product Listing ")
2546            print("    e: Sales Log ")
2547            print("    f: Export Sales Data as CSV ")
2548            print("    g: Invoice Deep Archive \n")
2549            storeselected = input("What would you like to do? ")
2550            print("\n")
2551
2552            if storeselected in ("a", "A"):
2553                del3a()
2554
2555            elif storeselected in ("b", "B"):
2556                del3b()
2557
2558            elif storeselected in ("c", "C"):
2559                del3c()
2560
2561            elif storeselected in ("d", "D"):
2562                del3d()
2563
2564            elif storeselected in ("e", "E"):
2565                del3e()
2566                logger.write("\n-------------------------------------- \n")
2567                logger.write("Sales log accessed! ")
2568
2569            elif storeselected in ("f", "F"):
2570                del3f()
2571                logger.write("\n\n-------------------------------------- \n")
2572                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2573                with HiddenPrints():
2574                    try:
2575                        sender = telegram_bot_sendtext(dt_string + "\n" + "Customer
data exported to CSV- REDFLAG Urgent Security Notice!")
2576                        print(sender)
2577                    except Exception:
2578                        pass
2579                logger.write("Sales data exported to CSV! ")
2580                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2581                logger.write("-------------------------------------- \n\n\n")
2582
2583            elif storeselected in ("g", "G"):
2584                print("Deep Archival Engine")
2585                del3g()
2586
2587            elif storeselected not in ("a", "b", "c", "d", "e", "f", "g", "A", "B",
"C", "D", "E", "F", "G"):
2588                print("Please select a valid option! ")
2589                time.sleep(1)
2590                mainmenu()
2591
```

```
2592
2593
2594        #Reports
2595        elif decfac == "4":
2596            command = "cls"
2597            os.system(command)
2598            print("-- Generating store report --")
2599            repstockfetch()
2600            print("Please wait...")
2601            repdatafetch()
2602            findMaximum = "select max(prodsales) from recmasterx"
2603            recx.execute(findMaximum)
2604            # Print the maximum score
2605            netr = recx.fetchone()[0]
2606            findin = "select prodid, prodname, prodprofit, prodsales, netprof from
    recmasterx WHERE prodsales = ?"
2607            arterx = (str(int(netr)),)
2608            recx.execute(findin, arterx)
2609            arterxout = recx.fetchall()
2610
2611            findMaximumProf = "select max(netprof) from recmasterx"
2612            recx.execute(findMaximumProf)
2613            xnetr = recx.fetchone()[0]
2614            findin = "select prodid, prodname, prodprofit, prodsales, netprof from
    recmasterx WHERE netprof = ?"
2615            xarterx = str(int(xnetr))
2616            xxarterx = (xarterx,)
2617            recx.execute(findin, xxarterx)
2618            xarterxout = recx.fetchall()
2619
2620            isol = sqlite3.connect(r'DBFA_vend.db')
2621            isolx = isol.cursor()
2622            isolx = isol.cursor()
2623            isolx.execute(("SELECT prodid, prodname, ordqty, delstat, vendor from
    stock WHERE delstat = ?"), ("TBD", ))
2624            rowsrec = isolx.fetchall()
2625            col_labels = [("P. ID", "P. Name", "Qty. to be delivered", "Status",
    "Vendor")]
2626            rowsxtb = col_labels + rowsrec
2627
2628            def add_page_number(canvas, doc):
2629                canvas.saveState()
2630                canvas.setFont('Times-Roman', 10)
2631                page_number_text = "%d" % (doc.page)
2632                canvas.drawCentredString(
2633                    0.75 * inch,
2634                    0.75 * inch,
2635                    page_number_text
2636                )
2637                canvas.restoreState()
2638
2639            import sqlite3 as sql
2640
 csvexx=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_CUST
    CC.db')
2641            print("Fetching data from database - II...")
2642            cursorx = csvexx.cursor()
2643            axct = cursorx.execute("select * from custcc")
2644            axctx = []
2645            for i in axct:
```

```
2646                axctx.append(i)
2647            axctx = [("C. ID","C. Name","Purchases Made","Total Amount","Loyalty
    Points")] + axctx
2648            doc = SimpleDocTemplate("dbfastorerep.pdf", pagesize=A4,
2649                                                rightMargin=2*cm,leftMargin=1.5*cm,
2650                                                topMargin=1*cm,bottomMargin=2*cm)
2651            # container for the 'Flowable' objects
2652            elements = []
2653            telegram_bot_sendtext("Access alert: Store Report")
2654            t1dot = ("<b>DBFA Automatic Store Report: </b> <br />This report has
    been automatically generated. This lists the profit earned, stock analytics and
    customer records as logged by DBFA.<br /><br />")
2655            t2dot = ("DBFA synchronously updates its database alongwith algorithmic
    data interpretation to deliver these reports. <br />This report contains information
    from the start of using DBFA on this system.<br /><br />")
2656            t6dot = ("Report generated on: " + dt_string)
2657            t3dot = ("<br /><br /><b>Most sold listing: </b><br />")
2658            t4dot = ("<br /><br /><b>Total profit per listing: </b><br /><br />")
2659            t5dot = ("<br /><br /><b>Most profit making listing: </b><br /><br />")
2660            t8dot = ("<br /><br /><b>Customer purchases: </b><br /><br />")
2661            t10dot = ("<br /><br /><br /><br /><br /><br /><b>DBFA Stock Orders
    Report: </b><br />Product stock yet to be recieved: <br /><br />")
2662            t11dot = ("<br /><br /><b>DBFA Sales Analysis Plotter: </b><br />DBFA
    uses advanced data analysis algorithms to generate this plot. This is as per the
    latest data sets available. <br />")
2663            colas = (30, 300, 60, 50, 50)
2664            rowheights = (20, 20, 20, 20, 20,  20, 20, 20, 20, 20, 20, 20, 20, 20,
    20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,20,20,20, 20, 20, 20, 20, 20,
    20, 20, 20, 20, 20 )
2665            text1=Paragraph(t1dot)
2666            text2=Paragraph(t2dot)
2667            text3=Paragraph(t3dot)
2668            text4=Paragraph(t4dot)
2669            text5=Paragraph(t5dot)
2670            text6=Paragraph(t6dot)
2671            text8=Paragraph(t8dot)
2672            text10=Paragraph(t10dot)
2673            text11=Paragraph(t11dot)
2674            x=Table(rows, colas, rowheights)
2675            t=Table(arterxout)
2676            t2=Table(xarterxout)
2677            t4=Table(axctx)
2678            t5=Table(rowsxtb)
2679            if tabarter == ['--']:
2680                t7dot = ("<br /><br /><b>All listings currently in stock!</b><br />
    <br />")
2681            else:
2682                t7dot = ("<br /><br /><b>Products running low on stock: </b><br />
    <br />")
2683
2684
2685            tblStyle = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
2686                                   ('VALIGN',(0,0),(-1,-1),'TOP'),
2687                                   ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
2688                                   ('BOX',(0,0),(-1,-1),1,colors.black),
2689                                   ('BOX',(0,0),(0,-1),1,colors.black)])
2690            tblStyle.add('BACKGROUND',(0,0),(1,0),colors.lightblue)
2691            tblStyle.add('BACKGROUND',(0,1),(-1,-1),colors.lightblue)
2692            GRID_STYLE = TableStyle(
2693                [('GRID', (0,0), (-1,-1), 0.25, colors.black),
```

```
2694                ('ALIGN', (1,1), (-1,-1), 'LEFT')]
2695                )
2696            text7=Paragraph(t7dot)
2697            t3=Table(tabarter)
2698            t.setStyle(GRID_STYLE)
2699            t2.setStyle(GRID_STYLE)
2700            t3.setStyle(GRID_STYLE)
2701            t4.setStyle(GRID_STYLE)
2702            t5.setStyle(GRID_STYLE)
2703            x.setStyle(GRID_STYLE)
2704            # ------------------------------------------------------------------------
---------
2705            import sqlite3, time
2706            import matplotlib.pyplot as plt
2707            salesr = sqlite3.connect(r'dbfasales.db')
2708            salesx = salesr.cursor()
2709            datefetch = []
2710
2711            salesx.execute("SELECT DISTINCT date FROM sales")
2712            for i in salesx.fetchall():
2713                datefetch.append((i[0]))
2714
2715            netray = []
2716            for i in datefetch:
2717                salesx.execute(("SELECT sum(prof) FROM sales WHERE date = ?"), (i,
))
2718                netray.append(salesx.fetchall()[0][0])
2719
2720            # Plotting
2721            plt.plot(datefetch, netray, color='purple', linestyle='dashed',
linewidth = 3,
2722                     marker='o', markerfacecolor='magenta', markersize=12)
2723            # naming the x axis
2724            plt.xlabel('Date')
2725            # naming the y axis
2726            plt.ylabel('Profit')
2727            # Graph Title
2728            plt.title('DBFA Profit Report')
2729            time.sleep(1)
2730            # Finally, display
2731            plt.savefig('DBFAplot.png', dpi=300, bbox_inches='tight')
2732            # ------------------------------------------------------------------------
---------
2733
2734            delI = Image('DBFAplot.png')
2735            delI.drawHeight =  4.2*inch
2736            delI.drawWidth = 5.5*inch
2737            elements.append(text1)
2738            elements.append(text2)
2739            elements.append(text6)
2740            elements.append(text3)
2741            elements.append(t)
2742            elements.append(text5)
2743            elements.append(t2)
2744            elements.append(text7)
2745            elements.append(t3)
2746            elements.append(text4)
2747            elements.append(x)
2748            elements.append(text8)
2749            elements.append(t4)
```

```
            elements.append(text10)
            elements.append(t5)
            elements.append(text11)
            elements.append(delI)
            # write the document to disk
            doc.build(elements,
                onFirstPage=add_page_number,
                onLaterPages=add_page_number,)
            print("Report Created. ")
            for i in tqdm (range (100), desc="Publishing Report: "):
                time.sleep(0.00001)
            print("Opening store report now")
            os.startfile('dbfastorerep.pdf')
            time.sleep(2)


        #DBFA Backup&Switch
        elif decfac == "7":
            os.startfile(r'delauth.py')
            time.sleep(0.3)
            os._exit(0)


        #License
        elif decfac == "9":
            print("Fetching latest licensing information.......")
            print(" ")
            print(" ")
            logoprintxrt()
            time.sleep(1.5)
            print(" ")
            print(" ")
            print("DBFA by Pranav Balaji, 2020")
            print(" ")
            print("_______ Licensing _______")
            print("           GNU PUBLIC LICENSE - TERMS AND CONDITIONS")
            print("    <deltaBillingFramework>  Copyright (C) 2020 Pranav Balaji and
Sushant Gupta")
            print("    This program comes with ABSOLUTELY NO WARRANTY; for details
type *show w*.")
            print("    This is free software, and you are welcome to redistribute
it")
            print("    under certain conditions; type *show c* for details. ")
            toaster.show_toast("DFBA Framework Runtime Broker", "©2020: DBFA by
Pranav Balaji and Sushant Gupta", duration = 1.5)
            print(" ")
            print(" ")
            print("Visit: www.github.com/deltaonealpha/deltaBillingFramework for
complete licensing terms. ")
            print(" ")
            print(" ")
            time.sleep(2)
            webbrowser.open('https://telegra.ph/DBFA-Licensing-Information-08-16')
            print("-------------------------------------------------")


        #Stock Ordering Option
        elif decfac == "5":
            print("------------------ DBFA DELIVERY MANAGER ------------------")
            time.sleep(2)
```

```
2805            print("For issuing new delivery orders, use the invoicing option")
2806            print("------------------------------------------------------------")
2807            print("a: View existing deliveries")
2808            print("b: Show delivery count")
2809            print("c: Confirm a delivery")
2810            delfacx = input("Select: ")
2811            if delfacx in ("a", "A"):
2812                # Fetch Data
2813                print("Currently pending deliveries::  \n")
2814                filedel = open('./DBFAdeliveries.txt', 'r')
2815                for line in filedel:
2816                    print(line)
2817                print("\n\n")
2818                time.sleep(2)
2819            if delfacx in ("b", "B"):
2820                # Pending Delivery Count
2821                delcount = 0
2822                filedel = open('./DBFAdeliveries.txt', 'r+')
2823                for line in filedel:
2824                    delcount+=1
2825                filedel.close()
2826                if delcount != 0:
2827                    print("Number of pending deliveries: ", delcount)
2828                else:
2829                    print("No deliveries pending! ")
2830                time.sleep(2)
2831            if delfacx in ("c", "C"):
2832                # Show data and Remove delivery record
2833                print("Current delivery data::  \n")
2834                filedel = open('./DBFAdeliveries.txt', 'r')
2835                for line in filedel:
2836                    print(line)
2837                print("\n\n")
2838                cleanstr = ""
2839                deledid = input("Enter the delivery ID to remove (EXAMPLE: *del2*):
")
2840                delcount = 0
2841                filedel = open('./DBFAdeliveries.txt', 'r')
2842                for line in filedel:
2843                    if deledid in line:
2844                        line = ""
2845                    else:
2846                        delcount+=1
2847                    cleanstr+=line
2848                filedel.close()
2849                filedel = open('./DBFAdeliveries.txt', 'w+')
2850                filedel.write(cleanstr)
2851                filedel.close()
2852                filedel = open('./DBFAdeliveries.txt', 'r')
2853                print("Delivery", deledid, "completed! ")
2854                time.sleep(2)
2855
2856            elif delfacx not in ("a", "b", "c", "A", "B", "C"):
2857                print("Invalid option selected! \n\n")
2858                time.sleep(2)
2859                mainmenu()
2860
2861        #DevChangelog Option
2862        elif decfac == "10":
2863            print("\n\nLatest Development Changelog: \n")
```

```
            webbrowser.open('https://telegra.ph/DBFA-8-RC2-Highlights-08-17')
            webbrowser.open('https://telegra.ph/DBFA-8-Release-Candidate---1-08-16')
            print("-------------------------------------------------")
            time.sleep(2)

        #DBFA Settings - Currently in development
        elif decfac == "6":
            def transitionprogress():
                from colorama import init, Fore, Back, Style
                os.system("cls")
                time.sleep(1)
                print(Fore.WHITE+'|'+Fore.RED+'███ OFF |')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.RED+'██     '+Fore.GREEN+'█|')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.RED+'█     '+Fore.GREEN+'██|')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.RED+'     '+Fore.GREEN+'███|')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.GREEN+ ' ON  ███|'+Fore.WHITE)
                time.sleep(1.24)
                os.system("cls")


            def transitionprogressneg():
                from colorama import init, Fore, Back, Style
                os.system("cls")
                time.sleep(1)
                print(Fore.WHITE+'|'+Fore.GREEN+ ' ON  ███|')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.RED+'     '+Fore.GREEN+'███|')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.RED+'█     '+Fore.GREEN+'██|')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.RED+'██     '+Fore.GREEN+'█|')
                time.sleep(0.3)
                print(Fore.WHITE+'|'+Fore.RED+'███ OFF |'+Fore.WHITE)
                time.sleep(1.24)
                os.system("cls")

            os.system("cls")

            def settingsmenu():
                from colorama import init, Fore, Back, Style
                while(1):
                    import time
                    time.sleep(0.2)

 print("
")
                    print(" -------------------------  DBFA Settings  ------------
-------------    ")

                    if (settingscommonfetch(1)) == 1:
                        print(" 1:    Display boot image
  :", '| ON '+Fore.GREEN+'███'+Fore.WHITE+'|      ')
                    else:
                        print(" 1:    Display boot image
  :", ('|'+Fore.RED+'███'+Fore.WHITE+' OFF|      '))
                    if (settingscommonfetch(2)) == 1:
```

```
                        print(" 2:    Email invoice to registered customers
 :", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      ')
                    else:
                        print(" 2:    Email invoice to registered customers
 :", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      '))
                    if (settingscommonfetch(3)) == 1:
                        print(" 3:    Enable DBFA Music Controls (beta):
:", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      ')
                    else:
                        print(" 3:    Enable DBFA Music Controls (beta):
:", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      '))
                    if (settingscommonfetch(4)) == 1:
                        print(" 4:    Open CSV when exported
:", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      ')
                    else:
                        print(" 4:    Open CSV when exported
:", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      '))
                    if (settingscommonfetch(5)) == 1:
                        print(" 5:    Enable database encryption
:", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      '+Fore.RED)
                        print(" ")
                    else:
                        print(" 5:    Enable database encryption
:", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      ')+Fore.RED)
                        print(" ")
                    if (settingscommonfetch(6)) == 1:
                        print(" 6:    Enable DBFA Secure Two-Factor-Authenication
 :", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      ')
                        print(" ")
                    else:
                        print(" 6:    Enable DBFA Secure Two-Factor-Authenication
 :", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      '))
                        print(" ")
                    if (settingscommonfetch(7)) == 1:
                        print(" 7:    Use new DBFA Menu style
 :", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      '+Fore.RED)
                        print(" ")
                    else:
                        print(" 7:    Use new DBFA Menu style
 :", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      ')+Fore.RED)
                        print(" ")

                    print(Fore.MAGENTA+" 8:    Create DBFA Desktop Shortcut
          :"+Fore.WHITE, '|'+Fore.MAGENTA+"[redacted] Proceed > "+Fore.WHITE+"| ")

                    print(Fore.RED+" 9:    Delete customer records
       :"+Fore.WHITE, '|'+Fore.RED+"[redacted] Proceed > "+Fore.WHITE+"| ")
                    print(Fore.RED+" 10:   Delete store records
        :"+Fore.WHITE, '|'+Fore.RED+"[redacted] Proceed > "+Fore.WHITE+"| ")
                    print(Fore.MAGENTA+" 11:   Check for updates
           :"+' |'+"[redacted] Proceed > "+Fore.WHITE+"|  " )
                    print("
                ")
                    print(Fore.RED+" 12:   Return to Main Menu
      :"+' |'+"[redacted] Proceed > "+Fore.WHITE+"| " )
                    print("
                ")

#print("[redacted]")
```

```
2958                    settfac = input("What would you like to do? ")
2959                    if settfac == "1":
2960                        print('''DBFA displays an image for 2 seconds when it is
    started.
2961                        This image changes with each major iteration of DBFA.
2962                        Displaying this image let's us prepare files in the
    background so that DBFA runs smoothly once its started.
2963                        Disabling this option may lead to errors. Continue? ''')
2964                        print(" ")
2965                        print("Display DBFA boot image? ")
2966                        print("y:    ",  '| ON '+Fore.GREEN+'███'+Fore.WHITE+'|')
2967                        settfac1x = input(("n:     "+
    '|'+Fore.RED+'███'+Fore.WHITE+' OFF|: '))
2968                        if settfac1x == "y":
2969                            settingsmodifier(1, 1)
2970                            transitionprogress()
2971                            print("DBFA will now display its boot image when it
    prepares the backend on boot. ")
2972                            print("")
2973                            time.sleep(1)
2974                            settingsmenu()
2975                        elif settfac1x == "n":
2976                            settingsmodifier(1, 0)
2977                            transitionprogressneg()
2978                            print("DBFA won't display its boot image when it
    prepares the backend on boot from now. ")
2979                            print("")
2980                            time.sleep(1)
2981                            settingsmenu()
2982                        else:
2983                            print("That's an invalid input... ")
2984                            print("")
2985                            time.sleep(1)
2986                            settingsmenu()
2987
2988                    elif settfac == "2":
2989                        print('''DBFA creates an invoice on each billing cycle
2990                        If the customer account in-use is registered with DBFA, the
    invoice is E-Mailed to the same.
2991                        Disabling this option will stop DBFA from E-Mailing
    customers with their invoice from now.''')
2992                        print(" ")
2993                        print("E-Mail registered customers their invoice? ")
2994                        print("y:    ",  '| ON '+Fore.GREEN+'███'+Fore.WHITE+'|')
2995                        settfac1x = input(("n:     "+
    '|'+Fore.RED+'███'+Fore.WHITE+' OFF|: '))
2996                        if settfac1x == "y":
2997                            settingsmodifier(2, 1)
2998                            transitionprogress()
2999                            print("DBFA will continue E-Mailing customers with their
    invoice. ")
3000                            print("")
3001                            time.sleep(1)
3002                            settingsmenu()
3003                        elif settfac1x == "n":
3004                            settingsmodifier(2, 0)
3005                            transitionprogressneg()
3006                            print("DBFA will stop E-Mailing customers their invoice
    from now on. ")
3007                            print("")
```

```
3008                            time.sleep(1)
3009                            settingsmenu()
3010                        else:
3011                            print("That's an invalid input... ")
3012                            print("")
3013                            time.sleep(1)
3014                            settingsmenu()
3015
3016                    elif settfac == "3":
3017                        print('''In our mission of making DBFA the ultimate space to
    control your entire store and its functioning,
3018                        we keep adding tiny tid-bits to make that process even
    easier.
3019                        DBFA Music Controls is one such feature introduced in DBFA 8
    RC3x (IB3).
3020                        When you disable this functionality:
3021                                - The currently-playing track will no longer be
    displayed.
3022                                - DBFA Music Controls, including but not limited to
    pause/play, prev and next will be restricted. ''')
3023                        print(" ")
3024                        print("Enable DBFA Music Controls? ")
3025                        print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
3026                        settfac1x = input(("n:     "+
    '|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|: '))
3027                        if settfac1x == "y":
3028                            settingsmodifier(3, 1)
3029                            transitionprogress()
3030                            print("DBFA Music Controls Service will be started with
    the next menu-cycle. ")
3031                            print("")
3032                            time.sleep(1)
3033                            settingsmenu()
3034                        elif settfac1x == "n":
3035                            settingsmodifier(3, 0)
3036                            transitionprogressneg()
3037                            print("DBFA Music Controls Service will be restricted
    from the next menu-cycle.")
3038                            print("")
3039                            time.sleep(1)
3040                            settingsmenu()
3041                        else:
3042                            print("That's an invalid input... ")
3043                            print("")
3044                            time.sleep(1)
3045                            settingsmenu()
3046
3047
3048                    elif settfac == "4":
3049                        print("CSV files once generated are auto-opened in your
    default worksheet app")
3050                        print("Example: Microsoft Excel, LibreOffice Calc, Google
    Docs, et cetera.")
3051                        print(" ")
3052                        print("Open file after export? ")
3053                        print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
3054                        settfac1x = input(("n:     "+
    '|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|: '))
3055                        if settfac1x == "y":
3056                            settingsmodifier(4, 1)
```

```
3057                            transitionprogress()
3058                            print("DBFA will now open CSV files when exported on
request. ")
3059                            print("")
3060                            time.sleep(1)
3061                            settingsmenu()
3062                        elif settfac1x == "n":
3063                            settingsmodifier(4, 0)
3064                            transitionprogressneg()
3065                            print("DBFA will not open CSV files when exported from
now on. ")
3066                            print("")
3067                            time.sleep(1)
3068                            settingsmenu()
3069                        else:
3070                            print("That's an invalid input... ")
3071                            print("")
3072                            time.sleep(1)
3073                            settingsmenu()
3074
3075                    elif settfac == "5":
3076                        print('''In our process of phasing-out .txt based storage in
favour of sqlite storage,
3077                        we at DBFA are trying to make our files even tougher to
access than ever before without valid credentials.
3078
3079                        DBFA is currently experimenting with sqlcipher encryption
for it's sqlite databases.
3080                        Please note that this functionality is a part of DBFA
internal test builds for now,
3081                        and is not ready for public rollout.
3082
3083                        This process might impact DBFA's data integrity. We
recommend you to run *DBFA Backup&Switch* from option *5*
3084                        before you attempt to encrypt/ decrypt DBFA databases by
running this command.''')
3085                        print(" ")
3086                        print("Enable DBFA database encryption? ")
3087                        print("y:    ",  '| ON '+Fore.GREEN+'   '+Fore.WHITE+'|')
3088                        settfac1x = input(("n:     "+
'|'+Fore.RED+'   '+Fore.WHITE+' OFF|: '))
3089                        if settfac1x == "y":
3090                            settingsmodifier(5, 1)
3091                            transitionprogress()
3092                            print('''DBFA will attempt to encrypt it's databases
when restarted.
3093                            This process may fail, as this *internal test build* of
DBFA currently has encryption as a beta feature.''')
3094                            print("")
3095                            time.sleep(1)
3096                            settingsmenu()
3097                        elif settfac1x == "n":
3098                            settingsmodifier(5, 0)
3099                            transitionprogressneg()
3100                            print('''DBFA will attempt to de-crypt it's databases on
the next restart.
3101                            This process may fail, as this *internal test build* of
DBFA currently has encryption as a beta feature.
3102
```

```
                            If DBFA databases are already decrypted, no change will
take place and data integrity will be untouched.''')
                            print("")
                            time.sleep(1)
                            settingsmenu()
                        else:
                            print("That's an invalid input... ")
                            print("")
                            time.sleep(1)
                            settingsmenu()

                    elif settfac == "6":
                        print("----DBFA 2FA MANAGER----")
                        print("Two-factor authenication is a widely-used method
helpful in securing accounts when their passwords get compromised.")
                        print("With DBFA, you can choose between Telegram and Google
Authenicator as a medium to recieve these 2FA requests. ")
                        print(" ")
                        print("DBFA randomly generates these OTPs/ requests and
sends them via a secure and encrypted connection.")
                        time.sleep(2)
                        print(" ")
                        print(" ")
                        print("Please do note that enabling/ disabling 2FA will
reboot DBFA Store Manager!!")
                        print(" ")
                        print(" ")
                        print("Available authenication methods: ")
                        print("1: Telegram Authenication")
                        import os
                        print("2: Google Authenicator (alpha; experimental)")
                        print("3/ skip: Exit to settings menu")
                        authfac = input("What would you like?: ")
                        if authfac == "1":
                            print("Connecting to the Telegram Web API..")
                            print("To turn on/ off DBFA 2FA, you need to authenicate
with 2FA first.")
                            time.sleep(0.5)
                            os.startfile('modif2fa.py')
                            time.sleep(1)
                            os._exit(0)

                        if authfac == "2":
                            print("Loading Django framework..")
                            print("This option is currently under development!")
                            print(" ")
                            time.sleep(2)
                        else:
                            print("Please choose a valid option! ")
                            print(" ")

                    elif settfac == "7":
                        print("This option let's you switch between the older DBFA
menu-style")
                        print("and the newer one as introduced with DBFA 8.12")
                        print("\nFor the best visual experience with DBFA, we
recommend you to use the newer design.\n\n")
                        print("DBFA Menu-Style: ")
                        print("1: Use new style (recommended)")
                        print("2: Use old style")
```

```
3155                        msdsfac = input("Please make a choice: ")
3156                        if msdsfac == "1":
3157                            settingsmodifier(7, 1)
3158                            transitionprogress()
3159                            print("New menu style applied! ")
3160                        if msdsfac == "2":
3161                            print("Old menu style")
3162                            settingsmodifier(7, 0)
3163                            transitionprogressneg()
3164                            print("Old menu style applied..! ")
3165                        else:
3166                            print("Please choose a valid option! ")
3167                            print(" ")
3168
3169
3170
3171                    elif settfac == "8":
3172                        import shutil
3173                        import os
3174                        desktop =
os.path.join(os.path.join(os.environ['USERPROFILE']), 'OneDrive\Desktop')
3175                        # Prints: C:\Users\sdkca\Desktop
3176                        print("Shortcut will be created at: " + desktop)
3177                        try:
3178                            original =
r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\Assets\run_DBFA.lnk'
3179                            shutil.copy(original, desktop)
3180                            print("Executed. ")
3181                        except:
3182                            print("DBFA Permission Error: Can't get perms to execute
in directory! ")
3183
3184                    elif settfac == "9":
3185                        print('''This option PERMANENTLY CLEARS ALL DBFA CUSTOMER
RECORDS.
3186                        This includes their registration data, purchase records, and
loyalty points.
3187
3188                        This execution can NOT BE REVERSED.
3189                        DATA INTEGRITY MAY BE LOST during this process.
3190                        Proceed with caution! ''')
3191                        print(" ")
3192                        print("ERASE DBFA customer records PERMANENTLY? ")
3193                        print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
3194                        settfac1x = input(("n:     "+
'|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|: '))
3195
3196                        if settfac1x == "y":
3197                            print("DBFA will now reboot itself to finish applying
changes.")
3198                            time.sleep(0.5)
3199                            transitionprogress()
3200                            # window.close()
3201                            os.startfile(r'securepack.py')
3202                            time.sleep(1)
3203                            os._exit(0)
3204
3205
3206                            settingsmenu()
3207                        elif settfac1x == "n":
```

```
3208                            print("Denied. ")
3209                            settingsmenu()
3210                        else:
3211                            print("That's an invalid input... ")
3212                            print("")
3213                            time.sleep(1)
3214                            settingsmenu()
3215
3216                    elif settfac == "10":
3217                        print('''This option PERMANENTLY CLEARS ALL DBFA VOUCHERS/
    COUPONS
3218                        All current vouchers/ coupons WILL BE LOST.
3219                        Vouchers already issued will become redundant unless
    manually re-added again.
3220                        Validity and usage limits will be lost for all voucher/
    coupon instanced recorded by DBFA.
3221
3222                        However, DBFA's logged voucher/ coupon usage will continue
    to exist in memory and will not be erased.
3223
3224                        This execution can NOT BE REVERSED.
3225                        DATA INTEGRITY MAY BE LOST during this process.
3226                        Proceed with caution! ''')
3227                        print(" ")
3228                        print("ERASE DBFA voucher/ coupon records PERMANENTLY? ")
3229                        print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
3230                        settfac1x = input(("n:     "+
    '|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|: '))
3231                        if settfac1x == "y":
3232                            print("DBFA will now reboot itself to finish applying
    changes.")
3233                            time.sleep(0.5)
3234                            transitionprogress()
3235                            # window.close()
3236                            os.startfile(r'securepackxvc.py')
3237                            time.sleep(1)
3238                            os._exit(0)
3239
3240
3241                            settingsmenu()
3242                        elif settfac1x == "n":
3243                            print("Denied.")
3244
3245
3246                            settingsmenu()
3247                        else:
3248                            print("That's an invalid input... ")
3249                            print("")
3250                            time.sleep(1)
3251                            settingsmenu()
3252
3253
3254                    elif settfac == "11":
3255                        print("DBFA Updater is currently in the making. ")
3256                        print("You'll be notified immediately this feature is
    enabled. ")
3257                        print("Support for this will come with a future update. ")
3258                        settingsmenu()
3259
3260
```

```
3261                    elif settfac == "11":
3262                        break
3263                        break
3264                        break
3265
3266                    else:
3267                        print("That's an invalid input... ")
3268                        print("")
3269                        time.sleep(1)
3270                        break
3271                        break
3272                        break
3273
3274            settingsmenu()
3275
3276        #Profit Graph Plotter
3277        elif decfac == "8":
3278            time.sleep(0.5)
3279            print("---- DBFA Sales Analyzer Engine v1 ----")
3280            time.sleep(0.5)
3281            print("In DBFA's plotter, you can zoom in/out of the graph, adjust plot
    dimensions and export the plot to a .png file\n")
3282            time.sleep(1)
3283            print("Please wait while we analyze store sales..\n\n")
3284            time.sleep(2)
3285            print("A new window will be shortly opened. ")
3286            print("You're requested to close the same when you want to return to
    DBFA's main menu.\n")
3287            time.sleep(1.7)
3288            os.startfile(r'plotter.pyw')
3289
3290
3291        #DBFA Updater
3292        elif decfac == "11":
3293            import requests, os, time, shutil, oschmod
3294            os.system('cls')
3295            print("---------------------------------\n\n( つ ◕_◕ )つ delta Update
    Utility\n\n---------------------------------")
3296            time.sleep(1)
3297            url = "https://raw.github.com/deltaonealpha/DBFA/master/updates.txt"
3298            r = requests.get(url)
3299            dbfaver = ((str(r.content.decode('utf-8'))))[4:]
3300            xdbfaver = ((str(r.content.decode('utf-8'))))
3301
3302            with open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\updates.txt',
    'r+') as upread:
3303                upread = (str(upread.read())).strip()
3304
3305            print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
3306            time.sleep(1)
3307            spass1 = []
3308            spass2 = []
3309            for i in dbfaver:
3310                spass1.append(i)
3311            for j in upread[4: ]:
3312                spass2.append(j)
3313            if float(upread[4: ]) > float(dbfaver):
3314                pass
3315
3316            else:
```

```
3317                if xdbfaver == upread:
3318                    print("This installation of DBFA is up-to date! ")
3319
3320                elif spass1 != spass2:
3321                    time.sleep(1)
3322                    print("A new DBFA update is available: DBFA", dbfaver)
3323                    time.sleep(3)
3324                    updateconfo =  input("Update DBFA now? (y/n): ")
3325                    if updateconfo == "y":
3326                        try:
3327
 oschmod.set_mode(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_Update
Handler", "777")
3328                        except:
3329                            pass
3330
 shutil.rmtree(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHan
dler', ignore_errors=True)
3331                        try:
3332
 os.rmdir(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHandler'
)
3333                        except:
3334                            pass
3335                        if
os.path.isdir(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHand
ler') == True:
3336
 shutil.rmtree(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHan
dler', ignore_errors=True)
3337                            print("Cleaning-up previous update package... ")
3338
 shutil.rmtree(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHan
dler', ignore_errors=True)
3339                            try:
3340
 os.rmdir(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHandler'
)
3341                            except:
3342                                pass
3343                        else:
3344                            pass
3345                        try:
3346                            os.system('git clone
https://github.com/deltaonealpha/DBFA_UpdateHandler')
3347                        except:
3348                            print("The directory 'DBFA_UpdateHandler' already
exists.")
3349                            print("Please delete it from DBFA's installation
location and re-run the updater.")
3350                        #os.system('git log')
3351                        print("Commit: ")
3352                        os.system(r'git rev-parse HEAD')
3353
3354                        print("The new package has been downloaded to your DBFA
installation > master > DBFA_UpdateHandler")
3355                        print("Please replace *only the required* files manually.
There's a 'Update Instructions.txt' file inside the 'DBFA_UpdateHandler' folder with
the required steps to update DBFA.")
3356
```

```
                    if updateconfo == "n":
                        time.sleep(1)
                        print("Use this, (option 11) whenever you want to update DBFA. We recommend doing so on urgent grounds. DBFA updates bring better security and new ground-breaking features with them!")

        #Exit System
        elif decfac == "12":
            if os.path.exists(r'userblock.txt'):
                userblock.close()
                os.remove(r'userblock.txt')
            if os.path.exists(r'userblock.zconf'):
                userblock.close()
                os.remove(r'userblock.zconf')
            toaster.show_toast("DFBA Framework Runtime Broker", "Obsufcating program...", duration = 1)
            logoprintxrt()
            floodscreen()
            #os.close('securepack.pyw')
            os._exit(0)

        #DBFA EMPLOYEE MANAGER
        elif decfac in ("emp", "EMP", "EMPLOYEE", "employee", "manager", "MANAGER", "empm", "EMPM"):
            print("Continuing to DBFA Employee Manager - -")
            time.sleep(2)
            with HiddenPrints():
                try:
                    sender = telegram_bot_sendtext(dt_string + "\n" + "DBFA Employee Manager accessed! \n\n - DBFA Security")
                    print(sender)
                except Exception:
                    pass

            import os, time, sqlite3, requests, json
            os.system('cls')
            print("-------DBFA Employee Manager-------")
            time.sleep(0.5)
            print("ʃ つ ◕_◕ ʅつ ")
            time.sleep(0.5)
            print("         ʃ つ ◕_◕ ʅつ ")
            time.sleep(0.5)
            print("                  ʃ つ ◕_◕ ʅつ ")
            time.sleep(1)
            os.system('cls')

            def empmenu():
                print('''-------DBFA Employee Manager-------
                Options:
                    1. Hire an employee
                    2. View employee records
                    3. Change employee details
                    4. Fire an employee ʃ •'◡'• ʅつ

                    5. Mark attendance
                    6. Attendance records - All
                    7. Attendance records - OiD-specific

                    8. Attendance records - All (THIS MONTH)
                    9. Attendance records - OiD-specific (THIS MONTH)
```

```
3412
3413                    10. Pay salary
3414
3415                    11. <<< Back to DBFA menu
3416
3417                '''')
3418
3419            while (1):
3420                empmenu()
3421                empfac = input("What would you like to do? ")
3422
3423                if empfac == "1":
3424                    print("DBFA will now be opening a seperate window due to GUI-
restrictions.")
3425                    time.sleep(2)
3426                    with HiddenPrints():
3427                        try:
3428                            sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Hire Employee - deltaDBFA")
3429                            print(sender)
3430                        except Exception:
3431                            pass
3432                    os.startfile(r'dbfaempman.py')
3433
3434
3435                if empfac == "2":
3436                    print("\n2. View employee records\n----------------------------
------------------")
3437                    empmas = sqlite3.connect(r'dbfaempmaster.db')
3438                    empmascur = empmas.cursor()
3439                    print("Employee records as maintained by DBFA: ")
3440                    with HiddenPrints():
3441                        try:
3442                            sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Records - deltaDBFA")
3443                            print(sender)
3444                        except Exception:
3445                            pass
3446                    empmascur.execute("SELECT * FROM emp")
3447                    emprows = empmascur.fetchall()
3448                    time.sleep(1)
3449                    for emprow in emprows:
3450                        print(emprow, "\n")
3451                    print("\n\n-----------------------------------------------")
3452                    time.sleep(3)
3453
3454                if empfac == "3":
3455                    print("3. Change employee details")
3456                    empmas = sqlite3.connect(r'dbfaempmaster.db')
3457                    empmascur = empmas.cursor()
3458                    empmascur.execute("SELECT * FROM emp")
3459                    emprows = empmascur.fetchall()
3460                    time.sleep(1)
3461                    for emprow in emprows:
3462                        print(emprow, "\n")
3463                    print("\n")
3464                    emppay = str(input("Enter the Oid (Employee ID) to change
details for: "))
3465                    empmascur.execute("SELECT Name FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
```

```
3466                    firename = str(empmascur.fetchall()[0][0])
3467                    confofac = input("CONFIRM: Change details for "+firename+"?
(y/n): ")
3468                    if confofac == "y":
3469                        print('''Options:
3470                                1. Change Name
3471                                2. Change Email
3472                                3. Change Mobile Contact
3473
3474                                4. Change Residential Address
3475
3476                                5. Change UPI Payments ID
3477
3478                                6. Change Department
3479                                7. Change Designation (POST)
3480                                8. Change Salary
3481                                ''')
3482                        subfac = input("What would you like to do? ")
3483                        if subfac == "1":
3484                            print("1. Change Name")
3485                            empmas = sqlite3.connect(r'dbfaempmaster.db')
3486                            empmascur = empmas.cursor()
3487                            newmodif = input("Enter the changed name: ")
3488                            time.sleep(1)
3489                            empmascur.execute("UPDATE emp SET Name = ? WHERE Oid =
?", (newmodif, emppay))
3490                            empmas.commit()
3491                            print("Name changed for OID", emppay, " from ",
firename, "to ", newmodif)
3492                            time.sleep(1)
3493                            with HiddenPrints():
3494                                try:
3495                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "Accessed: Employee Details Changed - deltaDBFA")
3496                                    print(sender)
3497                                except Exception:
3498                                    pass
3499                            print("--------")
3500                            time.sleep(1)
3501
3502                        if subfac == "2":
3503                            print("2. Change Email")
3504                            empmas = sqlite3.connect(r'dbfaempmaster.db')
3505                            empmascur = empmas.cursor()
3506                            empmascur.execute("SELECT Email FROM emp WHERE Oid LIKE
?", ("%"+emppay+"%", ))
3507                            firename = str(empmascur.fetchall()[0][0])
3508                            newmodif = input("Enter the changed e-mail: ")
3509                            time.sleep(1)
3510                            empmascur.execute("UPDATE emp SET Email = ? WHERE Oid =
?", (newmodif, emppay))
3511                            empmas.commit()
3512                            with HiddenPrints():
3513                                try:
3514                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "Accessed: Employee Details Changed - deltaDBFA")
3515                                    print(sender)
3516                                except Exception:
3517                                    pass
```

```
3518                             print("Email changed for OID", emppay, " from ",
    firename, "to ", newmodif)
3519                             time.sleep(1)
3520                             print("--------")
3521                             time.sleep(1)
3522 
3523                         if subfac == "3":
3524                             print("3. Change Mobile Contact")
3525                             empmas = sqlite3.connect(r'dbfaempmaster.db')
3526                             empmascur = empmas.cursor()
3527                             empmascur.execute("SELECT Mobile FROM emp WHERE Oid LIKE
    ?", ("%"+emppay+"%", ))
3528                             firename = str(empmascur.fetchall()[0][0])
3529                             newmodif = input("Enter the changed mobile contact: ")
3530                             time.sleep(1)
3531                             empmascur.execute("UPDATE emp SET Mobile = ? WHERE Oid =
    ?", (newmodif, emppay))
3532                             empmas.commit()
3533                             with HiddenPrints():
3534                                 try:
3535                                     sender = telegram_bot_sendtext(dt_string + "\n"
    + "Accessed: Employee Details Changed - deltaDBFA")
3536                                     print(sender)
3537                                 except Exception:
3538                                     pass
3539                             print("Mobile contact changed for OID", emppay, " from
    ", firename, "to ", newmodif)
3540                             time.sleep(1)
3541                             print("--------")
3542                             time.sleep(1)
3543 
3544                         if subfac == "4":
3545                             print("4. Change Residential Address")
3546                             empmas = sqlite3.connect(r'dbfaempmaster.db')
3547                             empmascur = empmas.cursor()
3548                             empmascur.execute("SELECT Address FROM emp WHERE Oid
    LIKE ?", ("%"+emppay+"%", ))
3549                             firename = str(empmascur.fetchall()[0][0])
3550                             newmodif = input("Enter the changed address (in one-
    line): ")
3551                             time.sleep(1)
3552                             empmascur.execute("UPDATE emp SET Address = ? WHERE Oid
    = ?", (newmodif, emppay))
3553                             empmas.commit()
3554                             with HiddenPrints():
3555                                 try:
3556                                     sender = telegram_bot_sendtext(dt_string + "\n"
    + "Accessed: Employee Details Changed - deltaDBFA")
3557                                     print(sender)
3558                                 except Exception:
3559                                     pass
3560                             print("Address changed for OID", emppay, " from ",
    firename, "to ", newmodif)
3561                             time.sleep(1)
3562                             print("--------")
3563                             time.sleep(1)
3564 
3565                         if subfac == "5":
3566                             print("5. Change UPI Payments ID")
3567                             empmas = sqlite3.connect(r'dbfaempmaster.db')
```

```
                            empmascur = empmas.cursor()
                            empmascur.execute("SELECT UPI FROM emp WHERE Oid LIKE
?", ("%"+emppay+"%", ))
                            firename = str(empmascur.fetchall()[0][0])
                            newmodif = input("Enter the changed UPI ID: ")
                            time.sleep(1)
                            empmascur.execute("UPDATE emp SET UPI = ? WHERE Oid =
?", (newmodif, emppay))
                            empmas.commit()
                            with HiddenPrints():
                                try:
                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "Accessed: Employee Details Changed - deltaDBFA")
                                    print(sender)
                                except Exception:
                                    pass
                            print("UPI ID changed for OID", emppay, " from ",
firename, "to ", newmodif)
                            time.sleep(1)
                            print("--------")
                            time.sleep(1)

                        if subfac == "6":
                            print("6. Change Department")
                            time.sleep(1)
                            empmas = sqlite3.connect(r'dbfaempmaster.db')
                            empmascur = empmas.cursor()
                            empmascur.execute("SELECT Dept FROM emp WHERE Oid LIKE
?", ("%"+emppay+"%", ))
                            firename = str(empmascur.fetchall()[0][0])
                            print("-----------------------------------\nCurrent
Dept: ", firename)
                            print('''Options:
                                        IT,
                                        Administration,
                                        Sales,
                                        Care-taking,
                                        Logistics ''')
                            print("-----------------------------------\n\n")
                            time.sleep(2)
                            newmodif = input("Enter the changed department: ")
                            time.sleep(1)
                            empmascur.execute("UPDATE emp SET Dept = ? WHERE Oid =
?", (newmodif, emppay))
                            empmas.commit()
                            with HiddenPrints():
                                try:
                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "Accessed: Employee Details Changed - deltaDBFA")
                                    print(sender)
                                except Exception:
                                    pass
                            print("Department changed for OID", emppay, " from ",
firename, "to ", newmodif)
                            time.sleep(1)
                            print("--------")
                            time.sleep(1)

                        if subfac == "7":
                            print("7. Change Designation (POST)")
```

```
3619                            empmas = sqlite3.connect(r'dbfaempmaster.db')
3620                            empmascur = empmas.cursor()
3621                            empmascur.execute("SELECT Post FROM emp WHERE Oid LIKE
?", ("%"+emppay+"%", ))
3622                            firename = str(empmascur.fetchall()[0][0])
3623                            newmodif = input("Enter the new designation: ")
3624                            time.sleep(1)
3625                            empmascur.execute("UPDATE emp SET Post = ? WHERE Oid =
?", (newmodif, emppay))
3626                            empmas.commit()
3627                            with HiddenPrints():
3628                                try:
3629                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "Accessed: Employee Details Changed - deltaDBFA")
3630                                    print(sender)
3631                                except Exception:
3632                                    pass
3633                            print("Designation (Post) changed for OID", emppay, "
from ", firename, "to ", newmodif)
3634                            time.sleep(1)
3635                            print("--------")
3636                            time.sleep(1)
3637
3638                        if subfac == "8":
3639                            print("8. Change Salary")
3640                            empmas = sqlite3.connect(r'dbfaempmaster.db')
3641                            empmascur = empmas.cursor()
3642                            empmascur.execute("SELECT Salary FROM emp WHERE Oid LIKE
?", ("%"+emppay+"%", ))
3643                            firename = str(empmascur.fetchall()[0][0])
3644                            newmodif = input("Enter the new salary (net; not
incremental): ")
3645                            time.sleep(1)
3646                            empmascur.execute("UPDATE emp SET Salary = ? WHERE Oid =
?", (newmodif, emppay))
3647                            empmas.commit()
3648                            with HiddenPrints():
3649                                try:
3650                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "Accessed: Employee Details Changed - deltaDBFA")
3651                                    print(sender)
3652                                except Exception:
3653                                    pass
3654                            print("Salary changed for OID", emppay, " from ",
firename, "to ", newmodif)
3655                            time.sleep(1)
3656                            print("--------")
3657                            time.sleep(1)
3658                        else:
3659                            print("Invalid option! \n\n")
3660                            time.sleep(1)
3661                    else:
3662                        print("Invalid option! \n\n")
3663                        time.sleep(1)
3664
3665
3666                if empfac == "4":
3667                    empmas = sqlite3.connect(r'dbfaempmaster.db')
3668                    empmascur = empmas.cursor()
3669                    print("4. Fire an employee ʃ •'◡'• ʅつ \n")
```

```
                    empmascur.execute("SELECT * FROM emp")
                    emprows = empmascur.fetchall()
                    time.sleep(1)
                    for emprow in emprows:
                        print(emprow, "\n")
                    print("\n")
                    emppay = str(input("Enter the Oid (Employee ID) for the employee to fire: "))
                    empmascur.execute("SELECT Name FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                    firename = str(empmascur.fetchall()[0][0])
                    confofac = input("CONFIRM: Fire "+firename+"? (y/n): ")
                    if confofac == "y":
                        reasonfire = input("Enter the reason for firing: ")
                        print("FIRING EMPLOYEE! ")
                        empmascur.execute("DELETE FROM emp WHERE Oid = ?", ("%"+emppay+"%", ))
                        empmas.commit()
                        telethon = ""
                        print("\n\n")
                        print("-------------------------------------------")
                        telethon += ("---------------------------\n")
                        print("DBFA Employee Firing Record        deltaDBFA")
                        telethon += ("DBFA Employee Firing Record\n")
                        print("- - - - - - - - - - - - - - - - - - - - - - -")
                        telethon += ("- - - - - - - - - - - - - - - -\n")
                        print("Name  :", '%s'%firename)
                        telethon += ("Name  : "+ '%s'%firename)
                        print("Reason:", '%s'%reasonfire)
                        telethon += ("\nReason: "+ '%s'%reasonfire)
                        print("- - - - - - - - - - - - - - - - - - - - - - -")
                        telethon += ("\n- - - - - - - - - - - - - - - -\n")
                        telethon += ("~ deltaDBFA\n")
                        telethon += ("---------------------------\n\n")
                        telegram_bot_sendtext(telethon)
                        print("~ Event logged in Infinity Logger & Telegram.")
                        print("-------------------------------------------\n\n")

                        time.sleep(2)
                    else:
                        print("Cancelled op..")

                if empfac == "5":
                    import sqlite3, time, os, requests
                    from datetime import datetime  #for reporting the billing time and date
                    empmas = sqlite3.connect(r'dbfaempmaster.db')
                    empmascur = empmas.cursor()
                    empmascur.execute("SELECT DISTINCT * FROM emp")
                    dump = empmascur.fetchall()
                    Oiddump = []
                    for row in dump:
                        Oiddump.append(row[0])
                    print("OiDs registered: ", Oiddump)
                    try:
                        Oid = int(input("DBFA MARK ATTENDANCE- Enter your OiD: "))
                        trypass = 1
                    except:
                        print("OiDs are integer-only. Please retry using valid credentials! \n")
```

```
3725                        trypass = 0
3726
3727                    if trypass == 1:
3728                        if Oid in Oiddump:
3729                            print("OiD found ")
3730                            time.sleep(0.5)
3731                            now = datetime.now()
3732                            dt_string = now.strftime("%Y/%m/%d")  #datetime object
containing current date and time
3733                            tm_string = now.strftime("%H:%M:%S")  #datetime object
containing current date and time
3734                            empmascur.execute("SELECT count(*) FROM attendance WHERE
Date = ? AND OiD = ?", (dt_string, Oid,))
3735                            data = empmascur.fetchone()[0]
3736                            if data==0:
3737                                #print('No record on %s'%dt_string+' for Employee
%s'%Oid)
3738                                #print("insert into attendance(Date, OiD, Time, YN,
IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
3739                                empmascur.execute("insert into attendance(Date, OiD,
Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
3740                                #empmascur.execute("UPDATE attendance SET Oid = 'Y'
WHERE DATE = ?", (dt_string))
3741                                empmas.commit()
3742                                print("\n----------------DBFA-----------------")
3743                                print("C1 ENTRY: Marked Attendance! OiD: ", Oid)
3744                                with HiddenPrints():
3745                                    try:
3746                                        sender = telegram_bot_sendtext(dt_string +
"\n" + "C1 ENTRY: Marked attendance - OiD"+'%s'%Oid)
3747                                        print(sender)
3748                                    except Exception:
3749                                        pass
3750                                print("-------------------------------------\n")
3751                            elif data==1:
3752                                #print('Component %s found in %s row(s)'%(dt_string,
data))
3753                                empmascur.execute("insert into attendance(Date, OiD,
Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'O'))
3754                                empmas.commit()
3755                                print("\n----------------DBFA-----------------")
3756                                print("C2 DAY END: Marked Attendance! OiD: ", Oid)
3757                                with HiddenPrints():
3758                                    try:
3759                                        sender = telegram_bot_sendtext(dt_string +
"\n" + "C2 DAY END: Marked attendance - OiD"+'%s'%Oid)
3760                                        print(sender)
3761                                    except Exception:
3762                                        pass
3763                                print("-------------------------------------\n")
3764                            elif data > 1:
3765                                print("You can only mark in/out attendance once a
day! \n")
3766                        else:
3767                            print("OiD not found! \n")
3768
3769                if empfac == "6":
3770                    print("6. Attendance records - All")
3771                    with HiddenPrints():
3772                        try:
```

```
                            sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Attendance Records - deltaDBFA")
                            print(sender)
                        except Exception:
                            pass
                    import sqlite3, time, os, requests, datetime
                    from datetime import datetime, date

                    empmas = sqlite3.connect(r'dbfaempmaster.db')
                    empmascur = empmas.cursor()

                    empmascur.execute("SELECT DISTINCT * FROM emp")
                    dump = empmascur.fetchall()
                    Oiddump = []
                    for row in dump:
                        Oiddump.append(row[0])

                    print("OiDs registered: ", Oiddump)


                    now = datetime.now()
                    dt_string = now.strftime("%Y/%m/%d")  #datetime object
containing current date and time

                    month = datetime.now().month - 1
                    if month < 1:
                        month = 12 + month  # At this point month is 0 or a negative
number so we add
                    if len(str(month)) == 1:
                        month = "0"+str(month)
                    dt1mb =
('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))


                    time.sleep(0.5)
                    print("\nLoading ALL attendance data recorded since the start of
using DBFA\n")
                    empmascur.execute("SELECT * FROM attendance ORDER BY Date ASC")
                    returned = empmascur.fetchall()
                    for row in returned:
                        print(row)


                if empfac == "7":
                    print("7. Attendance records - OiD-specific")
                    with HiddenPrints():
                        try:
                            sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Attendance Records - deltaDBFA")
                            print(sender)
                        except Exception:
                            pass
                    import sqlite3, time, os, requests
                    from datetime import datetime  #for reporting the billing time
and date

                    empmas = sqlite3.connect(r'dbfaempmaster.db')
                    empmascur = empmas.cursor()

                    empmascur.execute("SELECT DISTINCT * FROM emp")
```

```
3826                    dump = empmascur.fetchall()
3827                    Oiddump = []
3828                    for row in dump:
3829                        Oiddump.append(row[0])
3830
3831                    print("OiDs registered: ", Oiddump)
3832
3833                    try:
3834                        Oid = int(input("DBFA ATTENDANCE RECORDS- Enter the OiD: "))
3835                        trypass = 1
3836                    except:
3837                        print("OiDs are integer-only. Please retry using valid
    credentials! \n")
3838                        trypass = 0
3839
3840                    if trypass == 1:
3841                        if Oid in Oiddump:
3842                            print("OiD found ")
3843                            time.sleep(0.5)
3844                            empmascur.execute("SELECT * FROM attendance WHERE Oid =
    ? ORDER BY Date ASC, Time, IO", ('%s'%Oid))
3845                            returned = empmascur.fetchall()
3846                            for row in returned:
3847                                print(row)
3848
3849
3850                if empfac == "8":
3851                    print("8. Attendance records - All (THIS MONTH)")
3852                    with HiddenPrints():
3853                        try:
3854                            sender = telegram_bot_sendtext(dt_string + "\n" +
    "Accessed: Employee Attendance Records - deltaDBFA")
3855                            print(sender)
3856                        except Exception:
3857                            pass
3858                    print("Coming soon! ")
3859                    import sqlite3, time, os, requests, datetime
3860                    from datetime import datetime, date
3861
3862                    empmas = sqlite3.connect(r'dbfaempmaster.db')
3863                    empmascur = empmas.cursor()
3864
3865                    empmascur.execute("SELECT DISTINCT * FROM emp")
3866                    dump = empmascur.fetchall()
3867                    Oiddump = []
3868                    for row in dump:
3869                        Oiddump.append(row[0])
3870
3871                    print("OiDs registered: ", Oiddump)
3872
3873
3874                    now = datetime.now()
3875                    dt_string = now.strftime("%Y/%m/%d")  #datetime object
    containing current date and time
3876
3877                    month = datetime.now().month - 1
3878                    if month < 1:
3879                        month = 12 + month  # At this point month is 0 or a negative
    number so we add
3880                    if len(str(month)) == 1:
```

```
3881                        month = "0"+str(month)
3882                    dt1mb =
('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
3883
3884
3885                    time.sleep(0.5)
3886                    print("\nLoading data for days between "+'%s'%dt1mb+" and
"+'%s'%dt_string+"\n")
3887                    empmascur.execute("SELECT * FROM attendance WHERE Date BETWEEN ?
AND ? ORDER BY Date ASC", (dt1mb, dt_string))
3888                    returned = empmascur.fetchall()
3889                    for row in returned:
3890                        print(row)
3891
3892
3893                if empfac == "9":
3894                    print("9. Attendance records - OiD-specific (THIS MONTH)")
3895                    with HiddenPrints():
3896                        try:
3897                            sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Attendance Records - deltaDBFA")
3898                            print(sender)
3899                        except Exception:
3900                            pass
3901                    import sqlite3, time, os, requests, datetime
3902                    from datetime import datetime, date
3903
3904                    empmas = sqlite3.connect(r'dbfaempmaster.db')
3905                    empmascur = empmas.cursor()
3906
3907                    empmascur.execute("SELECT DISTINCT * FROM emp")
3908                    dump = empmascur.fetchall()
3909                    Oiddump = []
3910                    for row in dump:
3911                        Oiddump.append(row[0])
3912
3913                    print("OiDs registered: ", Oiddump)
3914
3915                    try:
3916                        Oid = int(input("DBFA ATTENDANCE RECORDS- Enter the OiD: "))
3917                        trypass = 1
3918                    except:
3919                        print("OiDs are integer-only. Please retry using valid
credentials! \n")
3920                        trypass = 0
3921
3922                    now = datetime.now()
3923                    dt_string = now.strftime("%Y/%m/%d")  #datetime object
containing current date and time
3924
3925                    month = datetime.now().month - 1
3926                    if month < 1:
3927                        month = 12 + month  # At this point month is 0 or a negative
number so we add
3928                    if len(str(month)) == 1:
3929                        month = "0"+str(month)
3930                    dt1mb =
('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
3931
3932                    if trypass == 1:
```

```
3933                         if Oid in Oiddump:
3934                             print("OiD found ")
3935                             time.sleep(0.5)
3936                             print("\nLoading data for days between "+'%s'%dt1mb+"
    and "+'%s'%dt_string+"\n")
3937                             empmascur.execute("SELECT * FROM attendance WHERE OiD =
    ? AND Date BETWEEN ? AND ? ORDER BY Date ASC", ('%s'%Oid, dt1mb, dt_string))
3938                             returned = empmascur.fetchall()
3939                             for row in returned:
3940                                 print(row)
3941
3942
3943                 if empfac == "10":
3944                     import pyqrcode, png, os
3945                     from pyqrcode import QRCode
3946
3947                     empmas = sqlite3.connect(r'dbfaempmaster.db')
3948                     empmascur = empmas.cursor()
3949
3950                     empmascur.execute("SELECT * FROM emp")
3951                     emprows = empmascur.fetchall()
3952                     for emprow in emprows:
3953                         print(emprow)
3954
3955                     time.sleep(1)
3956
3957                     emppay = str(input("Enter the Oid (Employee ID) to pay salary
    for: "))
3958                     empmascur.execute("SELECT * FROM emp WHERE Oid LIKE ?",
    ("%"+emppay+"%", ))
3959                     print((empmascur.fetchall()[0]))
3960                     time.sleep(1)
3961                     emppayconfox = input("\n\nPay salary? (y/n): ")
3962                     if emppayconfox == "y":
3963                         #emarpay = str("%"+'%s'%emppay+"%")
3964                         empmascur.execute("SELECT Name, UPI FROM emp WHERE Oid LIKE
    ?", (emppay,) )
3965                         tempemppay = empmascur.fetchall()
3966                         print("Paying ", list(tempemppay[0])[0], "at ",
    list(tempemppay[0])[1])
3967                         name = list(tempemppay[0])[0]
3968                         upid = list(tempemppay[0])[1]
3969                     else:
3970                         empmenu()
3971                         break
3972                         print("Aaaa")
3973
3974                     #upid = '9810141714@upi'
3975                     #name = 'KPBalaji'
3976
3977                     s = "upi://pay?pa="+'%s'%upid+"&pn="+'%s'%name+"&cu=INR"
3978
3979                     # Generate QR code
3980                     url = pyqrcode.create(s)
3981
3982                     url.png('payqr.png', scale = 6)
3983                     from PIL import Image, ImageDraw, ImageFont
3984                     image = Image.open('payqr.png')
3985                     with HiddenPrints():
3986                         try:
```

```
                            sender = telegram_bot_sendtext(dt_string + "\n" +
"Started Process: Issue Salary - deltaDBFA")
                            print(sender)
                        except Exception:
                            pass
                    draw = ImageDraw.Draw(image)
                    font =
ImageFont.truetype(r'C:\Users\balaj\AppData\Local\Microsoft\Windows\Fonts\MiLanProVF
.ttf', size=200)
                    (x, y) = (5, 250)
                    xname = 'Scan to pay with UPI              deltaDBFA'
                    draw.text((x, y), xname) #, fill=color)
                    (x, y) = (5, 5)

                    name = "Paying "+'%s'%name+" "*(28-len(str(name)))+"deltaPay"
                    draw.text((x, y), name) #, fill=color)
                    image.save('payqr.png', optimize=True, quality=120)

                    print("-------------------------------------")
                    time.sleep(1)
                    print("DBFA will now open a QR code for UPI payment to the
registered UPI address of the employee.")
                    time.sleep(1)
                    print("Scan the code in a UPI app to pay")
                    time.sleep(1)


 os.system(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\payqr.png')

                    time.sleep(5)
                    paycheck = input("Mark salary as 'PAID'? (y/n): ")
                    if paycheck == "y":
                        print("Salary paid!")
                    else:
                        print("Not paid. ")

                if empfac == "11":
                    print("Returning to DBFA Main.. ")
                    time.sleep(1)
                    break



        #CIT
        elif decfac == "113":
            print("INTERNAL TESTING MODE")
            ffxfac = str(input("Enter CIT Testing Mode? (y/n):: "))
            if ffxfac == "y":
                ffrxfac = str(input("Entering CIT may lead to data loss. Confirm
entering CIT? (y/n):: "))
                if ffrxfac == "y":
                    print("DNSS CIT MODE")
                    print(" ")
                    print(" ")
                    print("NOTE: DBFA will restart to execute CIT options. ")
                    print("CIT Options::")
                    print("Enter '1' to CLEAR ALL CUSTOMER RECORDS")
                    print("Enter '2' to CLEAR ALL VOUCHERS/ COUPONS")
                    print("Enter '3' to exit CIT")
                    citfacin = int(input("Waiting for input:: "))
```

```
                    if citfacin == 1:
                        # window.close()
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: CIT del cust recs - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                        os.startfile(r'securepack.py')
                        time.sleep(1)
                        os._exit(0)
                    if citfacin == 2:
                        # window.close()
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: CIT del voucher recs - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                        os.startfile(r'securepackxvc.py')
                        time.sleep(1)
                        os._exit(0)
                    else:
                        continue

                else:
                    continue

            elif ffxfac == "3":
                print("Exiting CIT")
                time.sleep(1)
                continue
            else:
                print("Invalid input. . . . ")
                time.sleep(1)


        # Direct Calls Section - 2
        elif decfac in ("2a", "2A", "2 a", "2 A"):
            del2a()
            logger.write("---------------------------------------- \n")
            logger.write("  \n")
            logger.write("Date and time: ") #including the date and time of billing
(as taken from the system)
            logger.write(dt_string)
            logger.write(" \n")
            logger.write("New customer registered! ")
            #x = " custname: " + custname + " custemail: " + email + "\n"
            logger.write("---------------------------------------- \n")

        elif decfac in ("2b", "2B", "2 b", "2 B"):
            del2b()
            logger.write("---------------------------------------- \n")
            logger.write("  \n")
            logger.write("Date and time: ") #including the date and time of billing
(as taken from the system)
            logger.write(dt_string)
            logger.write(" \n")
```

```
4097            logger.write("Customer registry accessed! \n")
4098            logger.write("---------------------------------------- \n")
4099
4100        elif decfac in ("2c", "2C", "2 c", "2 C"):
4101            del2c()
4102            logger.write("---------------------------------------- \n")
4103            logger.write("\nDBFA Customer Purchase Records accessed! \n")
4104
4105        elif decfac in ("2d", "2D", "2 d", "2 D"):
4106            del2d()
4107            logger.write("---------------------------------------- \n")
4108            logger.write("\nCustomer search used. \n")
4109
4110        elif decfac in ("2e", "2E", "2 e", "2 E"):
4111            del2e()
4112            logger.write("\n\n---------------------------------------- \n")
4113            logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
4114            logger.write("Customer data exported to CSV! ")
4115            with HiddenPrints():
4116                try:
4117                    sender = telegram_bot_sendtext(dt_string + "\n" + "Sales data
    exported to CSV- REDFLAG Urgent Security Notice!")
4118                    print(sender)
4119                except Exception:
4120                    pass
4121            logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
4122            logger.write("---------------------------------------- \n\n\n")
4123
4124
4125
4126        # Direct Calls Section - 3
4127        elif decfac in ("3a", "3A", "3 a", "3 A"):
4128            del3a()
4129
4130        elif decfac in ("3b", "3B", "3 b", "3 B"):
4131            del3b()
4132
4133        elif decfac in ("3c", "3C", "3 c", "3 C"):
4134            del3c()
4135
4136        elif decfac in ("3d", "3D", "3 d", "3 D"):
4137            del3d()
4138
4139        elif decfac in ("3e", "3E", "3 e", "3 E"):
4140            del3e()
4141            logger.write("\n---------------------------------------- \n")
4142            logger.write("Sales log accessed! ")
4143
4144
4145        elif decfac in ("3f", "3F", "3 f", "3 F"):
4146            del3f()
4147            logger.write("\n\n---------------------------------------- \n")
4148            logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
4149            with HiddenPrints():
4150                try:
4151                    sender = telegram_bot_sendtext(dt_string + "\n" + "Customer data
    exported to CSV- REDFLAG Urgent Security Notice!")
4152                    print(sender)
4153                except Exception:
4154                    pass
```

```
            logger.write("Sales data exported to CSV! ")
            logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
            logger.write("-------------------------------------- \n\n\n")

        elif decfac in ("3g", "3G", "3 f", "3 F"):
            print("Deep Archival Engine")
            del3g()

        elif decfac in (None, "", " "):
            print("Please select a valid main-menu option. erc101\n\n")
            time.sleep(0.8)
            continue

        else:
            print("Please select a valid main-menu option. erc101\n\n")
            time.sleep(0.8)
            continue

except:
    import os, requests, time
    os.system('cls')
    error_message = traceback.format_exc()
    print("DBFA has crashed due to unexpected reasons. ")
    print("DBFA will now automatically check its installation for integrity issues::
")
    time.sleep(1)
    print("Error recieved from interpreter follows:")
    print("    ", error_message)
    time.sleep(3)
    time.sleep(2)
    from md5checker import make_hash

    print("-------------------------------------------------")
    print(" delta Installation Integrity Validation Service")
    print("-------------------------------------------------")
    time.sleep(1)

    listing = ('bleading_edge', 'modif2fa', 'dbfaempman', 'plotter', 'delauth',
'dbfabackupper', 'authtimeout', 'securepack', 'securepackxvc', 'wrelogin',
'run_DBFA')

    print("Fetching MD5 from server.")
    servermd5 =
((requests.get("https://raw.githubusercontent.com/deltaonealpha/DBFA/master/md5")).c
ontent).decode('utf-8')
    #.replace(" ", "").replace("\n", "")
    serverdump = (servermd5.split())
    print("Arranging hashes ###")
    server_md5 = []
    server_md5.append(serverdump[0].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[1].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[2].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[3].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[4].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[5].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[6].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[7].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[8].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[9].replace(" ", "").replace("\n", ""))
    server_md5.append(serverdump[10].replace(" ", "").replace("\n", ""))
```

```
4210
4211    print("Fetching logged MD5.")
4212    try:
4213        localdump =
str(open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\md5').read())
4214        sorteddump = (localdump.split())
4215        print("Arranging hashes ###")
4216        local_md5 = []
4217        local_md5.append(sorteddump[0].replace(" ", "").replace("\n", ""))
4218        local_md5.append(sorteddump[1].replace(" ", "").replace("\n", ""))
4219        local_md5.append(sorteddump[2].replace(" ", "").replace("\n", ""))
4220        local_md5.append(sorteddump[3].replace(" ", "").replace("\n", ""))
4221        local_md5.append(sorteddump[4].replace(" ", "").replace("\n", ""))
4222        local_md5.append(sorteddump[5].replace(" ", "").replace("\n", ""))
4223        local_md5.append(sorteddump[6].replace(" ", "").replace("\n", ""))
4224        local_md5.append(sorteddump[7].replace(" ", "").replace("\n", ""))
4225        local_md5.append(sorteddump[8].replace(" ", "").replace("\n", ""))
4226        local_md5.append(sorteddump[9].replace(" ", "").replace("\n", ""))
4227        local_md5.append(sorteddump[10].replace(" ", "").replace("\n", ""))
4228    except:
4229        print("DBFA's code has been tampered with! Please rectify this to avoid such
program crashes!")
4230        print("    rrtt - Master Copy Error: dta=intl.err_instldir?
imp=md5lognotfound")
4231
4232
4233    print("Hashing MD5 from deployed code.")
4234    live_md5 = []
4235    live_md5.append(make_hash('bleading_edge.py', algo='md5'))
4236    live_md5.append(make_hash('modif2fa.py', algo='md5'))
4237    live_md5.append(make_hash('dbfaempman.py', algo='md5'))
4238    live_md5.append(make_hash('plotter.pyw', algo='md5'))
4239    live_md5.append(make_hash('delauth.py', algo='md5'))
4240    live_md5.append(make_hash('dbfabackupper.py', algo='md5'))
4241    live_md5.append(make_hash('authtimeout.pyw', algo='md5'))
4242    live_md5.append(make_hash('securepack.py', algo='md5'))
4243    live_md5.append(make_hash('securepackxvc.py', algo='md5'))
4244    live_md5.append(make_hash('wrelogin.pyw', algo='md5'))
4245    live_md5.append(make_hash('run_DBFA.pyw', algo='md5'))
4246    print("Arranging hashes ###")
4247
4248
4249    alphas = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
4250
4251    for i in range(len(listing)):
4252        if live_md5[i] != local_md5[i]:
4253            print("! - Misaligned MD5 at")
4254            print(live_md5[i], local_md5[i], "////\n")
4255    print("Alignment check done p1x")
4256
4257    for i in server_md5:
4258        arter = ""
4259        for j in i.replace(" ", "").replace("\n", ""):
4260            if str(j).isalpha() is True:
4261                arter += '%d'%(alphas.index(j.lower()))
4262            else:
4263                arter += j
4264        server_md5[server_md5.index(i)] = int(arter)
4265
```

```
    for i in local_md5:
        arter = ""
        for j in i.replace(" ", "").replace("\n", ""):
            if str(j).isalpha() is True:
                arter += '%d'%(alphas.index(j.lower()))
            else:
                arter += j
        local_md5[local_md5.index(i)] = int(arter)

    for i in live_md5:
        arter = ""
        for j in i.replace(" ", "").replace("\n", ""):
            if str(j).isalpha() is True:
                arter += '%d'%(alphas.index(j.lower()))
            else:
                arter += j
        live_md5[live_md5.index(i)] = int(arter)

    for i in range(len(listing)):
        if live_md5[i] != local_md5[i]:
            print("! - Misaligned MD5 at")
            print(live_md5[i], local_md5[i], "////\n")
    print("Alignment check done p2x")

    print("\n")
    if live_md5 != local_md5:
        marker = 1
        issue = i
    elif live_md5 == local_md5:
        marker = 2

    time.sleep(3)

    if marker == 1:
        if sum(server_md5) == sum(live_md5):
            os.startfile(r'ep1.pyw')
        elif sum(server_md5) > sum(live_md5):
            os.startfile(r'ep2.pyw')
        elif sum(server_md5) < sum(live_md5):
            os.startfile(r'ep3.pyw')
    elif marker == 2:
            os.startfile(r'ep4.pyw')
    else:
        print("Integrity checker service failed to execute properly \\ | / |  -_- -_- -_-  \\ | / | ")
        time.sleep(10)


# End of program
# Available on github: www.github.com/deltaonealpha/DBFA
# https://deltaonealpha.github.io/DBFA/
```

<Deep Archival Services Engine>

// The Code 2/10

```python
#Data Format: <order ids>///<customer name>///<customer id>///<date and time string>///tax///discount///loyalty///net

# DD/MM/YYYY_HH/MM/SS
# DD000MM000YYYY000HH000MM000SS

import time
def alphadecoder(datalist):
    alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
    out = ""
    for i in (datalist):
        if i.isnumeric():
            print(i, int(i)-1)
            out += alpha[(int(i)-1)]
    del alpha
    return(out)

def dttdecoder(datalist):
    templist1 = datalist.split('y')
    out = templist1[0] + '/'
    print(templist1[0])
    out += templist1[1] + '/'
    print(templist1[1])
    out += templist1[2] + ' '
    print(templist1[2])
    out += templist1[3] + ':'
    print(templist1[3])
    out += templist1[4] + ':'
    out += templist1[5]
    print(templist1[5])
    print(templist1[3])
    return(out)

def encoder_deeparchival(data, invid):
    deeparchival_input = data
    orders, cust_name, cust_id, dt_string = deeparchival_input[0], deeparchival_input[1], deeparchival_input[2], deeparchival_input[3]
    tax_bar, disct_bar, lylredemp, net_process = deeparchival_input[4], deeparchival_input[5], deeparchival_input[6], deeparchival_input[7]
    templist = (dt_string.split(" "))
    templist2 = templist[0].split('/')
    xdatetime = templist2[0] + 'y' + templist2[1] + 'y' + templist2[2] + 'y'
    templist3 = templist[1].split(':')
    xdatetime += templist3[0] + 'y' + templist3[1] + 'y' + templist3[2] + 'y'
    dt_string = xdatetime

    ordernet = orders.split('.')

    deeparchival_key = ""
    for i in ordernet:
        deeparchival_key += str(i)+"00"
    alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
    out = ""
    for i in str(cust_name):
        if i.isalpha():
            out += str((alpha.index(i.lower())) + 1) + 'z'
        else:
            out += str(i) + 'z'
```

```
    deeparchival_key += ("000" + out + "00000" + str(cust_id) + "00000" + str(dt_string))
    deeparchival_key += ("00000" + str(tax_bar) + "00000" + str(disct_bar) + "00000" + str(lylredemp) + "00000" + str(net_process))
    import os, sqlite3
    from datetime import datetime
    now = datetime.now()
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object containing current date and time
    time_string = now.strftime("%H:%M:%S")  #datetime object containing current date and time
    dde = sqlite3.connect(r'DeepArchivalVault.db')
    ddex = dde.cursor()
    ddex.execute("INSERT INTO archive(Invid, Date, Time, Key) VALUES (?, ?, ?, ?)", (invid, dt_string, time_string, deeparchival_key))
    dde.commit()
    print("Data stored to VAULT.")
    return deeparchival_key


def decoder_deeparchival(key):
    print(key)
    if((key is None) or (len(key) == 0)):
        print("deeparchival-error:key-none?")
    else:
        print(key)
        init_data = key.split("00000")
    if init_data is not None:
        print('Decoding')
    else:
        exit
    print('rrtt', init_data)
    orders, cust_name, cust_id, dt_string = init_data[0], init_data[1], init_data[2], init_data[3]
    tax_bar, disct_bar, lylredemp, net_process = init_data[4], init_data[5], init_data[6], init_data[7]

    ordernet = str(orders).split('00')
    print(cust_name)
    if disct_bar in (None, "", " "):
        disct_bar = '0'
    dt_string = dttdecoder(dt_string)
    cust_name = alphadecoder(cust_name.split('z'))
    out = []
    xttemp = ""
    for i in ordernet:
        xttemp += (str(i) + '.')
    xttemp = xttemp[:-1]
    print('rrtt', lylredemp)
    if str(lylredemp) in ('0', 0):
        lylredemp = (str(lylredemp[1:]))
    import os
    os.system('cls')
    return (xttemp, cust_name, cust_id, dt_string, tax_bar, disct_bar, lylredemp, net_process)

def deepfetch_deeparchival():
    import sqlite3, os
    from sqlite3 import Error
    import os, time
```

```
    from datetime import datetime  #for reporting the billing time and date
    from reportlab.pdfbase import pdfmetrics
    from reportlab.pdfbase.ttfonts import TTFont
    pdfmetrics.registerFont(TTFont('MiLanProVF',
r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\MiLanProVF.ttf'))
    from reportlab.pdfgen import canvas
    from reportlab.lib.pagesizes import A4
    from reportlab.platypus import SimpleDocTemplate, Paragraph
    from reportlab.lib.styles import getSampleStyleSheet
    from reportlab.lib.units import cm
    from reportlab.lib.enums import TA_JUSTIFY
    from reportlab.lib.pagesizes import letter
    from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
    from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
    from reportlab.lib.units import inch
    from tabulate import tabulate
    from reportlab.lib import colors
    from reportlab.lib.pagesizes import letter
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle

    dde = sqlite3.connect(r'DeepArchivalVault.db')
    ddex = dde.cursor()
    print("Starting deep archival fetch process ----\n")
    ddex.execute("SELECT * FROM archive GROUP BY Date")
    datastream = ddex.fetchall()

    from tabulate import tabulate

    print(r'''
''')
    print(tabulate(datastream, headers=['Inv. ID', 'Date', 'Time', 'Key'],
tablefmt='orgtbl'))
    inputmaster = input("\nEnter the invoice ID to fetch from the VAULT: ")
    print("Fetching details for invoice ID: ", inputmaster, "from the VAULT.")
    inputmaster = '%'+inputmaster+'%'
    ddex.execute("SELECT Key FROM archive WHERE Invid LIKE ?", (inputmaster, ))
    row = ddex.fetchall()
    if row in (None, [], [[]], [()], "", " "):
        print("Invalid invoice ID.")
    else:
        ddex.execute("SELECT Invid FROM archive WHERE Key = ?", (row[0][0], ))
        xkeyer = ddex.fetchall()[0][0]
        #delta DEEP Archival System Decoder
        from DBFADeepArchivalEngine import alphadecoder, dttdecoder,
decoder_deeparchival
        temp, cust_name, cust_id, dt_string, tax_bar, disct_bar, lylredemp,
net_process = decoder_deeparchival(row[0][0])
        print(temp, cust_name, cust_id, dt_string, tax_bar, disct_bar, lylredemp,
net_process)
        namiex = ["TV 4K OLED 50", "TV FHD OLED 50", "8K QLED 80", "Redmi K20 PRO",
"Redmi K20", "Redmi Note 9 PRO", "POCOPHONE F1", "Mi MIX ALPHA", "Wireless
Headphones", "Noise-Cancelling Wireless Headphones", "Essentials Headphones", "Gaming
Headphones", "Truly-Wireless Eadphones", "Neckband-Style Wireless Earphones",
"Essentials Earphones", "Gaming Earphones", "30W Bluetooth Speakers", "20W Bluetooth
Speakers", "Essentials Bluetooth Speaker", "BOSE QC35", "Essentials Home Theatre",
"Wired Speaker - 5.1", "Essentials Wired Speaker - STEREO", "Tactical Series Power
Bank 30000mah", "Essentials Power Bank 10000mah", "Essentials Mouse", "Logitech G604
LightSpeed Wireless", "Tactical Essentials Keyboard", "DROP GS21k RGB Gaming
```

```
Keyboard", "Polowski Tactical Flashlight", "OneFiber Wi-Fi Router AX7", "Mijia Mesh Wi-Fi Router", "lapcare 45W Laptop Adapter", "lapcare 60W Laptop Adapter","Spigen Phone Case(s)", "Essentials Phone Charger 15W", "HyperPower Type-C Gallium-Nitride Charger 120W", "ASUS Zephyrus G4 Gaming Laptop", "DELL XPS 5 Content Creator's Laptop", "Hewlett-Packard Essential's Student's Laptop (Chromebook)"]
        datax = [40000, 55000, 67000, 25000, 21000, 14000, 3000, 220000, 4500, 17000, 1200, 3700, 4500, 2200, 700, 2750, 6499, 1499, 799, 27000, 6750, 2100, 1199, 3210, 989, 750, 1700, 600, 2175, 890, 2100, 7158, 597, 347, 500, 300, 1097, 80000, 87900, 23790]
        writer = ("~DBFA DEEP ARCHIVAL VAULT~\n\nDBFA Billing Framework\nOne-stop solution for all your billing needs!\n\nBilling time:" + str(dt_string) + "\nCustomer ID: " + str(cust_id) + str(cust_name) + "\n----------------------------Invoice ID:" + str(xkeyer) + "\nPurchased: ")
        orders = temp.split('.')
        #print(temp, orders)
        for i in orders:
            writer += str(str(namiex[int(i)-1]) + '\n')
            writer += str(str(datax[int(i)-1]) + '\n\n')
        writer += ("----------------------------\nTax amount: " + str(tax_bar)+'%' + "\nDiscount: " + str(disct_bar) + "\nUsed DBFA loyalty points worth:" + str(lylredemp)  + "\nNET TOTAL:" + str(net_process))
        from datetime import datetime
        import datetime
        daterey = (dt_string.replace("/","")).replace(":", "")
        namer = 'DBFAinvid'+'%s'%xkeyer+"-"+daterey+'.pdf'
        can = SimpleDocTemplate(namer, pagesize=A4,
                                rightMargin=2*cm,leftMargin=2*cm,
                                topMargin=2*cm,bottomMargin=2*cm)
        #can.setFont("MiLanProVF", 24)
        can.build([Paragraph(writer.replace("\n", "<br />"), getSampleStyleSheet()['Normal']),])

        import shutil
        source = namer
        temp = str(r"C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\Generated_invoices\\ ")[:-1] + str(namer)
        dest = shutil.move(source, temp)
        os.startfile(temp)
        del temp
        print("DEEP ARCHIVAL VAULT: Invoice generated and moved to the 'Generated Invoices' directory ~ ")


```

<Login Handler Service Scripts – (2/2)>

// The Code 3/10

```
from datetime import datetime
import sqlite3, time, os

if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')

def OAuthset():
    now = datetime.now()
    try: #To avoid error when time is 00:00:00
        netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 + int(now.strftime("%S"))
    except:
        time.sleep(1)
        netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 + int(now.strftime("%S"))
    oauth = sqlite3.connect(r'dbfasettings.db')
    oauthx = oauth.cursor()

    oauthx.execute("SELECT count(*) FROM LoginHandler")
    rows = oauthx.fetchall()
    if (rows[0][0]) > 1:
        print("DBFA Authenticator has been tampered with! DBFA WILL EXIT NOW!")
        time.sleep(2)
        os._exit(0)
    oauthx.execute("SELECT OAuthID FROM LoginHandler")
    try: #To avoid error when there's no data in the table
        maxid = int(oauthx.fetchall()[0][0]) + 1
    except:
        maxid = 1
        oauthx.execute("insert into LoginHandler(OAuthID, Value, TimeMark) values(?, 1, ?)", (maxid, netr))
    oauthx.execute("UPDATE LoginHandler SET OAuthID = ?, Value = 1, TimeMark = ?", (maxid, netr))
    print("Login session dtalgnrt", netr, "-", maxid)
    oauth.commit()
    oauth.close()

def Login():
    creds = r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\tempfile.temp'
    with open(creds, 'r') as f:
        data = f.readlines() # This takes the entire document we put the info into and puts it into the data variable
        uname = data[0].rstrip() # Data[0], 0 is the first line, 1 is the second and so on.
        pword = data[1].rstrip() # Using .rstrip() will remove the \n (new line) word from before when we input it
        import PySimpleGUI as sgx
        sgx.theme('DarkTeal9')  # Add a touch of color
        # All the stuff inside your window.
        def CAPSLOCK_STATE():
            import ctypes
            hllDll = ctypes.WinDLL ("User32.dll")
            VK_CAPITAL = 0x14
            return hllDll.GetKeyState(VK_CAPITAL)

        CAPSLOCK = CAPSLOCK_STATE()

        if ((CAPSLOCK) & 0xffff) != 0:
```

```
            #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
            layout = [  [sgx.Text('Yo we know security...')],
                        [sgx.Text('Username: '), sgx.InputText()],
                        [sgx.Text('Password: '), sgx.InputText(password_char='*')],
                        [sgx.Button('Login'), sgx.Button('Cancel')],
                        [sgx.Text('WARNING:  CAPS LOCK IS ENABLED!')]]

        else:
            layout = [  [sgx.Text('Yo we know security...')],
                        [sgx.Text('Username: '), sgx.InputText()],
                        [sgx.Text('Password: '), sgx.InputText(password_char='*')],
                        [sgx.Button('Login'), sgx.Button('Cancel')]]
        # Create the Window
        window = sgx.Window('DNSS Authenication Service', layout)
        # Event Loop to process "events" and get the "values" of the inputs
    while True:
        event, values = window.read()
        if event in (None, 'Cancel'): # if user closes window or clicks cancel
            window.close
            break
        window.close()
        window.close()
        if values[0] == 'ed' and values[1] == 'edd':
            #os.close(r'DDD.py')
            window.close()
            userblock = open(r"userblock.txt","a+") #Opening / creating (if it
doesn't exist already) the .txt record file
            userblock.write('ed')
            #time.sleep(2)
            OAuthset()
            userblock.close()
            print("logging success")
            os.startfile('bleading_edge.py')
            window.close()
            window.close()
            exit
            exit
            exit
            break
        else:

 os.startfile(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\wrelogin.pyw')
        #window.close
        #erraise()
import PySimpleGUI as sg
if
os.path.exists(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.txt')
:
    os.remove(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.txt')
if
os.path.exists(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.zconf
'):

 os.remove(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.zconf')
sg.theme('DarkTeal9') # Add a touch of color
# All the stuff inside your window.
layout = [  [sg.Text("DBFA Security")],
            [sg.Text("This program requires you to login.")],
            [sg.Button('Login'), sg.Button('Exit')] ]
# Create the Window
```

```
window = sg.Window('DBFA', layout)
# Event Loop to process "events" and get the "values" of the inputs
while True:
    event, values = window.read()
    if event in (None, 'Exit'): # if user closes window or clicks cancel
        break
    window.close()
    Login()

```

```python
from datetime import datetime
import sqlite3, time, os

def OAuthset():
    now = datetime.now()
    try: #To avoid error when time is 00:00:00
        netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 + int(now.strftime("%S"))
    except:
        time.sleep(1)
        netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 + int(now.strftime("%S"))
    oauth = sqlite3.connect(r'dbfasettings.db')
    oauthx = oauth.cursor()

    oauthx.execute("SELECT count(*) FROM LoginHandler")
    rows = oauthx.fetchall()
    if (rows[0][0]) > 1:
        print("DBFA Authenticator has been tampered with! DBFA WILL EXIT NOW!")
        time.sleep(2)
        os._exit(0)
    oauthx.execute("SELECT OAuthID FROM LoginHandler")
    try: #To avoid error when there's no data in the table
        maxid = int(oauthx.fetchall()[0][0]) + 1
    except:
        maxid = 1
        oauthx.execute("insert into LoginHandler(OAuthID, Value, TimeMark) values(?, 1, ?)", (maxid, netr))
    oauthx.execute("UPDATE LoginHandler SET OAuthID = ?, Value = 1, TimeMark = ?", (maxid, netr))
    print("Login session dtalgnrt", netr, "-", maxid)
    oauth.commit()
    oauth.close()

if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')
def Login():
    import PySimpleGUI as sgx
    sgx.theme('DarkRed')
    def CAPSLOCK_STATE():
        import ctypes
        hllDll = ctypes.WinDLL ("User32.dll")
        VK_CAPITAL = 0x14
        return hllDll.GetKeyState(VK_CAPITAL)

    CAPSLOCK = CAPSLOCK_STATE()

    if ((CAPSLOCK) & 0xffff) != 0:
        #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
        layout = [  [sgx.Text('INVALID LOGIN. Please retry:')],
                    [sgx.Text('Username: '), sgx.InputText()],
                    [sgx.Text('Password: '), sgx.InputText(password_char='*')],
                    [sgx.Button('Authenicate'), sgx.Button('Cancel')],
                    [sgx.Text('WARNING:  CAPS LOCK IS ENABLED!')]]

    else:
        layout = [  [sgx.Text('INVALID LOGIN. Please retry:')],
                    [sgx.Text('Username: '), sgx.InputText()],
```

```
                    [sgx.Text('Password: '), sgx.InputText(password_char='*')],
                    [sgx.Button('Authenicate'), sgx.Button('Cancel')] ]

    window = sgx.Window('deltaAuthenication Service', layout)
    while True:
        event, values = window.read()
        if event in (None, 'Cancel'): # if user closes window or clicks cancel
            window.close
            break
        window.close()
        window.close()
        if values[0] == 'ed' and values[1] == 'edd':
            #os.close(r'DDD.py')
            window.close()
            userblock = open(r"userblock.txt","a+") #Opening / creating (if it
doesn't exist already) the .txt record file
            userblock.write('ed')
            #time.sleep(2)
            OAuthset()
            userblock.close()
            print("logging success")
            os.startfile('bleading_edge.py')
            window.close()
            window.close()
            exit
            exit
            exit
            break
        else:
            os.startfile("wrelogin.pyw")
            exit
        #window.close
        #erraise()
import PySimpleGUI as sg
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')

Login()

```

<Database Deletion Services>

// The Code 4/10

```python
print("Removing ALL registered customer records now...")
import os, time
print("        ___ ______ __   _____________    _____________     ______")
time.sleep(0.1)
print("       /  /_______/  /  /  /_______/  /  /  /________/    /  /_/ /")
time.sleep(0.1)
print("      /  /       /  /  /  /       /  /  /  /            /  /  / /")
time.sleep(0.1)
print("     /  /       /  /  /  /_______/  /  /  /  CLI        /  /   / /")
time.sleep(0.1)
print("    /  /       /  /  / // // // // /  /  /_________    /  /____/ /")
time.sleep(0.1)
print("   /  /       /  /  /  /-------/  /  /  /_________/   /  /_____/ /")
time.sleep(0.1)
print("  /  /       /  /  /  /       /  /  /  /            /  /      / /")
time.sleep(0.1)
print(" /  /_______/  /  /  /______ /  /  /  /            /  /       / /")
time.sleep(0.1)
print("/__/_______/__/  /__/_______/__/  /__/            /__/        /__/")
print(" ")
print(" ")
time.sleep(2)
print("Flushing record directory. ")
time.sleep(0.5)
print("Flushing record directory. . ")
time.sleep(0.5)
print("Flushing record directory. . . ")
time.sleep(0.5)

x = "cponmgmtsys.db"
if os.path.exists(r'DBFA.db'):
    file = open(x, 'rb')
    data = file.read()
    os.remove(os.path.normpath(r"DBFA.db"))
if os.path.exists(r'DBFA.db'):
    file = open(x, 'rb')
    data = file.read()
    os.remove(os.path.normpath(r"DBFA.db"))
print("Database flush completed!")
time.sleep(1)
print("Restarting DBFA")
time.sleep(1)
os.startfile(r"bleading_edge.py")

```

```
print("Removing ALL registered vouchers now...")
import os, time
print("        ___ ______ __   _____________    _____________     ______")
time.sleep(0.1)
print("       /  /_______/  /  /  /_______/  /  /  /________/    /  /_/ /")
time.sleep(0.1)
print("      /  /       /  /  /  /       /  /  /  /            /  /  / /")
time.sleep(0.1)
print("     /  /       /  /  /  /_______/  /  /  /  CLI        /  /   / /")
time.sleep(0.1)
print("    /  /       /  /  / // // // // /  /  /_________    /  /____/ /")
time.sleep(0.1)
print("   /  /       /  /  /  /-------/  /  /  /_________/   /  /_____/ /")
time.sleep(0.1)
print("  /  /       /  /  /  /       /  /  /  /            /  /      / /")
time.sleep(0.1)
print(" /  /_______/  /  /  /______ /  /  /  /            /  /       / /")
time.sleep(0.1)
print("/__/_______/__/  /__/_______/__/  /__/            /__/        /__/")
print(" ")
print(" ")
time.sleep(2)
print("Flushing record directory. ")
time.sleep(0.5)
print("Flushing record directory. . ")
time.sleep(0.5)
print("Flushing record directory. . . ")
time.sleep(0.5)

x = "cponmgmtsys.db"

try:
    if os.path.exists(r'cponmgmtsys.db'):
        os.remove(os.path.normpath(r"cponmgmtsys.db"))
    if os.path.exists(r'cponmgmtsys.db'):
        os.remove(os.path.normpath(r"cponmgmtsys.db"))
    print("Database flush completed!")
    time.sleep(1)
    print("Restarting DBFA")
    time.sleep(1)
    os.startfile(r"bleading_edge.py")
except:
    print("Critical error while operating on file.")
    time.sleep(0.5)
    print("Manually delete file - cponmgmtsys.db from your DBFA installation
directory")
    time.sleep(1)
    os.startfile(r"bleading_edge.py")
    time.sleep(19)

```

<Backup Authentication Provider Service (1/2)>

// The Code 5/10

```python
import requests, time, json, urllib, os
from tqdm import tqdm
import PySimpleGUI as sgx

global valn
valn = 0
#print(valn)

def telegram_bot_sendtext(bot_message):
    bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
    bot_chatID = '680917769'
    send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id='
+ bot_chatID + '&parse_mode=Markdown&text=' + bot_message
    response = requests.get(send_text)
    return response.json()


global last_update_id
lns = open(r"lastupdateid.txt", "r+")  #Opening / creating (if it doesn't exist
already) the .txt record file
last_update_id = (lns.read())

TOKEN = "1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis"
URL = "https://api.telegram.org/bot{}/".format(TOKEN)


def get_url(url):
    response = requests.get(url)
    content = response.content.decode("utf8")
    return content


def get_json_from_url(url):
    content = get_url(url)
    js = json.loads(content)
    return js


sgx.theme('DarkBlue')
def Login():
    layout = [  [sgx.Text('Login to authenicate: ')],
                [sgx.Text('Username: '), sgx.InputText()],
                [sgx.Text('Password: '), sgx.InputText(password_char='*')],
                [sgx.Button('Authenicate'), sgx.Button('Cancel')] ]
    window = sgx.Window('deltaAuthenication Service', layout)
    while True:
        event, values = window.read()
        if event in (None, 'Cancel'): # if user closes window or clicks cancel
            window.close
            os._exit(0)
            window.close()
            window.close()
            os._exit(1)
        if values[0] == 'ed' and values[1] == 'edd':
            window.close()
            window.close()
            print("As you're accessing sensitive information, we'll require you to
authenicate this request.")
            time.sleep(2)
            print("")
```

```
58              print("Please open your Telegram application and authenicate the request
    from the *DBFA Communicator* bot by following the instructions there.")
59              time.sleep(0.5)
60              for i in tqdm (range (10), desc="Waiting to detect authenication:  "):

61                  texter = "This is an authenication request for DBFA Backup and
    Reset." + "\n\n" + "Please send the passcode to authenicate this request." + "\n\n" +
    "DBFA Security"
62                  sender = telegram_bot_sendtext(texter)
63                  main()
64          else:
65              sgx.theme('DarkRed')
66              Login()
67
68
69  def get_updates(offset=None):
70      global updates
71      url = URL + "getUpdates"
72      if offset:
73          url += "?offset={}".format(offset)
74      js = get_json_from_url(url)
75      updates = js
76      return updates
77
78
79  def get_last_update_id(updates):
80      global last_update_id, valn
81      update_ids = []
82      for update in updates["result"]:
83          update_ids.append(int(update["update_id"]))
84      try:
85          last_update_id = max(update_ids)
86          lns = open(r"lastupdateid.txt", "r+")  #Opening / creating (if it doesn't
    exist already) the .txt record file
87          lns.truncate(0)
88          lns.write('%d'%last_update_id)
89      except ValueError:
90          lns = open(r"lastupdateid.txt", "r+")  #Opening / creating (if it doesn't
    exist already) the .txt record file
91          last_update_id = int(lns.read())+1
92
93
94
95
96  def echo_all(updates):
97      global last_update_id
98      for update in updates["result"]:
99          text = update["message"]["text"]
100         chat = update["message"]["chat"]["id"]
101         global valn
102         #print(valn)
103         if valn == 0:
104             if text == "ed":
105                 valn = 1
106                 send_message("You have authenicated a DBFA Backup & Switch
    request.\n\nThis allows your installation of DBFA to be backed-up.\n\nIf this wasn't
    you, contact support and revoke your Telegram bot login at the earliest.\n\nDBFA
    Security", chat)
107                 ins = open(r"delauth.txt", "a+")  #Opening / creating (if it doesn't
    exist already) the .txt record file
```

```
                ins.write('%s'%update)
                ins.close()
                get_updates(last_update_id)
                get_last_update_id(updates)
                os.startfile('dbfabackupper.py')
                time.sleep(0.5)
                echo_all(updates)

            else:
                texterx = text + ": That'd be wrong. Please try again."
                send_message(texterx, chat)
        elif valn == 1:
            get_updates(last_update_id)
            get_last_update_id(updates)
            time.sleep(2)
            os._exit(0)



def get_last_chat_id_and_text(updates):
    global last_update_id

    num_updates = len(updates["result"])
    last_update = num_updates - 1
    text = updates["result"][last_update]["message"]["text"]
    chat_id = updates["result"][last_update]["message"]["chat"]["id"]
    return (text, chat_id)


def send_message(text, chat_id):
    text = urllib.parse.quote_plus(text)
    url = URL + "sendMessage?text={}&chat_id={}".format(text, chat_id)
    get_url(url)



def main():
    global last_update_id
    url = URL + "setWebhook?url="
    get_url(url)
    get_updates(last_update_id)
    get_last_update_id(updates)
    while True:
        get_updates(last_update_id)

        if len((updates)["result"]) > 0:
            last_update_id += 1
            echo_all(updates)
        time.sleep(0.5)


Login()

```

<Backup Creation Service (2/2)>

// The Code 6/10

```python
from zipfile import ZipFile
import os, shutil, time
from tqdm import tqdm


command = "cls"
os.system(command)


print("DBFA Backup & Switch Utility")
time.sleep(0.7)
if
os.path.exists(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\delauth
.txt'):
    pass
else:
    print("Authenication bypased. Exiting.")
    time.sleep(1)
    os._exit(0)
print("-------------------------------------")
print("All previous backups will be removed.")
print("-------------------------------------")
print("")
time.sleep(2)
print("Fetching settings..")
print("")
time.sleep(0.5)


def copier():
    for i in tqdm (range (100), desc="Processing:                          "):
        slave = r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFATempc'
        if os.path.exists(r'delauth.txt'):
            master = r'cponmgmtsys.db'
            shutil.copy(master, slave)
            master = r'lastupdateid.txt'
            shutil.copy(master, slave)
            time.sleep(0.0000000001)
            master = r'DBFA.db'
            shutil.copy(master, slave)
            master = r'DBFA_CUSTCC.db'
            shutil.copy(master, slave)
            master = r'DBFA_handler.db'
            shutil.copy(master, slave)
            master = r'recmaster.db'
            shutil.copy(master, slave)
            time.sleep(0.0000000001)
            master = r'invoicemaster.db'
            shutil.copy(master, slave)
            master = r'registry.txt'
            shutil.copy(master, slave)
            master = r'stockature.txt'
            shutil.copy(master, slave)
            master = r'tempfile.temp'
            shutil.copy(master, slave)
            master = r'qr-code.png'
            shutil.copy(master, slave)
            time.sleep(0.0000000001)
            master = r'log.txt'
            shutil.copy(master, slave)
```

```
            os.remove(r'delauth.txt')
            master = r'DBFA_vend.db'
            shutil.copy(master, slave)
            master = r'client_secrets.json'
            shutil.copy(master, slave)
            master = r'DBFAdeliveries.txt'
            shutil.copy(master, slave)
            master = r'graphing_testbuild.py'
            shutil.copy(master, slave)
            master = r'wrelogin.pyw'
            shutil.copy(master, slave)
            master = r'run_DBFA.pyw'
            shutil.copy(master, slave)
            master = r'lastupdateid.txt'
            shutil.copy(master, slave)
            master = r'dlr.pyq'
            shutil.copy(master, slave)

try:
    for i in tqdm (range (100), desc="Creating File Structure:          "):
        if
os.path.exists(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATem
pc'):

 shutil.rmtree(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATem
pc', ignore_errors=True)
        if
os.path.exists(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Ba
ckup&Switchc'):

 shutil.rmtree(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Ba
ckup&Switchc', ignore_errors=True)
        if not
os.path.exists(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATem
pc'):

 os.mkdir(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATempc',
mode = 0o777, dir_fd = None)
        if not
os.path.exists(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Ba
ckup&Switchc'):

 os.mkdir(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Backup&
Switchc', mode = 0o777, dir_fd = None)
        time.sleep(0.000001)
    copier()









    def get_all_file_paths(directory):
        file_paths = []
        for root, directories, files in os.walk(directory):
            for filename in files:
                filepath = os.path.join(root, filename)
                file_paths.append(filepath)
        return file_paths

    def main():
```

```
        directory = r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATempc'
        file_paths = get_all_file_paths(directory)

        with ZipFile('DBFABackup.zip','w') as zip:
            for file in file_paths:
                zip.write(file)



    if __name__ == "__main__":
        for i in tqdm (range (100), desc="Zipping Files:                    "):
            main()
            time.sleep(0.000001)
    time.sleep(1)

    shutil.move('DBFABackup.zip', 'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Backup&Switchc')

    for i in tqdm (range (100), desc="Cleaning up:                      "):

 shutil.rmtree(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATempc', ignore_errors=True)
        time.sleep(0.000001)
    for i in tqdm (range (100), desc="Writing Restoration Instructions: "):
        slave = r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATempc'
        master = r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\Assets\\instructions.txt'
        shutil.copy(master, slave)
        time.sleep(0.000001)
    print("")
    print("Backup created succesfully.")
    time.sleep(2)

    drivefac = (input("Enter *1* to further backup the same to your Google Drive. To cancel, press *enter*"))
    if drivefac == "1":
        time.sleep(1)
        from pydrive.auth import GoogleAuth
        from pydrive.drive import GoogleDrive
        import os

        g_login = GoogleAuth()
        g_login.LocalWebserverAuth()
        drive = GoogleDrive(g_login)

        folderName = 'DBFA_Backup&Switchc'  # Please set the folder name.


        drive_folder = drive.CreateFile({
            'title': "DBFA_Backup&Switchc",
            "mimeType": "application/vnd.google-apps.folder"
        })
        drive_folder.Upload()

        directory = r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Backup&Switchc'

```

```
        folders = drive.ListFile(
            {'q': "title='" + folderName + "' and mimeType='application/vnd.google-apps.folder' and trashed=false"}).GetList()
        for folder in folders:
            if folder['title'] == folderName:
                file2 = drive.CreateFile({'parents': [{'id': folder['id']}]})

 file2.SetContentFile(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Backup&Switchc\\DBFABackup.zip')
                file2.Upload()
        os.startfile(r'delauth.py')
    else:
        time.sleep(2)
        os.startfile(r'delauth.py')
        os._exit(0)



        print("Directory: {} backed up successfully".format(directory))
except:
    time.sleep(1)
    print("PERMISSION ERROR: We couldn't get access to DBFA directories.")

```

<Hire-Employee GUI Form Script>

// The Code 7/10

```
import os, time, sqlite3, requests, json

os.system('cls')

def telegram_bot_sendtext(bot_message):
        bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
        bot_chatID = '680917769'
        send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
        response = requests.get(send_text)
        return response.json()




print("DBFA will now open a GUI-based form for you to enter the required details.")
time.sleep(1)

import PySimpleGUI as sg

SYMBOL_UP =    '▲'
SYMBOL_DOWN =  '▼'

def collapse(layout, key):
    return sg.pin(sg.Column(layout, key=key))


section1 = [[sg.Text('Name:')],
            [sg.Input(key='-IN1-')],
            #[sg.Input(key='-IN11-')],
            [sg.Text('Gender:')],
            [sg.Checkbox('Male', key='male'), sg.Checkbox('Female', key='female'), sg.Checkbox('Others', key='othergender')],
            [sg.Text('Date of Birth:')],
            [sg.InputCombo(('Select - ', 'January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'), size=(20, 1))],
            [sg.Text('Year:')],
            [sg.Input(key='-IN3-')],
            [sg.Text('Day:')],
            [sg.Input(key='-IN4-')]]

section2 = [[sg.Text('Email:')],
            [sg.I(k='-IN5-')],
            [sg.Text('Mobile Contact:')],
            [sg.I(k='-IN6-')],
            [sg.Text('Residential Address:')],
            [sg.I(k='-IN7-')],
            [sg.Text("Employee's UPI ID for salary payments:")],
            [sg.I(k='-IN8-')]]

section3 = [[sg.Text('Department Name:')],
            [sg.InputCombo(('Select - ', 'IT' ,'Administration', 'Sales', 'Care-taking', 'Logistics'), size=(20, 1))],
            #[sg.Input(key='-IN11-')],
            [sg.Text('Designation:')],
            [sg.Input(key='-IN8-')],
            [sg.Text('Salary:')],
            [sg.Input(key='-IN9-')]]

```

```
56
57 layout =   [[sg.Text('Hire an employee')],
58             #### Section 1 part ####
59             [sg.T(SYMBOL_DOWN, enable_events=True, k='-OPEN SEC1-',
   text_color='white'), sg.T('Personal Details', enable_events=True,
   text_color='yellow', k='-OPEN SEC1-TEXT')],
60             [collapse(section1, '-SEC1-')],
61             #### Section 2 part ####
62             [sg.T(SYMBOL_DOWN, enable_events=True, k='-OPEN SEC2-',
   text_color='white'),
63             sg.T('Personal Details', enable_events=True, text_color='white', k='-OPEN
   SEC2-TEXT')],
64             [collapse(section2, '-SEC2-')],
65             #### Buttons at bottom ####
66             [sg.Button('Proceed'), sg.Button('Exit')]]
67
68
69 layout3 = [[sg.Text('Hire an employee')],
70             [sg.T(SYMBOL_DOWN, enable_events=True, k='-OPEN SEC3-',
   text_color='white'), sg.T('Employment Details', enable_events=True,
   text_color='yellow', k='-OPEN SEC3-TEXT')],
71             [collapse(section3, '-SEC3-')],
72             [sg.Button('Complete Form >>>'), sg.Button('Exit')]]
73
74 window = sg.Window('deltaDBFA 8.2 - Add New Employee', layout)
75 arter = 0
76 opened1, opened2 = True, True
77
78 while True:             # Event Loop
79     event, values = window.read()
80     eventx, valuesx = event, values
81     #print(event, values)
82     if event == sg.WIN_CLOSED or event == 'Exit':
83         break
84
85     if event.startswith('-OPEN SEC1-'):
86         opened1 = not opened1
87         window['-OPEN SEC1-'].update(SYMBOL_DOWN if opened1 else SYMBOL_UP)
88         window['-SEC1-'].update(visible=opened1)
89
90     if event.startswith('-OPEN SEC2-'):
91         opened2 = not opened2
92         window['-OPEN SEC2-'].update(SYMBOL_DOWN if opened2 else SYMBOL_UP)
93         window['-OPEN SEC2-CHECKBOX'].update(not opened2)
94         window['-SEC2-'].update(visible=opened2)
95
96     if  event.startswith('Proceed'):
97         axt = open(r"dbfaempre.txt", "w+")
98         axt.write(str(eventx))
99         axt.write(str(valuesx))
100        #print(eventx, valuesx)
101        global ds1
102        ds1 = (eventx, valuesx)
103        window.close()
104        window = sg.Window('deltaDBFA 8.2 - Add New Employee', layout3)
105        opened1, opened2 = True, True
106        while True:             # Event Loop
107            event, values = window.read()
108            eventr, valuesr = event, values
109            global ds2
```

```
            ds2 = (eventr, valuesr)
            if event == sg.WIN_CLOSED or event == 'Exit':
                break
                break

            if event.startswith('Complete Form'):
                opened1 = not opened1
                arter = 1
                window.close()
                break

window.close()

if arter == 0:
    print("No data recieved! ")
if arter == 1:
    import os
    os.system('cls')
    print("Data sets recieved! ")
    ds1 = (dict(ds1[1]))
    ds2 = (dict(ds2[1]))
    print("----------------")
    print("Data from the filled form follows. These details will be sent to DBFA's
data repository for safekeeping: ")

    ds1a = list(ds1.keys())
    ds1b = list(ds1.values())

    ds2a = list(ds2.keys())
    ds2b = list(ds2.values())

    #print(ds1a, ds1b, ds2a, ds2b)

    print("\n\nName          :", ds1b[0])
    print("Gender        :", ds1a[ds1b.index(True)])
    #print("DOB: (Month)  :", ds1b[4])
    monthx = ["January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"]
    for i in monthx:
        if i == ds1b[4]:
            month = (int(monthx.index(i))+1)
    #print(month)
    if len(str(ds1b[6])) == 1:
        day = ("0"+'%s'%ds1b[6])
    else:
        day = (ds1b[6])
    dob = str(ds1b[5])+"-"+str(month)+"-"+str(day)
    print("DOB           :", dob)
    print("EMail         :", ds1b[8])
    print("Mobile Contact:", ds1b[9])
    print("Resd. Address :", ds1b[10])
    print("UPI Payment ID:", ds1b[11])

    print("\nDepartment    :", ds2b[0])
    print("Designation   :", ds2b[1])
    print("Salary        :", ds2b[2])
    if len(str(month)) == 1:
        month = "0"+'%s'%month
        #print(month)

```

```
168     import sqlite3, time
169
170     empmas = sqlite3.connect(r'dbfaempmaster.db')
171     empmascur = empmas.cursor()
172     empmascur.execute("SELECT MAX(Oid) FROM emp")
173     Oid = (int(empmascur.fetchall()[0][0]) + 1)
174     strix = ('insert into emp(Oid, Name, DOB, Email, Mobile, Address, UPI, Dept,
    Post, Salary) values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)')
175     io = (Oid, str(ds1b[0]), str(dob), str(ds1b[8]), int(ds1b[9]), str(ds1b[10]),
    ds1b[11], str(ds2b[0]), str(ds2b[1]), int(ds2b[2]))
176     empmascur.execute(strix, io)
177     empmas.commit()
178     #Oid = "Oid"+'%s'%"3"
179     #xstr = "ALTER TABLE attendance ADD COLUMN "+'%s'%Oid+" CHAR"
180     #empmascur.execute(xstr)
181     #empmas.commit()
182     print("Employee registered in all DBFA databases")
183     time.sleep(2)
184     print("\n\nData sets logged with DBFA..")
185     empmascur.execute("SELECT * FROM emp ORDER BY Oid DESC LIMIT 0, 1")
186     time.sleep(1)
187     print("\n\nNEW Employee Details:: ")
188     print("------------------------------------------------------------")
189     print('%s'%ds1b[0]+(30-len(str(ds1b[0])))*" "+"deltaDBFA Employee Identifier")
190     print("Employee OID: ", empmascur.fetchall()[0][0])
191     print("------------------------------------------------------------\n\n")
192     time.sleep(2)
193
194
195
```

<Two-Factor-Authentication Modification Service>

// The Code 8/10

```
import requests, time, json, urllib, os, math, random, sqlite3
from tqdm import tqdm
import PySimpleGUI as sgx



def settingscommonfetch(SettingsType):
    import sqlite3
    settings = sqlite3.connect(r'dbfasettings.db')
    settingsx = settings.cursor()
    settingsx.execute(("SELECT Value from settings WHERE SettingsType = ?"), (SettingsType,))
    settingsfetch = (settingsx.fetchall()[0][0])
    return settingsfetch

def settingsmodifier(SettingsType, NewValue):
    import sqlite3
    settings = sqlite3.connect(r'dbfasettings.db')
    settingsx = settings.cursor()
    settingsx.execute(("UPDATE settings SET Value = ? WHERE SettingsType = ?"), (NewValue, SettingsType))
    settings.commit()




def transitionprogress():
    from colorama import init, Fore, Back, Style
    os.system("cls")
    time.sleep(1)
    print(Fore.WHITE+'|'+Fore.RED+'███ OFF |')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.RED+'███     '+Fore.GREEN+'█|')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.RED+'██     '+Fore.GREEN+'██|')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.RED+'█     '+Fore.GREEN+'███|')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.GREEN+ ' ON  ███|'+Fore.WHITE)
    time.sleep(1.24)
    os.system("cls")


def transitionprogressneg():
    from colorama import init, Fore, Back, Style
    os.system("cls")
    time.sleep(1)
    print(Fore.WHITE+'|'+Fore.GREEN+ ' ON  ███|')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.RED+'█     '+Fore.GREEN+'███|')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.RED+'██     '+Fore.GREEN+'██|')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.RED+'███     '+Fore.GREEN+'█|')
    time.sleep(0.3)
    print(Fore.WHITE+'|'+Fore.RED+'███ OFF |'+Fore.WHITE)
    time.sleep(1.24)
    os.system("cls")


```

```
59 def telegram_bot_sendtext(bot_message):
60     bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
61     bot_chatID = '680917769'
62     send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id='
   + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
63     response = requests.get(send_text)
64     return response.json()
65 
66 def getOTP():
67     digits = "0123456789"
68     otp = ""
69     otp += (digits[math.floor(random.random() * 10)])
70     otp += (digits[math.floor(random.random() * 10)])
71     otp += (digits[math.floor(random.random() * 10)])
72     otp += (digits[math.floor(random.random() * 10)])
73     otp += (digits[math.floor(random.random() * 10)])
74     otp += (digits[math.floor(random.random() * 10)])
75     return otp
76 
77 def main():
78     # Create the Updater and pass it your bot's token.
79     # Make sure to set use_context=True to use the new context based callbacks
80     # Post version 12 this will no longer be necessary
81     updater = Updater("1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis",
   use_context=True)
82 
83     updater.dispatcher.add_handler(CommandHandler('auth', start))
84     updater.dispatcher.add_handler(CallbackQueryHandler(button))
85     updater.dispatcher.add_handler(CommandHandler('help', help_command))
86 
87     # Start the Bot
88     updater.start_polling()
89 
90     # Run the bot until the user presses Ctrl-C or the process receives SIGINT,
91     # SIGTERM or SIGABRT
92     updater.idle()
93 
94 def Login():
95     layout = [  [sgx.Text('Login to authenicate: ')],
96                 [sgx.Text('Username: '), sgx.InputText()],
97                 [sgx.Text('Password: '), sgx.InputText(password_char='*')],
98                 [sgx.Button('Authenicate'), sgx.Button('Cancel')] ]
99     window = sgx.Window('deltaAuthenication Service', layout)
100     while True:
101         event, values = window.read()
102         if event in (None, 'Cancel'): # if user closes window or clicks cancel
103             print("DBFA 2FA Modification cancelled!! ")
104             time.sleep(5)
105             window.close()
106             os._exit(0)
107             window.close()
108             window.close()
109             os._exit(1)
110         if values[0] == 'ed' and values[1] == 'edd':
111             window.close()
112             window.close()
113             print("delta2FAAuthenicationAPI-lite")
114             print("As you're accessing sensitive information, this request needs to
   be validated.")
115             time.sleep(2)
```

```
116             print("")
117             print("A one-time-password (OTP) has been sent to your Telegram account
    to validate this request")
118             time.sleep(0.5)
119             for i in tqdm (range (10), desc="Generating and sending OTP"):
120                 time.sleep(0.00001)
121             texter = "delta2FA Authenication Service" + "\n\n" + "A login request has
    been recieved from your DBFA installation. Entering this OTP in your DBFA
    installation will enable/ disable 2FA authenication aboard that installation." +
    "\n\n" + "Do not share this OTP with anyone. Use key: "+ '%s'%delsecox  + "\n\n" +
    "deltaAuthenication Service"
122             sender = telegram_bot_sendtext(texter)
123             window.close()
124             window.close()
125             mainprocess()
126 
127         else:
128             sgx.theme('DarkRed')
129             Login()
130 
131 def mainprocess():
132     time.sleep(1)
133     tries = 0
134     while(1):
135         passkeyinput = input("Enter the validation key recieved: ")
136         if tries in (0, 1, 2, 3, 4):
137             if passkeyinput == delsecox:
138                 telegram_bot_sendtext("You have validated a DBFA 2FA login
    request.\n\nThis allows your installation of DBFA and the data it stores, to be
    used.\n\nContact support and revoke your bot login at the earliest if this wasn't
    you!\n\ndeltaAuthenication Service")
139                 tries += 1
140                 time.sleep(1)
141                 from colorama import init, Fore, Back, Style
142                 print("\n\nEnable DBFA two-factor-authenication? ")
143                 print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
144                 settfac1x = input(("n:     "+ '|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|:
    '))
145                 if settfac1x == "y":
146                     if (settingscommonfetch(6)) != 1:
147                         settingsmodifier(6, 1)
148                         transitionprogress()
149                         time.sleep(3)
150                         print("Please wait while we restart DBFA main client")
151                         os.startfile('bleading_edge.py')
152                         time.sleep(1)
153                         os._exit(0)
154                     else:
155                         print("DBFA 2FA is already enabled!")
156                         time.sleep(3)
157                         print("Please wait while we restart DBFA main client")
158                         os.startfile('bleading_edge.py')
159                         time.sleep(1)
160                         os._exit(0)
161 
162                 if settfac1x == "n":
163                     if (settingscommonfetch(6)) != 0:
164                         settingsmodifier(6, 0)
165                         transitionprogressneg()
166                         time.sleep(3)
```

```
                        print("Please wait while we restart DBFA main client")
                        os.startfile('bleading_edge.py')
                        time.sleep(1)
                        os._exit(0)
                    else:
                        print("DBFA 2FA is already enabled!")
                        time.sleep(3)
                        print("Please wait while we restart DBFA main client")
                        os.startfile('bleading_edge.py')
                        time.sleep(1)
                        os._exit(0)

            else:
                print("Invalid validation key entered. Please retry! ")
                tries += 1
                telegram_bot_sendtext('%s'%tries+"/5: validation attempts; no valid key recieved.")
        else:
            telegram_bot_sendtext("(5) validation attempts completed, yet no valid key recieved." + "\n\n" + "2FA denied." +"\n\ndeltaAuthenication Service")
            print("(5) validation attempts completed, yet no valid key recieved. 2FA denied. ")
            print("DBFA 2FA Modification cancelled!! ")
            time.sleep(5)
            break

delsecox = getOTP()
Login()
```

<Sales Analysis Provider Service>

// The Code 9/10

```python
import sqlite3, time
import matplotlib.pyplot as plt
salesr = sqlite3.connect(r'dbfasales.db')
salesx = salesr.cursor()
datefetch = []

salesx.execute("SELECT DISTINCT date FROM sales")
for i in salesx.fetchall():
    datefetch.append((i[0]))

netray = []
for i in datefetch:
    salesx.execute(("SELECT sum(prof) FROM sales WHERE date = ?"), (i, ))
    netray.append(salesx.fetchall()[0][0])

# Plotting
plt.plot(datefetch, netray, color='purple', linestyle='dashed', linewidth = 3,
        marker='o', markerfacecolor='magenta', markersize=12)
# naming the x axis
plt.xlabel('Date')
# naming the y axis
plt.ylabel('Profit')
# Graph Title
plt.title('DBFA Profit Report')
time.sleep(1)
# Finally, display
plt.show()
```

<Process Handler GUI Prompts>

```
import os
import time
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')
def Login():
    import PySimpleGUI as sgx
    sgx.theme('DarkTeal10')
    #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
    layout = [  [sgx.Text('DBFA C')],
                [sgx.Text('_____')],
                [sgx.Text(' ')],
                [sgx.Text('DBFA Client exited as the timeout period was crossed.')],
                [sgx.Text('(Client needs to be launched within a minute of completing primary login.)')],
                [sgx.Text(' ')],
                [sgx.Text('Error code: dtaec1103-authtimeout')],
                [sgx.Button('Exit')], [sgx.Text('DBFA crashed?')], [sgx.Button('Submit Crash Report')]]


    window = sgx.Window('DBFA Security Service', layout)
    while True:
        event, values = window.read()
        if event in (None, 'Exit'): # if user closes window or clicks cancel
            window.close()
            delche = 1
            break
        if event in ('DBFA crashed?', 'Submit Crash Report'):
            print("crashreport")
            window.close()
            delche = 1
            import PySimpleGUI as sgx
            sgx.theme('DarkTeal10')
            #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
            layout = [  [sgx.Text('DBFA C')],
                        [sgx.Text('_____')],
                        [sgx.Text(' ')],
                        [sgx.Text('Crash report has been submited! ')],
                        [sgx.Button('Exit')]]
            window = sgx.Window('delta ClientOAuth Handler', layout)
            while True:
                event, values = window.read()
                if event in (None, 'Exit'): # if user closes window or clicks cancel
                    window.close()
                    break




import PySimpleGUI as sg
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')

Login()

```

```
import os
import time
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')
def Login():
    import PySimpleGUI as sgx
    sgx.theme('BlueMono')
    #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
    layout = [  [sgx.Text('[redacted]  ¯\_(⊙‿⊙)_/¯')],
                [sgx.Text('[redacted]  ¯\_(⊙‿⊙)_/¯')],
                [sgx.Text("delta Installation Integrity Validation Service")],
                [sgx.Text("[redacted]")],
                [sgx.Text('Contacting DBFA servers')],
                [sgx.Text('Integrity checker result: ')],
                [sgx.Text("     DBFA's code has been tampered with! Please rectify this!")],
                [sgx.Text("     Further diagnosis details: ")],
                [sgx.Text('          DBFA is updated with the latest source')],
                [sgx.Text('\nError code: dtaec1207-unxpcrs-updated\n')],
                [sgx.Text('Analysis not correct?'), sgx.Button('Submit crash report'), sgx.Button('Exit')]]
    window = sgx.Window('delta Process Handler', layout)
    while True:
        event, values = window.read()
        if event in (None, 'Exit'): # if user closes window or clicks cancel
            window.close()
            delche = 1
            break
        if event in ('Analysis not correct?', 'Submit crash report'):
            print("crashreport")
            window.close()
            delche = 1
            import PySimpleGUI as sgx
            sgx.theme('DarkTeal10')
            #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
            layout = [  [sgx.Text('[redacted]')],
                        [sgx.Text('[redacted]')],
                        [sgx.Text(' ')],
                        [sgx.Text('Crash report has been submited! ')],
                        [sgx.Button('Exit')]]
            window = sgx.Window('delta ClientOAuth Handler', layout)
            while True:
                event, values = window.read()
                if event in (None, 'Exit'): # if user closes window or clicks cancel
                    window.close()
                    break









import PySimpleGUI as sg
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')

Login()
```

```
import os
import time
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')
def Login():
    import PySimpleGUI as sgx
    sgx.theme('BlueMono')
    #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
    layout = [  [sgx.Text('[redacted]  ¯\_(⊙‿⊙)_/¯')],
                [sgx.Text('[redacted]  ¯\_(⊙‿⊙)_/¯')],
                [sgx.Text("delta Installation Integrity Validation Service")],

[sgx.Text("[redacted]")],
                [sgx.Text('Contacting DBFA servers')],
                [sgx.Text('Integrity checker result: ')],
                [sgx.Text("     DBFA's code has been tampered with! Please rectify this!")],
                [sgx.Text("     Further diagnosis details: ")],
                [sgx.Text('          DBFA is running on an older build.\n     Please update your installation!\n     New updated often pack new features, bug-fixes and security improvements.')],
                [sgx.Text("If you're still encountering errors:")],
                [sgx.Text("          - We request you to try rebooting your device and re-opening DBFA.")],
                [sgx.Text("          - If you encounter the same issue again, please contact DBFA support or re-install DBFA AFTER CREATING A DATA BACKUP FOR DBFA.")],
                [sgx.Text('\nError code: dtaec1207-unxpcr-upavail\n')],
                [sgx.Text('Analysis not correct?'), sgx.Button('Submit crash report'), sgx.Button('Exit')]]
    window = sgx.Window('delta Process Handler', layout)
    while True:
        event, values = window.read()
        if event in (None, 'Exit'): # if user closes window or clicks cancel
            window.close()
            delche = 1
            break
        if event in ('Analysis not correct?', 'Submit crash report'):
            print("crashreport")
            window.close()
            delche = 1
            import PySimpleGUI as sgx
            sgx.theme('DarkTeal10')
            #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
            layout = [  [sgx.Text('[redacted]')],
                        [sgx.Text('[redacted]')],
                        [sgx.Text(' ')],
                        [sgx.Text('Crash report has been submited! ')],
                        [sgx.Button('Exit')]]
            window = sgx.Window('delta ClientOAuth Handler', layout)
            while True:
                event, values = window.read()
                if event in (None, 'Exit'): # if user closes window or clicks cancel
                    window.close()
                    break


```

```
53
54
55 import PySimpleGUI as sg
56 if os.path.exists(r'userblock.txt'):
57     os.remove(r'userblock.txt')
58 if os.path.exists(r'userblock.zconf'):
59     os.remove(r'userblock.zconf')
60
61 Login()
62
63
64
```

```
import os
import time
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')
def Login():
    import PySimpleGUI as sgx
    sgx.theme('BlueMono')
    #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
    layout = [  [sgx.Text('[redacted]     ¯\_(⊙_⊙)_/¯')],
                [sgx.Text('[redacted]     ¯\_(⊙_⊙)_/¯')],
                [sgx.Text("delta Installation Integrity Validation Service")],
                [sgx.Text("[redacted]")],
                [sgx.Text('Contacting DBFA servers')],
                [sgx.Text('Integrity checker result: ')],
                [sgx.Text("     DBFA's code has been tampered with! Please rectify this!")],
                [sgx.Text("     Further diagnosis details: ")],
                [sgx.Text('          Master Copy Error: dta=intl.err_undercommit?imp=buildahead')],
                [sgx.Text('\nError code: dta=intl.err_undercommit?imp=buildahead\n')],
                [sgx.Text('Analysis not correct?'), sgx.Button('Submit crash report'), sgx.Button('Exit')]]
    window = sgx.Window('delta Process Handler', layout)
    while True:
        event, values = window.read()
        if event in (None, 'Exit'): # if user closes window or clicks cancel
            window.close()
            delche = 1
            break
        if event in ('Analysis not correct?', 'Submit crash report'):
            print("crashreport")
            window.close()
            delche = 1
            import PySimpleGUI as sgx
            sgx.theme('DarkTeal10')
            #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
            layout = [  [sgx.Text('[redacted]')],
                        [sgx.Text('[redacted]')],
                        [sgx.Text(' ')],
                        [sgx.Text('Crash report has been submited! ')],
                        [sgx.Button('Exit')]]
            window = sgx.Window('delta ClientOAuth Handler', layout)
            while True:
                event, values = window.read()
                if event in (None, 'Exit'): # if user closes window or clicks cancel
                    window.close()
                    break



import PySimpleGUI as sg
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')
```

```
57
58 Login()
59
60
61
```

```
import os
import time
if os.path.exists(r'userblock.txt'):
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')
def Login():
    import PySimpleGUI as sgx
    sgx.theme('BlueMono')
    #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
    layout = [  [sgx.Text('[redacted]  ¯\_(⊙‿⊙)_/¯')],
                [sgx.Text('[redacted]  ¯\_(⊙‿⊙)_/¯')],
                [sgx.Text("delta Installation Integrity Validation Service")],

[sgx.Text("[redacted]")],
                [sgx.Text('Contacting DBFA servers')],
                [sgx.Text('Integrity checker result: ')],
                [sgx.Text("     The driver code of this installation of DBFA seems to be fine.\nIf you're still encountering errors:")],
                [sgx.Text("          - We request you to try rebooting your device and re-opening DBFA.")],
                [sgx.Text("          - If you encounter the same issue again, please contact DBFA support or re-install DBFA AFTER CREATING A DATA BACKUP FOR DBFA.")],
                [sgx.Text('\nError code: dta=intl.err_undercommit?imp=buildahead\n')],
                [sgx.Text('Analysis not correct?'), sgx.Button('Submit crash report'), sgx.Button('Exit')]]
    window = sgx.Window('delta Process Handler', layout)
    while True:
        event, values = window.read()
        if event in (None, 'Exit'): # if user closes window or clicks cancel
            window.close()
            delche = 1
            break
        if event in ('Analysis not correct?', 'Submit crash report'):
            print("crashreport")
            window.close()
            delche = 1
            import PySimpleGUI as sgx
            sgx.theme('DarkTeal10')
            #print("\nWARNING:  CAPS LOCK IS ENABLED!\n")
            layout = [  [sgx.Text('[redacted]')],
                        [sgx.Text('[redacted]')],
                        [sgx.Text(' ')],
                        [sgx.Text('Crash report has been submited! ')],
                        [sgx.Button('Exit')]]
            window = sgx.Window('delta ClientOAuth Handler', layout)
            while True:
                event, values = window.read()
                if event in (None, 'Exit'): # if user closes window or clicks cancel
                    window.close()
                    break




import PySimpleGUI as sg
if os.path.exists(r'userblock.txt'):
```

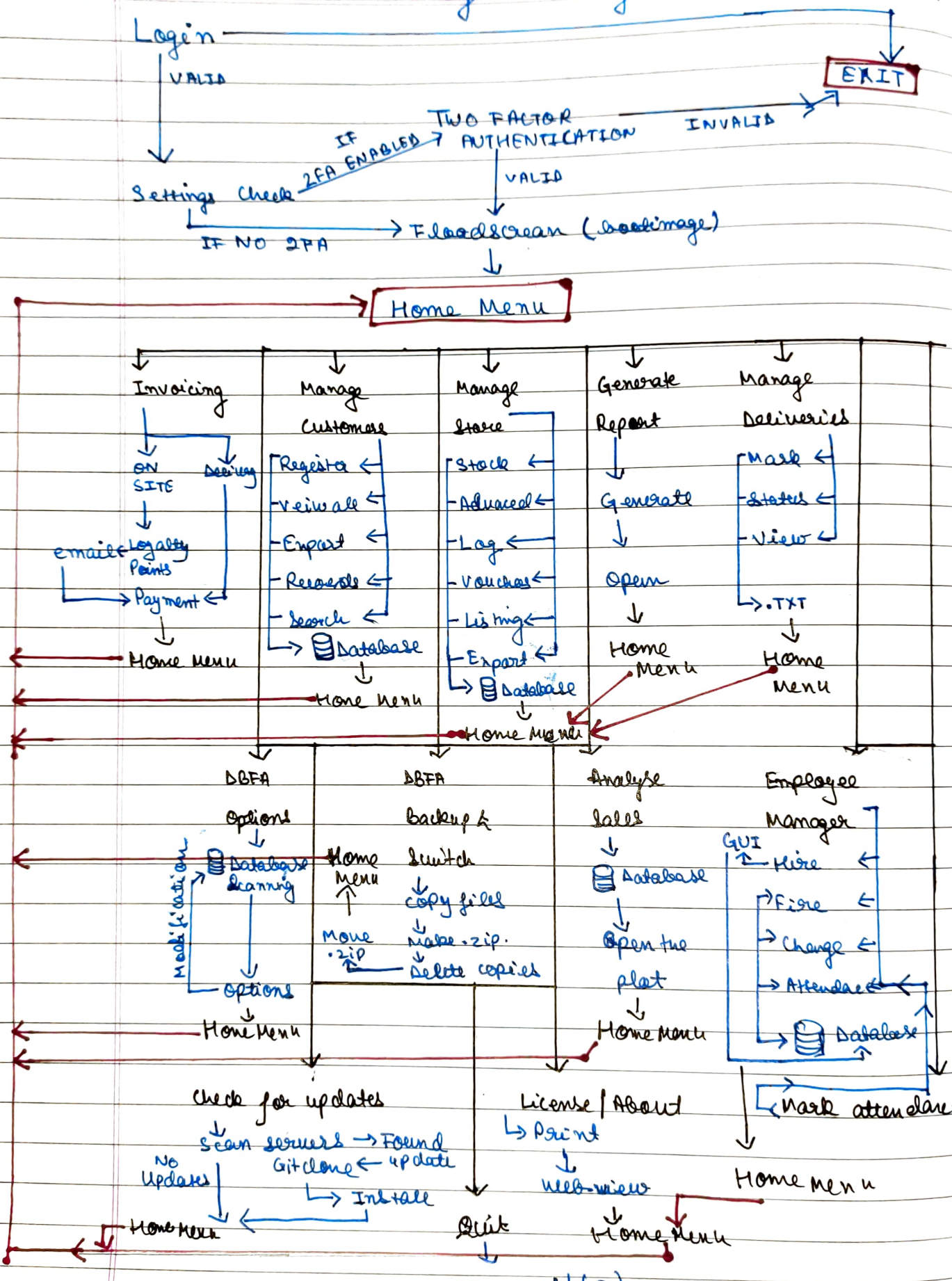
```
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')

Login()
```

(Please Turn Over)

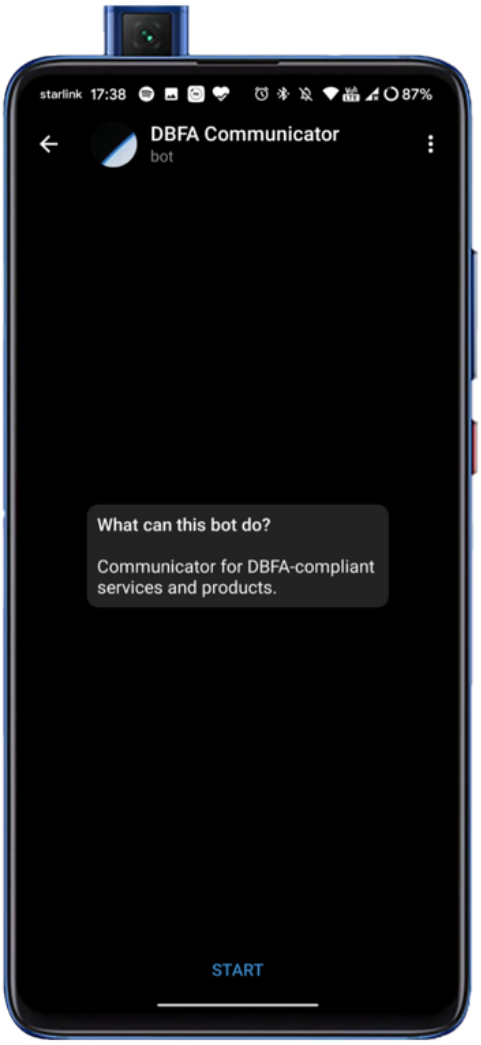
// Logic Chart

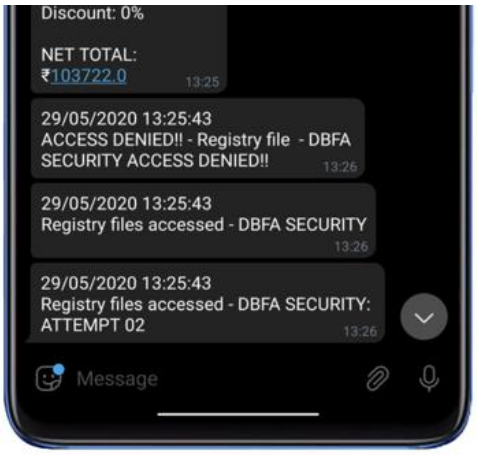
DBFA 8.34 Program Logic

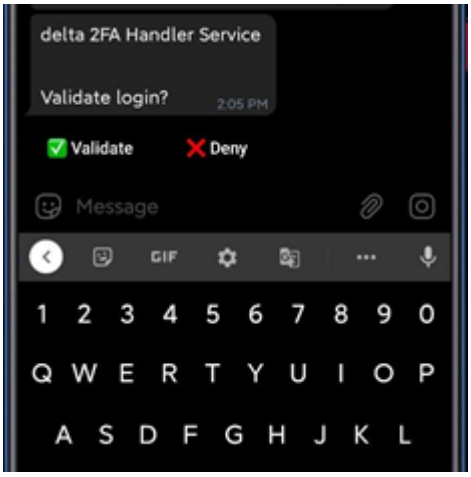


(Please Turn Over)

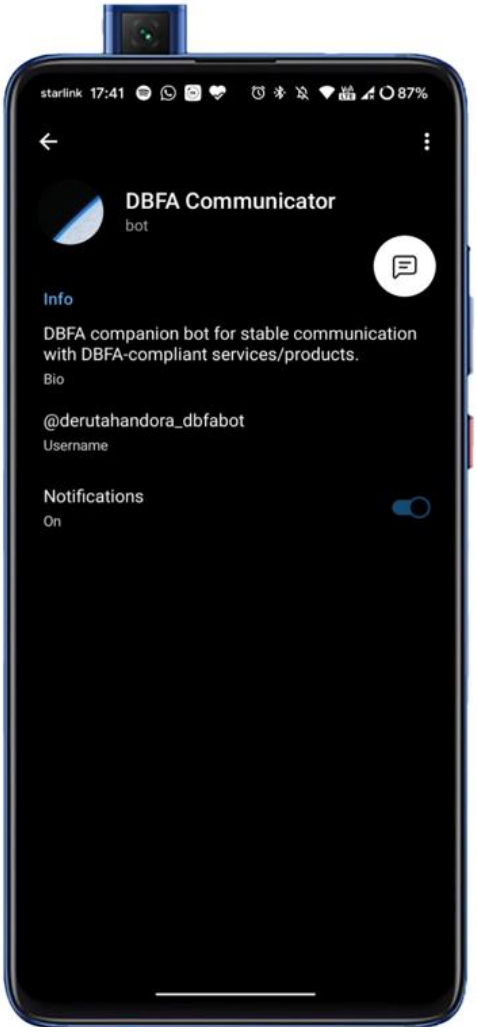
// Telegram Integration

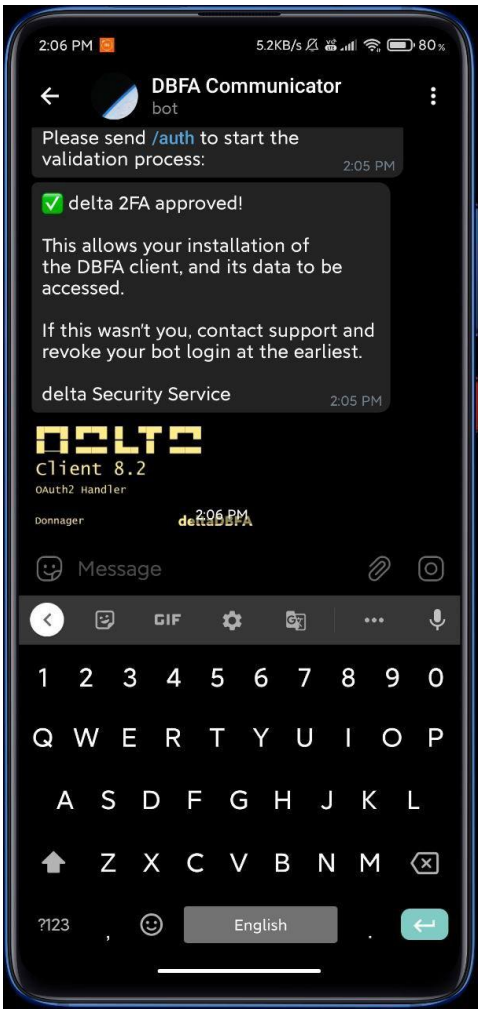

DBFA Communicator
bot
What can this bot do?
Communicator for DBFA-compliant services and products.
START

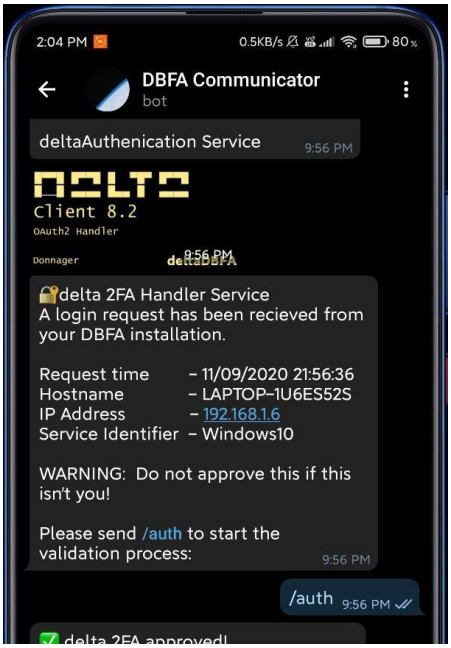

Discount: 0%
NET TOTAL:
₹103722.0
29/05/2020 13:25:43
ACCESS DENIED!! - Registry file - DBFA SECURITY ACCESS DENIED!!
29/05/2020 13:25:43
Registry files accessed - DBFA SECURITY
29/05/2020 13:25:43
Registry files accessed - DBFA SECURITY: ATTEMPT 02
Message

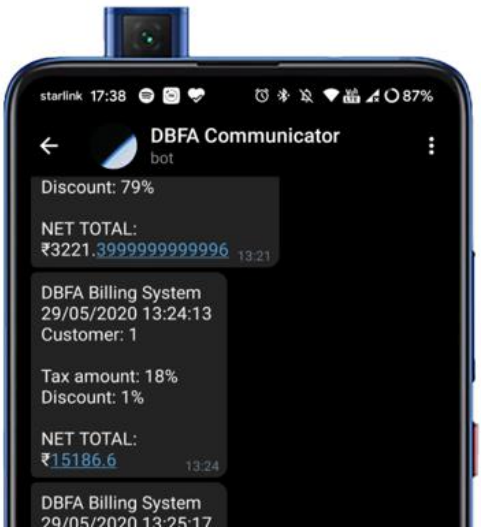

delta 2FA Handler Service
Validate login?
Validate
Deny
Message


DBFA Communicator
bot
Client 8.2
OAuth2 Handler
Donnager
Message


DBFA Communicator
bot
Info
DBFA companion bot for stable communication with DBFA-compliant services/products.
Bio
@derutahandora_dbfabot
Username
Notifications
On


DBFA Communicator
bot
Please send /auth to start the validation process:
delta 2FA approved!
This allows your installation of the DBFA client, and its data to be accessed.
If this wasn't you, contact support and revoke your bot login at the earliest.
delta Security Service
Client 8.2
OAuth2 Handler
Donnager
Message


DBFA Communicator
bot
deltaAuthenication Service
Client 8.2
OAuth2 Handler
Donnager
delta 2FA Handler Service
A login request has been recieved from your DBFA installation.
Request time - 11/09/2020 21:56:36
Hostname - LAPTOP-1U6ES52S
IP Address - 192.168.1.6
Service Identifier - Windows10
WARNING: Do not approve this if this isn't you!
Please send /auth to start the validation process:
/auth


DBFA Communicator
bot
Discount: 79%
NET TOTAL:
₹3221.3999999999996
DBFA Billing System
29/05/2020 13:24:13
Customer: 1
Tax amount: 18%
Discount: 1%
NET TOTAL:
₹15186.6
DBFA Billing System

(Please Turn Over)

// The Future

We're quite excited about DBFA's future. Once our CLI-based version is at its peak, we plan to expand upon it with a graphical-interface

We have already started work on the GUI version of DBFA. As you might have seen throughout the code, we have started transitioning certain elements of importance into their GUI form. We plan to complete the same soon!

Here's a sneak-peak into the earliest build of the future of DBFA:

DBFAx 0.2

(Please Turn Over)

// Bibliography

WORKS CITED FROM:

- Core Python Programming by R. Nageshwar Rao
- Python 3 Documentation
- StackOverflow – stackoverflow.com
- Google – google.com

- Python for Class – XII by Preeti Arora

```
def package_dbfadonager():
     /**
     * The program DBFAdonager implements an application that
     * houses a great store management software with many integrations.
     *
     * @author deltaonealpha
     */
     from DBFA import DBFAdocs
     if DBFAdocs.ExecutionStatus() == True:
          DBFAdocs.the_end()
          print("Thank You", DBFAdocs.endcredits())
          print("deltaDBFA8")
     else:
          print(DBFAdocs.HelloWorld())
          DBFAdocs.runExecution()
package_dbfadonager()
```

// deltaDBFA8

Developed by

Pranav Balaji and Sushant Gupta

Python Board Project