
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 8.4**
- **CHECK FOR UPDATES**
- **QUIT**

The home menu also contains two special triggers which are triggered upon entering special keywords which involve case-variations:

- **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.

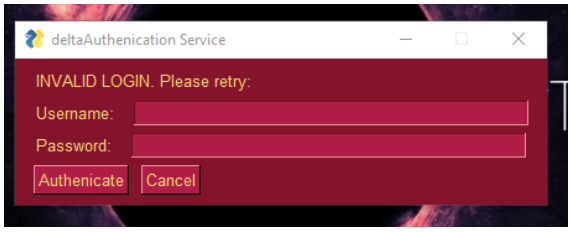
(Please Turn Over)

//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.

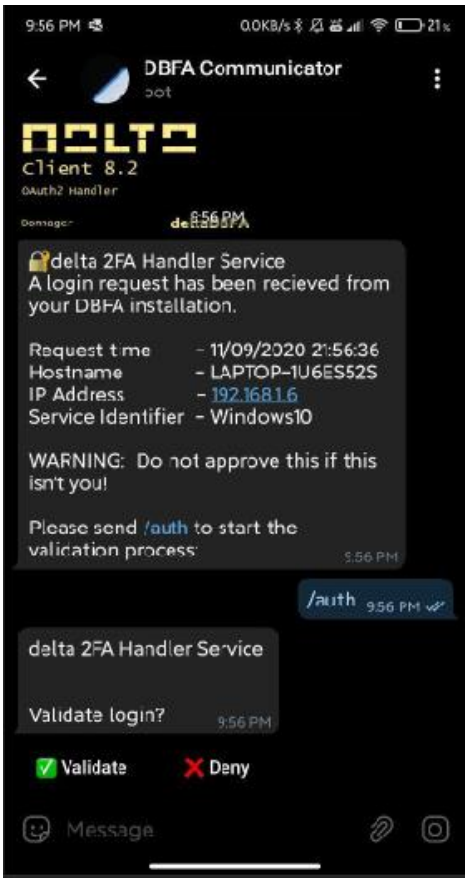
Upon successful login, a confirmation is provided and DBFA is subsequently launched. However, if the credentials entered are found to be incorrect, the same login page is relaunched, this time with a red background and text indicative of invalid creds.



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.

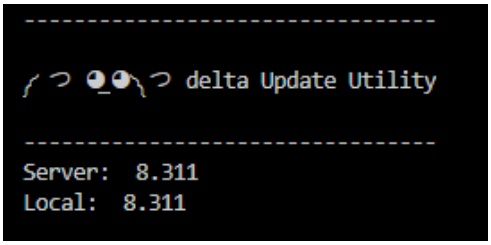


As this part makes use of an indefinite loop to be able to process each input from Telegram (including incorrect message replies to the request), DBFA uses SIGINT (Signal Interrupt) to break out of this loop. If the authentication is denied, DBFA exists gracefully.

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

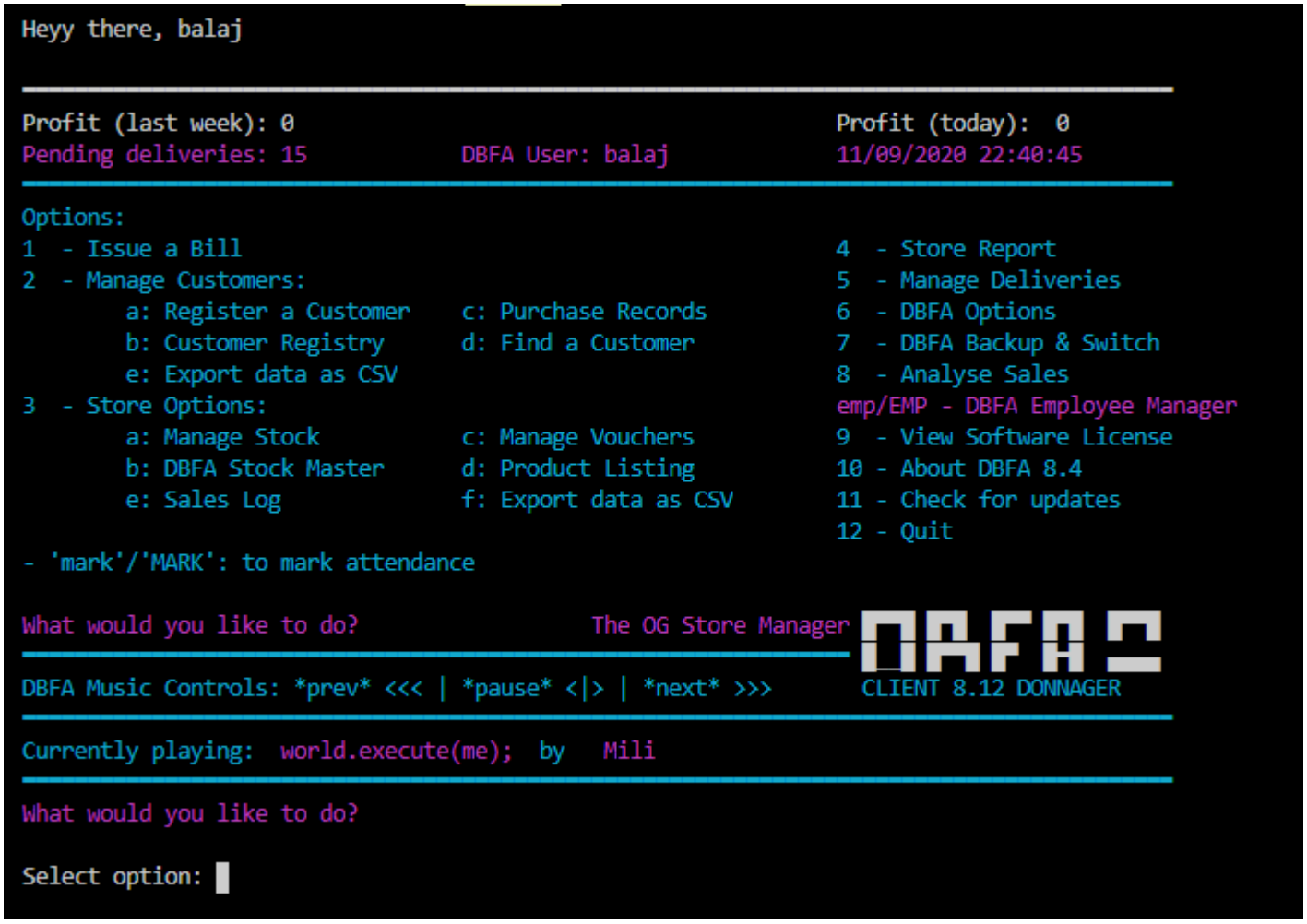
```
- telegram.ext.updater - INFO - Received signal 2 (SIGINT), stopping...
```

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.

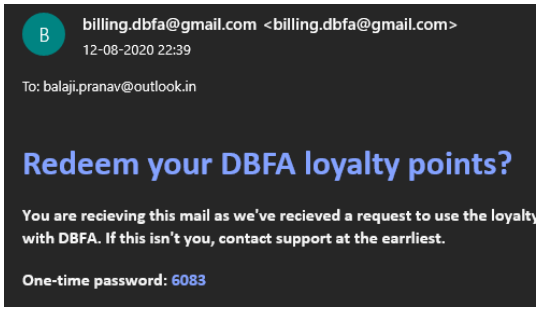
At this point, the user (presumably a store cashier) is given an option to either process the invoice as an in-store purchase, or as a delivery. In case of a delivery, the customer's name, and address (multi-line input) are enquired for and the same is stored in a file.

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:

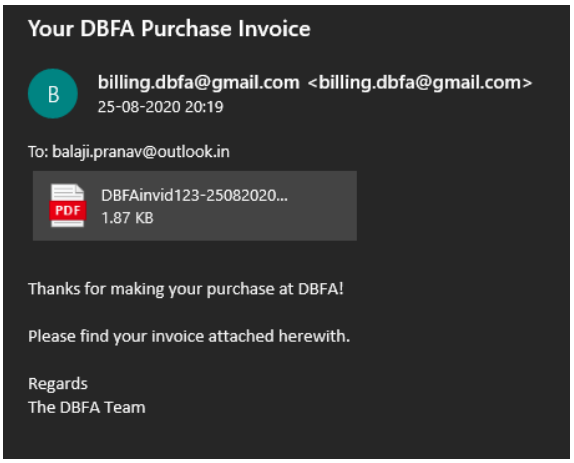
- Debit/ Credit Cards
- UPI
- Digital Wallets
- Cash

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.



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.

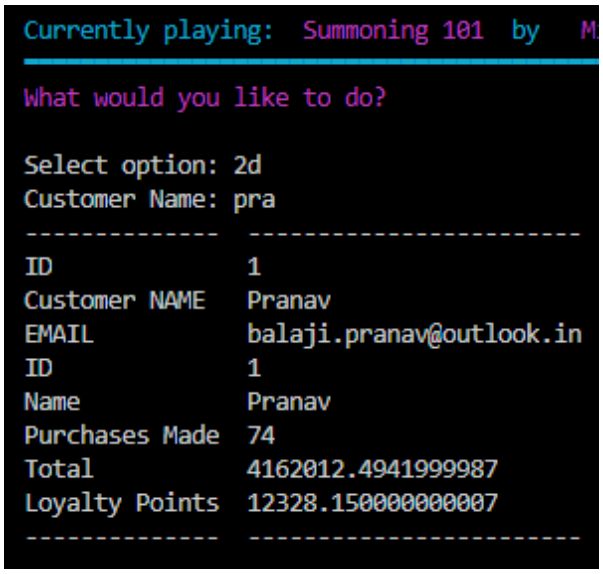
VIEW CUSTOMER-SPECIFIC PURCHASE RECORDS

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

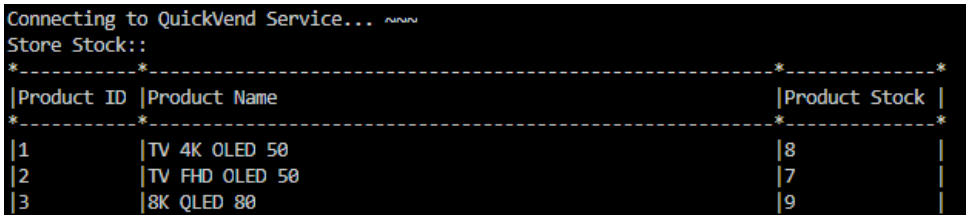
This option houses six sub-options:

- 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:

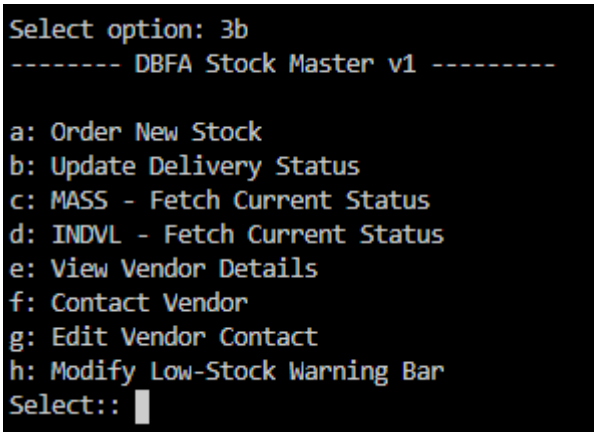
1. VIEW STOCK
   This prints out all stock information.



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 : | 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 AND Telegram is required for the same.

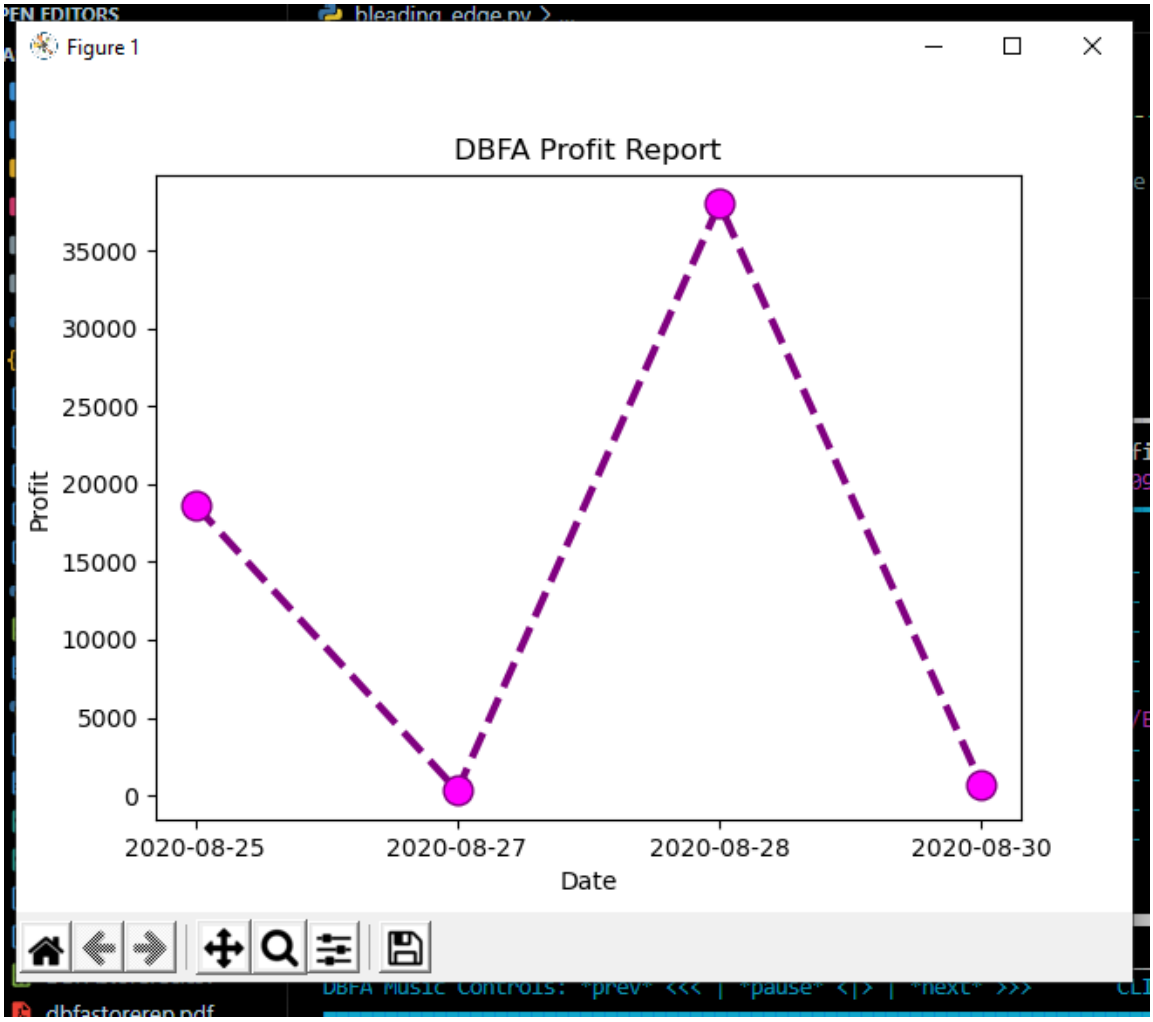
NOTE: 2FA CANNOT BE DISABLED FOR THIS SECURE OPERATION IN ANY CASE.

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.

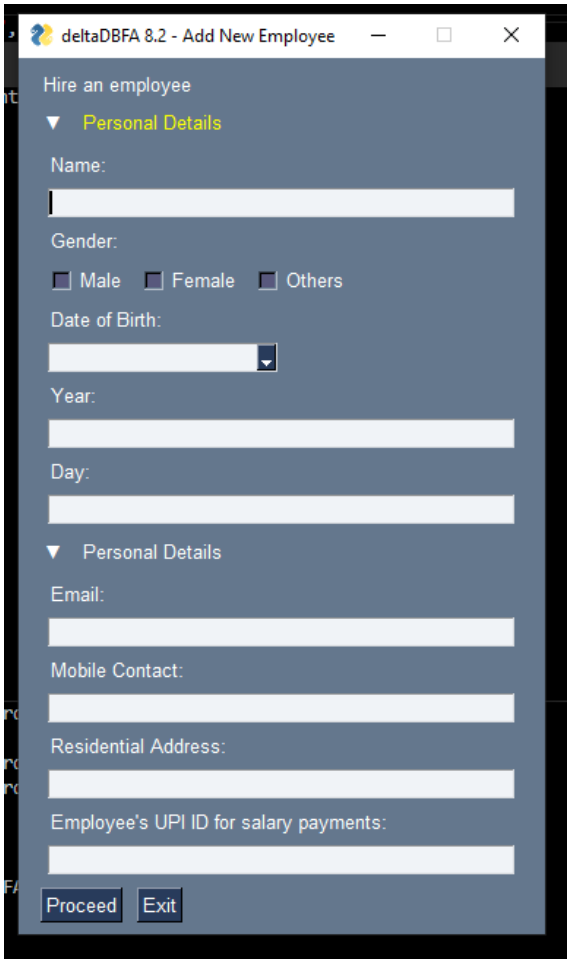
This is a high-level employee manager which provides a variety of functions, such as:

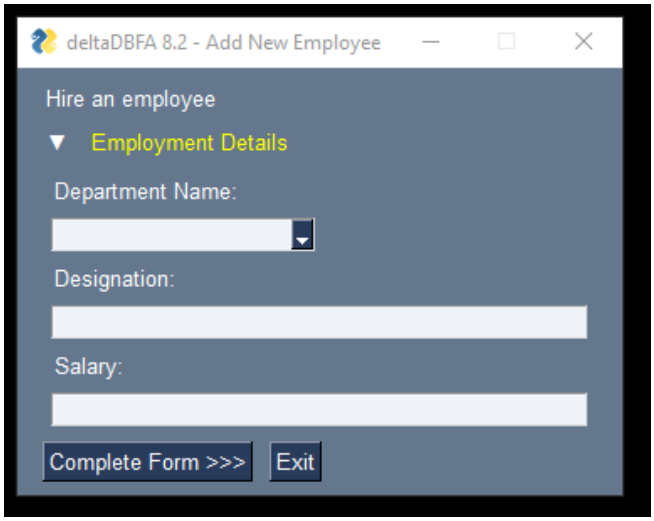
- HIRE AN EMPLOYEE
- VIEW EMPLOYEE RECORDS
- CHANGE EMPLOYEE DETAILS
- FIRE AN EMPLOYEE ʃ•'ᴗ'•ʖ⊃

- 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

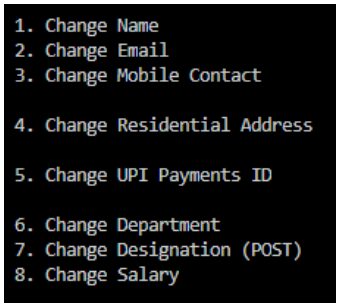
This option presents the user with a GUI-based form to enter details easily. These details are categorized automatically and sent to their respective storage spots.





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 *(8.4)*

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 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.

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 8.3** 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.

AUTHENTICATION BYPASS PROTECTION

As DBFA is based on Python and uses multiple scripts, one can bypass the main login script by opening multiple scripts in succession. This is taken care of as each successful login generates a login token, without which the main program won't boot.

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

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 ON CERTAIN FEATURES

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

(Please Turn Over)

//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 Pro (x64)
- Microsoft Visual Studio Code (for programming)
- Microsoft Windows Terminal (for testing)
- Microsoft Office Word 365 (for documentation)
- Microsoft GitHub (for code hosting and backups)
- Microsoft OneDrive (for code backups)

<Main Program>

// The Code 1/8


```
'''

by deltaonealpha and sushimuncher

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

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

```

```
60
61 def settingscommonfetch(SettingsType):
62     import sqlite3
63     settings = sqlite3.connect(r'dbfasettings.db')
64     settingsx = settings.cursor()
65     settingsx.execute(("SELECT Value from settings WHERE SettingsType = ?"),
(SettingsType,))
66     settingsfetch = (settingsx.fetchall()[0][0])
67     return settingsfetch
68
69 def settingsmodifier(SettingsType, NewValue):
70     import sqlite3
71     settings = sqlite3.connect(r'dbfasettings.db')
72     settingsx = settings.cursor()
73     settingsx.execute(("UPDATE settings SET Value = ? WHERE SettingsType = ?"),
(NewValue, SettingsType))
74     settings.commit()
75
76
77
78 logox = ('''
79    _____   ____    ____  ____  _____
80   / /  // / /  \\ /___  /__||  |--//
81  / /  // / /===| ///// ////||    //
82 /_/__// /_/__ / /     /    ||   //
83     ''')
84
85 global cccheck
86
87 #auth verification
88 if os.path.exists(r'userblock.zconf'):
89     print("Verifying login...")
90     print(" ")
91 if os.path.exists(r'userblock.txt'):
92     userblock = open(r"userblock.txt","r") #Opening / creating (if it doesn't exist
already) the .txt record file
93     redflag = 1
94 else:
95     redflag = 0
96 if os.path.exists(r'DBFA.zconf'):
97     pass
98 else:
99     print("Getting the database online.......")
100    time.sleep(0.2)
101    con = sqlite3.connect(r'DBFA.db')
102    print("Rebuilding database..")
103    c = con.cursor()
104    #c.execute("DROP TABLE cust;")
105    c.execute("""CREATE TABLE IF NOT EXISTS cust
106        (custt INTEGER PRIMARY KEY,
107        custname TEXT,
108        email TEXT);""")
109 try:
110    if os.path.exists(r'userblock.txt'):
111        os.remove(r'userblock.txt')
112    if os.path.exists(r'userblock.zconf'):
113        os.remove(r'userblock.zconf')
114 except PermissionError:
115    pass
116
```

```
117
118
119 #Stock Order Manager
120 xon = sqlite3.connect(r'DBFA_vend.db')
121 xbr7 = xon.cursor()
122 xbr7.execute("""CREATE TABLE IF NOT EXISTS stock
123     (prodid INTEGER PRIMARY KEY,
124     prodname CHAR,
125     ordqty VARCHAR(500),
126     delivered INT,
127     delstat CHAR,
128     vendor CHAR,
129     vendcont CHAR,
130     lowstock INT);""")
131 xon = sqlite3.connect(r'DBFA_vend.db')
132 xbr7 = xon.cursor()
133 if os.path.exists(r'DBFA_vend.db'):
134     pass
135 else:
136     xbr7.execute("""CREATE TABLE IF NOT EXISTS stock
137     (prodid INTEGER PRIMARY KEY,
138     prodname CHAR,
139     ordqty VARCHAR(500),
140     delivered INT,
141     delstat CHAR,
142     vendor CHAR,
143     vendcont CHAR
144     lowstock INT);""")
145
146 st = sqlite3.connect(r'DBFA_vend.db')
147 stx = st.cursor()
148 stockq = """UPDATE stock SET delstat = ? WHERE ordqty = 0"""
149 qans = ("DELIVERED", )
150 stx.execute(stockq, qans)
151 stockq = """UPDATE stock SET delstat = ? WHERE ordqty != 0"""
152 qans = ("TBD", )
153 stx.execute(stockq, qans)
154 st.commit()
155
156 #NEW Sales Report v2 DB Logger
157 sales = sqlite3.connect(r'dbfasales.db')
158 salesx = sales.cursor()
159 if os.path.exists(r'dbfasales.db'):
160     pass
161 else:
162     salesx.execute("""CREATE TABLE IF NOT EXISTS sales
163     (sno INT PRIMARY KEY,
164     custid INT,
165     prodid INT,
166     net INT,
167     prof INT,
168     date DATE);""")
169     sales.commit()
170     print("Table restructured! ")
171
172
173
174 class HiddenPrints:
175             def __enter__(self):
176                 self._original_stdout = sys.stdout
```

```
177                 sys.stdout = open(os.devnull, 'w')
178             def __exit__(self, exc_type, exc_val, exc_tb):
179                 sys.stdout.close()
180                 sys.stdout = self._original_stdout
181                 print()
182 
183 # TG Communicator
184 def telegram_bot_sendtext(bot_message):
185 
186     with HiddenPrints():
187         bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
188         bot_chatID = '680917769'
189         send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?
    chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
190         response = requests.get(send_text)
191         return response.json()
192 
193 def getOTP():
194     global otp
195     digits = "0123456789"
196     otp = ""
197     otp += (digits[math.floor(random.random() * 10)])
198     otp += (digits[math.floor(random.random() * 10)])
199     otp += (digits[math.floor(random.random() * 10)])
200     otp += (digits[math.floor(random.random() * 10)])
201 
202 
203 # DBFA Logo Printer
204 def logoprintxrt():
205             print("         ___ ______ __   _____________     _____________
    _______")
206             time.sleep(0.05)
207             print("       /  /_______/  /  /  /_______/  /  /  /________/    /  /_/
    /")
208             time.sleep(0.05)
209             print("      /  /       /  /  /  /       /  /  /  /            /  /  /
    /")
210             time.sleep(0.05)
211             print("     /  /       /  /  /  /_______/  /  /  /  CLI        /  /   /
    /")
212             time.sleep(0.05)
213             print("    /  /       /  /  / // // // // /  /  /_________    /  /____/
    /")
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             print(" ")
223             print(" ")
224 
225 
226 
```

```
227 # Database section
228 # Stock Records Master DB
229 ssh = sqlite3.connect(r'DBFA_handler.db')
230 ssh7 = ssh.cursor()
231 #c.execute("DROP TABLE cust;")
232 ssh7.execute("""CREATE TABLE IF NOT EXISTS sshandler
233     (prodid INTEGER,
234     prodname CHAR,
235     ssstock INTEGER);""")
236 ssh = sqlite3.connect('DBFA_handler.db')
237 ssh7 = ssh.cursor()
238 if os.path.exists(r'DBFA_handler.db'):
239     pass
240 else:
241     ssh7.execute("""CREATE TABLE IF NOT EXISTS sshandler
242         (prodid INTEGER,
243         prodname CHAR,
244         ssstock INTEGER);""")
245
246
247 # Invoice Master DB
248 inmas = sqlite3.connect(r'invoicemaster.db')
249 inmascur = inmas.cursor()
250 #c.execute("DROP TABLE cust;")
251 inmascur.execute("""CREATE TABLE IF NOT EXISTS inmas
252     (indid INTEGER);""")
253 inmas = sqlite3.connect('invoicemaster.db')
254 inmascur = inmas.cursor()
255 if os.path.exists(r'invoicemaster.db'):
256     pass
257 else:
258     inmascur.execute("""CREATE TABLE IF NOT EXISTS inmas
259     (indid INTEGER);""")
260
261
262 # Voucher Records Master DB
263 isol = sqlite3.connect(r'cponmgmtsys.db')
264 isolx = isol.cursor()
265 isolx.execute("""CREATE TABLE IF NOT EXISTS cponmaster
266     (cponid CHAR PRIMARY KEY,
267     cponlim INTEGER,
268     cponvalue INTEGER,
269     cpondtb DATE);""")
270 isol = sqlite3.connect('cponmgmtsys.db')
271 isolx = isol.cursor()
272 if os.path.exists(r'cponmgmtsys.db'):
273     pass
274 else:
275     isolx.execute("""CREATE TABLE IF NOT EXISTS cponmaster
276         (cponid CHAR PRIMARY KEY,
277         cponlim INTEGER,
278         cponvalue INTEGER,
279         cpondtb DATE);""")
280
281
282 # Customer Records Master DB
283 xon = sqlite3.connect(r'DBFA_CUSTCC.db')
284 xbr7 = xon.cursor()
285 xbr7.execute("""CREATE TABLE IF NOT EXISTS custcc
286     (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)"]
    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]
    profitmarker = 0
    for crrt in namiex:
        profvalue = profitx[profitmarker]
        gg = (namiex.index(crrt)) + 1
        strx = "insert into recmasterx(prodid, prodname, prodprofit, prodsales,
netprof) values(?, ?, ?, ?, ?)"
        io = (gg, crrt, profvalue, 0, 0)
        profitmarker += 1
        recx.execute(strx, io)
        rec.commit()
        print("Added record: ", crrt)









# Functions::

# DBFA Stock Master v1


def orderstock(prodid, qty):
    st = sqlite3.connect(r'DBFA_vend.db')
    stx = st.cursor()
    stockq = """UPDATE stock SET ordqty = ordqty + ? WHERE prodid = ?"""
    qans = (qty, prodid)
    stx.execute(stockq, qans)
    st.commit()
    print("Recieved order for product", prodid, "; qty", qty)
    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()


def delivered(qty, prodid):
    st = sqlite3.connect(r'DBFA_vend.db')
    stx = st.cursor()
    stockq = """SELECT ordqty FROM stock WHERE prodid = ?"""
    qans = (prodid, )
    stx.execute(stockq, qans)
    st.commit()
    aaa = (int(stx.fetchall()[0][0]))
```

```
391     if aaa >= qty:
392         st = sqlite3.connect(r'DBFA_vend.db')
393         stx = st.cursor()
394         xstockq = """UPDATE stock SET ordqty = ordqty - ? WHERE prodid = ?"""
395         xqans = (qty, prodid, )
396         stx.execute(xstockq, xqans)
397         xstockq = """UPDATE stock SET delivered = delivered + ? WHERE prodid = ?"""
398         xqans = (qty, prodid, )
399         stx.execute(xstockq, xqans)
400         st.commit()
401         print("Quantity ", qty, "recieved.")
402     else:
403         print("Cannot set status for qty:", qty, "for prod:", prodid, "as the
    ordered qty is lower than the delivered qty being set.")
404     st = sqlite3.connect(r'DBFA_vend.db')
405     stx = st.cursor()
406     stockq = """UPDATE stock SET delstat = ? WHERE ordqty = 0"""
407     qans = ("DELIVERED", )
408     stx.execute(stockq, qans)
409     stockq = """UPDATE stock SET delstat = ? WHERE ordqty != 0"""
410     qans = ("TBD", )
411     stx.execute(stockq, qans)
412     st.commit()
413
414
415 def delstatmass():
416     isol = sqlite3.connect(r'DBFA_vend.db')
417     isolx = isol.cursor()
418     print("\n\nProduct stock yet to be recieved: \n")
419     isolx = isol.cursor()
420     isolx.execute(("SELECT * from stock WHERE delstat = ?"), ("TBD", ))
421     rows = isolx.fetchall()
422     col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor",
    "Vendor", "Vendor Contact", "Lowstock Bar")
423     table(col_labels, rows)
424     print("\n\nProduct stock recieved: \n")
425     isolx.execute(("SELECT * from stock WHERE delstat = ?"), ("DELIVERED", ))
426     rows = isolx.fetchall()
427     col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor",
    "Vendor", "Vendor Contact", "Lowstock Bar")
428     table(col_labels, rows)
429
430 def delstatindvl(prodid):
431     isol = sqlite3.connect(r'DBFA_vend.db')
432     isolx = isol.cursor()
433     print("\n\nDetails for product ID", prodid, ": \n")
434     isolx.execute(("SELECT * from stock WHERE prodid = ?"), (prodid, ))
435     rows = isolx.fetchall()
436     if rows in ([], (), "", " ", None):
437         print("-------------------------------------------------------------")
438         print("| Entered product ID could not be located in DBFA's records |")
439         print("-------------------------------------------------------------\n\n")
440     else:
441         col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor",
    "Vendor", "Vendor Contact", "Lowstock Bar")
442         table(col_labels, rows)
443
444
445 def vendorfetch(prodid):
446     isol = sqlite3.connect(r'DBFA_vend.db')
```

```
447     isolx = isol.cursor()
448     print("\n\nVendor details for product ID", prodid, ":")
449     isolx.execute(("SELECT vendor from stock WHERE prodid = ?"), (prodid, ))
450     rows = isolx.fetchall()
451     if rows in ([], (), "", " ", None):
452         print("-------------------------------------------------------------")
453         print("| Entered product ID could not be located in DBFA's records |")
454         print("-------------------------------------------------------------\n")
455     else:
456         print("Vendor:", rows[0][0])
457 
458 
459 def vendorcontact(prodid):
460     isol = sqlite3.connect(r'DBFA_vend.db')
461     isolx = isol.cursor()
462     print("\n\nVendor contact for product ID", prodid, ":")
463     isolx.execute(("SELECT vendcont from stock WHERE prodid = ?"), (prodid, ))
464     rows = isolx.fetchall()
465     if rows in ([], (), "", " ", None):
466         print("-------------------------------------------------------------")
467         print("| Entered product ID could not be located in DBFA's records |")
468         print("-------------------------------------------------------------\n")
469     else:
470         print("Vendor contact:", rows[0][0])
471         confac = input("Contact vendor? (y/n): ")
472         if confac == "y":
473             import webbrowser
474             webbrowser.open(('mailto:'+'%s'%rows[0][0]), new=1)
475         elif confac == "n":
476             pass
477         else:
478             print("Invalid option entered. ")
479             vendorcontact(prodid)
480 
481 
482 def lowbarmodif(prodid):
483     isol = sqlite3.connect(r'DBFA_vend.db')
484     isolx = isol.cursor()
485     print("\n\nLow-stock bar for product ID", prodid, ":")
486     isolx.execute(("SELECT lowstock from stock WHERE prodid = ?"), (prodid, ))
487     rows = isolx.fetchall()
488     if rows in ([], (), "", " ", None):
489         print("-------------------------------------------------------------")
490         print("| Entered product ID could not be located in DBFA's records |")
491         print("-------------------------------------------------------------\n")
492     else:
493         print("Current low-stock bar:", rows[0][0])
494         modifier = int(input("Enter the new limit: "))
495         if modifier <=0:
496             print("Cannot set a negative/ null value as low-stock warning limit! ")
497             print("Try again: ")
498             time.sleep(1)
499             lowbarmodif(prodid)
500             print("Modified. New limit:", rows[0][0])
501         isol = sqlite3.connect(r'DBFA_vend.db')
502         isolx = isol.cursor()
503         isolx.execute(("UPDATE stock SET lowstock = ? WHERE prodid = ?"), (modifier,
prodid, ))
504         isol.commit()
505         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"
    time.sleep(1)
    recx.execute("SELECT DISTINCT prodid, prodname, prodprofit, prodsales, netprof FROM recmasterx")
    rows = recx.fetchall()
    '''
    for row in rows:
        print(row)
    '''
```

```
    ll = [("P.ID","Prod. Name","Profit P.U.","Qty. Sold","Net Profit")]
    rows = ll + rows
    time.sleep(0.1)
    #print(" ")


def repupdate(prodid):
    rec = sqlite3.connect(r'recmaster.db')
    recx = rec.cursor()
    # hidden prints here ig
    updatexr = ("UPDATE recmasterx SET prodsales = prodsales + 1 WHERE prodid = ?")
    updatexxr = ("UPDATE recmasterx SET netprof = prodsales*prodprofit WHERE prodid
= ?")
    indicator = (prodid, )
    recx.execute(updatexr, indicator)
    recx.execute(updatexxr, indicator)
    rec.commit()
    recx.close()


# Feature not released
# Invoice Master Record Maintainer
def inmaintainer():
    inmas = sqlite3.connect('invoicemaster.db')
    inmascur = inmas.cursor()
    updatetrtt = """UPDATE inmas SET indid = indid + 1"""
    inmascur.execute(updatetrtt)
    inmas.commit()
    inmascur.close()
    #time.sleep(1)
    #toaster.show_toast+("DBFA QuickVend Service - Background Sync", duration = 0.4)

def infetch():
    global inval
    inmas = sqlite3.connect('invoicemaster.db')
    inmascur = inmas.cursor()
    inmascur.execute("SELECT DISTINCT indid FROM inmas")
    rows = inmascur.fetchall()
    inval = int(rows[0][0])


# Stock System
# Mass Stock Allocator
def massmaintainer(inxstock):  #defining a function to input data into the SQL
database's table
    try:
        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]
        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()
```

```
606     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)"]
607     for crrt in namiex:
608         gg = (namiex.index(crrt)) + 1
609         str = "insert into sshandler(prodid, prodname, ssstock) values(?, ?, ?)"
610         strxx = (gg, crrt, inxstock,)
611         ssh7.execute(str, strxx)
612         ssh.commit()
613     ssh7.close()
614     time.sleep(1)
615     print("DBFA QuickVend service - Stock universally enforced to", inxstock)
616     time.sleep(1)
617 
618 
619 # Induvidual Stock Allocator
620 def ssxupdatescript(inxssincremental, prodid):
621     ssh = sqlite3.connect('DBFA_handler.db')
622     ssh7 = ssh.cursor()
623     updatetr = """UPDATE sshandler SET ssstock = ssstock + ? WHERE prodid = ?"""
624     xrindicator = (inxssincremental, prodid)
625     ssh7.execute(updatetr, xrindicator)
626     ssh.commit()
627     ssh7.close()
628     time.sleep(1)
629     print("DBFA QuickVend Service - Stock added for", prodid, "as",
    inxssincremental)
630 
631 # Stock Data Fetcher
632 def ssxsuperfetch():
633     ssh = sqlite3.connect('DBFA_handler.db')
634     ssh7 = ssh.cursor()
635     print("Connecting to QuickVend Service... ~~~")  #SQL connection prompt
636     print("Store Stock:: ")
637     time.sleep(1.5)
638     #Re-writing to refresh connection
639     ssh7 = ssh.cursor()
640     ssh7.execute("SELECT DISTINCT prodid, prodname, ssstock FROM sshandler")
641     rows = ssh7.fetchall()
642     col_labels = ("Product ID", "Product Name", "Product Stock")
643     table(col_labels, rows)
644 
645 
646 # Purchase-time Stock Handler
647 def ssxstockmaintainer(prodid):
648     ssh = sqlite3.connect('DBFA_handler.db')
649     ssh7 = ssh.cursor()
650     updatetrtt = """UPDATE sshandler SET ssstock = ssstock - 1 WHERE prodid = ?"""
651     xrindicatortt = (prodid,)
```

```
652     ssh7.execute(updatetrtt, xrindicatortt)
653     ssh.commit()
654     ssh7.close()
655     #time.sleep(1)
656     #toaster.show_toast("DBFA QuickVend Service - Background Sync", duration = 0.4)
657 
658 # Induvidual Stock Fetcher
659 def ssxstockmaster(prodid):
660     global ssxvarscheck
661     ssxvarscheck = 0
662     ssh = sqlite3.connect('DBFA_handler.db')
663     ssh.row_factory = lambda cursor, row: row[0]
664     ssh7 = ssh.cursor()
665     csrr = ("SELECT ssstock FROM sshandler WHERE prodid = (?);")
666     csrrxt = (prodid,)
667     ssh7.execute(csrr, csrrxt)
668     rows = ssh7.fetchall()
669     # print(rows) #debug point
670     values = ','.join(str(v) for v in rows)
671     ssxdsccheck = "1 2 3 4"
672     isolx = sqlite3.connect(r'DBFA_vend.db')
673     isolxx = isolx.cursor()
674     isolxx.execute(("SELECT lowstock from stock WHERE prodid = ?"), (prodid, ))
675     rowsr = isolxx.fetchall()
676     limiterx = []
677     for ix in range (1, int((rowsr[0][0]))+1):
678         limiterx.append(ix)
679     if int(values) in limiterx:
680         print("[Stock running out] Currently in stock: ", values, "pieces. Restock
    ASAP...")
681         ssxvarscheck = 1
682     elif int(values) == 0:
683         print("Current product stock: ", values)
684         ssxvarscheck = 2
685     elif int(values) < 1:
686         print("Current product stock: ", values)
687         ssxvarscheck = 2
688     else:
689         print("Current product stock: ", values)
690         ssxvarscheck = 1
691     time.sleep(0.2)
692     #toaster.show_toast("DBFA QuickVend Service - Background Sync", duration =
    0.3427)
693 
694 
695 
696 
697 # Voucher System
698 # Voucher User
699 def cponuse(cponid):
700     isol = sqlite3.connect('cponmgmtsys.db')
701     isolx = isol.cursor()
702     mod = """UPDATE cponmaster SET cponlim = cponlim - 1 WHERE cponid = ?"""
703     idler = (cponid, )
704     isolx.execute(mod, idler)
705     isol.commit()
706 
707 # Single Voucher Data Fetcher
708 def cpon_singlefetch(cponid):
709     isol = sqlite3.connect('cponmgmtsys.db')
```

```
710     isolx = isol.cursor()
711     isol.row_factory = lambda cursor, row: row[0]
712     csrr = ("SELECT cponid, cponlim, cponvalue FROM cponmaster WHERE cponid = (?);")
713     csrrxt = (cponid, )
714     isolx.execute(csrr, csrrxt)
715     rows = isolx.fetchall()
716     values = ','.join(str(v) for v in rows)
717     print("DNSS Coupon ", values)
718 
719 
720 # Single Voucher Data Fetcher For Billing
721 def cpon_ssinglefetch(cponid):
722     isol = sqlite3.connect('cponmgmtsys.db')
723     isolx = isol.cursor()
724     isol.row_factory = lambda cursor, row: row[0]
725     csrr = ("SELECT cponid FROM cponmaster WHERE cponid = (?);")
726     csrrxt = (cponid, )
727     isolx.execute(csrr, csrrxt)
728     rows = isolx.fetchall()
729     values = ','.join(str(v) for v in rows)
730     global sfetch_values
731     sfetch_values = values[2:-3]
732 
733 # Voucher Issuer
734 def cponissuer(cponid, cponlim, cponvalue):
735     isol = sqlite3.connect('cponmgmtsys.db')
736     isolx = isol.cursor()
737     #try:
738     str = "insert into cponmaster(cponid, cponlim, cponvalue) values('%s', '%s',
'%s')"
739     iox = (cponid, cponlim, cponvalue)
740     isolx.execute(str % iox)
741     isol.commit()
742     cpon_singlefetch(cponid)
743     print("DSNN: Coupon", cponid, "having discount %", cponvalue, "created for",
cponlim, "times of usage.")
744     #except sqlite3.IntegrityError:
745         #print("DNSS voucher already exists")
746         #cpon_singlefetch(cponid)
747 
748 # Single Voucher Value Fetcher
749 def cpon_valfetch(cponid):
750     global valdock
751     isol = sqlite3.connect('cponmgmtsys.db')
752     isolx = isol.cursor()
753     isol.row_factory = lambda cursor, row: row[0]
754     csrr = ("SELECT cponvalue FROM cponmaster WHERE cponid = (?);")
755     csrrxt = (cponid, )
756     isolx.execute(csrr, csrrxt)
757     rows = isolx.fetchall()
758     values = ''.join(str(v) for v in rows)
759     #print(values[1:-2])
760     valdock = values[1:-2]
761     print(valdock)
762 
763 #Voucher Limit Data Fetcher
764 def cpon_limfetch(cponid):
765     isol = sqlite3.connect('cponmgmtsys.db')
766     isolx = isol.cursor()
767     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")

# Customer Purchase Updater
def updatescript(custid, pincrement, billiemaster):
    try:
        xon = sqlite3.connect('DBFA_CUSTCC.db')
        xbr7 = xon.cursor()
        # hidden prints here ig
        points = (billiemaster/100)*1
        updatexr = """UPDATE custcc SET purchasecount = purchasecount + 1 WHERE
custid = ?"""
        updatexxr = """UPDATE custcc SET ptotalx = ptotalx + ? WHERE custid = ?"""
        updatexxxr = """UPDATE custcc SET points = points + ? WHERE custid = ?"""
```

```
824         indicator = (custid, )
825         xrindicator = (pincrement, custid)
826         pindicator = (points, custid)
827         xbr7.execute(updatexr, indicator)
828         xbr7.execute(updatexxr, xrindicator)
829         xbr7.execute(updatexxxr, pindicator)
830         xon.commit()
831         xbr7.close()
832     except sqlite3.Error as error:
833         pass
834 
835 
836 def pointfetch(custid):
837     global lylpoints
838     lylpoints = 0
839     xon = sqlite3.connect('DBFA_CUSTCC.db')
840     xbr7 = xon.cursor()
841     findinx = "select points from custcc WHERE custid = ?"
842     findinxx = (custid, )
843     xbr7.execute(findinx, findinxx)
844     arterxout = xbr7.fetchall()
845     lylpoints = int((arterxout[0])[0])
846 
847 
848 def pointmassfetch():
849     xon = sqlite3.connect('DBFA_CUSTCC.db')
850     xbr7 = xon.cursor()
851     findinx = "select DISTINCT points from custcc"
852     xbr7.execute(findinx)
853     rows = xbr7.fetchall()
854     for row in rows:
855         print(row[0])
856 
857 def pointsuse(custid, deduct):
858     xon = sqlite3.connect('DBFA_CUSTCC.db')
859     xbr7 = xon.cursor()
860     updatexxxr = """UPDATE custcc SET points = points - ? WHERE custid = ?"""
861     pindicator = (deduct, custid)
862     xbr7.execute(updatexxxr, pindicator)
863     xon.commit()
864     xbr7.close()
865 
866 def emailfetch(custid):
867     global custmail
868     con = sqlite3.connect(r'DBFA.db')
869     conn = con.cursor()
870     findinx = "select DISTINCT email from cust WHERE custt = ?"
871     findinxx = (custid, )
872     conn.execute(findinx, findinxx)
873     rows = conn.fetchall()
874     custmail = (rows[0][0])
875 
876 global custcheckindic
877 
878 global custt
879 
880 def custcheck(custt):
881     global cccheck, lylpoints
882     if custt in ("", " ", 0, None , "0"):
883         lylpoints = 0
```

```
884
885     cccheck = 0
886     custcheckindic = 0
887     con = sqlite3.connect(r'DBFA.db')
888     conn = con.cursor()
889     conn.execute("SELECT custt FROM cust WHERE custt = ?", (custt,))
890     data = conn.fetchall()
891     if len(data)==0:
892         custcheckindic = 0
893         print("Customer", custt, "NOT found. ")
894         print("- No customer selected -")
895         custt = 0
896         print("Using unregistered customer account")
897         cccheck = 0
898     else:
899         ccustcheckindic = 1
900         cccheck = 0
901         pass
902
903
904 # Customer Validity Checker
905 def cust_listfetch(custid):
906     clfetch = sqlite3.connect(r'DBFA_CUSTCC.db')
907     clfetchx = clfetch.cursor()
908     clfetchx.execute("SELECT custid FROM custcc")
909     rows = clfetchx.fetchall()
910     custyes = 1
911     custno = 2
912     custcount = 0
913     for row in rows:
914         row = row[0]
915         if custid == row:
916             custcount += 1
917         else:
918             pass
919     if custcount == 1:
920         return custyes
921     else:
922         return custno
923
924 def saleslogger(custid, prodid, netpay):  #defining a function to input data into
    the SQL database's table
925     sales = sqlite3.connect(r'dbfasales.db')
926     salesx = sales.cursor()
927
928     netprof = 0
929     from datetime import date
930     datex = date.today()
931
932     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]
933     for i in prodid:
934         netprof += profer[int(i)]
935
936     salesx.execute("SELECT MAX(sno) FROM SALES")
937     sno = (int(salesx.fetchall()[0][0]) + 1)
938
939     prodidxs = ""
940     for i in prodid:
```

```
941         prodidxs += '%s'%i + ", "
942 
943     str = "insert into sales(sno, custid, prodid, net, prof, date) values(?, ?, ?,
?, ?, ?)"
944     io = (sno, custid, prodidxs, netpay, netprof, datex)
945     salesx.execute(str, io)
946     sales.commit()
947     print("Sales activity logged. ")
948 
949 
950 def salesdatefetch():  #defining a function to input data into the SQL database's
table
951     from datetime import date
952     import datetime
953     sales = sqlite3.connect(r'dbfasales.db')
954     salesx = sales.cursor()
955     salesx.execute("SELECT prof FROM sales WHERE date BETWEEN datetime('now', '-6
days') AND datetime('now', 'localtime')")
956     sumer = 0
957     for i in salesx.fetchall():
958         sumer += int((i[0]))
959     return sumer
960 
961 def salestodayfetch():  #defining a function to input data into the SQL database's
table
962     from datetime import date
963     import datetime
964     sales = sqlite3.connect(r'dbfasales.db')
965     salesx = sales.cursor()
966     salesx.execute("SELECT prof FROM sales WHERE date = ?", (date.today(), ))
967     sumerx = 0
968     for i in salesx.fetchall():
969         sumerx += int((i[0]))
970     return sumerx
971 
972 
973 os.system('cls')
974 
975 
976 
977 def floodscreen():
978     image = cv2.imread("imagepx.png")
979     cv2.imshow("Loading.... ", image)
980     cv2.waitKey(2000)
981     cv2.destroyAllWindows()
982 
983 
984 def floodpay():
985     import cv2
986     image = cv2.imread("qr-code.png")
987     cv2.imshow("Pay With UPI", image)
988     cv2.waitKey(150000)
989     cv2.destroyAllWindows()
990 
991 
992 # Main Menu
993 # New Main Menu
994 def mainmenu(): #defining a function for the main menu
995     from colorama import init, Fore, Back, Style #color-settings for the
partner/sponsor adverts
```

```
    init(convert = True)
    url = "https://raw.github.com/deltaonealpha/DBFA/master/updates.txt"
    r = requests.get(url)
    dbfaver = ((str(r.content)[6:-1]).replace("\\n", "")).replace(" ", "")
    xdbfaver = ((str(r.content)[2:-3]).replace("\\n", "")).replace(" ", "")
    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)
    try:
        if float(upread[4: ]) > float(dbfaver):
            pass
    except:
        print("Error with updater. Error code : *bckshln12* ╭ つ ◕_◕ ╮つ  つ ╮ ◕_◕ つ╭")
    else:
        if spass1 != spass2:
            print("A new DBFA update is available: DBFA", dbfaver)
        #elif spass1 == spass2:
            #print("DBFA is up-to-date")
    # Count pending deliveries
    delcount = 0
    netprof = sqlite3.connect('recmaster.db')
    netprofx = netprof.cursor()
    netprofx.execute("SELECT netprof FROM recmasterx")
    rows = netprofx.fetchall()
    netproffetchx = []
    for i in rows:
        netproffetchx.append(i[0])
    print(" ")
    filedel = open('./DBFAdeliveries.txt', 'r+')
    for line in filedel:
        delcount+=1
    filedel.close()
    pro7d = salesdatefetch()
    protd = salestodayfetch()
    time.sleep(1)
    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
                                                                                                             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
                                                            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 8.12 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("━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━")
```

```
1084        else:
1085            print("DONNAGER 8.01 RC-2 Test Beta")
1086            print(Fore.BLACK + Back.CYAN + "No deliveries pending! " + Back.BLACK +
    Fore.CYAN)
1087        print(logoxnew)
1088    else:
1089        if delcount != 0:
1090            print("------------------------------------------------------------------
    ------------------------------------------------")
1091            lener1 = "Profit (last week): " + '%s'%pro7d
1092            print(lener1 + (95-len(lener1))*" ", "   Profit (today): ", protd)
1093            #pro7d, (56-len(str(pro7d)))*" ", "DONNAGER 8.01 RC-2 Test Beta")
1094            print(Back.BLACK + Fore.MAGENTA+ "Pending deliveries: " + str(delcount)
    + " "  + "       DBFA User: " + os.getlogin() + "              "+ dt_string +
    Fore.CYAN)
1095            print("------------------------------------------------------------------
    ------------------------------------------------")
1096        else:
1097            print("DONNAGER 8.01 RC-2 Test Beta")
1098            print(Fore.BLACK + Back.CYAN + "No deliveries pending! " + Back.BLACK +
    Fore.CYAN)
1099        print(logoxold)
1100    #Settings Checker
1101    if settingscommonfetch(3) == 1:
1102        time.sleep(0.2)
1103        try:
1104            if spotify.current() not in ("", " ", [], (), None):
1105                print("Currently playing:", Fore.MAGENTA , spotify.current()[0],
    Fore.CYAN, "by ", Fore.MAGENTA, spotify.current()[1], Fore.CYAN)
1106            else:
1107                print(Fore.MAGENTA, "No music playing. Use Spotify to play your
    favourite music and control it via DBFA", Fore.CYAN)
1108        except Exception as e:
1109            print(Fore.MAGENTA, "No music playing. Play your favourite music and
    control it via DBFA", Fore.CYAN)
1110        if settingscommonfetch(7) == 1:
1111
 print("==========================================================================
==========", Fore.MAGENTA)
1112        else:
1113            print("------------------------------------------------------------------
    ------------------------------------------------", Fore.MAGENTA)
1114    else:
1115
 print("==========================================================================
==========")
1116        print("Re-enable DBFA Music Controls Service from the settings to be able to
    control your music ")
1117
 print("==========================================================================
==========")
1118
1119    #underline_byte = b'\xcc\xb2'
1120    #underline = str(underline_byte,'utf-8')
1121    #for x in ("What would you like to do?"):
1122    #    if x.isspace() == False:
1123    #        print(x+underline,end='')
1124    #    else:
1125    #        print(x,end='')
1126    print("What would you like to do?", Fore.WHITE)
```

```
1127     print()
1128 
1129 global netpay
1130 
1131 def xpayboxie():
1132     command = "cls"
1133     os.system(command)
1134     global xrt, payindic
1135     xrt = 0
1136     from colorama import init, Fore, Back, Style #color-settings for the
     partner/sponsor adverts
1137     print("Amount to pay: ", netpay)
1138     print("Payment methods available: ")
1139     print("1. Credit/ Debit Card")
1140     print("2. Digital Wallet")
1141     print("3. UPI")
1142     print("4. Cash")
1143     print("*exit* to cancel this billing cycle")
1144     time.sleep(0.5)
1145     paycheck = input("Pay with: ")
1146     print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
1147     if paycheck == "1":
1148         payindic = "Paid with credit/ debit Card"
1149     elif paycheck == "2":
1150         payindic = "Paid with a digital wallet"
1151     elif paycheck == "3":
1152         payindic = "Paid with UPI"
1153         floodpay()
1154     elif paycheck == "4":
1155         payindic = "Paid with cash"
1156     elif paycheck == "exit":
1157         print("Cancelling this billing cycle")
1158         xrt = 1
1159     else:
1160         xpayboxie()
1161 
1162 global netpay
1163 
1164 # Payments Handler
1165 def payboxie(custid, total):
1166     global custcheckindic
1167     if custt not in (0, "0", "", " ", None) and cccheck == 0:
1168         command = "cls"
1169         os.system(command)
1170         global payindic, netpay, redeemindic, lylpoints
1171         xrt = 0
1172         redeemindic = 0
1173         payindic = 0
1174         from colorama import init, Fore, Back, Style #color-settings for the
     partner/sponsor adverts
1175         print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
1176         init(convert = True)
1177         print("Amount to be paid: ₹","%.2f" % total)
1178         print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
1179         pointfetch(custid)
1180         if lylpoints != 0:
1181             print("You have loyalty points: ", lylpoints, "worth: ", lylpoints)
1182             time.sleep(0.5)
1183             pointscheck = input("Use points (y/n)? ")
1184             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'))
                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:
```

```
1235                        total = total - lylpoints
1236                        pointsuse(custid, lylpoints)
1237                        print("New total: ", total)
1238                        time.sleep(2)
1239                        redeemindic = 1
1240                        netpay = total
1241                        print("Points redeemed worth: ", lylpoints)
1242
1243                    else:
1244                        redeemindic = 1
1245                        netpay = total
1246                        pointsuse(custid, total)
1247                        time.sleep(0.3)
1248                        total = 0
1249                        print("Points redeemed worth: ", lylpoints)
1250                    time.sleep(0.3)
1251
1252                else:
1253                    print("Wrong OTP. (1) attempt(s) remaining")
1254                    time.sleep(0.2)
1255                    otpverif = input("Enter the OTP recieved on " + "%s"%custmail +
    ": ")
1256                    if otpverif == otp:
1257                        if total > lylpoints:
1258                            total = total - lylpoints
1259                            pointsuse(custid, lylpoints)
1260                            print("New total: ", total)
1261                            time.sleep(2)
1262                            redeemindic = 1
1263                            netpay = total
1264                            print("Points redeemed worth: ", lylpoints)
1265                        else:
1266                            netpay = 0
1267                            pointsuse(custid, total)
1268                            time.sleep(0.3)
1269                            total = 0
1270                            print("Points redeemed worth: ", lylpoints)
1271                    else:
1272                        print("Wrong OTP. (0) attempt(s) remaining")
1273                        time.sleep(0.2)
1274            elif pointscheck == "n":
1275                redeemindic = 0
1276                netpay = total
1277            else:
1278                pass
1279            time.sleep(1)
1280            os.system(command)
1281        else:
1282            netpay = total
1283
1284    else:
1285        redeemindic = 0
1286        netpay = total
1287        redeemindic = 0
1288        lylpoints = 0
1289        netpay = total
1290
1291
1292
1293
```

```
1294 def del2a():
1295     try:
1296         if os.path.exists(r'userblock.txt'):
1297             os.remove(r'userblock.txt')
1298         if os.path.exists(r'userblock.zconf'):
1299             os.remove(r'userblock.zconf')
1300     except PermissionError:
1301             pass
1302     print("Connecting to server..")  #SQL connection prompt
1303     time.sleep(0.4)  #for a seamless experience
1304     #conn.execute("select * from cust")
1305     #takes values from the SQL database
1306     conn = sqlite3.connect('DBFA.db')
1307     cursor = conn.cursor()
1308     cursor.execute("select * from cust")
1309     results = cursor.fetchall()
1310     idd = len(results)+1
1311     print("Registering customer with ID: ", idd)
1312     custname = input("Customer Name: ")
1313     email = input("Customer's E-mail ID: ")
1314     inserter(idd, custname, email) #argumental function to insert values into the
     SQL database
1315     nullvalue = 0
1316     custcc(idd, custname, nullvalue, nullvalue)
1317     print(" ")
1318 
1319     print("Customer ID", idd, "registered in directory.")
1320     print("----------------------------------------")
1321     print(" ")
1322     print(" ")
1323     #conn.close()
1324     time.sleep(1) #for a seamless experience
1325 
1326 
1327 def del2b():
1328     try:
1329         if os.path.exists(r'userblock.txt'):
1330             os.remove(r'userblock.txt')
1331         if os.path.exists(r'userblock.zconf'):
1332             os.remove(r'userblock.zconf')
1333     except PermissionError:
1334             pass
1335     print()
1336     print("Connecting to server..")  #SQL connection prompt
1337     time.sleep(0.7) #for a seamless experience
1338     print("Registered customers are: ")
1339     #Re-writing to refresh connection
1340     conn = sqlite3.connect('DBFA.db')
1341     cur = conn.cursor()
1342     cur.execute("SELECT * FROM cust")
1343     rows = cur.fetchall()
1344     col_labels = ("ID", "Customer NAME", "EMAIL")
1345     table(col_labels, rows)
1346     toaster.show_toast("DNSS QuickSync", "Database acessed", duration = 2)
1347     #takes values from the SQL database
1348     '''
1349     while row is not None:
1350         print(row)
1351         #row = conn.fetchone()
1352     '''
```

```
1353
1354        conn.close()
1355        conn.close()
1356        print()
1357        print()
1358        time.sleep(2) #delay for easy-table viewing
1359
1360
1361    def del2c():
1362        try:
1363            if os.path.exists(r'userblock.txt'):
1364                os.remove(r'userblock.txt')
1365            if os.path.exists(r'userblock.zconf'):
1366                os.remove(r'userblock.zconf')
1367        except PermissionError:
1368                pass
1369        xon = sqlite3.connect(r'DBFA_CUSTCC.db')
1370        xbr7 = xon.cursor()
1371        xbr7.execute("SELECT * FROM custcc")
1372        l = xbr7.fetchall()
1373        print("\nCustomer Purchase Records: ")
1374
1375        import pandas as pd
1376
1377        flat_list = []
1378        print("-------------------------------------------------------------------")
1379        for sublist in l:
1380            flat_list.append(sublist)
1381        mydf = pd.DataFrame(flat_list, columns=['Customer ID', 'Name', 'Purchases Made',
    'Total', 'Loyalty Points'])
1382        mydf.pivot(index='Customer ID', columns='Purchases Made',
    values='Total').fillna(value='-')
1383        print(mydf)
1384        print("-------------------------------------------------------------------")
1385        time.sleep(2)
1386
1387        '''
1388        for row in rows:
1389            print(row)
1390            print(" ")
1391        '''
1392        xbr7.close()
1393        toaster.show_toast("DFBA QuickSync", "Database acessed", duration = 0.5)
1394
1395
1396
1397    def del2d():
1398        try:
1399            con = sqlite3.connect(r'DBFA.db')
1400            conn = con.cursor()
1401
1402            conx = sqlite3.connect(r'DBFA_CUSTCC.db')
1403            connx = conx.cursor()
1404
1405            searchcon = str(input("Customer Name: "))
1406            if " " in searchcon:
1407                for i in searchcon:
1408                    sconsplit = searchcon.split(" ")
1409                    for j in sconsplit:
```

```
1410                    conn.execute("SELECT custt FROM cust WHERE custname LIKE ?",
(("%"+searchcon+"%"), ))
1411                    searchdata = conn.fetchall()
1412                else:
1413                    searchcon = searchcon.replace(" ", "")
1414                    conn.execute("SELECT custt FROM cust WHERE custname LIKE ?",
(("%"+searchcon+"%"), ))
1415                    searchdata = conn.fetchall()
1416                    if len(searchdata) != 0:
1417                        pass
1418                    else:
1419                        searchdata = "No such customer found."
1420        else:
1421            conn.execute("SELECT custt FROM cust WHERE custname LIKE ?",
(("%"+searchcon+"%"), ))
1422            searchdata = conn.fetchall()
1423
1424        if len(searchdata) != 0:
1425            if len(searchdata) > 1:
1426                for i in searchdata:
1427                    conn.execute("SELECT * FROM cust WHERE custt = ?", (i[0], ))
1428                    custdata = conn.fetchall()
1429                    #col_labels = ("ID", "Customer NAME", "EMAIL")
1430                    #table(col_labels, custdata)
1431
1432                    connx.execute("SELECT * FROM custcc WHERE custid = ?", (i[0], ))
1433                    custdatax = connx.fetchall()
1434                    ccrt = []
1435                    for jk in custdata:
1436                        for jkx in jk:
1437                            ccrt.append(str(jkx))
1438                    for jk in custdatax:
1439                        for jkx in jk:
1440                            ccrt.append(str(jkx))
1441
1442                    col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name',
'Purchases Made', 'Total', 'Loyalty Points')
1443                    print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f"))
1444
1445                    print(" ")
1446            else:
1447                conn.execute("SELECT * FROM cust WHERE custt = ?", (searchdata[0]
[0], ))
1448                custdata = conn.fetchall()
1449
1450                connx.execute("SELECT * FROM custcc WHERE custid = ?",
(searchdata[0][0], ))
1451                custdatax = connx.fetchall()
1452                ccrt = []
1453                for jk in custdata:
1454                    for jkx in jk:
1455                        ccrt.append(str(jkx))
1456                for jk in custdatax:
1457                    for jkx in jk:
1458                        ccrt.append(str(jkx))
1459
1460                col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name',
'Purchases Made', 'Total', 'Loyalty Points')
1461                print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f"))
1462        else:
```

```
1463            srtx = "%"
1464            for i in searchcon:
1465                srtx += '%s'%i+"%"
1466            conn.execute("SELECT custt FROM cust WHERE custname LIKE ?", ((srtx), ))
1467            searchdata = conn.fetchall()
1468            if len(searchdata) != 0:
1469                conn.execute("SELECT * FROM cust WHERE custt = ?", (i[0], ))
1470                custdata = conn.fetchall()
1471                #col_labels = ("ID", "Customer NAME", "EMAIL")
1472                #table(col_labels, custdata)
1473
1474                connx.execute("SELECT * FROM custcc WHERE custid = ?", (i[0], ))
1475                custdatax = connx.fetchall()
1476                ccrt = []
1477                for jk in custdata:
1478                    for jkx in jk:
1479                        ccrt.append(str(jkx))
1480                for jk in custdatax:
1481                    for jkx in jk:
1482                        ccrt.append(str(jkx))
1483
1484                col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name',
     'Purchases Made', 'Total', 'Loyalty Points')
1485                print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f"))
1486            else:
1487                print("Customer not found.")
1488    except:
1489        print("Error encountered.  ")
1490
1491
1492
1493
1494 def del2e():
1495     import sqlite3 as sql
1496     import os
1497     import csv
1498     from sqlite3 import Error
1499
1500     try:
1501
      csvex=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA.db')
1502         cursor = csvex.cursor()
1503         cursor.execute("select * from cust")
1504         print("Fetching data from database - I...")
1505         with open("DBFAcustrec.csv", "w") as csv_file:
1506             csv_writer = csv.writer(csv_file, delimiter="\t")
1507             csv_writer.writerow([i[0] for i in cursor.description])
1508             csv_writer.writerows(cursor)
1509
      csvexx=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_CUST
     CC.db')
1510             print("Fetching data from database - II...")
1511             cursorx = csvexx.cursor()
1512             cursorx.execute("select * from custcc")
1513             csv_writer = csv.writer(csv_file, delimiter="\t")
1514             csv_writer.writerow([i[0] for i in cursorx.description])
1515             csv_writer.writerows(cursorx)
1516         dirpath = os.getcwd() + "/DBFAcustrec.csv"
1517         print("Data exported Successfully into {}".format(dirpath))
1518         time.sleep(2)
```

```
1519
1520        #Settings Checker
1521        if settingscommonfetch(4) == 1:
1522            os.startfile(r"DBFAcustrec.csv")
1523        else:
1524            pass
1525
1526    except Error as e:
1527        print(e)
1528
1529    finally:
1530        csvex.close()
1531
1532
1533
1534 def del3a():
1535     decsfacter = str(input("Enter 'a' to VIEW STOCK, 'b' to ADD STOCK and 'c' to
      ENFORCE MASS STOCK: "))
1536     if decsfacter == "a":
1537         ssxsuperfetch()
1538     elif decsfacter == "b":
1539         objid = int(input("Enter the product ID to add stock for: "))
1540         stockincrement = int(input("Enter the amount of stock to be added: "))
1541         ssxupdatescript(stockincrement, objid)
1542         print("Stock updated by", stockincrement, "for product ID:", objid)
1543     elif decsfacter == "c":
1544         ggrtrr = int(input("Enter stock to universally enforce: "))
1545         massmaintainer(ggrtrr)
1546
1547
1548 def del3b():
1549     print("-------- DBFA Stock Master v1 ---------")
1550     print(" ")
1551     time.sleep(1)
1552     print("a: Order New Stock")
1553     print("b: Update Delivery Status")
1554     print("c: MASS - Fetch Current Status")
1555     print("d: INDVL - Fetch Current Status")
1556     print("e: View Vendor Details")
1557     print("f: Contact Vendor")
1558     print("g: Edit Vendor Contact")
1559     print("h: Modify Low-Stock Warning Bar")
1560     stkmaster = input("Select:: ")
1561     if stkmaster in ("a", "A"):
1562         idquery = int(input("Enter the product ID: "))
1563         qtyquery = int(input("Enter the amount to order: "))
1564         orderstock(idquery, qtyquery)
1565         print("\n-----------------------------------------------------")
1566     elif stkmaster in ("b", "B"):
1567         idquery = int(input("Product ID of the product recieved: "))
1568         qtyquery = int(input("Quantity recieved: "))
1569         delivered(qtyquery, idquery)
1570         ssxupdatescript(qtyquery, idquery)
1571         print("\nChanges made in all related databases. \n")
1572         print("\n-----------------------------------------------------")
1573     elif stkmaster in ("c", "C"):
1574         delstatmass()
1575         print("\n-----------------------------------------------------")
1576     elif stkmaster in ("d", "D"):
1577         idquery = int(input("Product ID to locate: "))
```

```
1578         delstatindvl(idquery)
1579         print("\n-------------------------------------------------------")
1580     elif stkmaster in ("e", "E"):
1581         idquery = int(input("Product ID to get vendor details for: "))
1582         vendorfetch(idquery)
1583         print("\n-------------------------------------------------------")
1584     elif stkmaster in ("f", "F"):
1585         idquery = int(input("Product ID to contact vendor for: "))
1586         vendorcontact(idquery)
1587         print("\n-------------------------------------------------------")
1588     elif stkmaster in ("g", "G"):
1589         print("-- Change Vendor Contact --")
1590         time.sleep(0.5)
1591         try:
1592             prodidvendc = int(input("Enter product ID to change vendor contact for:
"))
1593             if prodidvendc > 40:
1594                 print("Please enter a valid product ID")
1595                 prodidvendc = input("Enter product ID to change vendor contact for:
")
1596 
1597         except:
1598             print("Please enter a valid product ID")
1599             time.sleep(0.3)
1600             prodidvendc = input("Enter product ID to change vendor contact for: ")
1601             if prodidvendc > 40:
1602                 print("Please enter a valid product ID")
1603                 prodidvendc = input("Enter product ID to change vendor contact for:
")
1604         while(1):
1605             vendorchange = input("Enter the new E-Mail ID: ")
1606             if "@" in vendorchange:
1607                 if "." in vendorchange:
1608                     st = sqlite3.connect(r'DBFA_vend.db')
1609                     stx = st.cursor()
1610                     stx.execute("""UPDATE stock SET vendcont = ? WHERE prodid =
?""", (vendorchange, prodidvendc,))
1611                     st.commit()
1612                     print("Vendor contact changed for product ID: ", prodidvendc)
1613                     time.sleep(1.5)
1614                     break
1615                     mainmenu()
1616                 else:
1617                     print("Please enter a valid E-Mail format! ")
1618             else:
1619                 print("Please enter a valid E-Mail format! ")
1620 
1621     elif stkmaster in ("h", "H"):
1622         idquery = int(input("Product ID to alter bar for: "))
1623         lowbarmodif(idquery)
1624         print("\n-------------------------------------------------------")
1625     else:
1626         print("Please select a valid option! ")
1627         print("\n-------------------------------------------------------")
1628         time.sleep(1)
1629         mainmenu()
1630 
1631 
1632 
1633 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 - ")
    passw = getpass.getpass(prompt='Enter root password to view store activity registry: ', stream=None)
    if passw == "root":
            time.sleep(1) #for a seamless experience
            print("Hold on, moneybags.")
            with HiddenPrints():
                try:
                    sender = telegram_bot_sendtext(dt_string + "\n" + "Registry files and sales DB records accessed - DBFA SECURITY")
                    print(sender)
                except Exception:
                    pass
            time.sleep(0.2) #for a seamless experience
            print()
            sales = sqlite3.connect(r'dbfasales.db')
            salesx = sales.cursor()
            salesx.execute("SELECT * FROM sales")
            salesrows = salesx.fetchall()
            col_labels = ("SalesID", "CustomerID", "Product Codes Purchased", "Total", "Profit Earned", "Date of Purchase")
```

```
1691            table(col_labels, salesrows)
1692            toaster.show_toast("DNSS QuickSync", "Database acessed", duration = 2)
1693            time.sleep(1.4) #for a seamless experience
1694            os.startfile('registry.txt') #to open the external notepad application
1695    else:
1696        print("Ehh that'd be wrong, sneaky-hat. Try again: ")
1697        print(" ")
1698        print(" - Echo-supressed input - ")
1699        passw = getpass.getpass(prompt='Enter root password to view store activity
registry: ', stream=None)
1700        if passw == "root":
1701                time.sleep(1) #for a seamless experience
1702                print("Hold on, moneybags.")
1703                with HiddenPrints():
1704                    try:
1705                        sender = telegram_bot_sendtext(dt_string + "\n" + "Registry
files and sales DB records accessed accessed - DBFA SECURITY: ATTEMPT 02")
1706                        print(sender)
1707                    except Exception:
1708                        pass
1709                print("There ya go:: ")
1710                print()
1711                sales = sqlite3.connect(r'dbfasales.db')
1712                salesx = sales.cursor()
1713                salesx.execute("SELECT * FROM sales")
1714                salesrows = salesx.fetchall()
1715                col_labels = ("SalesID", "CustomerID", "Product Codes Purchased",
"Total", "Profit Earned", "Date of Purchase")
1716                table(col_labels, salesrows)
1717                toaster.show_toast("DNSS QuickSync", "Database acessed", duration =
2)
1718                time.sleep(0.6) #for a seamless experience
1719                # print(logger.read())
1720                # print()
1721                # print("Opening sales log externally now. ")
1722                time.sleep(1.4) #for a seamless experience
1723                os.startfile('registry.txt')
1724        else:
1725            with HiddenPrints():
1726                try:
1727                    sender = telegram_bot_sendtext(dt_string + "\n" + "[ACCESS
DENIED!!] - Registry file  - DBFA SECURITY [ACCESS DENIED!!]")
1728                    print(sender)
1729                except Exception:
1730                    pass
1731            print("Multiple Unsuccesfull Attempts Detected. Re-run the program to
login now. ")
1732            logger.write("(MULTIPLE ATTEMPTS!): Log file access attempt -
AUTHORIZATION FAILED!!! \n")
1733            time.sleep(1.4) #for a seamless experience
1734            print()
1735            print()
1736
1737
1738
1739
1740 def del3f():
1741     import sqlite3 as sql
1742     import os
1743     import csv
```

```
    from sqlite3 import Error

    try:

 csvex=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\recmaster.db')
        print("Exporting sales data to CSV....")
        cursor = csvex.cursor()
        cursor.execute("select * from recmasterx")
        with open("DBFAstorerec.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)

        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:
        pass




print("--------------------------------------------------------------------------------")

# Store listing::
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}
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)"}
```

```
1780 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)"]
1781 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]
1782
1783 # dataxr is currently redundant
1784 dataxr = []
1785 for i in datax:
1786     i = "₹" + '%d' % i
1787     dataxr.append(i)
1788 tablx = zip(namiex, dataxr)
1789 titlex = ["Product:", "Pricing:"]
1790
1791 print('''
1792 DBFA Billing Framework 7 [Bellaire] (stable)
1793 <GNU Public License> Copyright (C) 2020 Pranav Balaji and Sushant Gupta
1794 View the license file in the installation dir for more info.
1795 \n\n
1796 Fetching Windows login details..
1797 Fetching Windows login details....
1798 \n''')
1799
1800 logoprintxrt()
1801 time.sleep(0.3)
1802 command = "cls"
1803 os.system(command)
1804
1805
1806 #vs2
1807 from datetime import datetime  #for reporting the billing time and date
1808 now = datetime.now()
1809 dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object containing current
     date and time
1810 logger = open(r"registry.txt", "a+")  #Opening / creating (if it doesn't exist
     already) the .txt record file
1811 logger.write("----------------------------------------- \n")
1812 logger.write('ed')
1813 logger.write("\n")
1814 logger.write("Automated Store Registry:\n")
1815
1816 #Settings Checker
1817 if settingscommonfetch(1) == 1:
1818     floodscreen() #comment to disable boot-flash screen
1819 else:
1820     pass
1821
```

```
1822
1823 import requests, time, json, urllib, os, math, random, sqlite3
1824 from tqdm import tqdm
1825 import logging, os, time, requests, socket
1826 import telegram_send
1827 from pynput.keyboard import Key, Controller
1828 # Telegram BOT API 2 (FULL)
1829 from telegram import InlineKeyboardButton, InlineKeyboardMarkup
1830 from telegram.ext import Updater, CommandHandler, CallbackQueryHandler
1831
1832 logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
1833                     level=logging.INFO)
1834 xlogger = logging.getLogger(__name__)
1835
1836 def telegram_bot_sendtext(bot_message):
1837     bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
1838     bot_chatID = '680917769'
1839     send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id='
+ bot_chatID + '&parse_mode=Markdown&text=' + bot_message
1840     response = requests.get(send_text)
1841     return response.json()
1842
1843 def start(update, context):
1844     keyboard = [[InlineKeyboardButton("✅ Validate", callback_data='1'),
1845                  InlineKeyboardButton("❌ Deny", callback_data='2')]]
1846
1847     reply_markup = InlineKeyboardMarkup(keyboard)
1848
1849     update.message.reply_text("delta 2FA Handler Service\n\n\nValidate login?",
reply_markup=reply_markup)
1850
1851
1852 def button(update, context):
1853     query = update.callback_query
1854     # CallbackQueries need to be answered, even if no notification to the user is
needed
1855     # Some clients may have trouble otherwise. See
https://core.telegram.org/bots/api#callbackquery
1856     inlet = ("{}".format(query.data))
1857     if inlet in (1, "1"):
1858         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")
1859         with
open(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\deltatgstickerlogonew.we
bp", "rb") as f:
1860             telegram_send.send(stickers=[f])
1861             keyboard = Controller()
1862             print("\n\nValidation recieved! DBFA Client will start in a moment\n\n")
1863             print("telegram_extended.updtr_pushreq(deltaonealpha, set.webhook: (on,
getUpdated.redir(servers.gokku.com/deltaonealpha/arterxt1, callback=False)))")
1864             print("\n\nif there's no print below for around 5 secs from now, press
ctrl+c to continue\n\n")
1865             keyboard.press(Key.ctrl)
1866             keyboard.press('c')
1867             keyboard.release('c')
1868             keyboard.release(Key.ctrl)
1869
1870     if inlet in (2, "2"):
```

```
1871        query.edit_message_text(text="❌🔐 Denied delta 2FA request.\n\ndelta
    Security Service")
1872        with
    open(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\deltatgstickerlogonew.we
    bp", "rb") as f:
1873            telegram_send.send(stickers=[f])
1874        keyboard = Controller()
1875        print("\n\nThe login request for this session has been DENIED.\n\n")
1876        time.sleep(1)
1877        print("telegram_extended.updtr_pushreq(deltaonealpha, set.webhook: (on,
    getUpdated.redir(servers.gokku.com/deltaonealpha/arterxt1, callback=False)))")
1878        print("\n\nif there's no print below for around 5 secs from now, press
    ctrl+c to continue\n\n")
1879        keyboard.press(Key.ctrl)
1880        keyboard.press('c')
1881        keyboard.release('c')
1882        keyboard.release(Key.ctrl)
1883        time.sleep(1)
1884        time.sleep(1)
1885        print("DBFA Client will now exit! ")
1886        time.sleep(5)
1887        os._exit(0)
1888    if inlet in (3, "3"):
1889        query.edit_message_text(text="Use */help*")
1890
1891
1892
1893 def help_command(update, context):
1894    update.message.reply_text("Use /auth when prompted. This bot will only respond
    when a delta service raises a request. ")
1895
1896
1897 def main():
1898    # Create the Updater and pass it your bot's token.
1899    # Make sure to set use_context=True to use the new context based callbacks
1900    # Post version 12 this will no longer be necessary
1901    updater = Updater("1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis",
    use_context=True)
1902
1903    updater.dispatcher.add_handler(CommandHandler('auth', start))
1904    updater.dispatcher.add_handler(CallbackQueryHandler(button))
1905    updater.dispatcher.add_handler(CommandHandler('help', help_command))
1906
1907    # Start the Bot
1908    updater.start_polling()
1909
1910    # Run the bot until the user presses Ctrl-C or the process receives SIGINT,
1911    # SIGTERM or SIGABRT
1912    updater.idle()
1913
1914
1915 def settingscommonfetch(SettingsType):
1916    import sqlite3
1917    settings = sqlite3.connect(r'dbfasettings.db')
1918    settingsx = settings.cursor()
1919    settingsx.execute(("SELECT Value from settings WHERE SettingsType = ?"),
    (SettingsType,))
1920    settingsfetch = (settingsx.fetchall()[0][0])
1921    return settingsfetch
1922
```

```
1923 if settingscommonfetch(6) == 1:
1924     def telegram_bot_sendtext(bot_message):
1925         bot_token = '1215404401:AAEvVBwzogEhOvBaW5iSpHRbz3Tnc7fCZis'
1926         bot_chatID = '680917769'
1927         send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?
     chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
1928         response = requests.get(send_text)
1929         return response.json()
1930 
1931     os.system('cls')
1932     print("delta2 Authenication Service")
1933     print("=======================================")
1934     print("DBFA 2FA Service  DBFA 2")
1935     print("==========")
1936     time.sleep(1)
1937     print("")
1938 
1939 
1940     print("You have DBFA 2FA activated. Please validate the login from your Telegram
     account. ")
1941     for i in tqdm (range (10), desc="Connecting.."):
1942         time.sleep(0.00001)
1943     if __name__ == '__main__':
1944         hostname = socket.gethostname()
1945         ip_address = socket.gethostbyname(hostname)
1946         import platform
1947         from datetime import datetime  #for reporting the billing time and date
1948         now = datetime.now()
1949         dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object containing
     current date and time
1950         with
     open(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\deltatgstickerlogonew.we
     bp", "rb") as f:
1951             telegram_send.send(stickers=[f])
1952         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: ")
1953         main()
1954     os.system('cls')
1955 
1956 else:
1957     print("DBFA 2FA is disabled. We recommend you to turn it on from the settings
     for a more secure experience with DBFA client.")
1958 
1959 
1960 print("--------------------------------\n\n( つ ◕_◕ )つ delta Update Utility\n\n---
     ------------------------------")
1961 time.sleep(0.5)
1962 print("DBFAIntellisense")
1963 print("Fetching update details from server : : : : ")
1964 url = "https://raw.github.com/deltaonealpha/DBFA/master/updates.txt"
1965 r = requests.get(url)
1966 dbfaver = ((str(r.content)[6:-1]).replace("\\n", "")).replace(" ", "")
1967 xdbfaver = ((str(r.content)[2:-3]).replace("\\n", "")).replace(" ", "")
1968 with open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\updates.txt', 'r+') as
     upread:
1969     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!")
    print("It has been detected that you have bypassed the login process.")
    time.sleep(1)
    print("Please obtain a genuine version of this program and/ or provide proper authenication.")
    time.sleep(1)
    print("The program shall now exit. Error code:013")
    time.sleep(2)
    print("----------------------------------------")
    time.sleep(5)
    exit()


# Main Program Starts Here
while(1): #while (always) true
    mainmenu() #mainmenu
    writer = ""
    telethon = ""
    time.sleep(0.037)  #for a seamless experience
    decfac = input("Select option: ")

    #DBFA Music Controls v1.2
    #All possible case-combinations; found using recursion
    if decfac in ('prev', 'preV', 'prEv', 'prEV', 'pRev', 'pReV', 'pREv', 'pREV', 'Prev', 'PreV', 'PrEv', 'PrEV', 'PRev', 'PReV', 'PREv', 'PREV'):
        try:
            spotilib.previous()
        except Exception as e:
            pass
    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'):
        try:
```

```
2024            spotilib.pause()
2025        except Exception as e:
2026            pass
2027
2028    elif decfac in ('next', 'nexT', 'neXt', 'neXT', 'nExt', 'nExT', 'nEXt', 'nEXT',
'Next', 'NexT', 'NeXt', 'NeXT', 'NExt', 'NExT', 'NEXt', 'NEXT'):
2029        try:
2030            spotilib.next()
2031        except Exception as e:
2032            pass
2033
2034    #DBFA Mark Attendance
2035    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'):
2036        import sqlite3, time, os, requests
2037        from datetime import datetime  #for reporting the billing time and date
2038        empmas = sqlite3.connect(r'dbfaempmaster.db')
2039        empmascur = empmas.cursor()
2040
2041        empmascur.execute("SELECT DISTINCT * FROM emp")
2042        dump = empmascur.fetchall()
2043        Oiddump = []
2044        for row in dump:
2045            Oiddump.append(row[0])
2046        print("OiDs registered: ", Oiddump)
2047        try:
2048            Oid = int(input("DBFA MARK ATTENDANCE- Enter your OiD: "))
2049            trypass = 1
2050        except:
2051            print("OiDs are integer-only. Please retry using valid credentials! \n")
2052            trypass = 0
2053
2054        if trypass == 1:
2055            if Oid in Oiddump:
2056                print("OiD found ")
2057                time.sleep(0.5)
2058                now = datetime.now()
2059                dt_string = now.strftime("%Y/%m/%d")  #datetime object containing
current date and time
2060                tm_string = now.strftime("%H:%M:%S")  #datetime object containing
current date and time
2061                empmascur.execute("SELECT count(*) FROM attendance WHERE Date = ?
AND OiD = ?", (dt_string, Oid,))
2062                data = empmascur.fetchone()[0]
2063                if data==0:
2064                    #print('No record on %s'%dt_string+' for Employee %s'%Oid)
2065                    #print("insert into attendance(Date, OiD, Time, YN, IO)
values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
2066                    empmascur.execute("insert into attendance(Date, OiD, Time, YN,
IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
2067                    #empmascur.execute("UPDATE attendance SET Oid = 'Y' WHERE DATE =
?", (dt_string))
2068                    empmas.commit()
2069                    print("\n----------------DBFA-----------------")
2070                    print("C1 ENTRY: Marked Attendance! OiD: ", Oid)
2071                    print("-------------------------------------\n")
2072                elif data==1:
2073                    #print('Component %s found in %s row(s)'%(dt_string, data))
2074                    empmascur.execute("insert into attendance(Date, OiD, Time, YN,
IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'O'))
```

```
                    empmas.commit()
                    print("\n----------------DBFA-----------------")
                    print("C2 DAY END: Marked Attendance! OiD: ", Oid)
                    print("-------------------------------------\n")
                elif data > 1:
                    print("You can only mark in/out attendance once a day! \n")

            else:
                print("OiD not found! \n")

    #Billing Mode
    elif decfac == "1":
        print()
        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("--- BIlling ---")
        print()
        custt = input("Customer ID (optional): ")
        if custt in ("0", 0, "", " "):
            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"
```

```
2130        global billiemaster
2131        billiemaster = 0 #variable for totalling the price
2132        time.sleep(0.0247) #for a seamless experience
2133        afac = 0
2134        while(1):
2135            item = input("Enter product code: ")
2136            if item == "0":
2137                break
2138            elif item in data:
2139                ssxstockmaster(item)
2140                if ssxvarscheck == 1:
2141                    billiemaster+=data[item]
2142                    print("Purchased: ", namie[item], " for: ", data[item])
2143                    repupdate(item)
2144                    lenxr = len(namie[item])
2145                    costlenxr = len(str(data[item]))
2146                    cj = 10 - costlenxr
2147                    pi = 60 - lenxr
2148                    idlerxx = namie[item] + " "*pi + "₹"+'%d'%data[item] + " "*cj +
"1 qty. ~"
2149                    purcheck += idlerxx
2150                    print("---")
2151                    priceprod = "₹" + '%d' % data[item]
2152                    logger.write("Appending product to order: \n")  #writing to file
2153                    profer.append(item)
2154                    logger.write(namie[item])
2155                    ssxstockmaintainer(item)
2156                    logger.write(" \n")
2157                    writer = writer + "\n Purchased: " + "\n" + namie[item] + "\n" +
priceprod + "\n"
2158                    afac+=1
2159                else:
2160                    print("Product currently out-of-stock. The inconvenience is
regretted..\n")
2161                    print("---")
2162                    continue
2163            else:
2164                print("Product not found. Please retry ")
2165                print("---")
2166
2167        #tax = int(input("Enter the net tax %: "))  #comment and uncomment tkinter
lines to use GUI-based input
2168        time.sleep(0.15)  #for a seamless experience
2169        try:
2170            cponid = str(input("Enter voucher code (if any): "))
2171        except (EOFError, ValueError):
2172            pass
2173
2174        if cponid != "":
2175            isol = sqlite3.connect(r'cponmgmtsys.db')
2176            isolx = isol.cursor()
2177            isolx.execute(("SELECT DISTINCT cponid from cponmaster WHERE cponid =
?"), (cponid, ))
2178            if isolx.fetchall() in ("", " ", [], (), None):
2179                print("Invalid voucher identifier! Not applying any!")
2180                discount = int(input("Enter discount % (if any): "))
2181            else:
2182                cpon_limfetch(cponid)
2183                print("")
2184                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))
                logger.write("\n")
                #regin.write(str(total))
                #regin.write("\n")
                updatescript(custt, total, billiemaster) #adds billed amount to the
customer's record
                abcd1+=1
                afac+=1
                now = datetime.now()
                dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object
containing current date and time
                daterey = (dt_string.replace("/","")).replace(":", "")
                namer = 'DBFAinvid'+'%s'%inval+"-"+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
                destination =
r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\Generated_invoices'
                dest = shutil.move(source, destination)
                time.sleep(1.5) #for a seamless experience


                if custt not in ("", " ", 0, "0") and cccheck == 0:
                    #Settings Checker
                    if settingscommonfetch(2) == 1:
                        emailfetch(custt)
                        print("Please wait..")
                        fromaddr = "billing.dbfa@gmail.com"
                        toaddr = custmail
                        msg = MIMEMultipart()
                        msg['From'] = fromaddr
                        msg['To'] = toaddr
                        msg['Subject'] = "Your DBFA Purchase Invoice"
                        body = """Thanks for making your purchase at DBFA!\n\nPlease
find your invoice attached herewith.\n\nRegards\nThe DBFA Team"""
                        msg.attach(MIMEText(body, 'plain'))
```

```
2347                        filename = namer
2348                        attachment =
open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\\Generated_invoices\\'+'
%s'%namer, "rb")
2349                        attac= MIMEBase('application', 'octet-stream')
2350                        attac.set_payload((attachment).read())
2351                        encoders.encode_base64(attac)
2352                        attac.add_header('Content-Disposition', "attachment;
filename= %s" % filename)
2353                        msg.attach(attac)
2354                        email = smtplib.SMTP('smtp.gmail.com', 587)
2355                        email.starttls()
2356                        email.login(fromaddr, "dbfaidlepass")
2357                        message = msg.as_string()
2358                        email.sendmail(fromaddr, toaddr, message)
2359                        print("Invoice mailed. ")
2360                        print("------------------------------------------------------
--------------------\n\n")
2361                    #Settings Checker
2362                    elif settingscommonfetch(1) == 2:
2363                        pass
2364                #regin.close()
2365                with HiddenPrints():
2366                    try:
2367                        sender = telegram_bot_sendtext(telethon)
2368                        print(sender)
2369                    except Exception:
2370                        pass
2371
2372                print()
2373
2374        else:
2375            print("\n\n-----------------------------------------------------------
------------")
2376            print("No purchase made.")
2377            print("-----------------------------------------------------------
--------")
2378            print("Amount           : ₹",billiemaster)
2379            print("-----------------------------------------------------------
--------")
2380            print("Amount to be paid: ₹","%.2f" % total)
2381            print("-----------------------------------------------------------
--------")
2382            toaster.show_toast("DBFA: No purchase made  ",str(total), duration = 1)
2383            logger.write("Billing cancelled: Reason: Net amount null\n") #writing to
file
2384            #regin.write("NET TOTAL: \n") #writing to file
2385            telethon = telethon + "Billing cancelled. Net amount null." + "\n"
2386            writer = writer + "BILLING CALCELLED. No purchase made." + "\n"
2387            logger.write(str(total))
2388            logger.write("\n")
2389            abcd1+=1
2390            afac+=1
2391            now = datetime.now()
2392
2393            #regin.close()
2394            with HiddenPrints():
2395                try:
2396                    sender = telegram_bot_sendtext(telethon)
2397                    print(sender)
```

```
2398                 except Exception:
2399                     pass
2400 
2401 
2402     #Manage Customers
2403     elif decfac == "2":
2404         print("Select: ")
2405         print("    a: Register a Customer ")
2406         print("    b: Customer Registry ")
2407         print("    c: Customer Purchase Records ")
2408         print("    d: Find a Customer ")
2409         print("    e: Export Records as CSV \n")
2410         selected = input("What would you like to do? ")
2411         print("\n")
2412         if selected in ("a", "A"):
2413             del2a()
2414             logger.write("---------------------------------------- \n")
2415             logger.write("  \n")
2416             logger.write("Date and time: ") #including the date and time of billing
    (as taken from the system)
2417             logger.write(dt_string)
2418             logger.write(" \n")
2419             logger.write("New customer registered: ")
2420             x = " custname: " + custname + " custemail: " + email + "\n"
2421             logger.write(x)
2422             logger.write("---------------------------------------- \n")
2423 
2424         if selected in ("b", "B"):
2425             del2b()
2426             logger.write("---------------------------------------- \n")
2427             logger.write("  \n")
2428             logger.write("Date and time: ") #including the date and time of billing
    (as taken from the system)
2429             logger.write(dt_string)
2430             logger.write(" \n")
2431             logger.write("Customer registry accessed! \n")
2432             logger.write("---------------------------------------- \n")
2433 
2434         if selected in ("c", "C"):
2435             del2c()
2436             logger.write("---------------------------------------- \n")
2437             logger.write("\nDBFA Customer Purchase Records accessed! \n")
2438 
2439         elif selected in ("d", "D"):
2440             del2d()
2441             logger.write("---------------------------------------- \n")
2442             logger.write("\nCustomer search used. \n")
2443 
2444         elif selected in ("e", "E"):
2445             del2e()
2446             logger.write("\n\n---------------------------------------- \n")
2447             logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2448             logger.write("Customer data exported to CSV! ")
2449             with HiddenPrints():
2450                 try:
2451                     sender = telegram_bot_sendtext(dt_string + "\n" + "Sales data
    exported to CSV- REDFLAG Urgent Security Notice!")
2452                     print(sender)
2453                 except Exception:
2454                     pass
```

```
2455            logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2456            logger.write("-------------------------------------- \n\n\n")
2457
2458        elif selected not in ("a", "b", "c", "d", "e", "A", "B", "C", "D", "E"):
2459            print("Please chose a valid option!")
2460            time.sleep(1)
2461
2462    #Store Options:
2463    elif decfac == "3":
2464        print("Select: ")
2465        print("    a: Manage Stock ")
2466        print("    b: DBFA Stock Master ")
2467        print("    c: Manage Vouchers ")
2468        print("    d: Product Listing ")
2469        print("    e: Sales Log ")
2470        print("    f: Export Sales Data as CSV \n")
2471        storeselected = input("What would you like to do? ")
2472        print("\n")
2473
2474        if storeselected in ("a", "A"):
2475            del3a()
2476
2477        elif storeselected in ("b", "B"):
2478            del3b()
2479
2480        elif storeselected in ("c", "C"):
2481            del3c()
2482
2483        elif storeselected in ("d", "D"):
2484            del3d()
2485
2486        elif storeselected in ("e", "E"):
2487            del3e()
2488            logger.write("\n-------------------------------------- \n")
2489            logger.write("Sales log accessed! ")
2490
2491        elif storeselected in ("f", "F"):
2492            del3f()
2493            logger.write("\n\n-------------------------------------- \n")
2494            logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2495            with HiddenPrints():
2496                try:
2497                    sender = telegram_bot_sendtext(dt_string + "\n" + "Customer data
exported to CSV- REDFLAG Urgent Security Notice!")
2498                    print(sender)
2499                except Exception:
2500                    pass
2501            logger.write("Sales data exported to CSV! ")
2502            logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
2503            logger.write("-------------------------------------- \n\n\n")
2504
2505        elif storeselected not in ("a", "b", "c", "d", "e", "f", "A", "B", "C", "D",
"E", "F"):
2506            print("Please select a valid option! ")
2507            time.sleep(1)
2508            mainmenu()
2509
2510
2511
2512    #Reports
```

```
2513    elif decfac == "4":
2514        command = "cls"
2515        os.system(command)
2516        print("-- Generating store report --")
2517        repstockfetch()
2518        print("Please wait...")
2519        repdatafetch()
2520        findMaximum = "select max(prodsales) from recmasterx"
2521        recx.execute(findMaximum)
2522        # Print the maximum score
2523        netr = recx.fetchone()[0]
2524        findin = "select prodid, prodname, prodprofit, prodsales, netprof from
recmasterx WHERE prodsales = ?"
2525        arterx = (str(int(netr)),)
2526        recx.execute(findin, arterx)
2527        arterxout = recx.fetchall()
2528
2529        findMaximumProf = "select max(netprof) from recmasterx"
2530        recx.execute(findMaximumProf)
2531        xnetr = recx.fetchone()[0]
2532        findin = "select prodid, prodname, prodprofit, prodsales, netprof from
recmasterx WHERE netprof = ?"
2533        xarterx = str(int(xnetr))
2534        xxarterx = (xarterx,)
2535        recx.execute(findin, xxarterx)
2536        xarterxout = recx.fetchall()
2537
2538        isol = sqlite3.connect(r'DBFA_vend.db')
2539        isolx = isol.cursor()
2540        isolx = isol.cursor()
2541        isolx.execute(("SELECT prodid, prodname, ordqty, delstat, vendor from stock
WHERE delstat = ?"), ("TBD", ))
2542        rowsrec = isolx.fetchall()
2543        col_labels = [("P. ID", "P. Name", "Qty. to be delivered", "Status",
"Vendor")]
2544        rowsxtb = col_labels + rowsrec
2545
2546        def add_page_number(canvas, doc):
2547            canvas.saveState()
2548            canvas.setFont('Times-Roman', 10)
2549            page_number_text = "%d" % (doc.page)
2550            canvas.drawCentredString(
2551                0.75 * inch,
2552                0.75 * inch,
2553                page_number_text
2554            )
2555            canvas.restoreState()
2556
2557        import sqlite3 as sql
2558
 csvexx=sql.connect(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_CUST
CC.db')
2559        print("Fetching data from database - II...")
2560        cursorx = csvexx.cursor()
2561        axct = cursorx.execute("select * from custcc")
2562        axctx = []
2563        for i in axct:
2564            axctx.append(i)
2565        axctx = [("C. ID","C. Name","Purchases Made","Total Amount","Loyalty
Points")] + axctx
```

```
2566        doc = SimpleDocTemplate("dbfastorerep.pdf", pagesize=A4,
2567                                            rightMargin=2*cm,leftMargin=1.5*cm,
2568                                            topMargin=1*cm,bottomMargin=2*cm)
2569        # container for the 'Flowable' objects
2570        elements = []
2571        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 />")
2572        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 />")
2573        t6dot = ("Report generated on: " + dt_string)
2574        t3dot = ("<br /><br /><b>Most sold listing: </b><br />")
2575        t4dot = ("<br /><br /><b>Total profit per listing: </b><br /><br />")
2576        t5dot = ("<br /><br /><b>Most profit making listing: </b><br /><br />")
2577        t8dot = ("<br /><br /><b>Customer purchases: </b><br /><br />")
2578        t10dot = ("<br /><br /><br /><br /><br /><br /><b>DBFA Stock Orders Report:
    </b><br />Product stock yet to be recieved: <br /><br />")
2579        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 />")
2580        colas = (30, 300, 60, 50, 50)
2581        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 )
2582        text1=Paragraph(t1dot)
2583        text2=Paragraph(t2dot)
2584        text3=Paragraph(t3dot)
2585        text4=Paragraph(t4dot)
2586        text5=Paragraph(t5dot)
2587        text6=Paragraph(t6dot)
2588        text8=Paragraph(t8dot)
2589        text10=Paragraph(t10dot)
2590        text11=Paragraph(t11dot)
2591        x=Table(rows, colas, rowheights)
2592        t=Table(arterxout)
2593        t2=Table(xarterxout)
2594        t4=Table(axctx)
2595        t5=Table(rowsxtb)
2596        if tabarter == ['--']:
2597            t7dot = ("<br /><br /><b>All listings currently in stock!</b><br /><br
    />")
2598        else:
2599            t7dot = ("<br /><br /><b>Products running low on stock: </b><br /><br
    />")
2600
2601
2602        tblStyle = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
2603                               ('VALIGN',(0,0),(-1,-1),'TOP'),
2604                               ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
2605                               ('BOX',(0,0),(-1,-1),1,colors.black),
2606                               ('BOX',(0,0),(0,-1),1,colors.black)])
2607        tblStyle.add('BACKGROUND',(0,0),(1,0),colors.lightblue)
2608        tblStyle.add('BACKGROUND',(0,1),(-1,-1),colors.lightblue)
2609        GRID_STYLE = TableStyle(
2610            [('GRID', (0,0), (-1,-1), 0.25, colors.black),
2611            ('ALIGN', (1,1), (-1,-1), 'LEFT')]
2612            )
2613        text7=Paragraph(t7dot)
2614        t3=Table(tabarter)
```

```
2615        t.setStyle(GRID_STYLE)
2616        t2.setStyle(GRID_STYLE)
2617        t3.setStyle(GRID_STYLE)
2618        t4.setStyle(GRID_STYLE)
2619        t5.setStyle(GRID_STYLE)
2620        x.setStyle(GRID_STYLE)
2621        # ------------------------------------------------------------------------
-----
2622        import sqlite3, time
2623        import matplotlib.pyplot as plt
2624        salesr = sqlite3.connect(r'dbfasales.db')
2625        salesx = salesr.cursor()
2626        datefetch = []
2627
2628        salesx.execute("SELECT DISTINCT date FROM sales")
2629        for i in salesx.fetchall():
2630            datefetch.append((i[0]))
2631
2632        netray = []
2633        for i in datefetch:
2634            salesx.execute(("SELECT sum(prof) FROM sales WHERE date = ?"), (i, ))
2635            netray.append(salesx.fetchall()[0][0])
2636
2637        # Plotting
2638        plt.plot(datefetch, netray, color='purple', linestyle='dashed', linewidth =
3,
2639                 marker='o', markerfacecolor='magenta', markersize=12)
2640        # naming the x axis
2641        plt.xlabel('Date')
2642        # naming the y axis
2643        plt.ylabel('Profit')
2644        # Graph Title
2645        plt.title('DBFA Profit Report')
2646        time.sleep(1)
2647        # Finally, display
2648        plt.savefig('DBFAplot.png', dpi=300, bbox_inches='tight')
2649        # ------------------------------------------------------------------------
-----
2650
2651        delI = Image('DBFAplot.png')
2652        delI.drawHeight =  4.2*inch
2653        delI.drawWidth = 5.5*inch
2654        elements.append(text1)
2655        elements.append(text2)
2656        elements.append(text6)
2657        elements.append(text3)
2658        elements.append(t)
2659        elements.append(text5)
2660        elements.append(t2)
2661        elements.append(text7)
2662        elements.append(t3)
2663        elements.append(text4)
2664        elements.append(x)
2665        elements.append(text8)
2666        elements.append(t4)
2667        elements.append(text10)
2668        elements.append(t5)
2669        elements.append(text11)
2670        elements.append(delI)
2671        # write the document to disk
```

```
2672        doc.build(elements,
2673            onFirstPage=add_page_number,
2674            onLaterPages=add_page_number,)
2675        print("Report Created. ")
2676        for i in tqdm (range (100), desc="Publishing Report: "):
2677            time.sleep(0.00001)
2678        print("Opening store report now")
2679        os.startfile('dbfastorerep.pdf')
2680        time.sleep(2)
2681
2682
2683    #DBFA Backup&Switch
2684    elif decfac == "7":
2685        os.startfile(r'delauth.py')
2686        time.sleep(0.3)
2687        os._exit(0)
2688
2689
2690    #License
2691    elif decfac == "9":
2692        print("Fetching latest licensing information.......")
2693        print(" ")
2694        print(" ")
2695        logoprintxrt()
2696        time.sleep(1.5)
2697        print(" ")
2698        print(" ")
2699        print("DBFA by Pranav Balaji, 2020")
2700        print(" ")
2701        print("_______ Licensing _______")
2702        print("           GNU PUBLIC LICENSE - TERMS AND CONDITIONS")
2703        print("    <deltaBillingFramework>  Copyright (C) 2020 Pranav Balaji and
    Sushant Gupta")
2704        print("    This program comes with ABSOLUTELY NO WARRANTY; for details type
    *show w*.")
2705        print("    This is free software, and you are welcome to redistribute it")
2706        print("    under certain conditions; type *show c* for details. ")
2707        toaster.show_toast("DFBA Framework Runtime Broker", "©2020: DBFA by Pranav
    Balaji and Sushant Gupta", duration = 1.5)
2708        print(" ")
2709        print(" ")
2710        print("Visit: www.github.com/deltaonealpha/deltaBillingFramework for
    complete licensing terms. ")
2711        print(" ")
2712        print(" ")
2713        time.sleep(2)
2714        webbrowser.open('https://telegra.ph/DBFA-Licensing-Information-08-16')
2715        print("-------------------------------------------------")
2716
2717
2718    #Stock Ordering Option
2719    elif decfac == "5":
2720        print("------------------ DBFA DELIVERY MANAGER ------------------")
2721        time.sleep(2)
2722        print("For issuing new delivery orders, use the invoicing option")
2723        print("-----------------------------------------------------------")
2724        print("a: View existing deliveries")
2725        print("b: Show delivery count")
2726        print("c: Confirm a delivery")
2727        delfacx = input("Select: ")
```

```
        if delfacx in ("a", "A"):
            # Fetch Data
            print("Currently pending deliveries::  \n")
            filedel = open('./DBFAdeliveries.txt', 'r')
            for line in filedel:
                print(line)
            print("\n\n")
            time.sleep(2)
        if delfacx in ("b", "B"):
            # Pending Delivery Count
            delcount = 0
            filedel = open('./DBFAdeliveries.txt', 'r+')
            for line in filedel:
                delcount+=1
            filedel.close()
            if delcount != 0:
                print("Number of pending deliveries: ", delcount)
            else:
                print("No deliveries pending! ")
            time.sleep(2)
        if delfacx in ("c", "C"):
            # Show data and Remove delivery record
            print("Current delivery data::  \n")
            filedel = open('./DBFAdeliveries.txt', 'r')
            for line in filedel:
                print(line)
            print("\n\n")
            cleanstr = ""
            deledid = input("Enter the delivery ID to remove (EXAMPLE: *del2*): ")
            delcount = 0
            filedel = open('./DBFAdeliveries.txt', 'r')
            for line in filedel:
                if deledid in line:
                    line = ""
                else:
                    delcount+=1
                cleanstr+=line
            filedel.close()
            filedel = open('./DBFAdeliveries.txt', 'w+')
            filedel.write(cleanstr)
            filedel.close()
            filedel = open('./DBFAdeliveries.txt', 'r')
            print("Delivery", deledid, "completed! ")
            time.sleep(2)

        elif delfacx not in ("a", "b", "c", "A", "B", "C"):
            print("Invalid option selected! \n\n")
            time.sleep(2)
            mainmenu()

    #DevChangelog Option
    elif decfac == "10":
        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":
```

```
2788        def transitionprogress():
2789            from colorama import init, Fore, Back, Style
2790            os.system("cls")
2791            time.sleep(1)
2792            print(Fore.WHITE+'|'+Fore.RED+'███ OFF |')
2793            time.sleep(0.3)
2794            print(Fore.WHITE+'|'+Fore.RED+'███     '+Fore.GREEN+'█|')
2795            time.sleep(0.3)
2796            print(Fore.WHITE+'|'+Fore.RED+'██     '+Fore.GREEN+'██|')
2797            time.sleep(0.3)
2798            print(Fore.WHITE+'|'+Fore.RED+'█     '+Fore.GREEN+'███|')
2799            time.sleep(0.3)
2800            print(Fore.WHITE+'|'+Fore.GREEN+ ' ON  ███|'+Fore.WHITE)
2801            time.sleep(1.24)
2802            os.system("cls")
2803
2804
2805        def transitionprogressneg():
2806            from colorama import init, Fore, Back, Style
2807            os.system("cls")
2808            time.sleep(1)
2809            print(Fore.WHITE+'|'+Fore.GREEN+ ' ON  ███|')
2810            time.sleep(0.3)
2811            print(Fore.WHITE+'|'+Fore.RED+'█     '+Fore.GREEN+'███|')
2812            time.sleep(0.3)
2813            print(Fore.WHITE+'|'+Fore.RED+'██     '+Fore.GREEN+'██|')
2814            time.sleep(0.3)
2815            print(Fore.WHITE+'|'+Fore.RED+'███     '+Fore.GREEN+'█|')
2816            time.sleep(0.3)
2817            print(Fore.WHITE+'|'+Fore.RED+'███ OFF |'+Fore.WHITE)
2818            time.sleep(1.24)
2819            os.system("cls")
2820
2821        os.system("cls")
2822
2823        def settingsmenu():
2824            from colorama import init, Fore, Back, Style
2825            while(1):
2826                import time
2827                time.sleep(0.2)
2828
 print("████████████████████████████████████████████████████████████████████████████████
")
2829                print(" -------------------------  DBFA Settings  ----------------
---------    ")
2830
2831                if (settingscommonfetch(1)) == 1:
2832                    print(" 1:    Display boot image
:", '| ON '+Fore.GREEN+'███'+Fore.WHITE+'|      ')
2833                else:
2834                    print(" 1:    Display boot image
:", ('|'+Fore.RED+'███'+Fore.WHITE+' OFF|      '))
2835                if (settingscommonfetch(2)) == 1:
2836                    print(" 2:    Email invoice to registered customers
:", '| ON '+Fore.GREEN+'███'+Fore.WHITE+'|      ')
2837                else:
2838                    print(" 2:    Email invoice to registered customers
:", ('|'+Fore.RED+'███'+Fore.WHITE+' OFF|      '))
2839                if (settingscommonfetch(3)) == 1:
```

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

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

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

 #print("███████████████████████████████████████████████████████████████████████████
█")
                settfac = input("What would you like to do? ")
                if settfac == "1":
                    print('''DBFA displays an image for 2 seconds when it is started.
                    This image changes with each major iteration of DBFA.
```

```
                    Displaying this image let's us prepare files in the background
so that DBFA runs smoothly once its started.
                    Disabling this option may lead to errors. Continue? ''')
                    print(" ")
                    print("Display DBFA boot image? ")
                    print("y:    ",  '| ON '+Fore.GREEN+'    '+Fore.WHITE+'|')
                    settfac1x = input(("n:     "+ '|'+Fore.RED+'    '+Fore.WHITE+'
OFF|: '))
                    if settfac1x == "y":
                        settingsmodifier(1, 1)
                        transitionprogress()
                        print("DBFA will now display its boot image when it prepares
the backend on boot. ")
                        print("")
                        time.sleep(1)
                        settingsmenu()
                    elif settfac1x == "n":
                        settingsmodifier(1, 0)
                        transitionprogressneg()
                        print("DBFA won't display its boot image when it prepares
the backend on boot from now. ")
                        print("")
                        time.sleep(1)
                        settingsmenu()
                    else:
                        print("That's an invalid input... ")
                        print("")
                        time.sleep(1)
                        settingsmenu()

                elif settfac == "2":
                    print('''DBFA creates an invoice on each billing cycle
                    If the customer account in-use is registered with DBFA, the
invoice is E-Mailed to the same.
                    Disabling this option will stop DBFA from E-Mailing customers
with their invoice from now.''')
                    print(" ")
                    print("E-Mail registered customers their invoice? ")
                    print("y:    ",  '| ON '+Fore.GREEN+'    '+Fore.WHITE+'|')
                    settfac1x = input(("n:     "+ '|'+Fore.RED+'    '+Fore.WHITE+'
OFF|: '))
                    if settfac1x == "y":
                        settingsmodifier(2, 1)
                        transitionprogress()
                        print("DBFA will continue E-Mailing customers with their
invoice. ")
                        print("")
                        time.sleep(1)
                        settingsmenu()
                    elif settfac1x == "n":
                        settingsmodifier(2, 0)
                        transitionprogressneg()
                        print("DBFA will stop E-Mailing customers their invoice from
now on. ")
                        print("")
                        time.sleep(1)
                        settingsmenu()
                    else:
                        print("That's an invalid input... ")
                        print("")
```

```
2930                        time.sleep(1)
2931                        settingsmenu()
2932
2933                elif settfac == "3":
2934                    print('''In our mission of making DBFA the ultimate space to
    control your entire store and its functioning,
2935                    we keep adding tiny tid-bits to make that process even easier.
2936                    DBFA Music Controls is one such feature introduced in DBFA 8
    RC3x (IB3).
2937                    When you disable this functionality:
2938                            - The currently-playing track will no longer be
    displayed.
2939                            - DBFA Music Controls, including but not limited to
    pause/play, prev and next will be restricted. ''')
2940                    print(" ")
2941                    print("Enable DBFA Music Controls? ")
2942                    print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
2943                    settfac1x = input(("n:     "+ '|'+Fore.RED+'[redacted]'+Fore.WHITE+'
    OFF|: '))
2944                    if settfac1x == "y":
2945                        settingsmodifier(3, 1)
2946                        transitionprogress()
2947                        print("DBFA Music Controls Service will be started with the
    next menu-cycle. ")
2948                        print("")
2949                        time.sleep(1)
2950                        settingsmenu()
2951                    elif settfac1x == "n":
2952                        settingsmodifier(3, 0)
2953                        transitionprogressneg()
2954                        print("DBFA Music Controls Service will be restricted from
    the next menu-cycle.")
2955                        print("")
2956                        time.sleep(1)
2957                        settingsmenu()
2958                    else:
2959                        print("That's an invalid input... ")
2960                        print("")
2961                        time.sleep(1)
2962                        settingsmenu()
2963
2964
2965                elif settfac == "4":
2966                    print("CSV files once generated are auto-opened in your default
    worksheet app")
2967                    print("Example: Microsoft Excel, LibreOffice Calc, Google Docs,
    et cetera.")
2968                    print(" ")
2969                    print("Open file after export? ")
2970                    print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
2971                    settfac1x = input(("n:     "+ '|'+Fore.RED+'[redacted]'+Fore.WHITE+'
    OFF|: '))
2972                    if settfac1x == "y":
2973                        settingsmodifier(4, 1)
2974                        transitionprogress()
2975                        print("DBFA will now open CSV files when exported on
    request. ")
2976                        print("")
2977                        time.sleep(1)
2978                        settingsmenu()
```

```
                    elif settfac1x == "n":
                        settingsmodifier(4, 0)
                        transitionprogressneg()
                        print("DBFA will not open CSV files when exported from now
on. ")
                        print("")
                        time.sleep(1)
                        settingsmenu()
                    else:
                        print("That's an invalid input... ")
                        print("")
                        time.sleep(1)
                        settingsmenu()

                elif settfac == "5":
                    print('''In our process of phasing-out .txt based storage in
favour of sqlite storage,
                    we at DBFA are trying to make our files even tougher to access
than ever before without valid credentials.

                    DBFA is currently experimenting with sqlcipher encryption for
it's sqlite databases.
                    Please note that this functionality is a part of DBFA internal
test builds for now,
                    and is not ready for public rollout.

                    This process might impact DBFA's data integrity. We recommend
you to run *DBFA Backup&Switch* from option *5*
                    before you attempt to encrypt/ decrypt DBFA databases by running
this command.''')
                    print(" ")
                    print("Enable DBFA database encryption? ")
                    print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
                    settfac1x = input(("n:     "+ '|'+Fore.RED+'[redacted]'+Fore.WHITE+'
OFF|: '))
                    if settfac1x == "y":
                        settingsmodifier(5, 1)
                        transitionprogress()
                        print('''DBFA will attempt to encrypt it's databases when
restarted.
                        This process may fail, as this *internal test build* of DBFA
currently has encryption as a beta feature.''')
                        print("")
                        time.sleep(1)
                        settingsmenu()
                    elif settfac1x == "n":
                        settingsmodifier(5, 0)
                        transitionprogressneg()
                        print('''DBFA will attempt to de-crypt it's databases on the
next restart.
                        This process may fail, as this *internal test build* of DBFA
currently has encryption as a beta feature.

                        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... ")
```

```
3026                        print("")
3027                        time.sleep(1)
3028                        settingsmenu()
3029
3030                elif settfac == "6":
3031                    print("----DBFA 2FA MANAGER----")
3032                    print("Two-factor authenication is a widely-used method helpful
    in securing accounts when their passwords get compromised.")
3033                    print("With DBFA, you can choose between Telegram and Google
    Authenicator as a medium to recieve these 2FA requests. ")
3034                    print(" ")
3035                    print("DBFA randomly generates these OTPs/ requests and sends
    them via a secure and encrypted connection.")
3036                    time.sleep(2)
3037                    print(" ")
3038                    print(" ")
3039                    print("Please do note that enabling/ disabling 2FA will reboot
    DBFA Store Manager!!")
3040                    print(" ")
3041                    print(" ")
3042                    print("Available authenication methods: ")
3043                    print("1: Telegram Authenication")
3044                    import os
3045                    print("2: Google Authenicator (alpha; experimental)")
3046                    print("3/ skip: Exit to settings menu")
3047                    authfac = input("What would you like?: ")
3048                    if authfac == "1":
3049                        print("Connecting to the Telegram Web API..")
3050                        print("To turn on/ off DBFA 2FA, you need to authenicate
    with 2FA first.")
3051                        time.sleep(0.5)
3052                        os.startfile('modif2fa.py')
3053                        time.sleep(1)
3054                        os._exit(0)
3055
3056                    if authfac == "2":
3057                        print("Loading Django framework..")
3058                        print("This option is currently under development!")
3059                        print(" ")
3060                        time.sleep(2)
3061                    else:
3062                        print("Please choose a valid option! ")
3063                        print(" ")
3064
3065                elif settfac == "7":
3066                    print("This option let's you switch between the older DBFA menu-
    style")
3067                    print("and the newer one as introduced with DBFA 8.12")
3068                    print("\nFor the best visual experience with DBFA, we recommend
    you to use the newer design.\n\n")
3069                    print("DBFA Menu-Style: ")
3070                    print("1: Use new style (recommended)")
3071                    print("2: Use old style")
3072                    msdsfac = input("Please make a choice: ")
3073                    if msdsfac == "1":
3074                        settingsmodifier(7, 1)
3075                        transitionprogress()
3076                        print("New menu style applied! ")
3077                    if msdsfac == "2":
3078                        print("Old menu style")
```

```
                        settingsmodifier(7, 0)
                        transitionprogressneg()
                        print("Old menu style applied..! ")
                    else:
                        print("Please choose a valid option! ")
                        print(" ")



                elif settfac == "8":
                    import shutil
                    import os
                    desktop = os.path.join(os.path.join(os.environ['USERPROFILE']),
'OneDrive\Desktop')
                    # Prints: C:\Users\sdkca\Desktop
                    print("Shortcut will be created at: " + desktop)
                    try:
                        original =
r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\Assets\run_DBFA.lnk'
                        shutil.copy(original, desktop)
                        print("Executed. ")
                    except:
                        print("DBFA Permission Error: Can't get perms to execute in
directory! ")

                elif settfac == "9":
                    print('''This option PERMANENTLY CLEARS ALL DBFA CUSTOMER
RECORDS.
                    This includes their registration data, purchase records, and
loyalty points.

                    This execution can NOT BE REVERSED.
                    DATA INTEGRITY MAY BE LOST during this process.
                    Proceed with caution! ''')
                    print(" ")
                    print("ERASE DBFA customer records PERMANENTLY? ")
                    print("y:    ",  '| ON '+Fore.GREEN+'   '+Fore.WHITE+'|')
                    settfac1x = input(("n:     "+ '|'+Fore.RED+'   '+Fore.WHITE+'
OFF|: '))

                    if settfac1x == "y":
                        print("DBFA will now reboot itself to finish applying
changes.")
                        time.sleep(0.5)
                        transitionprogress()
                        # window.close()
                        os.startfile(r'securepack.py')
                        time.sleep(1)
                        os._exit(0)


                        settingsmenu()
                    elif settfac1x == "n":
                        print("Denied. ")
                        settingsmenu()
                    else:
                        print("That's an invalid input... ")
                        print("")
                        time.sleep(1)
                        settingsmenu()
```

```
3132
3133                elif settfac == "10":
3134                    print('''This option PERMANENTLY CLEARS ALL DBFA VOUCHERS/
    COUPONS
3135                    All current vouchers/ coupons WILL BE LOST.
3136                    Vouchers already issued will become redundant unless manually
    re-added again.
3137                    Validity and usage limits will be lost for all voucher/ coupon
    instanced recorded by DBFA.
3138
3139                    However, DBFA's logged voucher/ coupon usage will continue to
    exist in memory and will not be erased.
3140
3141                    This execution can NOT BE REVERSED.
3142                    DATA INTEGRITY MAY BE LOST during this process.
3143                    Proceed with caution! ''')
3144                    print(" ")
3145                    print("ERASE DBFA voucher/ coupon records PERMANENTLY? ")
3146                    print("y:    ",  '| ON '+Fore.GREEN+'███'+Fore.WHITE+'|')
3147                    settfac1x = input(("n:     "+ '|'+Fore.RED+'███'+Fore.WHITE+'
    OFF|: '))
3148                    if settfac1x == "y":
3149                        print("DBFA will now reboot itself to finish applying
    changes.")
3150                        time.sleep(0.5)
3151                        transitionprogress()
3152                        # window.close()
3153                        os.startfile(r'securepackxvc.py')
3154                        time.sleep(1)
3155                        os._exit(0)
3156
3157
3158                        settingsmenu()
3159                    elif settfac1x == "n":
3160                        print("Denied.")
3161
3162
3163                        settingsmenu()
3164                    else:
3165                        print("That's an invalid input... ")
3166                        print("")
3167                        time.sleep(1)
3168                        settingsmenu()
3169
3170
3171                elif settfac == "11":
3172                    print("DBFA Updater is currently in the making. ")
3173                    print("You'll be notified immediately this feature is enabled.
    ")
3174                    print("Support for this will come with a future update. ")
3175                    settingsmenu()
3176
3177
3178                elif settfac == "11":
3179                    break
3180                    break
3181                    break
3182
3183                else:
3184                    print("That's an invalid input... ")
```

```
3185                    print("")
3186                    time.sleep(1)
3187                    break
3188                    break
3189                    break
3190
3191        settingsmenu()
3192
3193    #Profit Graph Plotter
3194    elif decfac == "8":
3195        time.sleep(0.5)
3196        print("---- DBFA Sales Analyzer Engine v1 ----")
3197        time.sleep(0.5)
3198        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")
3199        time.sleep(1)
3200        print("Please wait while we analyze store sales..\n\n")
3201        time.sleep(2)
3202        print("A new window will be shortly opened. ")
3203        print("You're requested to close the same when you want to return to DBFA's
    main menu.\n")
3204        time.sleep(1.7)
3205        os.startfile(r'plotter.pyw')
3206
3207
3208    #DBFA Updater
3209    elif decfac == "11":
3210        import requests, os, time, shutil, oschmod
3211        os.system('cls')
3212        print("---------------------------------\n\n( ͡° ͜ʖ ͡°)つ delta Update
    Utility\n\n---------------------------------")
3213        time.sleep(1)
3214        url = "https://raw.github.com/deltaonealpha/DBFA/master/updates.txt"
3215        r = requests.get(url)
3216        dbfaver = ((str(r.content)[6:-1]).replace("\\n", "")).replace(" ", "")
3217        xdbfaver = ((str(r.content)[2:-3]).replace("\\n", "")).replace(" ", "")
3218
3219        with open(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\updates.txt',
    'r+') as upread:
3220            upread = (str(upread.read())).strip()
3221
3222        print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
3223        time.sleep(1)
3224        spass1 = []
3225        spass2 = []
3226        for i in dbfaver:
3227            spass1.append(i)
3228        for j in upread[4: ]:
3229            spass2.append(j)
3230        if float(upread[4: ]) > float(dbfaver):
3231            pass
3232
3233        else:
3234            if xdbfaver == upread:
3235                print("This installation of DBFA is up-to date! ")
3236
3237            elif spass1 != spass2:
3238                time.sleep(1)
3239                print("A new DBFA update is available: DBFA", dbfaver)
3240                time.sleep(3)
```

```
3241                updateconfo =  input("Update DBFA now? (y/n): ")
3242                if updateconfo == "y":
3243                    try:
3244
 oschmod.set_mode(r"C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_Update
Handler", "777")
3245                    except:
3246                        pass
3247
 shutil.rmtree(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHan
dler', ignore_errors=True)
3248                    try:
3249
 os.rmdir(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHandler'
)
3250                    except:
3251                        pass
3252                    if
os.path.isdir(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHand
ler') == True:
3253
 shutil.rmtree(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHan
dler', ignore_errors=True)
3254                        print("Cleaning-up previous update package... ")
3255
 shutil.rmtree(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHan
dler', ignore_errors=True)
3256                        try:
3257
 os.rmdir(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\DBFA_UpdateHandler'
)
3258                        except:
3259                            pass
3260                    else:
3261                        pass
3262                    try:
3263                        os.system('git clone
https://github.com/deltaonealpha/DBFA_UpdateHandler')
3264                    except:
3265                        print("The directory 'DBFA_UpdateHandler' already exists.")
3266                        print("Please delete it from DBFA's installation location
and re-run the updater.")
3267                    #os.system('git log')
3268                    print("Commit: ")
3269                    os.system(r'git rev-parse HEAD')
3270
3271                    print("The new package has been downloaded to your DBFA
installation > master > DBFA_UpdateHandler")
3272                    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.")
3273
3274                if updateconfo == "n":
3275                    time.sleep(1)
3276                    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!")
3277
3278    #Exit System
3279    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("ノつ ◕_◕ ヽつ ")
        time.sleep(0.5)
        print("         ノつ ◕_◕ ヽつ ")
        time.sleep(0.5)
        print("                  ノつ ◕_◕ ヽつ ")
        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 ノ•'◡'•ヽつ

                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)

                10. Pay salary

                11. <<< Back to DBFA menu

            ''')

        while (1):
```

```
3337            empmenu()
3338            empfac = input("What would you like to do? ")
3339
3340            if empfac == "1":
3341                print("DBFA will now be opening a seperate window due to GUI-
restrictions.")
3342                time.sleep(2)
3343                with HiddenPrints():
3344                    try:
3345                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed:
Hire Employee - deltaDBFA")
3346                        print(sender)
3347                    except Exception:
3348                        pass
3349                os.startfile(r'dbfaempman.py')
3350
3351
3352            if empfac == "2":
3353                print("\n2. View employee records\n--------------------------------
--------------")
3354                empmas = sqlite3.connect(r'dbfaempmaster.db')
3355                empmascur = empmas.cursor()
3356                print("Employee records as maintained by DBFA: ")
3357                with HiddenPrints():
3358                    try:
3359                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed:
Employee Records - deltaDBFA")
3360                        print(sender)
3361                    except Exception:
3362                        pass
3363                empmascur.execute("SELECT * FROM emp")
3364                emprows = empmascur.fetchall()
3365                time.sleep(1)
3366                for emprow in emprows:
3367                    print(emprow, "\n")
3368                print("\n\n-----------------------------------------------")
3369                time.sleep(3)
3370
3371            if empfac == "3":
3372                print("3. Change employee details")
3373                empmas = sqlite3.connect(r'dbfaempmaster.db')
3374                empmascur = empmas.cursor()
3375                empmascur.execute("SELECT * FROM emp")
3376                emprows = empmascur.fetchall()
3377                time.sleep(1)
3378                for emprow in emprows:
3379                    print(emprow, "\n")
3380                print("\n")
3381                emppay = str(input("Enter the Oid (Employee ID) to change details
for: "))
3382                empmascur.execute("SELECT Name FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
3383                firename = str(empmascur.fetchall()[0][0])
3384                confofac = input("CONFIRM: Change details for "+firename+"? (y/n):
")
3385                if confofac == "y":
3386                    print('''Options:
3387                            1. Change Name
3388                            2. Change Email
3389                            3. Change Mobile Contact
```

```
3390
3391                            4. Change Residential Address
3392
3393                            5. Change UPI Payments ID
3394
3395                            6. Change Department
3396                            7. Change Designation (POST)
3397                            8. Change Salary
3398                            ''')
3399                    subfac = input("What would you like to do? ")
3400                    if subfac == "1":
3401                        print("1. Change Name")
3402                        empmas = sqlite3.connect(r'dbfaempmaster.db')
3403                        empmascur = empmas.cursor()
3404                        newmodif = input("Enter the changed name: ")
3405                        time.sleep(1)
3406                        empmascur.execute("UPDATE emp SET Name = ? WHERE Oid = ?",
    (newmodif, emppay))
3407                        empmas.commit()
3408                        print("Name changed for OID", emppay, " from ", firename,
    "to ", newmodif)
3409                        time.sleep(1)
3410                        with HiddenPrints():
3411                            try:
3412                                sender = telegram_bot_sendtext(dt_string + "\n" +
    "Accessed: Employee Details Changed - deltaDBFA")
3413                                print(sender)
3414                            except Exception:
3415                                pass
3416                        print("--------")
3417                        time.sleep(1)
3418
3419                    if subfac == "2":
3420                        print("2. Change Email")
3421                        empmas = sqlite3.connect(r'dbfaempmaster.db')
3422                        empmascur = empmas.cursor()
3423                        empmascur.execute("SELECT Email FROM emp WHERE Oid LIKE ?",
    ("%"+emppay+"%", ))
3424                        firename = str(empmascur.fetchall()[0][0])
3425                        newmodif = input("Enter the changed e-mail: ")
3426                        time.sleep(1)
3427                        empmascur.execute("UPDATE emp SET Email = ? WHERE Oid = ?",
    (newmodif, emppay))
3428                        empmas.commit()
3429                        with HiddenPrints():
3430                            try:
3431                                sender = telegram_bot_sendtext(dt_string + "\n" +
    "Accessed: Employee Details Changed - deltaDBFA")
3432                                print(sender)
3433                            except Exception:
3434                                pass
3435                        print("Email changed for OID", emppay, " from ", firename,
    "to ", newmodif)
3436                        time.sleep(1)
3437                        print("--------")
3438                        time.sleep(1)
3439
3440                    if subfac == "3":
3441                        print("3. Change Mobile Contact")
3442                        empmas = sqlite3.connect(r'dbfaempmaster.db')
```

```
3443                         empmascur = empmas.cursor()
3444                         empmascur.execute("SELECT Mobile FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
3445                         firename = str(empmascur.fetchall()[0][0])
3446                         newmodif = input("Enter the changed mobile contact: ")
3447                         time.sleep(1)
3448                         empmascur.execute("UPDATE emp SET Mobile = ? WHERE Oid = ?",
(newmodif, emppay))
3449                         empmas.commit()
3450                         with HiddenPrints():
3451                             try:
3452                                 sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Details Changed - deltaDBFA")
3453                                 print(sender)
3454                             except Exception:
3455                                 pass
3456                         print("Mobile contact changed for OID", emppay, " from ",
firename, "to ", newmodif)
3457                         time.sleep(1)
3458                         print("--------")
3459                         time.sleep(1)
3460
3461                     if subfac == "4":
3462                         print("4. Change Residential Address")
3463                         empmas = sqlite3.connect(r'dbfaempmaster.db')
3464                         empmascur = empmas.cursor()
3465                         empmascur.execute("SELECT Address FROM emp WHERE Oid LIKE
?", ("%"+emppay+"%", ))
3466                         firename = str(empmascur.fetchall()[0][0])
3467                         newmodif = input("Enter the changed address (in one-line):
")
3468                         time.sleep(1)
3469                         empmascur.execute("UPDATE emp SET Address = ? WHERE Oid =
?", (newmodif, emppay))
3470                         empmas.commit()
3471                         with HiddenPrints():
3472                             try:
3473                                 sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Details Changed - deltaDBFA")
3474                                 print(sender)
3475                             except Exception:
3476                                 pass
3477                         print("Address changed for OID", emppay, " from ", firename,
"to ", newmodif)
3478                         time.sleep(1)
3479                         print("--------")
3480                         time.sleep(1)
3481
3482                     if subfac == "5":
3483                         print("5. Change UPI Payments ID")
3484                         empmas = sqlite3.connect(r'dbfaempmaster.db')
3485                         empmascur = empmas.cursor()
3486                         empmascur.execute("SELECT UPI FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
3487                         firename = str(empmascur.fetchall()[0][0])
3488                         newmodif = input("Enter the changed UPI ID: ")
3489                         time.sleep(1)
3490                         empmascur.execute("UPDATE emp SET UPI = ? WHERE Oid = ?",
(newmodif, emppay))
3491                         empmas.commit()
```

```
3492                        with HiddenPrints():
3493                            try:
3494                                sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Details Changed - deltaDBFA")
3495                                print(sender)
3496                            except Exception:
3497                                pass
3498                        print("UPI ID changed for OID", emppay, " from ", firename,
"to ", newmodif)
3499                        time.sleep(1)
3500                        print("--------")
3501                        time.sleep(1)
3502
3503                    if subfac == "6":
3504                        print("6. Change Department")
3505                        time.sleep(1)
3506                        empmas = sqlite3.connect(r'dbfaempmaster.db')
3507                        empmascur = empmas.cursor()
3508                        empmascur.execute("SELECT Dept FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
3509                        firename = str(empmascur.fetchall()[0][0])
3510                        print("-----------------------------------\nCurrent Dept: ",
firename)
3511                        print('''Options:
3512                                   IT,
3513                                   Administration,
3514                                   Sales,
3515                                   Care-taking,
3516                                   Logistics ''')
3517                        print("-----------------------------------\n\n")
3518                        time.sleep(2)
3519                        newmodif = input("Enter the changed department: ")
3520                        time.sleep(1)
3521                        empmascur.execute("UPDATE emp SET Dept = ? WHERE Oid = ?",
(newmodif, emppay))
3522                        empmas.commit()
3523                        with HiddenPrints():
3524                            try:
3525                                sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Details Changed - deltaDBFA")
3526                                print(sender)
3527                            except Exception:
3528                                pass
3529                        print("Department changed for OID", emppay, " from ",
firename, "to ", newmodif)
3530                        time.sleep(1)
3531                        print("--------")
3532                        time.sleep(1)
3533
3534                    if subfac == "7":
3535                        print("7. Change Designation (POST)")
3536                        empmas = sqlite3.connect(r'dbfaempmaster.db')
3537                        empmascur = empmas.cursor()
3538                        empmascur.execute("SELECT Post FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
3539                        firename = str(empmascur.fetchall()[0][0])
3540                        newmodif = input("Enter the new designation: ")
3541                        time.sleep(1)
3542                        empmascur.execute("UPDATE emp SET Post = ? WHERE Oid = ?",
(newmodif, emppay))
```

```
3543                        empmas.commit()
3544                        with HiddenPrints():
3545                            try:
3546                                sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Details Changed - deltaDBFA")
3547                                print(sender)
3548                            except Exception:
3549                                pass
3550                        print("Designation (Post) changed for OID", emppay, " from
", firename, "to ", newmodif)
3551                        time.sleep(1)
3552                        print("--------")
3553                        time.sleep(1)
3554
3555                    if subfac == "8":
3556                        print("8. Change Salary")
3557                        empmas = sqlite3.connect(r'dbfaempmaster.db')
3558                        empmascur = empmas.cursor()
3559                        empmascur.execute("SELECT Salary FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
3560                        firename = str(empmascur.fetchall()[0][0])
3561                        newmodif = input("Enter the new salary (net; not
incremental): ")
3562                        time.sleep(1)
3563                        empmascur.execute("UPDATE emp SET Salary = ? WHERE Oid = ?",
(newmodif, emppay))
3564                        empmas.commit()
3565                        with HiddenPrints():
3566                            try:
3567                                sender = telegram_bot_sendtext(dt_string + "\n" +
"Accessed: Employee Details Changed - deltaDBFA")
3568                                print(sender)
3569                            except Exception:
3570                                pass
3571                        print("Salary changed for OID", emppay, " from ", firename,
"to ", newmodif)
3572                        time.sleep(1)
3573                        print("--------")
3574                        time.sleep(1)
3575                    else:
3576                        print("Invalid option! \n\n")
3577                        time.sleep(1)
3578                else:
3579                    print("Invalid option! \n\n")
3580                    time.sleep(1)
3581
3582
3583            if empfac == "4":
3584                empmas = sqlite3.connect(r'dbfaempmaster.db')
3585                empmascur = empmas.cursor()
3586                print("4. Fire an employee ノ●'◡'●ヽつ \n")
3587                empmascur.execute("SELECT * FROM emp")
3588                emprows = empmascur.fetchall()
3589                time.sleep(1)
3590                for emprow in emprows:
3591                    print(emprow, "\n")
3592                print("\n")
3593                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")
                    trypass = 0

                if trypass == 1:
                    if Oid in Oiddump:
                        print("OiD found ")
                        time.sleep(0.5)
                        now = datetime.now()
```

```
3649                        dt_string = now.strftime("%Y/%m/%d")  #datetime object
containing current date and time
3650                        tm_string = now.strftime("%H:%M:%S")  #datetime object
containing current date and time
3651                        empmascur.execute("SELECT count(*) FROM attendance WHERE
Date = ? AND OiD = ?", (dt_string, Oid,))
3652                        data = empmascur.fetchone()[0]
3653                        if data==0:
3654                            #print('No record on %s'%dt_string+' for Employee
%s'%Oid)
3655                            #print("insert into attendance(Date, OiD, Time, YN, IO)
values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
3656                            empmascur.execute("insert into attendance(Date, OiD,
Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
3657                            #empmascur.execute("UPDATE attendance SET Oid = 'Y'
WHERE DATE = ?", (dt_string))
3658                            empmas.commit()
3659                            print("\n----------------DBFA-----------------")
3660                            print("C1 ENTRY: Marked Attendance! OiD: ", Oid)
3661                            with HiddenPrints():
3662                                try:
3663                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "C1 ENTRY: Marked attendance - OiD"+'%s'%Oid)
3664                                    print(sender)
3665                                except Exception:
3666                                    pass
3667                            print("-------------------------------------\n")
3668                        elif data==1:
3669                            #print('Component %s found in %s row(s)'%(dt_string,
data))
3670                            empmascur.execute("insert into attendance(Date, OiD,
Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'O'))
3671                            empmas.commit()
3672                            print("\n----------------DBFA-----------------")
3673                            print("C2 DAY END: Marked Attendance! OiD: ", Oid)
3674                            with HiddenPrints():
3675                                try:
3676                                    sender = telegram_bot_sendtext(dt_string + "\n"
+ "C2 DAY END: Marked attendance - OiD"+'%s'%Oid)
3677                                    print(sender)
3678                                except Exception:
3679                                    pass
3680                            print("-------------------------------------\n")
3681                        elif data > 1:
3682                            print("You can only mark in/out attendance once a day!
\n")
3683                    else:
3684                        print("OiD not found! \n")
3685
3686            if empfac == "6":
3687                print("6. Attendance records - All")
3688                with HiddenPrints():
3689                    try:
3690                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed:
Employee Attendance Records - deltaDBFA")
3691                        print(sender)
3692                    except Exception:
3693                        pass
3694                import sqlite3, time, os, requests, datetime
3695                from datetime import datetime, date
```

```
3696
3697                empmas = sqlite3.connect(r'dbfaempmaster.db')
3698                empmascur = empmas.cursor()
3699
3700                empmascur.execute("SELECT DISTINCT * FROM emp")
3701                dump = empmascur.fetchall()
3702                Oiddump = []
3703                for row in dump:
3704                    Oiddump.append(row[0])
3705
3706                print("OiDs registered: ", Oiddump)
3707
3708
3709                now = datetime.now()
3710                dt_string = now.strftime("%Y/%m/%d")  #datetime object containing
    current date and time
3711
3712                month = datetime.now().month - 1
3713                if month < 1:
3714                    month = 12 + month  # At this point month is 0 or a negative
    number so we add
3715                if len(str(month)) == 1:
3716                    month = "0"+str(month)
3717                dt1mb =
    ('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
3718
3719
3720                time.sleep(0.5)
3721                print("\nLoading ALL attendance data recorded since the start of
    using DBFA\n")
3722                empmascur.execute("SELECT * FROM attendance ORDER BY Date ASC")
3723                returned = empmascur.fetchall()
3724                for row in returned:
3725                    print(row)
3726
3727
3728            if empfac == "7":
3729                print("7. Attendance records - OiD-specific")
3730                with HiddenPrints():
3731                    try:
3732                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed:
    Employee Attendance Records - deltaDBFA")
3733                        print(sender)
3734                    except Exception:
3735                        pass
3736                import sqlite3, time, os, requests
3737                from datetime import datetime  #for reporting the billing time and
    date
3738
3739                empmas = sqlite3.connect(r'dbfaempmaster.db')
3740                empmascur = empmas.cursor()
3741
3742                empmascur.execute("SELECT DISTINCT * FROM emp")
3743                dump = empmascur.fetchall()
3744                Oiddump = []
3745                for row in dump:
3746                    Oiddump.append(row[0])
3747
3748                print("OiDs registered: ", Oiddump)
3749
```

```
3750                 try:
3751                     Oid = int(input("DBFA ATTENDANCE RECORDS- Enter the OiD: "))
3752                     trypass = 1
3753                 except:
3754                     print("OiDs are integer-only. Please retry using valid
credentials! \n")
3755                     trypass = 0
3756 
3757                 if trypass == 1:
3758                     if Oid in Oiddump:
3759                         print("OiD found ")
3760                         time.sleep(0.5)
3761                         empmascur.execute("SELECT * FROM attendance WHERE Oid = ?
ORDER BY Date ASC, Time, IO", ('%s'%Oid))
3762                         returned = empmascur.fetchall()
3763                         for row in returned:
3764                             print(row)
3765 
3766 
3767             if empfac == "8":
3768                 print("8. Attendance records - All (THIS MONTH)")
3769                 with HiddenPrints():
3770                     try:
3771                         sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed:
Employee Attendance Records - deltaDBFA")
3772                         print(sender)
3773                     except Exception:
3774                         pass
3775                 print("Coming soon! ")
3776                 import sqlite3, time, os, requests, datetime
3777                 from datetime import datetime, date
3778 
3779                 empmas = sqlite3.connect(r'dbfaempmaster.db')
3780                 empmascur = empmas.cursor()
3781 
3782                 empmascur.execute("SELECT DISTINCT * FROM emp")
3783                 dump = empmascur.fetchall()
3784                 Oiddump = []
3785                 for row in dump:
3786                     Oiddump.append(row[0])
3787 
3788                 print("OiDs registered: ", Oiddump)
3789 
3790 
3791                 now = datetime.now()
3792                 dt_string = now.strftime("%Y/%m/%d")  #datetime object containing
current date and time
3793 
3794                 month = datetime.now().month - 1
3795                 if month < 1:
3796                     month = 12 + month  # At this point month is 0 or a negative
number so we add
3797                 if len(str(month)) == 1:
3798                     month = "0"+str(month)
3799                 dt1mb =
('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
3800 
3801 
3802                 time.sleep(0.5)
```

```
3803                 print("\nLoading data for days between "+'%s'%dt1mb+" and
"+'%s'%dt_string+"\n")
3804                 empmascur.execute("SELECT * FROM attendance WHERE Date BETWEEN ? AND
? ORDER BY Date ASC", (dt1mb, dt_string))
3805                 returned = empmascur.fetchall()
3806                 for row in returned:
3807                     print(row)
3808 
3809 
3810             if empfac == "9":
3811                 print("9. Attendance records - OiD-specific (THIS MONTH)")
3812                 with HiddenPrints():
3813                     try:
3814                         sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed:
Employee Attendance Records - deltaDBFA")
3815                         print(sender)
3816                     except Exception:
3817                         pass
3818                 import sqlite3, time, os, requests, datetime
3819                 from datetime import datetime, date
3820 
3821                 empmas = sqlite3.connect(r'dbfaempmaster.db')
3822                 empmascur = empmas.cursor()
3823 
3824                 empmascur.execute("SELECT DISTINCT * FROM emp")
3825                 dump = empmascur.fetchall()
3826                 Oiddump = []
3827                 for row in dump:
3828                     Oiddump.append(row[0])
3829 
3830                 print("OiDs registered: ", Oiddump)
3831 
3832                 try:
3833                     Oid = int(input("DBFA ATTENDANCE RECORDS- Enter the OiD: "))
3834                     trypass = 1
3835                 except:
3836                     print("OiDs are integer-only. Please retry using valid
credentials! \n")
3837                     trypass = 0
3838 
3839                 now = datetime.now()
3840                 dt_string = now.strftime("%Y/%m/%d")  #datetime object containing
current date and time
3841 
3842                 month = datetime.now().month - 1
3843                 if month < 1:
3844                     month = 12 + month  # At this point month is 0 or a negative
number so we add
3845                 if len(str(month)) == 1:
3846                     month = "0"+str(month)
3847                 dt1mb =
('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
3848 
3849                 if trypass == 1:
3850                     if Oid in Oiddump:
3851                         print("OiD found ")
3852                         time.sleep(0.5)
3853                         print("\nLoading data for days between "+'%s'%dt1mb+" and
"+'%s'%dt_string+"\n")
```

```
3854                        empmascur.execute("SELECT * FROM attendance WHERE OiD = ?
AND Date BETWEEN ? AND ? ORDER BY Date ASC", ('%s'%Oid, dt1mb, dt_string))
3855                        returned = empmascur.fetchall()
3856                        for row in returned:
3857                            print(row)
3858
3859
3860            if empfac == "10":
3861                import pyqrcode, png, os
3862                from pyqrcode import QRCode
3863
3864                empmas = sqlite3.connect(r'dbfaempmaster.db')
3865                empmascur = empmas.cursor()
3866
3867                empmascur.execute("SELECT * FROM emp")
3868                emprows = empmascur.fetchall()
3869                for emprow in emprows:
3870                    print(emprow)
3871
3872                time.sleep(1)
3873
3874                emppay = str(input("Enter the Oid (Employee ID) to pay salary for:
"))
3875                empmascur.execute("SELECT * FROM emp WHERE Oid LIKE ?",
("%"+emppay+"%", ))
3876                print((empmascur.fetchall()[0]))
3877                time.sleep(1)
3878                emppayconfox = input("\n\nPay salary? (y/n): ")
3879                if emppayconfox == "y":
3880                    #emarpay = str("%"+'%s'%emppay+"%")
3881                    empmascur.execute("SELECT Name, UPI FROM emp WHERE Oid LIKE ?",
(emppay,) )
3882                    tempemppay = empmascur.fetchall()
3883                    print("Paying ", list(tempemppay[0])[0], "at ",
list(tempemppay[0])[1])
3884                    name = list(tempemppay[0])[0]
3885                    upid = list(tempemppay[0])[1]
3886                else:
3887                    empmenu()
3888                    break
3889                    print("Aaaa")
3890
3891                #upid = '9810141714@upi'
3892                #name = 'KPBalaji'
3893
3894                s = "upi://pay?pa="+'%s'%upid+"&pn="+'%s'%name+"&cu=INR"
3895
3896                # Generate QR code
3897                url = pyqrcode.create(s)
3898
3899                url.png('payqr.png', scale = 6)
3900                from PIL import Image, ImageDraw, ImageFont
3901                image = Image.open('payqr.png')
3902                with HiddenPrints():
3903                    try:
3904                        sender = telegram_bot_sendtext(dt_string + "\n" + "Started
Process: Issue Salary - deltaDBFA")
3905                        print(sender)
3906                    except Exception:
3907                        pass
```

```
3908                draw = ImageDraw.Draw(image)
3909                font =
ImageFont.truetype(r'C:\Users\balaj\AppData\Local\Microsoft\Windows\Fonts\MiLanProVF
.ttf', size=200)
3910                (x, y) = (5, 250)
3911                xname = 'Scan to pay with UPI              deltaDBFA'
3912                draw.text((x, y), xname) #, fill=color)
3913                (x, y) = (5, 5)
3914
3915                name = "Paying "+'%s'%name+" "*(28-len(str(name)))+"deltaPay"
3916                draw.text((x, y), name) #, fill=color)
3917                image.save('payqr.png', optimize=True, quality=120)
3918
3919                print("-------------------------------------")
3920                time.sleep(1)
3921                print("DBFA will now open a QR code for UPI payment to the
registered UPI address of the employee.")
3922                time.sleep(1)
3923                print("Scan the code in a UPI app to pay")
3924                time.sleep(1)
3925
3926
 os.system(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\payqr.png')
3927
3928                time.sleep(5)
3929                paycheck = input("Mark salary as 'PAID'? (y/n): ")
3930                if paycheck == "y":
3931                    print("Salary paid!")
3932                else:
3933                    print("Not paid. ")
3934
3935            if empfac == "11":
3936                print("Returning to DBFA Main.. ")
3937                time.sleep(1)
3938                break
3939
3940
3941
3942    #CIT
3943    elif decfac == "113":
3944        print("INTERNAL TESTING MODE")
3945        ffxfac = str(input("Enter CIT Testing Mode? (y/n):: "))
3946        if ffxfac == "y":
3947            ffrxfac = str(input("Entering CIT may lead to data loss. Confirm
entering CIT? (y/n):: "))
3948            if ffrxfac == "y":
3949                print("DNSS CIT MODE")
3950                print(" ")
3951                print(" ")
3952                print("NOTE: DBFA will restart to execute CIT options. ")
3953                print("CIT Options::")
3954                print("Enter '1' to CLEAR ALL CUSTOMER RECORDS")
3955                print("Enter '2' to CLEAR ALL VOUCHERS/ COUPONS")
3956                print("Enter '3' to exit CIT")
3957                citfacin = int(input("Waiting for input:: "))
3958                if citfacin == 1:
3959                    # window.close()
3960                    with HiddenPrints():
3961                        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(x)
        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")
        logger.write("Customer registry accessed! \n")
        logger.write("---------------------------------------- \n")

```

```
4018    elif decfac in ("2c", "2C", "2 c", "2 C"):
4019        del2c()
4020        logger.write("-------------------------------------- \n")
4021        logger.write("\nDBFA Customer Purchase Records accessed! \n")
4022
4023    elif decfac in ("2d", "2D", "2 d", "2 D"):
4024        del2d()
4025        logger.write("-------------------------------------- \n")
4026        logger.write("\nCustomer search used. \n")
4027
4028    elif decfac in ("2e", "2E", "2 e", "2 E"):
4029        del2e()
4030        logger.write("\n\n-------------------------------------- \n")
4031        logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
4032        logger.write("Customer data exported to CSV! ")
4033        with HiddenPrints():
4034            try:
4035                sender = telegram_bot_sendtext(dt_string + "\n" + "Sales data
exported to CSV- REDFLAG Urgent Security Notice!")
4036                print(sender)
4037            except Exception:
4038                pass
4039        logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
4040        logger.write("-------------------------------------- \n\n\n")
4041
4042
4043
4044    # Direct Calls Section - 3
4045    elif decfac in ("3a", "3A", "3 a", "3 A"):
4046        del3a()
4047
4048    elif decfac in ("3b", "3B", "3 b", "3 B"):
4049        del3b()
4050
4051    elif decfac in ("3c", "3C", "3 c", "3 C"):
4052        del3c()
4053
4054    elif decfac in ("3d", "3D", "3 d", "3 D"):
4055        del3d()
4056
4057    elif decfac in ("3e", "3E", "3 e", "3 E"):
4058        del3e()
4059        logger.write("\n-------------------------------------- \n")
4060        logger.write("Sales log accessed! ")
4061
4062
4063    elif decfac in ("3f", "3F", "3 f", "3 F"):
4064        del3f()
4065        logger.write("\n\n-------------------------------------- \n")
4066        logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
4067        with HiddenPrints():
4068            try:
4069                sender = telegram_bot_sendtext(dt_string + "\n" + "Customer data
exported to CSV- REDFLAG Urgent Security Notice!")
4070                print(sender)
4071            except Exception:
4072                pass
4073        logger.write("Sales data exported to CSV! ")
4074        logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
4075        logger.write("-------------------------------------- \n\n\n")
```

```
4076
4077
4078
4079     elif decfac in (None, "", " "):
4080         print("Please select a valid main-menu option. erc101\n\n")
4081         time.sleep(0.8)
4082         continue
4083
4084     else:
4085         print("Please select a valid main-menu option. erc101\n\n")
4086         time.sleep(0.8)
4087         continue
4088
4089 # End of program
4090 # Available on github: www.github.com/deltaonealpha/DBFA
4091 # https://deltaonealpha.github.io/DBFA/
```

<Login Service Scripts – (3/3)>

// The Code 2/8

```
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():
    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()
            window.close

 os.startfile(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\dlr.pyw')
        else:

 os.startfile(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\wrelogin.pyw')
        #window.close
        #erraise()
```

```
56 import PySimpleGUI as sg
57 if
os.path.exists(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.txt')
:
58     os.remove(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.txt')
59 if
os.path.exists(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.zconf
'):
60
 os.remove(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.zconf')
61 sg.theme('DarkTeal9') # Add a touch of color
62 # All the stuff inside your window.
63 layout = [  [sg.Text("DBFA Security")],
64             [sg.Text("This program requires you to login.")],
65             [sg.Button('Login'), sg.Button('Exit')] ]
66 # Create the Window
67 window = sg.Window('DBFA', layout)
68 # Event Loop to process "events" and get the "values" of the inputs
69 while True:
70     event, values = window.read()
71     if event in (None, 'Exit'): # if user closes window or clicks cancel
72         break
73     window.close()
74     Login()
75
```

```
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('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()
            window.close()
            os.startfile('dlr.pyw')
            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()

```

```
import 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')

import PySimpleGUI as sg
sg.theme('DarkTeal7') # Add a touch of color
# All the stuff inside your window.
layout = [  [sg.Text('Login succesfull!')],
            [sg.Button('Proceed')] ]
# Create the Window
window = sg.Window('Login conf.', layout)
# Event Loop to process "events" and get the "values" of the inputs
while True:
    Proceed = "Proceed"
    event, values = window.read()
    if event in ('Proceed'):
        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)
        userblock.close()
        print("logging success")
        os.startfile('bleading_edge.py')
        window.close()
        window.close()
        exit
        exit
        exit
        break
```

<Database Deletion Services>

// The Code 3/8

```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 Creation Authenticator Service (1/2)>

// The Code 4/8

```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 5/8

```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 Service>

// The Code 6/8

```
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)

```

```
    import sqlite3, time

    empmas = sqlite3.connect(r'dbfaempmaster.db')
    empmascur = empmas.cursor()
    empmascur.execute("SELECT MAX(Oid) FROM emp")
    Oid = (int(empmascur.fetchall()[0][0]) + 1)
    strix = ('insert into emp(Oid, Name, DOB, Email, Mobile, Address, UPI, Dept,
Post, Salary) values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)')
    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]))
    empmascur.execute(strix, io)
    empmas.commit()
    #Oid = "Oid"+'%s'%"3"
    #xstr = "ALTER TABLE attendance ADD COLUMN "+'%s'%Oid+" CHAR"
    #empmascur.execute(xstr)
    #empmas.commit()
    print("Employee registered in all DBFA databases")
    time.sleep(2)
    print("\n\nData sets logged with DBFA..")
    empmascur.execute("SELECT * FROM emp ORDER BY Oid DESC LIMIT 0, 1")
    time.sleep(1)
    print("\n\nNEW Employee Details:: ")
    print("-------------------------------------------------------------")
    print('%s'%ds1b[0]+(30-len(str(ds1b[0])))*" "+"deltaDBFA Employee Identifier")
    print("Employee OID: ", empmascur.fetchall()[0][0])
    print("-------------------------------------------------------------\n\n")
    time.sleep(2)
```

<Two-Factor-Authentication Modification Service>

// The Code 7/8

```python
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 Plotter Service>

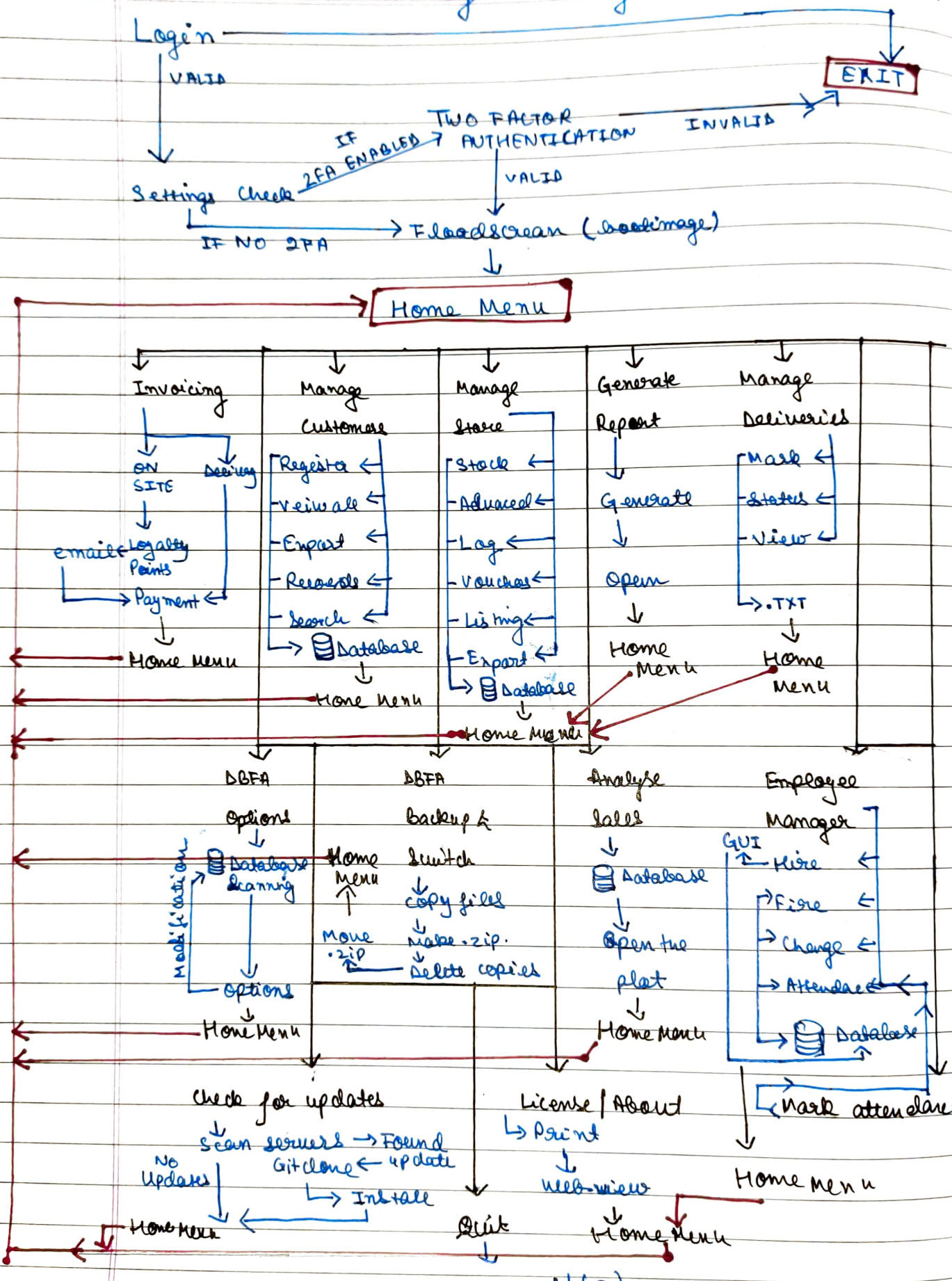
// The Code 8/8

```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()
```

(Please Turn Over)

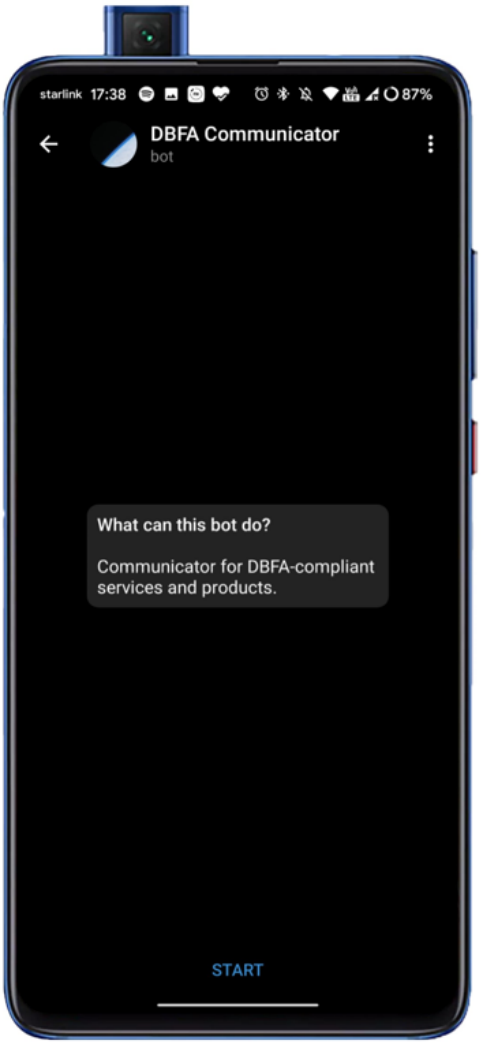
// Logic Chart

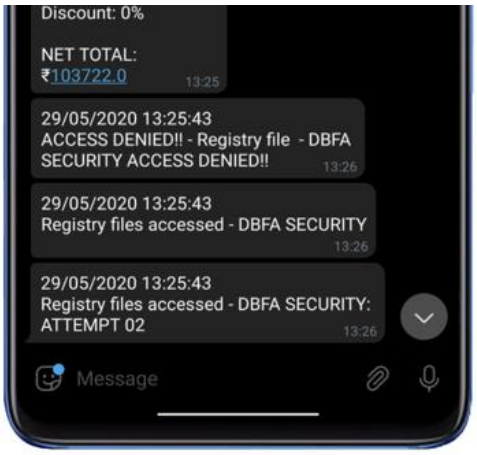
DBFA 8.34 Program Logic

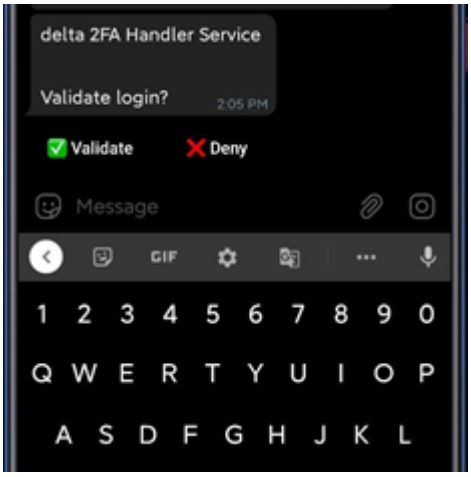


(Please Turn Over)

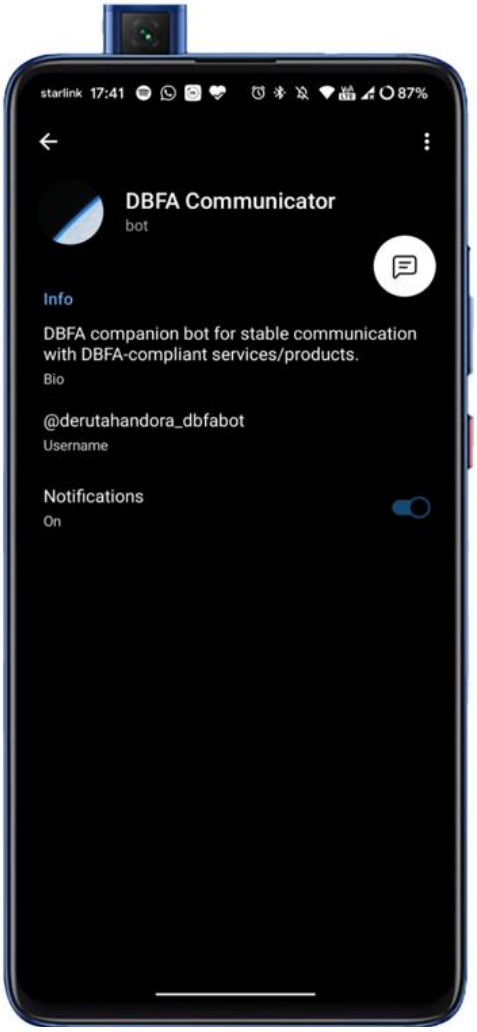
// Telegram Integration

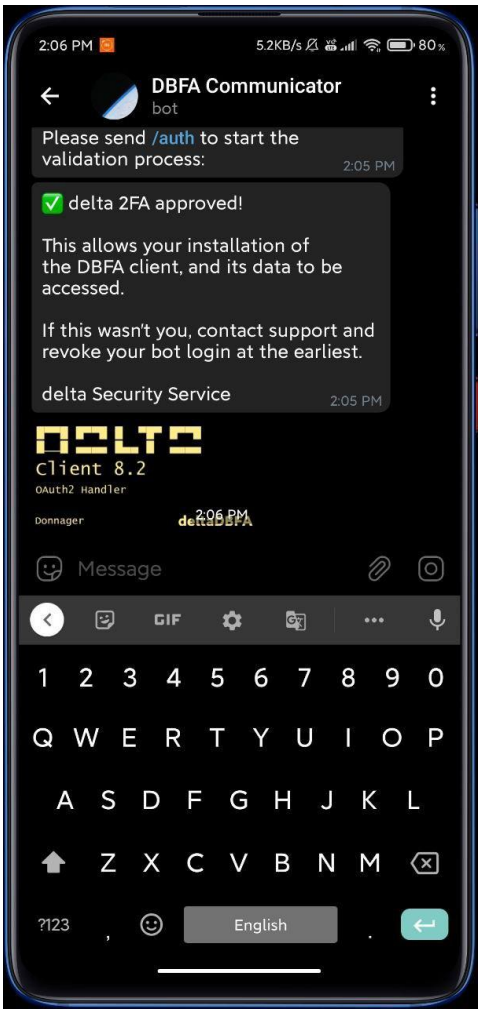

DBFA Communicator
bot
What can this bot do?
Communicator for DBFA-compliant services and products.
START

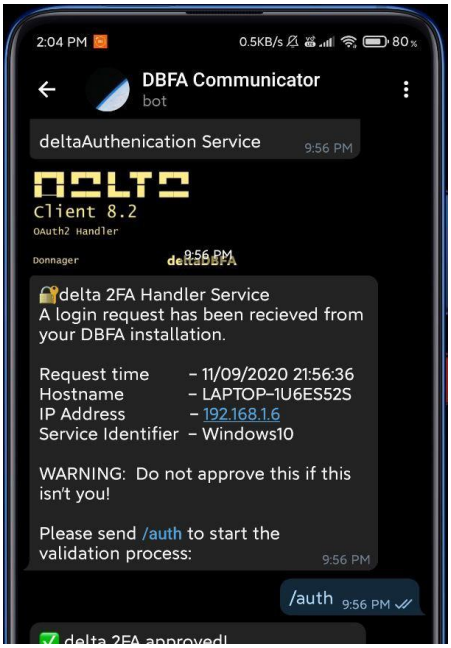

Discount: 0%
NET TOTAL:
₹103722.0
29/05/2020 13:25:43
ACCESS DENIED!! - Registry file - DBFA SECURITY ACCESS DENIED!!
29/05/2020 13:25:43
Registry files accessed - DBFA SECURITY
29/05/2020 13:25:43
Registry files accessed - DBFA SECURITY: ATTEMPT 02
Message

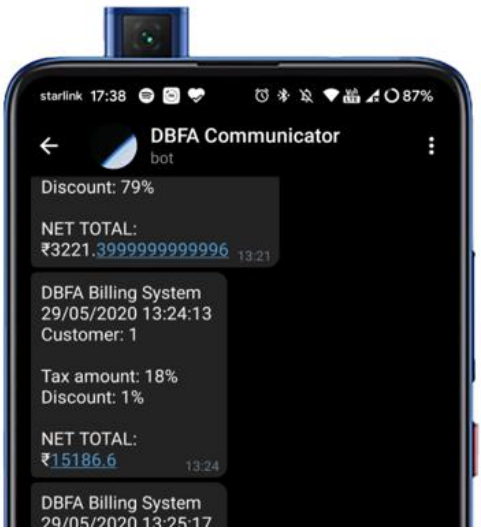

delta 2FA Handler Service
Validate login?
Validate
Deny
Message


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)

// 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