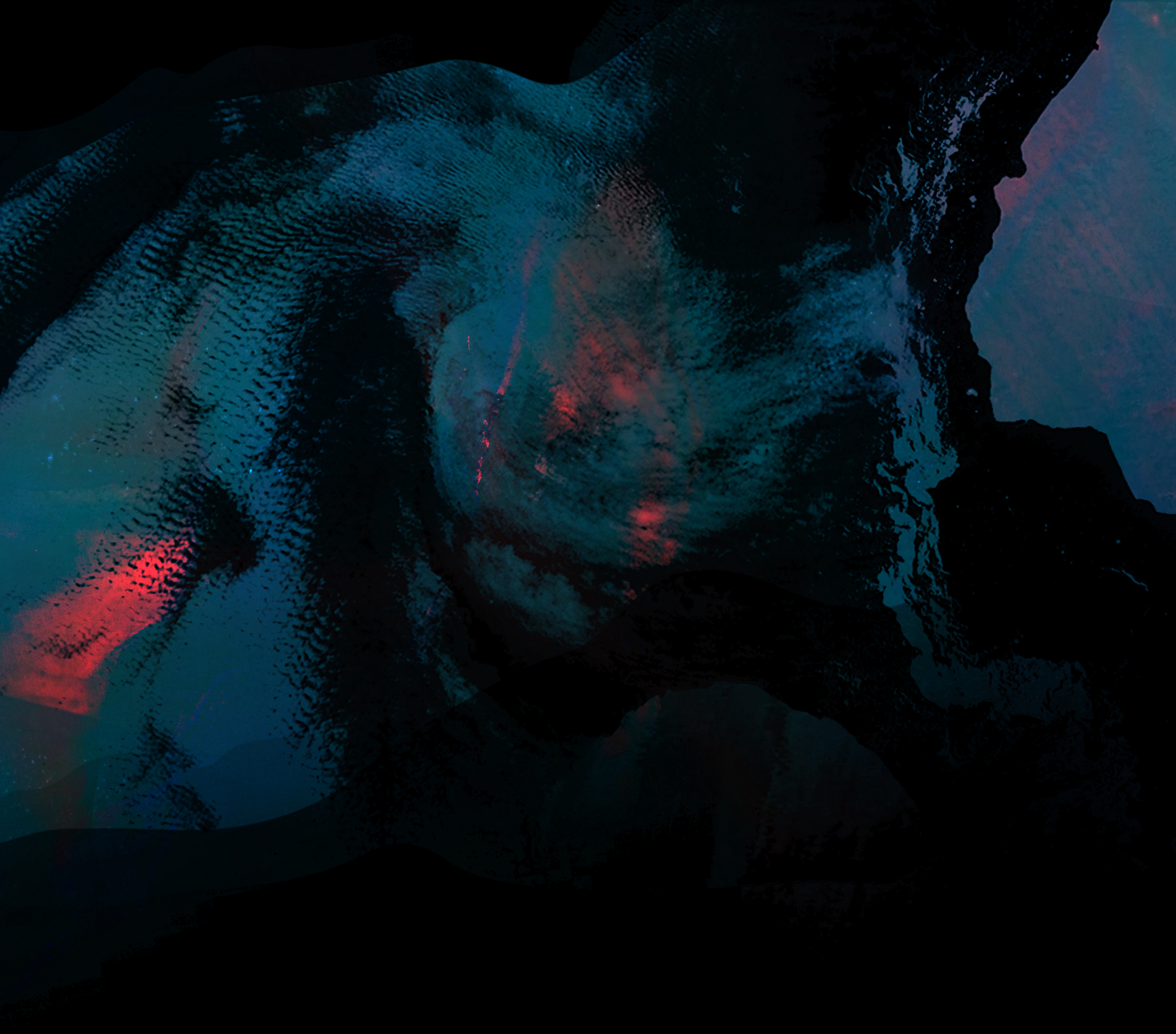


Inception 1.0

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.52**
- **CHECK FOR UPDATES**
- **QUIT**

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

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

(Please Turn Over)

//Libraries

Libraries used by DBFA:

from datetime import datetime

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

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

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

import urllib, json, math, random

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

import telegram_send

- Library which leverages the Telegram BOT API v2

from telegram import InlineKeyboardButton, InlineKeyboardMarkup

- Library which leverages the Telegram BOT API v2

from telegram.ext import Updater, CommandHandler, CallbackQueryHandler

- Library which leverages the Telegram BOT API v2

import getpass

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

from PySimpleGUI import PySimpleGUI

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

from win10toast import ToastNotifier

- Module to display Windows 10 UWP toast notifications.

import sqlite3

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

import cv2

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

import colorama

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

import reportlab

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

import requests

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

import pyqrcode, png

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

from pyqrcode import QRCode

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

from PIL import Image, ImageDraw, ImageFont

- Module used to interface with image files.

from tabularprint import table

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

from tqdm import tqdm

- Module used to display neat progress bars.

import webbrowser

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

import spotilib

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

from SwSpotify import spotify

- Another module used to interface with Spotify.

import email

- Module used to send and receive emails.

import pandas

- A very advanced library used for data analysis

import csv

- An inbuilt library used to interface with CSV files.

from pynput.keyboard import Key, Controller

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

import platform

- A module used to extract system information.

import matplotlib.pyplot

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

import oschmod

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

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

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

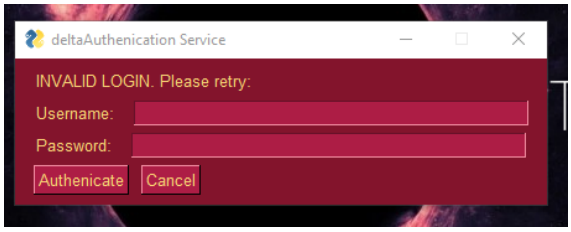
(Please Turn Over)

(Please Turn Over)

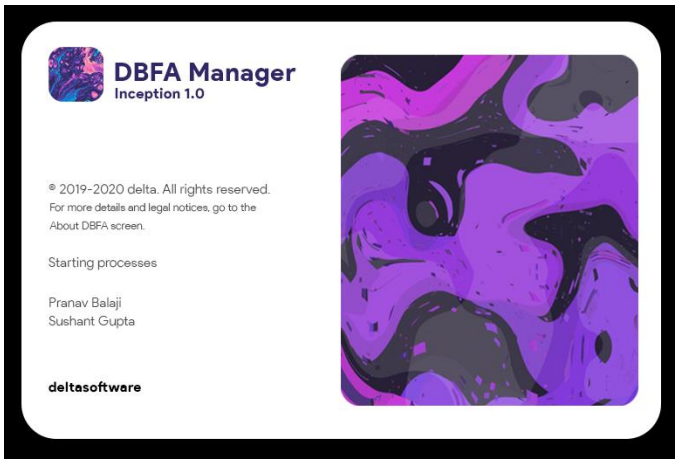
//Working

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.



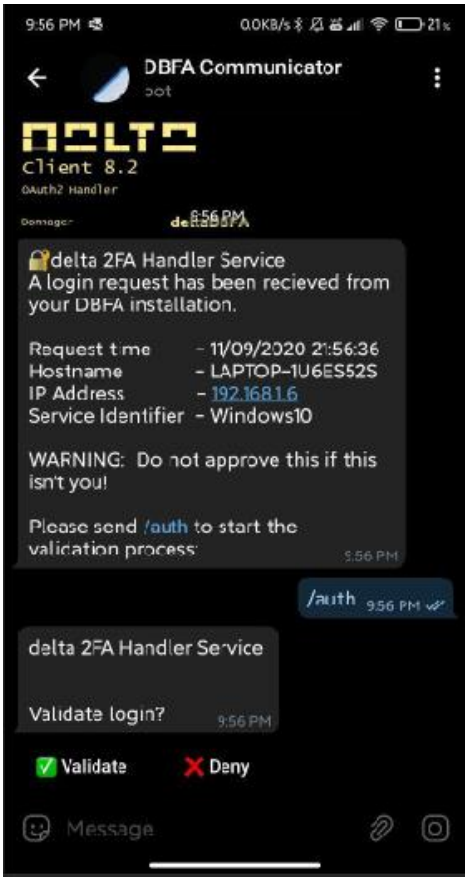
Upon successful login, a confirmation is provided and DBFA is subsequently launched. However, if the credentials entered are found to be incorrect, the same login page is relaunched, this time with a red background and text indicative of invalid creds.



With DBFA, 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.

```
--------------------------------
delta Update Utility
--------------------------------
Server:  8.311
Local:  8.311
```

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.

```
Heyy there, balaj

Profit (last week): 0                                   Profit (today):  0
Pending deliveries: 15          DBFA User: balaj        11/09/2020 22:40:45

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:                                     emp/EMP - DBFA Employee Manager
       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

What would you like to do?                 The OG Store Manager
DBFA Music Controls: *prev* <<< | *pause* <|> | *next* >>>      CLIENT 8.12 DONNAGER

Currently playing:  world.execute(me);  by   Mili

What would you like to do?

Select option:
```

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

Now we'll be discussing each menu option separately.

INVOICING

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

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

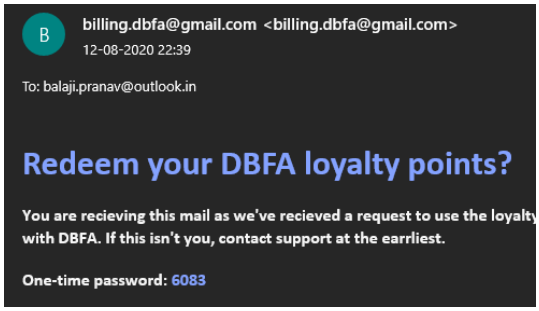
At this point, the user (presumably a store cashier) is given an option to either process the invoice as an in-store purchase, or as a delivery. In case of a delivery, the customer's name, and address (multi-line input) are enquired for and the same is stored in a file.

NOTE: Deliveries do not support loyalty point redemption.

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

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

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



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

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

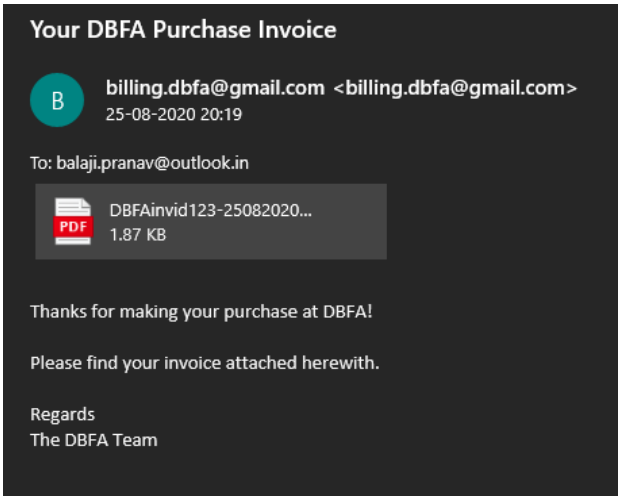
NOTE: Deliveries do not support any form of payment except pay-on-delivery.

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

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

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

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



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

CUSTOMER MANAGEMENT

This option houses five sub-options:

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

REGISTER A CUSTOMER

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

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

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

VIEW THE CUSTOMER REGISTRY

This option simply displays customer records as stored with DBFA.

VIEW CUSTOMER-SPECIFIC PURCHASE RECORDS

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

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

FIND A CUSTOMER

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

```
Currently playing:  Summoning 101  by   M

What would you like to do?

Select option: 2d
Customer Name: pra
--------------  ------------------------
ID              1
Customer NAME   Pranav
EMAIL           balaji.pranav@outlook.in
ID              1
Name            Pranav
Purchases Made  74
Total           4162012.4941999987
Loyalty Points  12328.150000000007
--------------  ------------------------
```

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.

```
Connecting to QuickVend Service... ~~~
Store Stock::
*-----------*-------------------------------------------------------*--------------*
|Product ID |Product Name                                           |Product Stock |
*-----------*-------------------------------------------------------*--------------*
|1          |TV 4K OLED 50                                          |8             |
|2          |TV FHD OLED 50                                         |7             |
|3          |8K QLED 80                                             |9             |
```

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.

```
Select option: 3b
-------- DBFA Stock Master v1 ---------

a: Order New Stock
b: Update Delivery Status
c: MASS - Fetch Current Status
d: INDVL - Fetch Current Status
e: View Vendor Details
f: Contact Vendor
g: Edit Vendor Contact
h: Modify Low-Stock Warning Bar
Select::
```

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 are collected, published into a CSV, saved, and opened in the device's default CSV-viewer, using this option.

(Please Turn Over)

GENERATE STORE REPORT

This is one of the flagship offerings of DBFA.

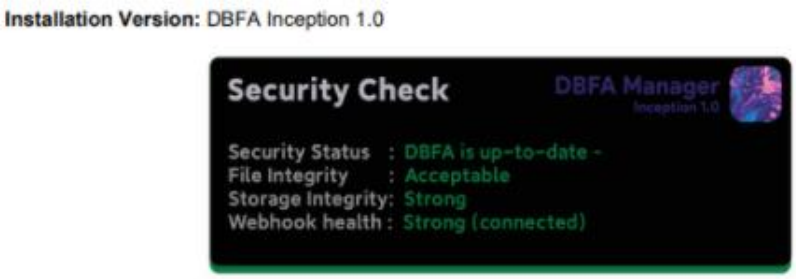
This option automatically picks data from multiple sources, analyses it and segregates useful data from the rest, and creates an artfully arranged PDF file.

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)
- DBFA SECURITY STATUS
- EMPLOYEE ATTENDANCE METRICS
- SALES RECORD OVERVIEW
- 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
- DEEP ARCHIVAL ENGINE DATA (with protected keys)

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

DELVERY MANAGER

VIEW PENDING DELIVERIES

A list of pending delivery tickets is displayed.

VIEW PENDING DELIVERIES COUNT

The number of pending deliveries is counted and displayed.

MARK A DELIVERY TICKET AS DELIVERED

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

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

DBFA OPTIONS

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

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

1: Display boot image : | ON |

2: Email invoice to registered customers : | ON |

3: Enable DBFA Music Controls (beta): : | ON |

4: Open CSV when exported : | ON |

5: Enable database encryption (under development) : | OFF|

6: Enable DBFA Secure Two-Factor-Authentication : | OFF|

7: Use new DBFA Menu style : | ON |

8: Create DBFA Desktop Shortcut : | Proceed > |

9: Delete customer records : | Proceed > |

10: Delete store records : | Proceed > |

11: Check for updates : | Proceed > |

12: Return to Main Menu : | Proceed > |

DISPLAY BOOT IMAGE

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

EMAIL INVOICE TO REGISTERED CUSTOMERS

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

ENABLE DBFA MUSIC CONTROLS (BETA)

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

OPEN CSVs WHEN EXPORTED

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

ENABLE DATABASE ENCRYPTION

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

ENABLE DBFA SECURE TWO-FACTOR AUTHENTICATION

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

(NOT RECOMMENDED TO DISABLE)

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

USE NEW DBFA MENU STYLE

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

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

CREATE DBFA DESKTOP SHORTCUT

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

DELETE ALL CUSTOMER RECORDS/ DELETE ALL STORE RECORDS

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

CHECK FOR UPDATES

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

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

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

DBFA BACKUP & SWITCH v2.1

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

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

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

With version 2.1, DBFA can now also restore itself from backup packages. Since this is a secure operation, the backed-up installation's admin password is required to restore. DBFA opens an os-level file dialog. Upon selecting the .zip file, DBFA automatically restores all settings, data sources, keys, credentials, IDs and log files.

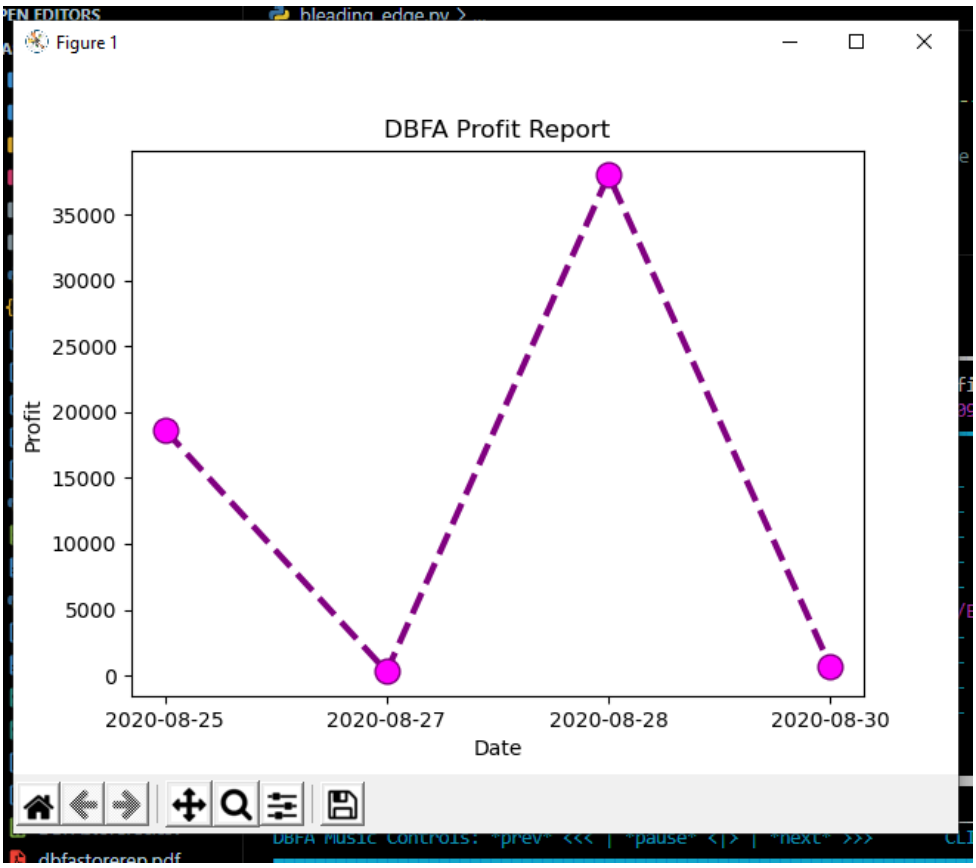
NOTE: 2FA CANNOT BE DISABLED FOR THIS IN ANY CASE.

ANALYSE SALES (SALES PLOTTER)

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

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

SAMPLE:



DBFA EMPLOYEE MANAGER

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

(Note: Employee Manager now required administrator login [18/10/2020 ISU])

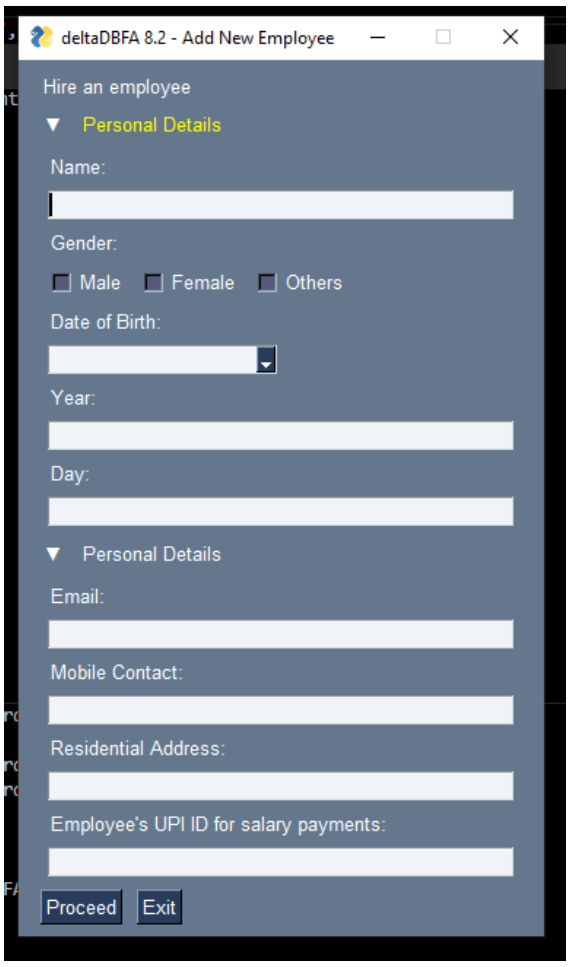
DBFA Employee Manager can be triggered on typing “emp”, “EMP”, or other random case variations of the same from the home menu itself.

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

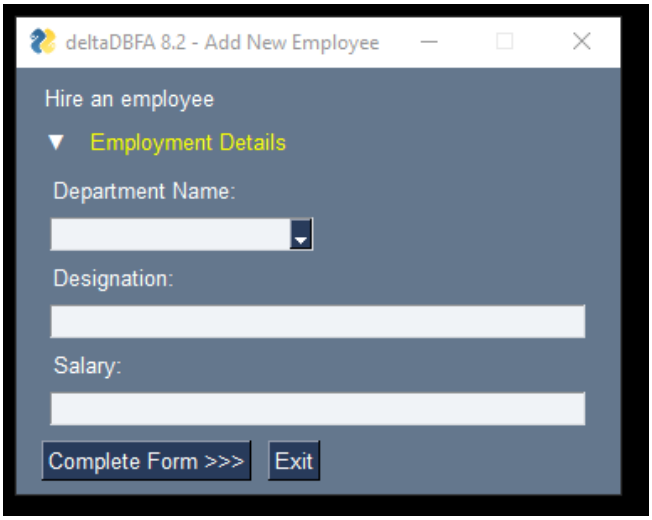
- HIRE AN EMPLOYEE
- VIEW EMPLOYEE RECORDS
- CHANGE EMPLOYEE DETAILS
- FIRE AN EMPLOYEE ʃ•'ᴗ'•ʄ⊃

- 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.

```
2. View employee records
---------------------------------------------
Employee records as maintained by DBFA:

(0, None, None, None, None, None, None, None, None, None)

(1, 'delta', '2003-9-01', 'contactdelta@delta.com', 8736937193, '(cl

(2, 'Sushant', '2003-3-12', 'sushant@delta.com', 9234567891, '(class

(3, 'rwfd', '2333-2-23', 'ed', 23232323223, 'ed', '23434323232', 'Sa
```

```
1. Change Name
2. Change Email
3. Change Mobile Contact

4. Change Residential Address

5. Change UPI Payments ID

6. Change Department
7. Change Designation (POST)
8. Change Salary
```

CHANGE EMPLOYEE DETAILS

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

FIRE AN EMPLOYEE ʃ•'ᴗ'•ʎ⊃

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

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.

Departing from the novice-nature of UPDATER v1.0, UPDATER v2.0 brings along automatic installations. DBFA now automatically replaces files, makes required data structure changes and reboots itself!

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

Improvements to Billing & Employee Manager

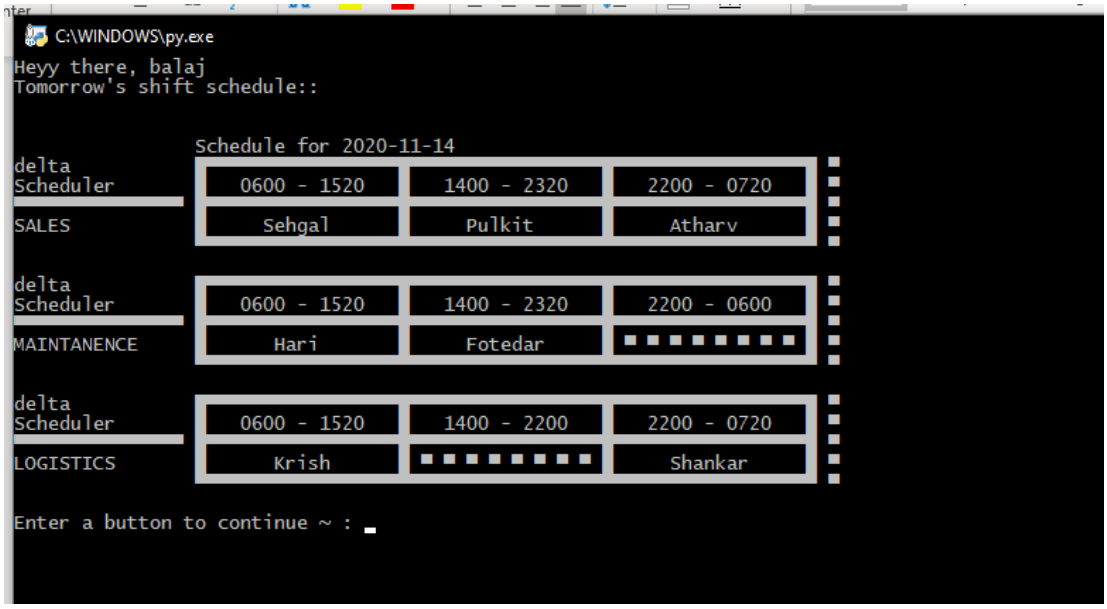
Presenting BILL SPLIT

DBFA Billing now has an option to split-bill. Decimal values are rounded off to two digits. Bills can be split between an unlimited number of payees.

AUTOMATIC EMPLOYEE SHIFT SCHEDULING

With build 8.6CE, DBFA can now automatically schedule employee shifts daily.

The next day's schedule is generated everyday whenever DBFA is booted and the same is displayed (only once a day).



This schedule is automatically sent to the store owner and a Telegram group housing all employees. Shifts are randomized and non-repetitive for the most part.

DBFA can now also take leave applications and dynamically reschedule shifts to accommodate even full shift leaves (shifts will be divided between other shift employees then).

The same schedule can also be re-seen via the "Employee Manager" option. This displayed schedule remains consistent with the one shown earlier.

Presenting SUPERCACHING

Unlike other applications which refer databases every time, for each product, thereby adding time-delay, DBFA fetches ALL listing details for each billing cycle, and caches the same in its memory. This saves a lot of querying time.

SUPERCACHED data is also updated every time the data sources are changed.

LIVE CART LISTING & PRODUCT REMOVAL FROM CART

With DBFA Inception – 1, the cart is displayed and updated every time a new product is added. Users can now also remove the last cart item!

```
Logitech G604 LightSpeed Wireless                          1700      1 qty.
Tactical Essentials Keyboard                               600       1 qty.
DROP GS21k RGB Gaming Keyboard                             2175      1 qty.
DROP GS21k RGB Gaming Keyboard                             2175      1 qty.

Type 'lastrem' to remove last added produt

Enter product code: lastrem

Last cart product removed ~


Currently in cart:
-------
Wired Speaker - 5.1                                        2100      1 qty.
Wired Speaker - 5.1                                        2100      1 qty.
Tactical Series Power Bank 30000mah                        3210      1 qty.
Tactical Series Power Bank 30000mah                        3210      1 qty.
Logitech G604 LightSpeed Wireless                          1700      1 qty.
Tactical Essentials Keyboard                               600       1 qty.
DROP GS21k RGB Gaming Keyboard                             2175      1 qty.

Enter product code:
```

INVOICE ID INDEXING

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

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.

AUTOMATIC SALARY CALCULATION

Stores can specify their minimum salary, extra-hour salary et cetera.

Whenever salary is to be paid, DBFA uses its data to calculate the salary amount that is to be paid. DBFA considers multiple factors while doing this as listed below:

- Designation
- Days worked in the current month
- Weekly average hours worked
- Overtime hours worked
- Applicable bonuses

DBFA INVOICE DEEP ARCHIVAL ENGINE

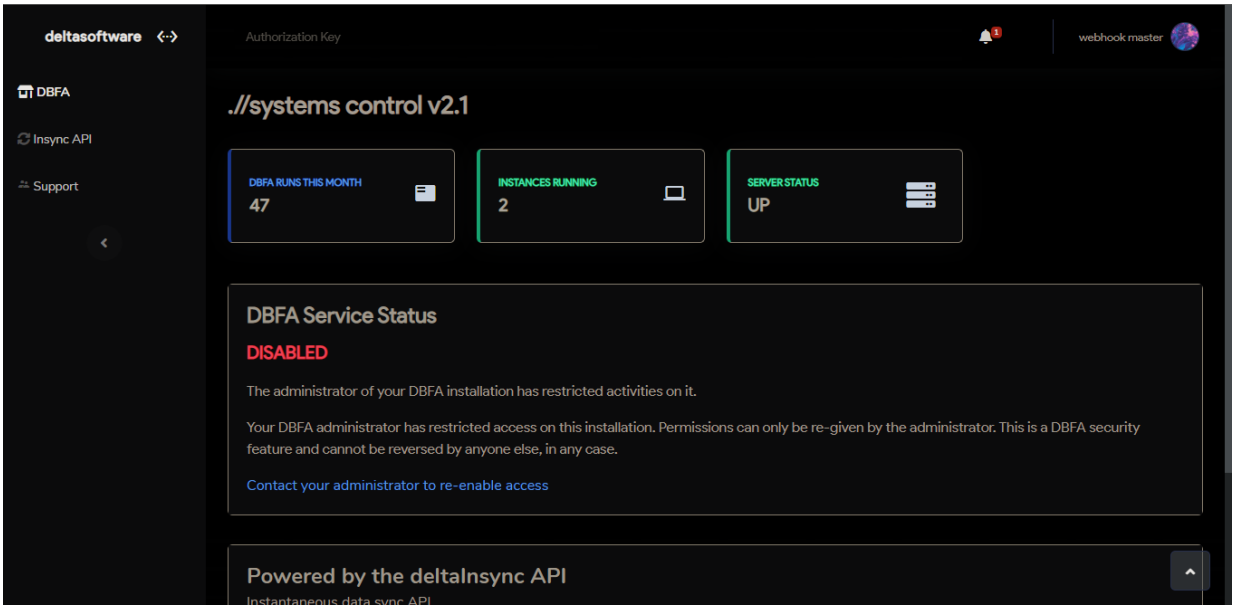
- As you know, DBFA generated PDF invoices upon each purchase. These can be immediately printed and are also automatically mailed to registered customers.
- However, maintaining invoice records by storing these PDF invoices can be gruesome when dealing with large sale volumes.
- To deal with this, DBFA houses a ***custom-developed deep archival algorithm.*** With this, DBFA can encode the complete data of an invoice into a string, randomize its characters for security and store this in a secure database in a matter of milliseconds!
- Whenever a back-dated invoice is required, DBFA can instantly fetch the invoice's encoded key, decode it, and pass the details to the same trusty PDF engine that typically generated its invoices. As always, this invoice is auto moved to a separate directory within DBFA's installation.
- This not only helps stores to maintain decades and centuries of invoice information, but also helps them save space compared to storing full-blown PDF invoices!
- ***NOTE:*** *The deep archival engine can be triggered via the "Invoice Manager" option.*

Improvements to Security

DBFA WEBHOOK SECURITY

DBFA features autonomous and continuous logging of feature access/ purchases/ actions on the installation.

Since this only informs the store owner of any possible malpractices, we wanted the owner to have a way of shut down operations if something seems our-of-order.



Presenting, DBFA Webhook Security. The store owner has to just send a message containing "disableDBFA" to the Telegram bot, and DBFA will completely stop all operations starting the next menu-cycle/ boot.

The best part is that this cannot be bypassed since DBFA refuses to boot without an active internet connection (raises alert and exits).

Whenever "disableDBFA" is triggered, a webpage showing operational status is opened.

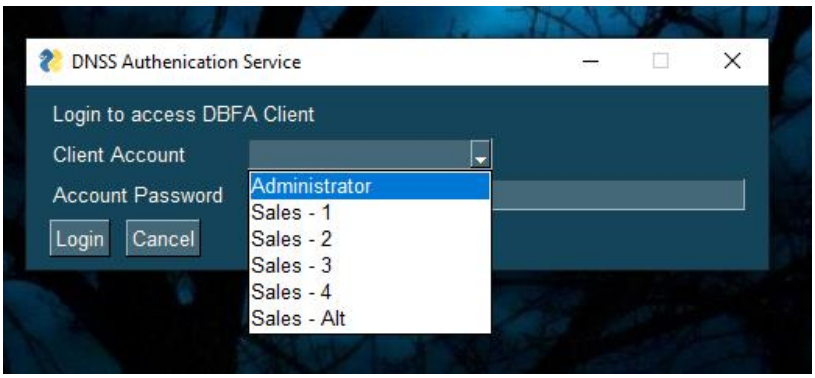
Subsequent attempts to use DBFA are suppressed until the store owner sends "enableDBFA" twice to the bot. On doing so, all restrictions are lifted automatically (user-account-level restrictions still apply).

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.

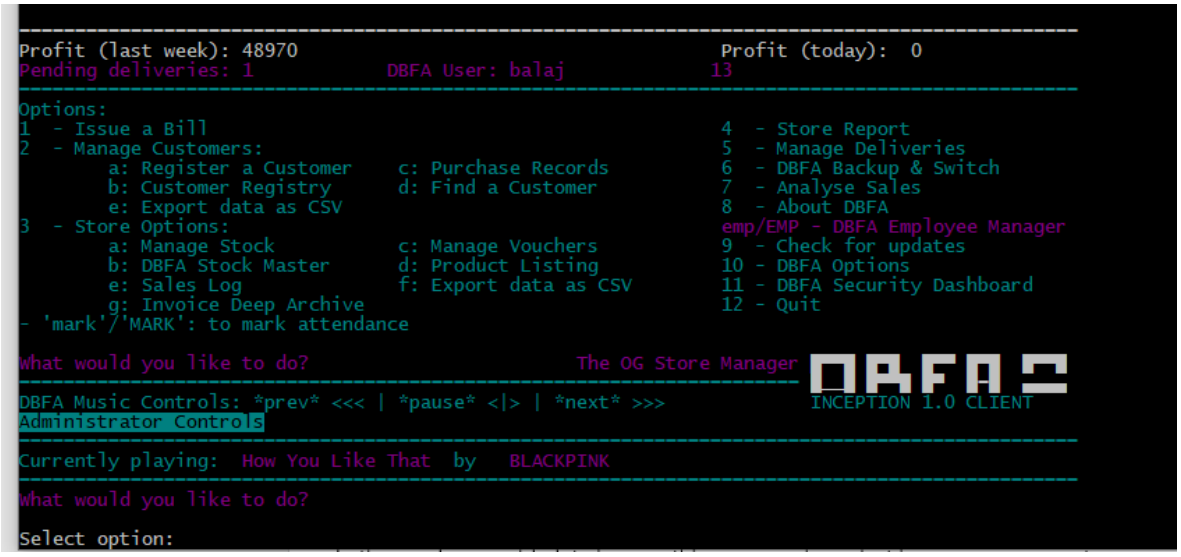
USER ACCOUNT CONTROLS AND RESTRICTIONS

With DBFA versions Inception – 1 and above, DBFA now allows having multiple user accounts. This is another stride in our pursuit of prime security, as not only does this allow for person-specific logging, but it also allows the owner to restrict functions on non-admin accounts.

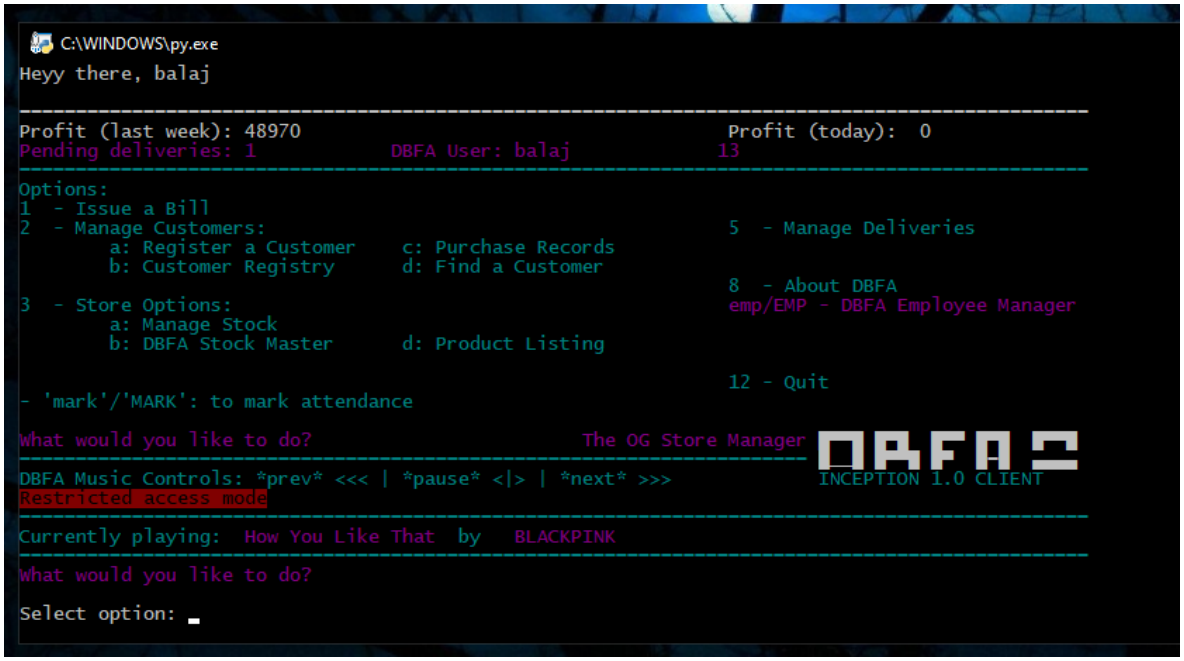


There are two account types, “administrator” and “sales-class”. There can only be one admin account, but infinite sales-class accounts, where all admin-level functions are restricted. These accounts can be created via the Employee Manager.

Administrator-level account:

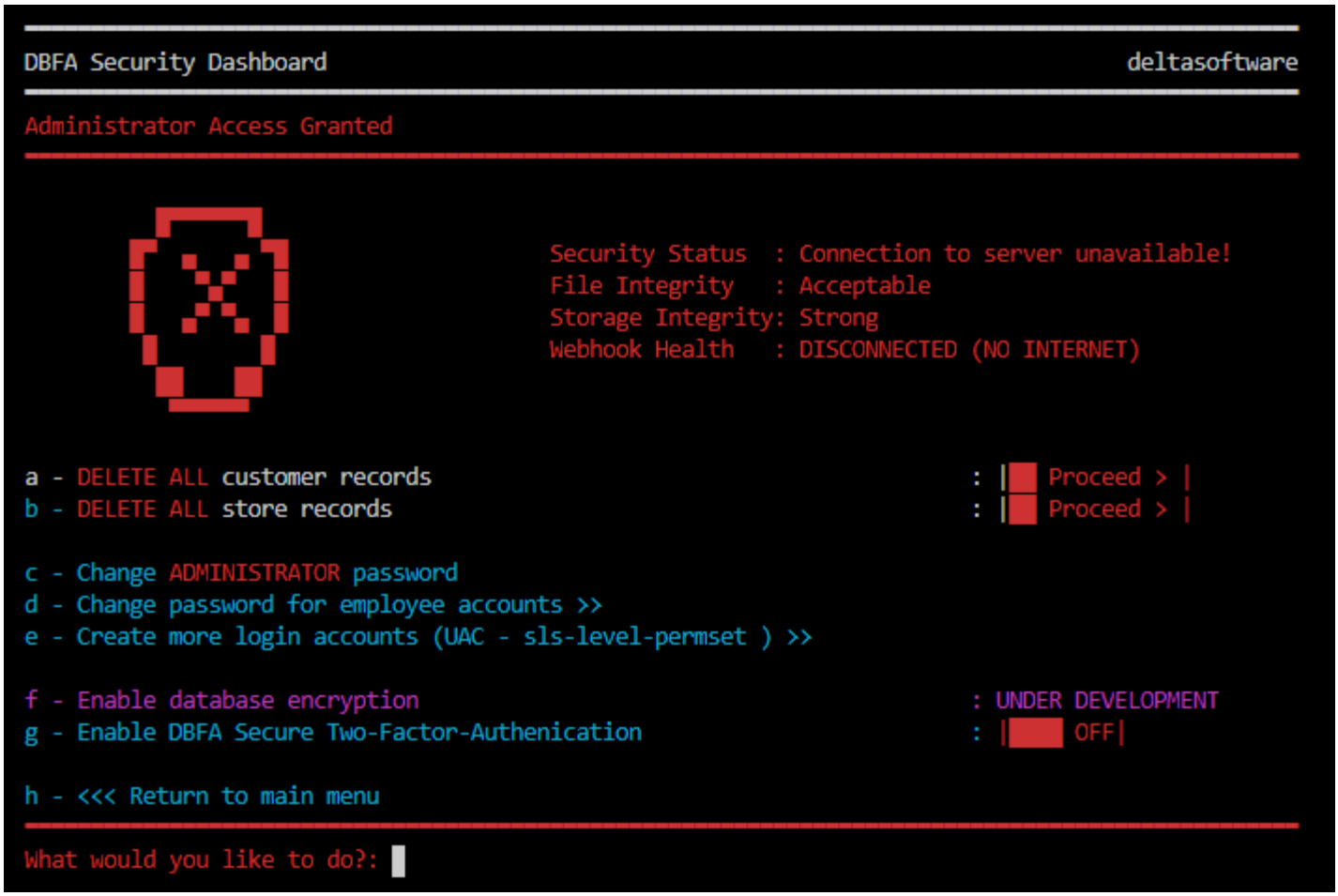


“Sales-class” accounts (with restrictions):



Presenting THE SECURITY DASHBOARD

Gives you a visual representation of program and dependencies health, with security-related options (clear records/ change passwords/ encryption toggle, 2FA toggle/ create login accounts.



BYPASS PREVENTION SERVICE v2

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

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

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

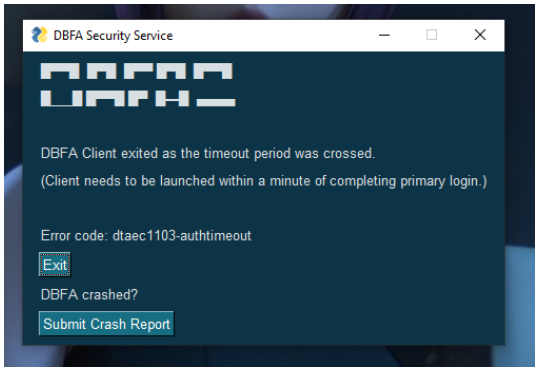
OTP-BASED 2FA MODIFICATION PROCESS

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.

LOGIN-CLIENT-RUN TIMEOUT

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



TWO-FACTOR-AUTHENTICATION

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

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

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

SECURE PASSWORDS; AUTOMATIC PATH REFERENCES

DBFA does not house any password/ key/ ID/ credentials in its main code. All these are fetched from multiple database sources. The key advantage with such a secure approach is that DBFA can be easily scaled for a large store/ chain, and databases can also be securely hosted on cloud servers.

AUTOMATED CODE INTEGRITY SCANS ON CRASH

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

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

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

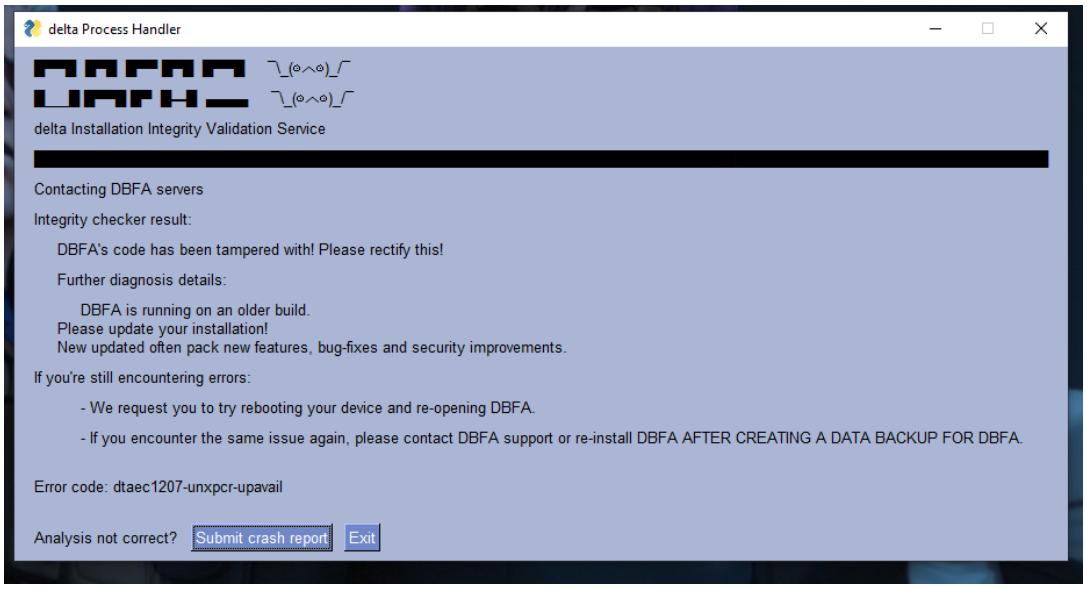
This scan proceeds in multiple steps, as detailed below:

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

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

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

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



AUTHENICATION FOR EMPLOYEE MANAGER

The Employee Manager now requires administrator login.

COMPLETE CONTROL OVER OPERATIONS

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

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

Improvements to Updater, Backups and Deployments

DBFA FIRST-TIME SETUP

Whenever DBFA is freshly installed (via DBFA Deployment: available on GitHub releases) or installed via source, an initialization wizard starts which generated database files, walks the user through getting Telegram and email requisite ready and asks the user for admin/sales passwords.

DBFA AUTOMATIC UPDATER v2.0

Departing from the novice-nature of UPDATER v1.0, UPDATER v2.0 brings along automatic installations. DBFA now automatically replaces files, makes required data structure changes and reboots itself!

BACKUP & SWITCH v2.1 (with automatic restoration)

With version 2.1, DBFA can now also restore itself from backup packages. Since this is a secure operation, the backed-up installation's admin password is required to restore. DBFA opens an OS-level file dialog. Upon selecting the .zip file, DBFA automatically restores all settings, data sources, keys, credentials, IDs, and log files.

General Improvements to Speed, Ease-of-use, Integrity & Stability

AUTOMATED DATABASE STRUCTURE REBUILDING

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

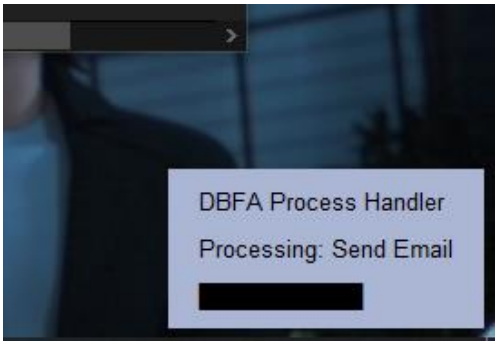
Disk failures, accidental deletions, malwares, viruses, ransomwares, or even internal errors can lead to corrupted databases. DBFA overcomes this by searching for its databases each time the software is booted.

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

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.

GUI PROGRESS BAR FOR VARIOUS PROCESSES



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

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

ERROR CATCHING ACROSS THE CODE

DBFA has almost its whole under a try... except block to avoid ungraceful crashes.

FASTER BOOT & MENU TIMES WITH INCEPTION – 1

With DBFA version Inception 1.0 Patch – I, DBFA now boots 39% faster with 60% faster menu times!

//Information

HARDWARE USED

- **CPU**

  INTEL CORE i5 7200U

- **SOLID STATE DRIVE**

  SAMSUNG 950 Evo

- **RAM**

  SK Hynix 8GB DDR4-2600mhz

- **NETWORK**

  GIGABIT WLAN

SOFTWARE USED

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

<Main Program – DBFA Client>

// The Code 1/12

```python
'''
DBFA Store Manager
by deltaonealpha and sushimuncher

DBFA Store Manager Client - Build Details:
-------------------------------------
Version Information: INCEPTION 1.0
Date               : 06-11-2020
Build Branch       : Source
Release Identifier : INCEPTION

ds-ddbfa.incp.client:101

https://github.com/deltaonealpha/DBFA
https://gitlab.com/deltaonealpha/DBFA
https://software.deltaone.tech
-------------------------------------
-------------------------------------
'''
##########################################################################
# vs                                                                     #
# Administrator == all                                                   #
# Sales = ['1', "2a", '2b', '2c', '2d', '3a', '3b', '3d', 5, 9, 10, 12 ]  #
##########################################################################

import traceback
import json, requests, time, urllib, sys

import os, time
global curdir, currdir, parentdir, fontsdir
currdir = str(os.getcwd())
from pathlib import Path
path = Path(os.getcwd())
parentdir = str(Path(path.parent))
userdir = os.path.expanduser('~')
fontsdir = str(os.path.join(userdir, 'AppData\Local\Microsoft\Windows\Fonts\MiLanProVF.ttf'))

class HiddenPrints:
    def __enter__(self):
        self._original_stdout = sys.stdout
        sys.stdout = open(os.devnull, 'w')
    def __exit__(self, exc_type, exc_val, exc_tb):
        sys.stdout.close()
        sys.stdout = self._original_stdout
        print()

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 settingsdatafetch(SettingsType):
    import sqlite3
```

```
    settings = sqlite3.connect(r'dbfasettings.db')
    settingsx = settings.cursor()
    settingsx.execute(("SELECT Notes from settings WHERE SettingsType = ?"), (SettingsType,))
    settingsfetch = (settingsx.fetchall()[0][0])
    return settingsfetch

def getlisting():
    import sqlite3
    settings = sqlite3.connect(r'dbfasettings.db')
    settingsx = settings.cursor()

    xnamiex = []
    xdatax = []
    settingsx.execute("SELECT Name FROM listing")
    for i in settingsx.fetchall():
        xnamiex.append(i[0])
    settingsx.execute("SELECT Cost FROM listing")
    for i in settingsx.fetchall():
        xdatax.append(i[0])
    return (xnamiex, xdatax)

def dictgetlisting():
    import sqlite3
    settings = sqlite3.connect(r'dbfasettings.db')
    settingsx = settings.cursor()

    xnamiex = []
    xdatax = []
    settingsx.execute("SELECT Name FROM listing")
    for i in settingsx.fetchall():
        xnamiex.append(i[0])
    settingsx.execute("SELECT Cost FROM listing")
    for i in settingsx.fetchall():
        xdatax.append(i[0])
    namiex, datax = xnamiex, xdatax
    lenx = []
    for i in range(1, len(namiex)+1):
        lenx.append(i)
    listres = {}
    for key in lenx:
        for value in namiex:
            listres[int(key)] = str(value)
            namiex.remove(value)
            break
    costres = {}
    for key in lenx:
        for value in datax:
            costres[int(key)] = str(value)
            datax.remove(value)
            break
    return(listres, costres)


def securedatafetch(SettingsType):
    import sqlite3
    settings = sqlite3.connect(r'dbfasettings.db')
    settingsx = settings.cursor()
    settingsx.execute(("SELECT Col1 from passkeyhandler WHERE Sno = ?"), (SettingsType,))
```

```python
    settingsfetch = (settingsx.fetchall()[0][0])
    return settingsfetch

def telegram_bot_sendtext(bot_message):
    with HiddenPrints():
        bot_token = str(securedatafetch(2))
        bot_chatID = str(securedatafetch(3))
        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()

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

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 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

def get_updates(offset=None):
    url = URL + "getUpdates"
    if offset:
        url += "?offset={}".format(offset)
    js = get_json_from_url(url)
    return js

def get_last_update_id(updates):
    update_ids = []
    for update in updates["result"]:
        update_ids.append(int(update["update_id"]))
    return max(update_ids)

def echo_all(updates):
    for update in updates["result"]:
        text = update["message"]["text"]
        chat = update["message"]["chat"]["id"]
        send_message(text, chat)

def get_last_chat_id_and_text(updates):
    num_updates = len(updates["result"])
    last_update = num_updates - 1
    text = updates["result"][last_update]["message"]["text"]
    chat_id = '680917769'
    return (text, chat_id)

```

```python

def send_message(text, chat_id):
    text = urllib.parse.quote_plus(text)
    url = URL + "sendMessage?text={}&chat_id={}".format(text, '680917769')
    get_url(url)


def dmain():
    #print(settingscommonfetch(9))
    #get_url('https://api.telegram.org/bot/deleteWebhook')
    with open('lastupdateid2.txt', 'a+') as file:
        file.close()
    with open('lastupdateid2.txt', 'r+') as file:
        xid = file.read()
    last_update_id = int(xid)
    #print(last_update_id)
    import webbrowser
    updates = get_updates(last_update_id)
    #print(updates)
    #time.sleep(5)
    #print(len(updates["result"]))
    if len(updates["result"]) > 0:
        #print("ddcc")
        last_update_id = get_last_update_id(updates)
        #print(last_update_id)
        for update in updates["result"]:
            #print(update["message"]["text"])
            try:
                if (update["message"]["text"]).replace(" ", "") == "disableDBFA":
                    settingsmodifier(9, 0)
                    time.sleep(0.5)
                    telegram_bot_sendtext("delta Webhook Services\nUsage permissions have been revoked from your installation of DBFA.\n\nExpect access to be stopped from the next boot/ menu cycle.\nhttps://dashboard.deltaone.tech/dbfa.html")
                    print("delta Webhook Prompt: ")
                    print("A webbrowser window will shortly open ~")
                    webbrowser.open('https://dashboard.deltaone.tech/dbfa.html')
                    with open('lastupdateid2.txt', 'a+') as file:
                        file.close()
                    with open('lastupdateid2.txt', 'r+') as file:
                        settingsmodifier(9, 0)
                        time.sleep(0.5)
                        file.truncate(0)
                        file.write('%d'%last_update_id)
                    time.sleep(5)
                    settingsmodifier(9, 0)
                    print("DBFA will now exit.")
                    time.sleep(0.5)
                    os._exit(0)
                    os._exit(1)
                    os._exit(0)
                elif (update["message"]["text"]).replace(" ", "") == "enableDBFA":
                    if settingscommonfetch(9) == 0:
                        settingsmodifier(9, 1)
                        print("delta Webhook Prompt: ")
                        telegram_bot_sendtext("delta Webhook Services\n\nAccess has been re-granted on your installation of DBFA.")
                        print("The administrator of this DBFA installation has re-allowed access to DBFA on this device!")
                        time.sleep(3)
```

```
                else:
                    if settingscommonfetch(9) == 0:
                        with open('lastupdateid2.txt', 'a+') as file:
                            file.close()
                        with open('lastupdateid2.txt', 'r+') as file:
                            file.truncate(0)
                            file.write('%d'%last_update_id)
                        print("A webbrowser window will shortly open ~")
                        print("delta Webhook Prompt: ")
                        webbrowser.open('https://dashboard.deltaone.tech/dbfa.html')
                        print("DBFA will now exit.")
                        time.sleep(5)
                        os._exit(0)
                        os._exit(1)
                        os._exit(0)
                    else:
                        pass
            except KeyError:
                print("Do not send images to DBFA communicator on Telegram!")
        with open('lastupdateid2.txt', 'a+') as file:
            file.close()
        with open('lastupdateid2.txt', 'r+') as file:
            file.truncate(0)
            file.write('%d'%last_update_id)
        #echo_all(updates)

    else:
        if settingscommonfetch(9) == 0:
            with open('lastupdateid2.txt', 'a+') as file:
                file.close()
            with open('lastupdateid2.txt', 'r+') as file:
                file.truncate(0)
                file.write('%d'%last_update_id)
            print("A webbrowser window will shortly open ~")
            print("delta Webhook Prompt: ")
            webbrowser.open('https://dashboard.deltaone.tech/dbfa.html')
            print("DBFA will now exit.")
            time.sleep(5)
            os._exit(0)
            os._exit(1)
            os._exit(0)

try:
    import getpass, time, pathlib, sqlite3, sys, os #sys, os for system-level ops
    from tabularprint import table
    from tqdm import tqdm
    import webbrowser
    dmain()
    # 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', str(currdir+r'\\MiLanProVF.ttf')))
    from reportlab.pdfgen import canvas
    from reportlab.lib.pagesizes import A4
    from reportlab.platypus import SimpleDocTemplate, Paragraph
    from reportlab.lib.styles import getSampleStyleSheet
    from reportlab.lib.units import cm
    from reportlab.lib.enums import TA_JUSTIFY
    from reportlab.lib.pagesizes import letter
    from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
    from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
    from reportlab.lib.units import inch
    from tabulate import tabulate
    from reportlab.lib import colors
    from reportlab.lib.pagesizes import letter
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
    import requests
    import cv2


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

    def settingscommondumpfetch(SettingsType):
        import sqlite3
        settings = sqlite3.connect(r'dbfasettings.db')
        settingsx = settings.cursor()
        settingsx.execute(("SELECT Notes 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 permssetter(NewValue):
        import sqlite3
        settings = sqlite3.connect(r'dbfasettings.db')
        settingsx = settings.cursor()
        settingsx.execute(("UPDATE settings SET Notes = ? WHERE SettingsType = 10"), (NewValue, ))
```

```
        settings.commit()



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

    global cccheck

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

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

```
    else:
        xbr7.execute("""CREATE TABLE IF NOT EXISTS stock
        (prodid INTEGER PRIMARY KEY,
        prodname CHAR,
        ordqty VARCHAR(500),
        delivered INT,
        delstat CHAR,
        vendor CHAR,
        vendcont CHAR
        lowstock INT);""")

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

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



    class HiddenPrints:
                def __enter__(self):
                    self._original_stdout = sys.stdout
                    sys.stdout = open(os.devnull, 'w')
                def __exit__(self, exc_type, exc_val, exc_tb):
                    sys.stdout.close()
                    sys.stdout = self._original_stdout
                    print()

    # TG Communicator
    def telegram_bot_sendtext(bot_message):
        with HiddenPrints():
            bot_token = str(securedatafetch(2))
            bot_chatID = str(securedatafetch(3))
            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()

    def telegram_bot_grouptext(bot_message):
        with HiddenPrints():
```

```
            bot_token = str(securedatafetch(2))
            bot_chatID = str(securedatafetch(4))
            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()

    def getOTP():
        global otp
        digits = "0123456789"
        otp = ""
        otp += (digits[math.floor(random.random() * 10)])
        otp += (digits[math.floor(random.random() * 10)])
        otp += (digits[math.floor(random.random() * 10)])
        otp += (digits[math.floor(random.random() * 10)])


    # DBFA Logo Printer
    def logoprintxrt():
        #VID: DBFA Inception 1.0
        print("        ___ ______ ___   _____________    _____________     _______")
        time.sleep(0.01)
        print("       /  /_______/  /  /  /_______/  /  /  /________/    /  /_/ /")
        time.sleep(0.01)
        print("      /  /       /  /  /  /       /  /  /  /            /  /  / /")
        time.sleep(0.01)
        print("     /  /       /  /  /  /_______/  /  /  /  CLI        /  /   / /")
        time.sleep(0.01)
        print("    /  /       /  /  / // // // // /  /  /_________    /  /____/ /")
        time.sleep(0.01)
        print("   /  /       /  /  /  /-------/  /  /  /_________/   /  /_____/ /")
        time.sleep(0.01)
        print("  /  /       /  /  /  /       /  /  /  /            /  /      / /")
        time.sleep(0.01)
        print(" /  /_______/  /  /  /______ /  /  /  /            /  /       / /")
        time.sleep(0.01)
        print("/__/_______/__/  /__/_______/__/  /__/            /__/        /__/")
        print(" ")
        print(" ")



    # Database section
    # Stock Records Master DB
    ssh = sqlite3.connect(r'DBFA_handler.db')
    ssh7 = ssh.cursor()
    #c.execute("DROP TABLE cust;")
    ssh7.execute("""CREATE TABLE IF NOT EXISTS sshandler
        (prodid INTEGER,
        prodname CHAR,
        ssstock INTEGER);""")
    ssh = sqlite3.connect('DBFA_handler.db')
    ssh7 = ssh.cursor()
    if os.path.exists(r'DBFA_handler.db'):
        pass
    else:
        ssh7.execute("""CREATE TABLE IF NOT EXISTS sshandler
            (prodid INTEGER,
            prodname CHAR,
            ssstock INTEGER);""")
```

```python


# Invoice Master DB
inmas = sqlite3.connect(r'invoicemaster.db')
inmascur = inmas.cursor()
#c.execute("DROP TABLE cust;")
inmascur.execute("""CREATE TABLE IF NOT EXISTS inmas
    (indid INTEGER);""")
inmas = sqlite3.connect('invoicemaster.db')
inmascur = inmas.cursor()
if os.path.exists(r'invoicemaster.db'):
    pass
else:
    inmascur.execute("""CREATE TABLE IF NOT EXISTS inmas
    (indid INTEGER);""")


# Voucher Records Master DB
isol = sqlite3.connect(r'cponmgmtsys.db')
isolx = isol.cursor()
isolx.execute("""CREATE TABLE IF NOT EXISTS cponmaster
    (cponid CHAR PRIMARY KEY,
    cponlim INTEGER,
    cponvalue INTEGER,
    cpondtb DATE);""")
isol = sqlite3.connect('cponmgmtsys.db')
isolx = isol.cursor()
if os.path.exists(r'cponmgmtsys.db'):
    pass
else:
    isolx.execute("""CREATE TABLE IF NOT EXISTS cponmaster
        (cponid CHAR PRIMARY KEY,
        cponlim INTEGER,
        cponvalue INTEGER,
        cpondtb DATE);""")


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



```

```python

    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, datax = getlisting()
        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]))
        if aaa >= qty:
            st = sqlite3.connect(r'DBFA_vend.db')
            stx = st.cursor()
            xstockq = """UPDATE stock SET ordqty = ordqty - ? WHERE prodid = ?"""
            xqans = (qty, prodid, )
            stx.execute(xstockq, xqans)
            xstockq = """UPDATE stock SET delivered = delivered + ? WHERE prodid = ?"""
            xqans = (qty, prodid, )
            stx.execute(xstockq, xqans)
            st.commit()
            print("Quantity ", qty, "recieved.")
        else:
            print("Cannot set status for qty:", qty, "for prod:", prodid, "as the ordered qty is lower than the delivered qty being set.")
        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 delstatmass():
        isol = sqlite3.connect(r'DBFA_vend.db')
        isolx = isol.cursor()
        print("\n\nProduct stock yet to be recieved: \n")
        isolx = isol.cursor()
        isolx.execute(("SELECT * from stock WHERE delstat = ?"), ("TBD", ))
        rows = isolx.fetchall()
```

```
        col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor", "Vendor", "Vendor Contact", "Lowstock Bar")
        table(col_labels, rows)
        print("\n\nProduct stock recieved: \n")
        isolx.execute(("SELECT * from stock WHERE delstat = ?"), ("DELIVERED", ))
        rows = isolx.fetchall()
        col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor", "Vendor", "Vendor Contact", "Lowstock Bar")
        table(col_labels, rows)

    def delstatindvl(prodid):
        isol = sqlite3.connect(r'DBFA_vend.db')
        isolx = isol.cursor()
        print("\n\nDetails for product ID", prodid, ": \n")
        isolx.execute(("SELECT * from stock WHERE prodid = ?"), (prodid, ))
        rows = isolx.fetchall()
        if rows in ([], (), "", " ", None):
            print("-------------------------------------------------------------")
            print("| Entered product ID could not be located in DBFA's records |")
            print("-------------------------------------------------------------\n\n")
        else:
            col_labels = ("P. ID", "P. Name", "Qty. Ordered", "Delivered","Vendor", "Vendor", "Vendor Contact", "Lowstock Bar")
            table(col_labels, rows)


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


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

```
                print("Invalid option entered. ")
                vendorcontact(prodid)


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


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


    # Report Stock Fetcher
    namiex, datax = getlisting()
    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-1)], "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()

    def repremupdate(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()
        namiex, datax = getlisting()
        for crrt in namiex:
            gg = (namiex.index(crrt)) + 1
            str = "insert into sshandler(prodid, prodname, ssstock) values(?, ?, ?)"
            strxx = (gg, crrt, inxstock,)
            ssh7.execute(str, strxx)
            ssh.commit()
        ssh7.close()
        time.sleep(1)
        print("DBFA QuickVend service - Stock universally enforced to", inxstock)
        time.sleep(1)


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

    # Stock Data Fetcher
```

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


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

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

    # Induvidual Stock Fetcher
    def ssxstockmaster(prodid):
        global ssxvarscheck
        ssxvarscheck = 0
        ssh = sqlite3.connect('DBFA_handler.db')
        ssh.row_factory = lambda cursor, row: row[0]
        ssh7 = ssh.cursor()
        csrr = ("SELECT ssstock FROM sshandler WHERE prodid = (?);")
        csrrxt = (prodid,)
        ssh7.execute(csrr, csrrxt)
        rows = ssh7.fetchall()
        # print(rows) #debug point
        values = ','.join(str(v) for v in rows)
        ssxdsccheck = "1 2 3 4"
        isolx = sqlite3.connect(r'DBFA_vend.db')
        isolxx = isolx.cursor()
        isolxx.execute(("SELECT lowstock from stock WHERE prodid = ?"), (prodid, ))
        rowsr = isolxx.fetchall()
        limiterx = []
```

```
        for ix in range (1, int((rowsr[0][0]))+1):
            limiterx.append(ix)
        if int(values) in limiterx:
            print("[Stock running out] Currently in stock: ", values, "pieces. Restock ASAP...")
            ssxvarscheck = 1
        elif int(values) == 0:
            print("Current product stock: ", values)
            ssxvarscheck = 2
        elif int(values) < 1:
            print("Current product stock: ", values)
            ssxvarscheck = 2
        else:
            print("Current product stock: ", values)
            ssxvarscheck = 1
        time.sleep(0.2)
        #toaster.show_toast("DBFA QuickVend Service - Background Sync", duration = 0.3427)




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

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


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

    # Voucher Issuer
```

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

    # Single Voucher Value Fetcher
    def cpon_valfetch(cponid):
        global valdock
        isol = sqlite3.connect('cponmgmtsys.db')
        isolx = isol.cursor()
        isol.row_factory = lambda cursor, row: row[0]
        csrr = ("SELECT cponvalue FROM cponmaster WHERE cponid = (?);")
        csrrxt = (cponid, )
        isolx.execute(csrr, csrrxt)
        rows = isolx.fetchall()
        values = ''.join(str(v) for v in rows)
        #print(values[1:-2])
        valdock = values[1:-2]
        print(valdock)

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

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




    # Customer System
```

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

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

    # 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 = ?"""
            indicator = (custid, )
            xrindicator = (pincrement, custid)
            pindicator = (points, custid)
            xbr7.execute(updatexr, indicator)
            xbr7.execute(updatexxr, xrindicator)
            xbr7.execute(updatexxxr, pindicator)
            xon.commit()
            xbr7.close()
        except sqlite3.Error as error:
            pass


    def pointfetch(custid):
        global lylpoints
        lylpoints = 0
        xon = sqlite3.connect('DBFA_CUSTCC.db')
        xbr7 = xon.cursor()
        findinx = "select points from custcc WHERE custid = ?"
        findinxx = (custid, )
        xbr7.execute(findinx, findinxx)
        arterxout = xbr7.fetchall()
        lylpoints = int((arterxout[0])[0])
```

```python


def pointmassfetch():
    xon = sqlite3.connect('DBFA_CUSTCC.db')
    xbr7 = xon.cursor()
    findinx = "select DISTINCT points from custcc"
    xbr7.execute(findinx)
    rows = xbr7.fetchall()
    for row in rows:
        print(row[0])

def pointsuse(custid, deduct):
    xon = sqlite3.connect('DBFA_CUSTCC.db')
    xbr7 = xon.cursor()
    updatexxxr = """UPDATE custcc SET points = points - ? WHERE custid = ?"""
    pindicator = (deduct, custid)
    xbr7.execute(updatexxxr, pindicator)
    xon.commit()
    xbr7.close()

def emailfetch(custid):
    global custmail
    con = sqlite3.connect(r'DBFA.db')
    conn = con.cursor()
    findinx = "select DISTINCT email from cust WHERE custt = ?"
    findinxx = (custid, )
    conn.execute(findinx, findinxx)
    rows = conn.fetchall()
    custmail = (rows[0][0])

global custcheckindic

global custt

def custcheck(custt):
    global cccheck, lylpoints
    if custt in ("", " ", 0, None , "0"):
        lylpoints = 0

    cccheck = 0
    custcheckindic = 0
    con = sqlite3.connect(r'DBFA.db')
    conn = con.cursor()
    conn.execute("SELECT custt FROM cust WHERE custt = ?", (custt,))
    data = conn.fetchall()
    if len(data)==0:
        custcheckindic = 0
        print("Customer", custt, "NOT found. ")
        print("- No customer selected -")
        custt = 0
        print("Using unregistered customer account")
        cccheck = 0
    else:
        ccustcheckindic = 1
        cccheck = 0
        pass


# Customer Validity Checker
def cust_listfetch(custid):
```

```
        clfetch = sqlite3.connect(r'DBFA_CUSTCC.db')
        clfetchx = clfetch.cursor()
        clfetchx.execute("SELECT custid FROM custcc")
        rows = clfetchx.fetchall()
        custyes = 1
        custno = 2
        custcount = 0
        for row in rows:
            row = row[0]
            if custid == row:
                custcount += 1
            else:
                pass
        if custcount == 1:
            return custyes
        else:
            return custno

    def saleslogger(custid, prodid, netpay):  #defining a function to input data into the SQL database's table
        sales = sqlite3.connect(r'dbfasales.db')
        salesx = sales.cursor()

        netprof = 0
        from datetime import date
        datex = date.today()

        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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        for i in prodid:
            netprof += profer[int(i)]

        salesx.execute("SELECT MAX(sno) FROM SALES")
        sno = (int(salesx.fetchall()[0][0]) + 1)

        prodidxs = ""
        for i in prodid:
            prodidxs += '%s'%i + ", "

        str = "insert into sales(sno, custid, prodid, net, prof, date) values(?, ?, ?, ?, ?, ?)"
        io = (sno, custid, prodidxs, netpay, netprof, datex)
        salesx.execute(str, io)
        sales.commit()
        print("Sales activity logged. ")


    def salesdatefetch():  #defining a function to input data into the SQL database's table
        from datetime import date
        import datetime
        sales = sqlite3.connect(r'dbfasales.db')
        salesx = sales.cursor()
        salesx.execute("SELECT prof FROM sales WHERE date BETWEEN datetime('now', '-6 days') AND datetime('now', 'localtime')")
        sumer = 0
        for i in salesx.fetchall():
            sumer += int((i[0]))
```

```
        return sumer

    def salestodayfetch():  #defining a function to input data into the SQL database's table
        from datetime import date
        import datetime
        sales = sqlite3.connect(r'dbfasales.db')
        salesx = sales.cursor()
        salesx.execute("SELECT prof FROM sales WHERE date = ?", (date.today(), ))
        sumerx = 0
        for i in salesx.fetchall():
            sumerx += int((i[0]))
        return sumerx


    os.system('cls')



    def floodscreen():
        image = cv2.imread("imagepx.png")
        cv2.imshow("Loading.... ", image)
        cv2.waitKey(2000)
        cv2.destroyAllWindows()


    def floodpay():
        import cv2
        image = cv2.imread("qr-code.png")
        cv2.imshow("Pay With UPI", image)
        cv2.waitKey(150000)
        cv2.destroyAllWindows()


    # Main Menu
    # New Main Menu
    def mainmenu(): #defining a function for the main menu
        import sqlite3, time, random, requests
        start = time.time()
        from datetime import datetime, timedelta

        from datetime import datetime, timedelta
        now = datetime.now()
        now = now + timedelta(days=1)
        dt_string = now.strftime("%Y-%m-%d")
        empmas = sqlite3.connect(r'dbfaempmaster.db')
        empmascur = empmas.cursor()
        empmascur.execute("SELECT MAX(Date) FROM scheddelivery")
        datie = empmascur.fetchall()[0][0]
        if datie != dt_string:
            def telegram_bot_sendtext(bot_message):
                bot_token = str(securedatafetch(2))
                bot_chatID = str(securedatafetch(3))
                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()

            today = datetime.now().date()
            start = today + timedelta(days=1)
```

```python

            def leaveget(Oid):
                today = datetime.now().date()
                empmas = sqlite3.connect(r'dbfaempmaster.db')
                empmascur = empmas.cursor()
                empmascur.execute("SELECT * FROM leave WHERE Oid = ? AND Date = ?", ('%s'%Oid, '%s'%start, ))
                datastream = empmascur.fetchall()
                if len(datastream) != 0:
                    return(datastream[0][0])
                else:
                    return None

            def gethoursavg(Oid):
                empmas = sqlite3.connect(r'dbfaempmaster.db')
                empmascur = empmas.cursor()
                empmascur.execute("SELECT * FROM attendance WHERE OiD = ? ORDER BY Date ASC, Time", ('%s'%Oid,))
                count = 0
                rows = empmascur.fetchall()
                datelist = []
                for i in rows:
                    datelist.append(i[0])
                a = [i for i in rows if datelist.count(i[0])>1]
                netr = (len(datelist) - len(a)) * 8
                ini_list = [i[2] for i in a]
                from datetime import timedelta
                diff_list = []
                for x, y in zip(ini_list[0::], ini_list[1::]):
                    t1 = str((timedelta(hours=int(y.split(':')[0]), minutes=int(y.split(':')[1])) - timedelta(hours=int(x.split(':')[0]), minutes=int(x.split(':')[1]))).split(':')[0]
                    diff_list.append(t1)
                del diff_list[1::2]
                for i in range(0, len(diff_list)):
                    diff_list[i] = int(diff_list[i])
                if int(netr) > 0:
                    diff_list.append(netr)
                return (sum(diff_list)/len(diff_list))


            def autospacer(word):
                #print(((17-len(word))//2))
                if len(word) % 2 == 0:
                    return (((17-len(word))//2)*" " + str(word) + ((17-len(word))//2)*" " + " ")
                else:
                    #print(word)
                    return ((17-len(word))//2)*" " + str(word) + ((17-len(word))//2)*" "


            def getschedvals(Oid):
                empmas = sqlite3.connect(r'dbfaempmaster.db')
                empmascur = empmas.cursor()
                empmascur.execute("SELECT Name, Post FROM emp WHERE OiD = ?", ('%s'%Oid,))
                daysrt = empmascur.fetchall()
                return daysrt[0][0]

```

```python

            def getSales():
                empmas = sqlite3.connect(r'dbfaempmaster.db')
                empmascur = empmas.cursor()
                empmascur.execute("SELECT OiD FROM emp WHERE Post = ?", ('Sales',))
                daysrt = empmascur.fetchall()
                return [daysrt[0][0], daysrt[1][0],  daysrt[2][0]]

            def getMaintanence():
                empmas = sqlite3.connect(r'dbfaempmaster.db')
                empmascur = empmas.cursor()
                empmascur.execute("SELECT OiD FROM emp WHERE Post = ?", ('Maintanence',))
                daysrt = empmascur.fetchall()
                return [daysrt[0][0], daysrt[1][0]]

            def getLogistics():
                empmas = sqlite3.connect(r'dbfaempmaster.db')
                empmascur = empmas.cursor()
                empmascur.execute("SELECT OiD FROM emp WHERE Post = ?", ('Logistics',))
                daysrt = empmascur.fetchall()
                return [daysrt[0][0], daysrt[1][0]]

            def timex(Oid):
                if gethoursavg(1) < 8:
                    return ((8-gethoursavg(1))/10)*600
                else:
                    return 0

            rand = (random.randint(1, 4))

            if rand == 1:
                if leaveget(getSales()[0]) is None:
                    r11 = getschedvals(getSales()[0])
                else:
                    r11 = getschedvals(getSales()[2])+ " & " +getschedvals(getSales()[1])
                t11 = str((int(1400+timex(3))))

                if leaveget(getSales()[1]) is None:
                    r12 = getschedvals(getSales()[1])
                else:
                    r12 = getschedvals(getSales()[0]) + " & " + getschedvals(getSales()[2])
                t12 = str(int(2200+timex(4)))

                if leaveget(getSales()[2]) is None:
                    r13 = getschedvals(getSales()[2])
                else:
                    r13 = getschedvals(getSales()[1]) + " & " + getschedvals(getSales()[0])
                t13 = str("0"+str(int(600+timex(5))))

                if leaveget(getMaintanence()[0]) is None:
                    r21 = getschedvals(getMaintanence()[0])
                else:
                    r21 = getschedvals(getMaintanence()[1])
                t21 = str((int(1400+timex(6))))

```

```
                if leaveget(getMaintanence()[1]) is None:
                    r22 = getschedvals(getMaintanence()[1])
                else:
                    r22 = getschedvals(getMaintanence()[0])
                t22 = str(int(2200+timex(7)))

                if leaveget(getLogistics()[0]) is None:
                    r31 = getschedvals(getLogistics()[0])
                else:
                    r31 = getschedvals(getLogistics()[1])
                t31 = str((int(1400+timex(8))))

                if leaveget(getLogistics()[1]) is None:
                    r33 = getschedvals(getLogistics()[1])
                else:
                    r31 = getschedvals(getLogistics()[0])
                t33 = str("0"+str((int(600+timex(9)))))


            if rand == 2:
                if leaveget(getSales()[2]) is None:
                    r11 = getschedvals(getSales()[2])
                else:
                    r11 = getschedvals(getSales()[0])+ " & " +getschedvals(getSales()[1])
                t11 = str((int(1400+timex(3))))

                if leaveget(getSales()[1]) is None:
                    r12 = getschedvals(getSales()[1])
                else:
                    r12 = getschedvals(getSales()[0]) + " & " + getschedvals(getSales()[2])
                t12 = str(int(2200+timex(4)))

                if leaveget(getSales()[0]) is None:
                    r13 = getschedvals(getSales()[0])
                else:
                    r13 = getschedvals(getSales()[2]) + " & " + getschedvals(getSales()[1])
                t13 = str("0"+str(int(600+timex(5))))

                if leaveget(getMaintanence()[1]) is None:
                    r21 = getschedvals(getMaintanence()[1])
                else:
                    r21 = getschedvals(getMaintanence()[0])
                t21 = str((int(1400+timex(6))))

                if leaveget(getMaintanence()[0]) is None:
                    r22 = getschedvals(getMaintanence()[0])
                else:
                    r22 = getschedvals(getMaintanence()[1])
                t22 = str(int(2200+timex(7)))

                if leaveget(getLogistics()[0]) is None:
                    r31 = getschedvals(getLogistics()[0])
                else:
                    r31 = getschedvals(getLogistics()[1])
                t31 = str((int(1400+timex(8))))

                if leaveget(getLogistics()[1]) is None:
```

```
                    r33 = getschedvals(getLogistics()[1])
                else:
                    r31 = getschedvals(getLogistics()[0])
                t33 = str("0"+str((int(600+timex(9)))))


            if rand == 3:
                if leaveget(getSales()[1]) is None:
                    r11 = getschedvals(getSales()[1])
                else:
                    r11 = getschedvals(getSales()[0])+ " & " +getschedvals(getSales()[2])
                t11 = str((int(1400+timex(3))))

                if leaveget(getSales()[0]) is None:
                    r12 = getschedvals(getSales()[0])
                else:
                    r12 = getschedvals(getSales()[2]) + " & " + getschedvals(getSales()[1])
                t12 = str(int(2200+timex(4)))

                if leaveget(getSales()[2]) is None:
                    r13 = getschedvals(getSales()[2])
                else:
                    r13 = getschedvals(getSales()[1]) + " & " + getschedvals(getSales()[0])
                t13 = str("0"+str(int(600+timex(5))))

                if leaveget(getMaintanence()[1]) is None:
                    r21 = getschedvals(getMaintanence()[1])
                else:
                    r21 = getschedvals(getMaintanence()[0])
                t21 = str((int(1400+timex(6))))

                if leaveget(getMaintanence()[0]) is None:
                    r22 = getschedvals(getMaintanence()[0])
                else:
                    r22 = getschedvals(getMaintanence()[1])
                t22 = str(int(2200+timex(7)))

                if leaveget(getLogistics()[1]) is None:
                    r31 = getschedvals(getLogistics()[1])
                else:
                    r31 = getschedvals(getLogistics()[0])
                t31 = str((int(1400+timex(8))))

                if leaveget(getLogistics()[0]) is None:
                    r33 = getschedvals(getLogistics()[0])
                else:
                    r31 = getschedvals(getLogistics()[1])
                t33 = str("0"+str((int(600+timex(9)))))


            if rand == 4:
                if leaveget(getSales()[2]) is None:
                    r11 = getschedvals(getSales()[2])
                else:
                    r11 = getschedvals(getSales()[0])+ " & " +getschedvals(getSales()[1])
                t11 = str((int(1400+timex(3))))
```

```
                if leaveget(getSales()[0]) is None:
                    r12 = getschedvals(getSales()[0])
                else:
                    r12 = getschedvals(getSales()[1]) + " & " + getschedvals(getSales()[2])
                t12 = str(int(2200+timex(4)))

                if leaveget(getSales()[1]) is None:
                    r13 = getschedvals(getSales()[1])
                else:
                    r13 = getschedvals(getSales()[2]) + " & " + getschedvals(getSales()[0])
                t13 = str("0"+str(int(600+timex(5))))

                if leaveget(getMaintanence()[0]) is None:
                    r21 = getschedvals(getMaintanence()[0])
                else:
                    r21 = getschedvals(getMaintanence()[1])
                t21 = str((int(1400+timex(6))))

                if leaveget(getMaintanence()[1]) is None:
                    r22 = getschedvals(getMaintanence()[1])
                else:
                    r22 = getschedvals(getMaintanence()[0])
                t22 = str(int(2200+timex(7)))

                if leaveget(getLogistics()[1]) is None:
                    r31 = getschedvals(getLogistics()[1])
                else:
                    r31 = getschedvals(getLogistics()[0])
                t31 = str((int(1400+timex(8))))

                if leaveget(getLogistics()[0]) is None:
                    r33 = getschedvals(getLogistics()[0])
                else:
                    r31 = getschedvals(getLogistics()[1])
                t33 = str("0"+str((int(600+timex(9)))))

            sched = ('''                Schedule for ''' + str(start) + '''
delta           ╔═══════════════╦═══════════════╦═══════════════╗
Scheduler       ╠   0600 - '''+t11+'''   ╠   1400 - '''+t12+'''   ╠   2200 - '''+t13+''' ╠
═══════════     ╠═══════════════╬═══════════════╬═══════════════╣
SALES
╠'''+autospacer(r11)+'''╠'''+autospacer(r12)+'''╠'''+autospacer(r13)+'''╠
                ╚═══════════════╩═══════════════╩═══════════════╝


delta           ╔═══════════════╦═══════════════╦═══════════════╗

Scheduler       ╠   0600 - '''+t21+'''   ╠   1400 - '''+t22+'''   ╠   2200 - 0600 ╠
═══════════     ╠═══════════════╬═══════════════╬═══════════════╣
MAINTANENCE     ╠'''+autospacer(r21)+'''╠'''+autospacer(r22)+'''╠ ═ ═ ╠ ═ ═ ═ ═ ═ ╠
                ╚═══════════════╩═══════════════╩═══════════════╝


delta           ╔═══════════════╦═══════════════╦═══════════════╗
```

```
Scheduler        0600 - '''+t31+'''    1400 - 2200    2200 - '''+t33+'''

LOGISTICS        '''+autospacer(r31)+'''          '''+autospacer(r33)+'''

            ''')

            print("Tomorrow's shift schedule:: \n")
            bot_message = '''DBFA Automated Scheduler
Schedule for tomorrow (''' + str(start) + ''')

Sales Staff
0600 - '''+t11+''': '''+r11+'''
1400 - '''+t12+''': '''+r12+'''
2200 - '''+t13+''': '''+r13+'''

Maintanence Staff
0600 - '''+t21+''': '''+r21+'''
1400 - '''+t22+''': '''+r22+'''

Logistics Staff
0600 - '''+t31+''': '''+r31+'''
2200 - '''+t33+''': '''+r33+'''

You are recieving this shift schedule as your store is serviced by DBFA.

This is a dynamically generated schedule with alternating shifts. Employees on leave are taken care of and adjusted accordingly.'''

            from datetime import datetime, timedelta
            now = datetime.now()
            now = now + timedelta(days=1)
            dt_string = now.strftime("%Y-%m-%d")
            empmas = sqlite3.connect(r'dbfaempmaster.db')
            empmascur = empmas.cursor()
            empmascur.execute("INSERT INTO scheddelivery(Date) VALUES (?)", (dt_string,))
            empmas.commit()

            telegram_bot_grouptext(bot_message.replace("&", "and"))
            time.sleep(0.5)
            telegram_bot_sendtext(bot_message.replace("&", "and"))
            print(sched)
            contfac = input("Enter a button to continue ~ : ")
            with open('lastsched.txt', 'a+') as file:
                file.close()
            with open('lastsched.txt', 'w+', encoding="utf-8") as file:
                file.truncate(0)
                file.write(sched)
                file.close()

        else:
            pass


        dmain()

        from datetime import datetime, date
```

```
        now = datetime.now()
        dt_string = now.strftime("%d")  #datetime object containing current date and time
        #print(dt_string)
        if dt_string in ("01", "02", "03", "04", "05"):
            print("Salary days: Pay salaries between 01st - 05th of every month. Open DBFA Employee Manager to pay ~")
        from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
        init(convert = True)
        url = "https://raw.githubusercontent.com/deltaonealpha/DBFA_UpdateHandler/master/updates.txt"
        r = requests.get(url)
        dbfaver = ((str(r.content.decode('utf-8'))))[4:]
        xdbfaver = ((str(r.content.decode('utf-8'))))
        with open(str(parentdir+'\\updates.txt'), 'r+') as upread:
            upread = (str(upread.read())).strip()
        #print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
        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:
            if line not in ("", " ", (), [], None, "\n"):
                delcount+=1
        filedel.close()
        pro7d = salesdatefetch()
        protd = salestodayfetch()
        logoxold = (Fore.CYAN+'''
                                Options:
 ▄▄▄
```

```
CLIENT Inception 1.0                a: Register a Customer    c: Purchase Records      6  - DBFA Backup & Switch
'''+Fore.MAGENTA+'''  The OG Store Manager'''+Fore.CYAN+'''              b: Customer Registry      d: Find a Customer         7  - Analyse Sales
                                    e: Export data as CSV                                                                  8  - About DBFA
                          3  - Store Options:
   '''+Fore.MAGENTA+'''emp/EMP - DBFA Employee Manager'''+Fore.CYAN+'''
                                          a: Manage Stock           c: Manage Vouchers         9 - Check for updates
                                    b: DBFA Stock Master      d: Product Listing      10 - DBFA Options
                                    e: Sales Log              f: Export data as CSV      11 - DBFA Security Dashboard
                                    g: Invoice Deep Archive                                                                12 - Quit
- 'mark'/'MARK': to mark attendance

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


        logoxnew = (Fore.CYAN+'''Options:
1  - Issue a Bill                                          4  - Store Report
2  - Manage Customers:                                     5  - Manage Deliveries
        a: Register a Customer    c: Purchase Records      6  - DBFA Backup & Switch
        b: Customer Registry      d: Find a Customer       7  - Analyse Sales
        e: Export data as CSV                              8  - About DBFA
3  - Store Options:
'''+Fore.MAGENTA+'''emp/EMP - DBFA Employee Manager'''+Fore.CYAN+'''
        a: Manage Stock           c: Manage Vouchers       9  - Check for updates
        b: DBFA Stock Master      d: Product Listing       10 - DBFA Options
        e: Sales Log              f: Export data as CSV    11 - DBFA Security Dashboard
        g: Invoice Deep Archive                            12 - Quit
- 'mark'/'MARK': to mark attendance
'''+Fore.MAGENTA+'''
What would you like to do?                    The OG Store Manager'''+Fore.WHITE+''' ▄▄ ▄▄ ▄▄ ▄▄ ▄▄'''+Fore.CYAN+'''
=================================================================''' +Fore.WHITE+'''▄▄ ▄▄ ▄▄ ▄▄ ▄▄ ▄▄'''+Fore.CYAN+'''
DBFA Music Controls: *prev* <<< | *pause* <|> | *next* >>>             INCEPTION 1.0 CLIENT   \n'''
+Back.CYAN+Fore.BLACK+'''Administrator Controls'''+Back.BLACK+Fore.CYAN+'''

=================================================================================================''')
#  '''+Fore.RED+'''
#  '''+Fore.CYAN+'''
        logoxnewrestrictedperms = (Fore.CYAN+'''Options:
1  - Issue a Bill
2  - Manage Customers:                                     5  - Manage Deliveries
        a: Register a Customer    c: Purchase Records
```

```
        b: Customer Registry      d: Find a Customer
                                                                8  - About DBFA
3  - Store Options:
'''+Fore.MAGENTA+'''emp/EMP - DBFA Employee Manager'''+Fore.CYAN+'''
        a: Manage Stock
        b: DBFA Stock Master      d: Product Listing

                                                                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+'''▐▌▐▌▐▌
```

```
            print(logoxold)
        #Settings Checker
        if settingscommonfetch(3) == 1:
            time.sleep(0.2)
            try:
                if spotify.current() not in ("", " ", [], (), None):
                    print("Currently playing:", Fore.MAGENTA , spotify.current()[0], Fore.CYAN, "by ", Fore.MAGENTA, spotify.current()[1], Fore.CYAN)
                else:
                    print(Fore.MAGENTA, "No music playing. Use Spotify to play your favourite music and control it via DBFA", Fore.CYAN)
            except Exception as e:
                print(Fore.MAGENTA, "No music playing. Play your favourite music and control it via DBFA", Fore.CYAN)
            if settingscommonfetch(7) == 1:
 print("==========================================================================================", Fore.MAGENTA)
            else:
                print("----------------------------------------------------------------------------------------------------", Fore.MAGENTA)
        else:
 print("==========================================================================================")
            print("Re-enable DBFA Music Controls Service from the settings to be able to control your music ")
 print("==========================================================================================")

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

    global netpay

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

```
        if paycheck == "1":
            payindic = "Paid with credit/ debit Card"
        elif paycheck == "2":
            payindic = "Paid with a digital wallet"
        elif paycheck == "3":
            payindic = "Paid with UPI"
            floodpay()
        elif paycheck == "4":
            payindic = "Paid with cash"
        elif paycheck == "exit":
            print("Cancelling this billing cycle")
            xrt = 1
        else:
            xpayboxie()

    global netpay

    # Payments Handler
    def payboxie(custid, total):
        global custcheckindic
        if custt not in (0, "0", "", " ", None) and cccheck == 0:
            command = "cls"
            os.system(command)
            global payindic, netpay, redeemindic, lylpoints
            xrt = 0
            redeemindic = 0
            payindic = 0
            from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
            print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
            init(convert = True)
            print("Amount to be paid: ₹","%.2f" % total)
            print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
            pointfetch(custid)
            if lylpoints != 0:
                print("You have loyalty points: ", lylpoints, "worth: ₹", lylpoints)
                time.sleep(0.5)
                pointscheck = input("Use " + '%s'%total + " points (y/n)? ")
                print(Fore.LIGHTBLUE_EX + "-----------------" + Fore.WHITE)
                if pointscheck == "y":
                    if total > lylpoints:
                        redeemam = lylpoints
                    else:
                        redeemam = total
                    getOTP()
                    emailfetch(custid)
                    print("Please wait..")
                    email = str(securedatafetch(5))
                    password = str(securedatafetch(1))
                    send_to_email = custmail
                    subject = 'Redeem your DBFA loyalty points'
                    messageHTML = '''
                    <h1><span style="color: #496dd0">Redeem your DBFA loyalty points?</span></h1>
                    <h6> </h6>
                    <h4>You are recieving this mail as we've recieved a request to use the loyalty points on your account for a purchase with DBFA. If this isn't you, contact support at the earrliest.</h4>
                    <h6> </h6>
```

```
                    <h4>One-time password: <span style="color: #496dd0">''' + "%s"%otp + '''</span></h4>
                    <h6> </h6>
                    <h6>Points in your account: ''' + "%s"%lylpoints + '''</h6>
                    <h6>Points that will be redeemed: ''' + "%s"%redeemam + '''</h6>
                    <h6>Each DBFA point is worth ₹1</h6>
                    <h6>This purchase will give you: ''' + "%s"%((billiemaster/100)*1) + ''' points</h6>
                    <h6></h6>
                    <h6></h6>
                    <h6> </h6>
                    <h6> </h6>
                    <h6> </h6>
                    <h5>When you share this OTP with a DBFA-store, you authorize us to use your loyalty points to discount your purchase.</h5>
                    <h6>Do not share this OTP with any third party.</h6>
                    <h6>DBFA or any of its affiliates will not be resposible in anyway for any implications this OTP or any part of it might have on anyone in any way.</h6>
                    <h6>Offering these reward points does not benefit DBFA in any immediate manner.</h6>
                    ''')
                    messagePlain = 'DBFA Security'
                    msg = MIMEMultipart('alternative')
                    msg['From'] = email
                    msg['To'] = send_to_email
                    msg['Subject'] = subject
                    # Attach both plain and HTML versions
                    msg.attach(MIMEText(messagePlain, 'plain'))
                    msg.attach(MIMEText(messageHTML, 'html'))
                    os.startfile(r'Process_Handlers\\emailprocesswindow.pyw')
                    server = smtplib.SMTP('smtp.gmail.com', 587)
                    server.starttls()
                    server.login(email, password)
                    text = msg.as_string()
                    server.sendmail(email, send_to_email, text)
                    server.quit()
                    otpverif = input("Enter the OTP recieved on " + "%s"%custmail + ": ")
                    if otpverif == otp:
                        if total > lylpoints:
                            total = total - lylpoints
                            pointsuse(custid, lylpoints)
                            print("New total: ", total)
                            time.sleep(2)
                            redeemindic = 1
                            netpay = total
                            print("Points redeemed worth: ", lylpoints)
                            return total

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

```

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

        else:
            redeemindic = 0
            netpay = total
            redeemindic = 0
            lylpoints = 0
            netpay = total
            return total


    global custname, email, idd

    def del2a():
        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("Connecting to server..")  #SQL connection prompt
        time.sleep(0.4)  #for a seamless experience
        #conn.execute("select * from cust")
        #takes values from the SQL database
```

```
        conn = sqlite3.connect('DBFA.db')
        cursor = conn.cursor()
        cursor.execute("select * from cust")
        results = cursor.fetchall()
        idd = len(results)+1
        print("Registering customer with ID: ", idd)
        custname = input("Customer Name: ")
        email = input("Customer's E-mail ID: ")
        inserter(idd, custname, email) #argumental function to insert values into the SQL database
        nullvalue = 0
        custcc(idd, custname, nullvalue, nullvalue)
        print(" ")

        print("Customer ID", idd, "registered in directory.")
        print("--------------------------------------")
        print(" ")
        print(" ")
        #conn.close()
        time.sleep(1) #for a seamless experience


    def del2b():
        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()
        print("Connecting to server..")  #SQL connection prompt
        time.sleep(0.7) #for a seamless experience
        print("Registered customers are: ")
        #Re-writing to refresh connection
        conn = sqlite3.connect('DBFA.db')
        cur = conn.cursor()
        cur.execute("SELECT * FROM cust")
        rows = cur.fetchall()
        col_labels = ("ID", "Customer NAME", "EMAIL")
        table(col_labels, rows)
        toaster.show_toast("DNSS QuickSync", "Database acessed", duration = 2)
        #takes values from the SQL database
        '''
        while row is not None:
            print(row)
            #row = conn.fetchone()
        '''

        conn.close()
        conn.close()
        print()
        print()
        time.sleep(2) #delay for easy-table viewing


    def del2c():
        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
        xon = sqlite3.connect(r'DBFA_CUSTCC.db')
        xbr7 = xon.cursor()
        xbr7.execute("SELECT * FROM custcc")
        l = xbr7.fetchall()
        print("\nCustomer Purchase Records: ")

        import pandas as pd

        flat_list = []
        print("----------------------------------------------------------------")
        for sublist in l:
            flat_list.append(sublist)
        mydf = pd.DataFrame(flat_list, columns=['Customer ID', 'Name', 'Purchases Made', 'Total', 'Loyalty Points'])
        mydf.pivot(index='Customer ID', columns='Purchases Made', values='Total').fillna(value='-')
        print(mydf)
        print("----------------------------------------------------------------")
        time.sleep(2)

        '''
        for row in rows:
            print(row)
            print(" ")
        '''
        xbr7.close()
        toaster.show_toast("DFBA QuickSync", "Database acessed", duration = 0.5)



    def del2d():
        try:
            con = sqlite3.connect(r'DBFA.db')
            conn = con.cursor()

            conx = sqlite3.connect(r'DBFA_CUSTCC.db')
            connx = conx.cursor()

            searchcon = str(input("Customer Name: "))
            if " " in searchcon:
                for i in searchcon:
                    sconsplit = searchcon.split(" ")
                    for j in sconsplit:
                        conn.execute("SELECT custt FROM cust WHERE custname LIKE ?", (("%"+searchcon+"%"), ))
                        searchdata = conn.fetchall()
                    else:
                        searchcon = searchcon.replace(" ", "")
                        conn.execute("SELECT custt FROM cust WHERE custname LIKE ?", (("%"+searchcon+"%"), ))
                        searchdata = conn.fetchall()
                        if len(searchdata) != 0:
                            pass
                        else:
                            searchdata = "No such customer found."
            else:
```

```
                conn.execute("SELECT custt FROM cust WHERE custname LIKE ?", (("%"+searchcon+"%"), ))
                searchdata = conn.fetchall()

            if len(searchdata) != 0:
                if len(searchdata) > 1:
                    for i in searchdata:
                        conn.execute("SELECT * FROM cust WHERE custt = ?", (i[0], ))
                        custdata = conn.fetchall()
                        #col_labels = ("ID", "Customer NAME", "EMAIL")
                        #table(col_labels, custdata)

                        connx.execute("SELECT * FROM custcc WHERE custid = ?", (i[0], ))
                        custdatax = connx.fetchall()
                        ccrt = []
                        for jk in custdata:
                            for jkx in jk:
                                ccrt.append(str(jkx))
                        for jk in custdatax:
                            for jkx in jk:
                                ccrt.append(str(jkx))

                        col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name', 'Purchases Made', 'Total', 'Loyalty Points')
                        print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f", tablefmt='fancy_grid'))

                        print(" ")
                else:
                    conn.execute("SELECT * FROM cust WHERE custt = ?", (searchdata[0][0], ))
                    custdata = conn.fetchall()

                    connx.execute("SELECT * FROM custcc WHERE custid = ?", (searchdata[0][0], ))
                    custdatax = connx.fetchall()
                    ccrt = []
                    for jk in custdata:
                        for jkx in jk:
                            ccrt.append(str(jkx))
                    for jk in custdatax:
                        for jkx in jk:
                            ccrt.append(str(jkx))

                    col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name', 'Purchases Made', 'Total', 'Loyalty Points')
                    print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f", tablefmt='fancy_grid'))
            else:
                srtx = "%"
                for i in searchcon:
                    srtx += '%s'%i+"%"
                conn.execute("SELECT custt FROM cust WHERE custname LIKE ?", ((srtx), ))
                searchdata = conn.fetchall()
                if len(searchdata) != 0:
                    conn.execute("SELECT * FROM cust WHERE custt = ?", (i[0], ))
                    custdata = conn.fetchall()
                    #col_labels = ("ID", "Customer NAME", "EMAIL")
```

```
                    #table(col_labels, custdata)

                    connx.execute("SELECT * FROM custcc WHERE custid = ?", (i[0], ))
                    custdatax = connx.fetchall()
                    ccrt = []
                    for jk in custdata:
                        for jkx in jk:
                            ccrt.append(str(jkx))
                    for jk in custdatax:
                        for jkx in jk:
                            ccrt.append(str(jkx))

                    col_labels = ('ID', 'Customer NAME', 'EMAIL', 'ID', 'Name', 'Purchases Made', 'Total', 'Loyalty Points')
                    print(tabulate(zip(col_labels, ccrt), floatfmt = ".4f", tablefmt='fancy_grid'))
                else:
                    print("Customer not found.")
        except:
            print("Error encountered.  ")




    def del2e():
        import sqlite3 as sql
        import os
        import csv
        from sqlite3 import Error

        try:
            csvex=sql.connect(currdir+'\\DBFA.db')
            cursor = csvex.cursor()
            cursor.execute("select * from cust")
            print("Fetching data from database - I...")
            with open("DBFAcustrec.csv", "w") as csv_file:
                csv_writer = csv.writer(csv_file)
                csv_writer.writerow([i[0] for i in cursor.description])
                csv_writer.writerows(cursor)
                csvexx=sql.connect(currdir+'\\DBFA_CUSTCC.db')
                print("Fetching data from database - II...")
                cursorx = csvexx.cursor()
                cursorx.execute("select * from custcc")
                csv_writer = csv.writer(csv_file)
                csv_writer.writerow([i[0] for i in cursorx.description])
                csv_writer.writerows(cursorx)
            dirpath = os.getcwd() + "/DBFAcustrec.csv"
            print("Data exported Successfully into {}".format(dirpath))
            time.sleep(2)

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

        except Error as e:
            print(e)

        finally:
```

```
            csvex.close()



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


    def del3b():
        from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
        init(convert = True)
        time.sleep(1)
        print(Fore.MAGENTA+'''
==============================================================================
Options:                                Store Options >>> DBFA Stock Master v1
==============================================================================
    a: Order New Stock
    b: Update Delivery Status
    c: MASS - Fetch Current Status
    d: INDVL - Fetch Current Status
    e: View Vendor Details
    f: Contact Vendor
    g: Edit Vendor Contact
    h: Modify Low-Stock Warning Bar
    i: Add new product listing ~~

What would you like to do?            '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
=================================  '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
'''+Fore.CYAN+'''Stock Master ~'''+Fore.MAGENTA+'''
=============================================================================='''+Fore.WHITE)
        stkmaster = input("Select:: ")
        if stkmaster in ("i", "I"):
            prodn = input("Enter new product name: ")
            costn = input("Enter the price to sell " + str(prodn) + " for (EXCLUDING TAX): ")
            categn = input("Enter category for product (refer docs for help): ")
            profn = input("Enter the amount of profit earned (included in price) for this product: ")
            conffac = input("\nList new product with name " + str(prodn) + " for (excluding tax) price ₹" + str(costn) + " under category " + str(categn) + "? (y/n): ")
            if conffac in ("y", "Y"):
                print(". . .")
                import sqlite3
                settings = sqlite3.connect(r'dbfasettings.db')
                settingsx = settings.cursor()
                settingsx.execute("SELECT MAX(PID) FROM listing")
```

```
                maxpid = settingsx.fetchall()[0][0]+1
                settingsx.execute("INSERT INTO listing(PID, Name, Cost, Category) VALUES (?, ?, ?, ?)", (maxpid, prodn, costn, categn))
                settings.commit()
                isolx = sqlite3.connect(r'DBFA_vend.db')
                isolxx = isolx.cursor()
                isolxx.execute("INSERT INTO stock(prodid, prodname, ordqty, delivered, delstat, vendor, vendcont, lowstock) VALUES(?, ?, ?, ?, ?, ?, ?, ?)", (maxpid, prodn, 0, 0, "DELIVERED", "deltaSystems", "vendor@dbfa.com", 5))
                isolx.commit()
                ssh = sqlite3.connect('DBFA_handler.db')
                ssh7 = ssh.cursor()
                ssh7.execute("INSERT INTO sshandler(prodid, prodname, ssstock) VALUES(?, ?, ?)", (maxpid, prodn, 0))
                ssh.commit()
                netprof = sqlite3.connect('recmaster.db')
                netprofx = netprof.cursor()
                netprofx.execute("INSERT INTO recmasterx(prodid, prodname, prodprofit, prodsales, netprof) VALUES (?, ?, ?, ?, ?)", (maxpid, prodn, profn, 0, 0))
                netprof.commit()
                settingsx.execute("SELECT * FROM listing WHERE PID = ?", (str(maxpid), ))
                print("Product listed: ", settingsx.fetchall())
            else:
                print("New listing cancelled! ~ ")


        if stkmaster in ("a", "A"):
            idquery = int(input("Enter the product ID: "))
            qtyquery = int(input("Enter the amount to order: "))
            orderstock(idquery, qtyquery)
            print("\n-------------------------------------------------------")
        elif stkmaster in ("b", "B"):
            idquery = int(input("Product ID of the product recieved: "))
            qtyquery = int(input("Quantity recieved: "))
            delivered(qtyquery, idquery)
            ssxupdatescript(qtyquery, idquery)
            print("\nChanges made in all related databases. \n")
            print("\n-------------------------------------------------------")
        elif stkmaster in ("c", "C"):
            delstatmass()
            print("\n-------------------------------------------------------")
        elif stkmaster in ("d", "D"):
            idquery = int(input("Product ID to locate: "))
            delstatindvl(idquery)
            print("\n-------------------------------------------------------")
        elif stkmaster in ("e", "E"):
            idquery = int(input("Product ID to get vendor details for: "))
            vendorfetch(idquery)
            print("\n-------------------------------------------------------")
        elif stkmaster in ("f", "F"):
            idquery = int(input("Product ID to contact vendor for: "))
            vendorcontact(idquery)
            print("\n-------------------------------------------------------")
        elif stkmaster in ("g", "G"):
            print("-- Change Vendor Contact --")
            time.sleep(0.5)
            try:
```

```
                prodidvendc = int(input("Enter product ID to change vendor contact for: "))
                if prodidvendc > 40:
                    print("Please enter a valid product ID")
                    prodidvendc = input("Enter product ID to change vendor contact for: ")

            except:
                print("Please enter a valid product ID")
                time.sleep(0.3)
                prodidvendc = input("Enter product ID to change vendor contact for: ")
                if prodidvendc > 40:
                    print("Please enter a valid product ID")
                    prodidvendc = input("Enter product ID to change vendor contact for: ")
            while(1):
                vendorchange = input("Enter the new E-Mail ID: ")
                if "@" in vendorchange:
                    if "." in vendorchange:
                        st = sqlite3.connect(r'DBFA_vend.db')
                        stx = st.cursor()
                        stx.execute("""UPDATE stock SET vendcont = ? WHERE prodid = ?""", (vendorchange, prodidvendc,))
                        st.commit()
                        print("Vendor contact changed for product ID: ", prodidvendc)
                        time.sleep(1.5)
                        break
                        mainmenu()
                    else:
                        print("Please enter a valid E-Mail format! ")
                else:
                    print("Please enter a valid E-Mail format! ")

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



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

```
            print("DNSS CouponMaster: Viewer")
            cpon_masterfetch()
            time.sleep(0.4)



    def del3d():
        try:
            if os.path.exists(r'userblock.txt'):
                os.remove(r'userblock.txt')
            if os.path.exists(r'userblock.zconf'):
                os.remove(r'userblock.zconf')
        except PermissionError:
                pass
        # Store listing::
        namie, data = dictgetlisting()
        namiex, datax = getlisting()
        # dataxr is currently redundant
        dataxr = []
        for i in datax:
            i = "₹" + '%d' % i
            dataxr.append(i)
        tablx = zip(namiex, dataxr)
        titlex = ["Product:", "Pricing:"]
        print("Store listing (as per updated records): ")
        print(tabulate(tablx, headers = titlex, floatfmt = ".4f", tablefmt='fancy_grid'))



    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")
```

```
                table(col_labels, salesrows)
                toaster.show_toast("DNSS QuickSync", "Database acessed", duration = 2)
                time.sleep(1.4) #for a seamless experience
                os.startfile('registry.txt') #to open the external notepad application
        else:
            print("Ehh that'd be wrong, sneaky-hat. Try again: ")
            print(" ")
            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 accessed - DBFA SECURITY: ATTEMPT 02")
                            print(sender)
                        except Exception:
                            pass
                    print("There ya go:: ")
                    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")
                    table(col_labels, salesrows)
                    toaster.show_toast("DNSS QuickSync", "Database acessed", duration = 2)
                    time.sleep(0.6) #for a seamless experience
                    # print(logger.read())
                    # print()
                    # print("Opening sales log externally now. ")
                    time.sleep(1.4) #for a seamless experience
                    os.startfile('registry.txt')
            else:
                with HiddenPrints():
                    try:
                        sender = telegram_bot_sendtext(dt_string + "\n" + "[ACCESS DENIED!!] - Registry file  - DBFA SECURITY [ACCESS DENIED!!]")
                        print(sender)
                    except Exception:
                        pass
                print("Multiple Unsuccesfull Attempts Detected. Re-run the program to login now. ")
                logger.write("(MULTIPLE ATTEMPTS!): Log file access attempt - AUTHORIZATION FAILED!!! \n")
                time.sleep(1.4) #for a seamless experience
                print()
                print()




    def del3f():
        import sqlite3 as sql
```

```
        import os
        import csv
        from sqlite3 import Error

        try:
            csvex=sql.connect(currdir+'\\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)
                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

    def del3g(): #delta DEEP ARCHIVAL VAULT ENGINE
        print("DBFA uses advanced algorithms to store invoice data in easily indecipherable strings.")
        print("These can be used at any point to generate an invoice from any date, given its key is stored with the deep archival VAULT.")
        print("This ensures data protection and enables us to practically store infinity invoices in a relatively non-existent amount of space\n\n")
        print("This option can be used to generate such back-dated invoices.")
        time.sleep(2)
        from DBFADeepArchivalEngine import deepfetch_deeparchival, encoder_deeparchival
        from DBFADeepArchivalEngine import alphadecoder, dttdecoder, decoder_deeparchival
        deepfetch_deeparchival()

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

    # Store listing::
    namie, data = dictgetlisting()
    namiex, datax = getlisting()
    # dataxr is currently redundant
    dataxr = []
    for i in datax:
        i = "₹" + '%d' % i
        dataxr.append(i)
    tablx = zip(namiex, dataxr)
    titlex = ["Product:", "Pricing:"]

    print('''
```

```
    DBFA Billing Framework 7 [Bellaire] (stable)
    <GNU Public License> Copyright (C) 2020 Pranav Balaji and Sushant Gupta
    View the license file in the installation dir for more info.
    \n\n
    Fetching Windows login details..
    Fetching Windows login details....
    \n''')

    logoprintxrt()
    time.sleep(0.3)
    command = "cls"
    os.system(command)


    #vs2
    from datetime import datetime  #for reporting the billing time and date
    now = datetime.now()
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object containing current date and time
    logger = open(r"registry.txt", "a+")  #Opening / creating (if it doesn't exist already) the .txt record file
    logger.write("---------------------------------------- \n")
    logger.write('ed')
    logger.write("\n")
    logger.write("Automated Store Registry:\n")

    #Settings Checker
    if settingscommonfetch(1) == 1:
        floodscreen() #comment to disable boot-flash screen
    else:
        pass


    import requests, time, json, urllib, os, math, random, sqlite3
    from tqdm import tqdm
    import logging, os, time, requests, socket
    import telegram_send
    from pynput.keyboard import Key, Controller
    # Telegram BOT API 2 (FULL)
    from telegram import InlineKeyboardButton, InlineKeyboardMarkup
    from telegram.ext import Updater, CommandHandler, CallbackQueryHandler

    logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                        level=logging.INFO)
    xlogger = logging.getLogger(__name__)

    def telegram_bot_sendtext(bot_message):
        bot_token = str(securedatafetch(2))
        bot_chatID = str(securedatafetch(3))
        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()

    def start(update, context):
        keyboard = [[InlineKeyboardButton("✅ Validate", callback_data='1'),
                     InlineKeyboardButton("❌ Deny", callback_data='2')]]

        reply_markup = InlineKeyboardMarkup(keyboard)
```

```python

        update.message.reply_text("delta 2FA Handler Service\n\n\nValidate login?", reply_markup=reply_markup)


    def button(update, context):
        query = update.callback_query
        # CallbackQueries need to be answered, even if no notification to the user is needed
        # Some clients may have trouble otherwise. See https://core.telegram.org/bots/api#callbackquery
        inlet = ("{}".format(query.data))
        if inlet in (1, "1"):
            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")
            with open(currdir+'\\deltatgstickerlogonew.webp', "rb") as f:
                telegram_send.send(stickers=[f])
                keyboard = Controller()
                print("\n\nValidation recieved! DBFA Client will start in a moment\n\n")
                print("telegram_extended.updtr_pushreq(deltaonealpha, set.webhook: (on, getUpdated.redir(servers.gokku.com/deltaonealpha/arterxt1, callback=False)))")
                print("\n\nif there's no print below for around 5 secs from now, press ctrl+c to continue\n\n")
                keyboard.press(Key.ctrl)
                keyboard.press('c')
                keyboard.release('c')
                keyboard.release(Key.ctrl)

        if inlet in (2, "2"):
            query.edit_message_text(text="❌🔏 Denied delta 2FA request.\n\ndelta Security Service")
            with open(currdir+'\\deltatgstickerlogonew.webp', "rb") as f:
                telegram_send.send(stickers=[f])
            keyboard = Controller()
            print("\n\nThe login request for this session has been DENIED.\n\n")
            time.sleep(1)
            print("telegram_extended.updtr_pushreq(deltaonealpha, set.webhook: (on, getUpdated.redir(servers.gokku.com/deltaonealpha/arterxt1, callback=False)))")
            print("\n\nif there's no print below for around 5 secs from now, press ctrl+c to continue\n\n")
            keyboard.press(Key.ctrl)
            keyboard.press('c')
            keyboard.release('c')
            keyboard.release(Key.ctrl)
            time.sleep(1)
            time.sleep(1)
            print("DBFA Client will now exit! ")
            time.sleep(5)
            os._exit(0)
        if inlet in (3, "3"):
            query.edit_message_text(text="Use */help*")


    def OAuthvalidate():
        from datetime import datetime
        import sqlite3, time, os
        currdir = str(os.getcwd())
```

```
        os.system('cls')
        now = datetime.now()
        try: #To avoid error when time is 00:00:00
            netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 + int(now.strftime("%S"))
        except:
            time.sleep(1)
            netr = int(now.strftime("%H"))*3600 + int(now.strftime("%M"))*60 + int(now.strftime("%S"))
        oauth = sqlite3.connect(r'dbfasettings.db')
        oauthx = oauth.cursor()
        oauthx.execute("SELECT Value FROM LoginHandler")
        check = oauthx.fetchall()[0][0]
        if check != 1:
            print("Login was bypassed! DBFA will now exit!")
            time.sleep(1)
            os._exit(0)
        oauthx.execute("SELECT count(*) FROM LoginHandler")
        rows = oauthx.fetchall()
        if (rows[0][0]) > 1:
            print("DBFA Authenticator has been tampered with! DBFA WILL EXIT NOW!")
            oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
            oauth.commit()
            oauth.close()
            time.sleep(2)
            os._exit(0)
        oauthx.execute("SELECT TimeMark FROM LoginHandler")
        stamp = oauthx.fetchall()[0][0]
        if int(netr)-int(stamp) > 60:
            print("netr")
            oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
            oauth.commit()
            oauth.close()
            os.startfile(currdir+'\\authtimeout.pyw')
            time.sleep(1)
            os._exit(0)
        elif int(netr)-int(stamp) < 60:
            print("- - - Login validated - - -\n\n")
            oauthx.execute("SELECT PermSet FROM LoginHandler")
            permset = oauthx.fetchall()[0][0]
            permssetter(permset)
            print("Permissions set: ", settingscommondumpfetch(10))

        else:
            oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
            oauth.commit()
            oauth.close()
            print("Login was bypassed! DBFA will now exit!")
            time.sleep(1)
            os._exit(0)

        oauthx.execute("UPDATE LoginHandler SET Value = 0, TimeMark = 0")
        oauth.commit()
        oauth.close()

    OAuthvalidate()


    def help_command(update, context):
```

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


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

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

        # Start the Bot
        updater.start_polling()

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


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



    if settingscommonfetch(6) == 1:
        def telegram_bot_sendtext(bot_message):
            bot_token = str(securedatafetch(2))
            bot_chatID = str(securedatafetch(3))
            send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' + bot_message
            response = requests.get(send_text)
            return response.json()

        print("delta2 Authenication Service")
        print("=========================================")
        print("DBFA 2FA Service")
        print("=================")
        time.sleep(0.5)
        print("")


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

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

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


    print("--------------------------------\n\n( ͡° ͜ʖ ͡°) delta IntelliSense Updater\n\n--------------------------------")
    time.sleep(0.1)
    print("Talking to server ~ ")
    url = "https://raw.githubusercontent.com/deltaonealpha/DBFA_UpdateHandler/master/updates.txt"
    r = requests.get(url)
    dbfaver = ((str(r.content.decode('utf-8'))))[4:]
    xdbfaver = ((str(r.content.decode('utf-8'))))
    with open(parentdir+'\\updates.txt', 'r+') as upread:
        upread = (str(upread.read())).strip()
    print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
    spass1 = []
    spass2 = []
    for i in dbfaver:
        spass1.append(i)
    for j in upread[4: ]:
        spass2.append(j)
    if float(upread[4: ]) > float(dbfaver):
        pass

    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
    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 = ""
        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:
                spotilib.pause()
            except Exception as e:
                pass

        elif decfac in ('next', 'nexT', 'neXt', 'neXT', 'nExt', 'nExT', 'nEXt', 'nEXT', 'Next', 'NexT', 'NeXt', 'NeXT', 'NExt', 'NExT', 'NEXt', 'NEXT'):
            try:
                spotilib.next()
            except Exception as e:
                pass

        #DBFA Mark Attendance
        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'):
            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()
                    dt_string = now.strftime("%Y/%m/%d")  #datetime object containing current date and time
                    tm_string = now.strftime("%H:%M:%S")  #datetime object containing current date and time
                    empmascur.execute("SELECT count(*) FROM attendance WHERE Date = ? AND OiD = ?", (dt_string, Oid,))
                    data = empmascur.fetchone()[0]
                    if data==0:
                        #print('No record on %s'%dt_string+' for Employee %s'%Oid)
                        #print("insert into attendance(Date, OiD, Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
                        empmascur.execute("insert into attendance(Date, OiD, Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
                        #empmascur.execute("UPDATE attendance SET Oid = 'Y' WHERE DATE = ?", (dt_string))
                        empmas.commit()
                        print("\n----------------DBFA-----------------")
                        print("C1 ENTRY: Marked Attendance! OiD: ", Oid)
                        print("-------------------------------------\n")
                    elif data==1:
                        #print('Component %s found in %s row(s)'%(dt_string, data))
                        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"
            global billiemaster
            billiemaster = 0 #variable for totalling the price
            time.sleep(0.0247) #for a seamless experience
            afac = 0
            namie, data = dictgetlisting()
            namiex, datax = getlisting()
            dde_productlist = ""
            xrinit = 0
            lstrst = 0
            ppped = 0
            from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
            while(1):
                runnedstat = 0
                print(Fore.YELLOW+"\n\n_________________________")
                print(Fore.YELLOW+"Currently in cart: ")
                print(Fore.YELLOW+"-------")
                printobj = purcheck.split("~")
                for i in printobj:
                    print(Fore.MAGENTA+i)
                if xrinit == 1 and ppped == 0:

 print(Fore.YELLOW+"_____________________________________________________________________\nType 'lastrem' to remove last added produt\n_____________________________________________________________\n"+Fore.WHITE)
                else:

 print(Fore.YELLOW+"_____________________________________________________________________\n"+Fore.WHITE)
```

```
                item = input("Enter product code: ")
                if item == "0":
                    break
                elif str(item) in ("lastrem", "LASTREM", "Lastrem", "LastRem", "lASTREM", "lASTrEM"):
                    runnedstat = 1
                    if xrinit == 1 and ppped == 0:
                        print("  ")
                        billiemaster = billiemaster - int(data[int(lstrst)])
                        dde_productlist = dde_productlist.replace((str(lstrst)+'00'), "")
                        repremupdate(int(lstrst))
                        lenxr = len(namie[int(lstrst)])
                        costlenxr = len(str(data[int(lstrst)]))
                        cj = 10 - costlenxr
                        pi = 60 - lenxr
                        idlerxx = namie[lstrst] + " "*pi + "₹"+'%d'%int(data[int(lstrst)]) + " "*cj + "1 qty. ~"
                        purcheck = purcheck.replace(idlerxx, "", 1)
                        profer = profer[:-1]
                        logger.write("Last cart item removed! ~ ")
                        priceprod = "₹" + '%d' % int(data[int(lstrst)])
                        writer = writer.replace(("\n Purchased: " + "\n" + namie[lstrst] + "\n" + priceprod + "\n"), "", 1)
                        remssxstockmaintainer(int(lstrst))
                        afac = afac - 1
                        print("Last cart product removed ~ ")
                        ppped = 1
                        time.sleep(2)
                    else:
                        print("No products in cart! ~")

                if item not in ("", " ", (), [], "\n") and runnedstat != 1:
                    if int(item) in data:
                        ssxstockmaster(item)
                        if ssxvarscheck == 1:
                            item = int(item)
                            #print(item)
                            #print(data)
                            #print(data[int(item)])
                            billiemaster+=int(data[int(item)])
                            dde_productlist += str(item) + '00'
                            print("Purchased: ", namie[item], " for: ", data[int(item)])
                            xrinit = 1
                            lstrst = int(item)
                            repupdate(item)
                            lenxr = len(namie[item])
                            costlenxr = len(str(data[int(item)]))
                            cj = 10 - costlenxr
                            pi = 60 - lenxr
                            idlerxx = namie[item] + " "*pi + "₹"+'%d'%int(data[int(item)]) + " "*cj + "1 qty. ~"
                            purcheck += idlerxx
                            print("---")
                            priceprod = "₹" + '%d' % int(data[int(item)])
                            logger.write("Appending product to order: \n")  #writing to file
                            profer.append(item)
                            logger.write(namie[item])
```

```
                            ssxstockmaintainer(item)
                            logger.write(" \n")
                            writer = writer + "\n Purchased: " + "\n" + namie[item] + "\n" + priceprod + "\n"
                            ppped = 0
                            afac+=1
                        else:
                            print("Product currently out-of-stock. The inconvenience is regretted..\n")
                            print("---")
                            continue
                            time.sleep(1)
                    else:
                        print("Product not found. Please retry ")
                        print("---")
                        time.sleep(1)
                else:
                    if runnedstat != 1:
                        print("Product not found. Please retry ")
                        print("---")
                        time.sleep(1)

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

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

```
            total = (tota + ((tota/100)*18))
            if total != 0:
                discountx = '%d' % discount
                telethon = telethon + "\n" + "Tax amount: 18%" + "\n"  + "Discount: " + discountx + "%" + "\n" + "\n"
                writer = writer + "\n" + "\n" + "----------------------------" + "\n" + "Tax amount: 18%"  + "\n"  + discountx + "\n"  + "\n"
                delxfac = input("Enter *d* for delivery; skip for in-store purchase: ")
                if delxfac != "d":
                    splitfac = input("Split bill? (y/n): ")
                    netpay = total
                    netsplitpay = total
                    if splitfac in ("Y", "y"):
                        try:
                            splitcountfac = int(input("Enter the number of people to split for: "))
                        except:
                            try:
                                print("Person count must be real! ~")
                                splitcountfac = int(input("Enter the number of people to split for: "))
                            except:
                                print("Billing splitting has been cancelled due to repetitive invalid inputs. Bill will be generated without a split.")
                                splitcountfac = 1
                        if "." in str(((netpay)/int(splitcountfac))):
                            splitpay = float(eval((str(((netpay)/int(splitcountfac))).split('.')[0] + '.' + (str(((netpay)/int(splitcountfac))).split('.')[1])[0:2])))
                        else:
                            splitpay = float(eval(netpay/int(splitcountfac)))
                        print("Net payable amount per person (for ", splitcountfac, " persons): ", splitpay)
                        splitmark = 1
                        redeemindic = 0
                        time.sleep(1)
                        netpay = total
                        xpayboxie()
                    else:
                        splitmark = 0
                        netpay = total
                        netpay = payboxie(custt, netpay)
                        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:
                            if line not in ("", " ", (), [], None, "\n"):
                                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:
                        if total > lylpoints:
                            print("Redeemed loyalty points worth: ₹", lylpoints)
                        else:
                            print("Redeemed loyalty points worth: ₹", total)
                    try:
                        if splitmark == 0:
                            print("------------------------------------------------------------------------")
                            print("Amount to be paid: ₹","%.2f" % netpay)
                            print("------------------------------------------------------------------------")
                        else:
                            print("------------------------------------------------------------------------")
                            print("Amount payable per person (POST SPLIT): ₹","%.2f" % splitpay)
                            print("Total payees                          : ", splitcountfac)
                            print("------------------------------------------------------------------------")
                            print("Amount to be paid: ₹","%.2f" % netpay)
                            print("------------------------------------------------------------------------")
                    except:
                        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:
                        try:
                            writer += "Used DBFA loyalty points worth: " + '%s'%lylpoints + "\n"
                        except:
                            writer += "Loyalty points were not redeemed on this transaction." + "\n"
```

```
                    #regin.write("NET TOTAL: \n") #writing to file
                    if splitmark == 1:
                        telethon = telethon + "NET TOTAL AFTER SPLIT PER PERSON: \n" + "₹" + str(splitpay) + "\n"
                        telethon = telethon + "Number of payees (after bill-split): \n" + str(splitcountfac) + "\n"
                        telethon = telethon + "Net Total: \n" + str(netpay) + "\n"
                        writer = writer + "NET TOTAL AFTER SPLIT PER PERSON: \n" + str(splitpay) + "\n"
                        writer = writer + "Number of payees (after bill-split): \n" + str(splitcountfac) + "\n"
                        writer = writer + "Net Total: \n" + str(netpay) + "\n"
                    else:
                        telethon = telethon + "NET TOTAL: \n" + "₹" + str(netpay) + "\n"
                        writer = writer + "NET TOTAL: \n" + str(netpay) + "\n"
                    logger.write(str(total))

                    # delta DDE - Deep Archival Engine
                    from DBFADeepArchivalEngine import encoder_deeparchival
                    #Data Format: <order ids>///<customer name>///<customer id>///<date and time string>///tax///discount///loyalty///net
                    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")  #datetime object containing current date and time
                    if redeemindic == 1:
                        data = [dde_productlist, 'dde', custt, dt_string, 18, discount, lylpoints, netpay]
                    else:
                        data = [dde_productlist[:-1], 'dde', custt, dt_string, 18, discount, 0, netpay]
                    encoder_deeparchival(data, inval)

                    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 = currdir+'\\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'))
                            filename = namer
                            attachment = open(currdir+'\\\Generated_invoices\\'+'%s'%namer, "rb")
                            attac= MIMEBase('application', 'octet-stream')
                            attac.set_payload((attachment).read())
                            encoders.encode_base64(attac)
                            attac.add_header('Content-Disposition', "attachment; filename= %s" % filename)
                            msg.attach(attac)
 os.startfile(r'Process_Handlers\\emailprocesswindow.pyw')
                            email = smtplib.SMTP('smtp.gmail.com', 587)
                            email.starttls()
                            email.login(fromaddr, "dbfaidlepass")
                            message = msg.as_string()
                            email.sendmail(fromaddr, toaddr, message)
                            print("Invoice mailed. ")
                            print("-----------------------------------------------------------------------------------------------------------\n\n")
                        #Settings Checker
                        elif settingscommonfetch(1) == 2:
                            pass
                    #regin.close()
                    with HiddenPrints():
                        try:
                            sender = telegram_bot_sendtext(telethon)
                            print(sender)
                        except Exception:
                            pass

                    print()

            else:
                print("\n\n-------------------------------------------------------------------------------------------------------------")
                print("No purchase made.")
                print("-------------------------------------------------------------------------------------------------------------")
                print("Amount           : ₹",billiemaster)
                print("-------------------------------------------------------------------------------------------------------------")
                print("Amount to be paid: ₹","%.2f" % total)
                print("-------------------------------------------------------------------------------------------------------------")
                toaster.show_toast("DBFA: No purchase made  ",str(total), duration = 1)
                logger.write("Billing cancelled: Reason: Net amount null\n") #writing to file
```

```
                #regin.write("NET TOTAL: \n") #writing to file
                telethon = telethon + "Billing cancelled. Net amount null." + "\n"
                writer = writer + "BILLING CALCELLED. No purchase made." + "\n"
                logger.write(str(total))
                logger.write("\n")
                abcd1+=1
                afac+=1
                now = datetime.now()

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


        #Manage Customers
        elif decfac == "2":
            from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
            init(convert = True)
            print(Fore.MAGENTA+'''
==========================================================================================
Options:                                                    Client >>> Manage Customers
==========================================================================================
    a: Register a Customer
    b: Customer Registry
    c: Customer Purchase Records
    d: Find a Customer
    e: Export Records as CSV

What would you like to do?                 '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
=========================================  '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
'''+Fore.CYAN+'''Manage Customers ~'''+Fore.MAGENTA+'''
==========================================================================================''' +Fore.WHITE)
            selected = input("What would you like to do? "+Fore.WHITE)
            print("\n")
            if selected in ("a", "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")

            if selected in ("b", "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")

            if selected in ("c", "C"):
                del2c()
                logger.write("------------------------------------- \n")
                logger.write("\nDBFA Customer Purchase Records accessed! \n")

            elif selected in ("d", "D"):
                del2d()
                logger.write("------------------------------------- \n")
                logger.write("\nCustomer search used. \n")

            elif selected in ("e", "E"):
                if str(settingscommondumpfetch(10)) == "Administrator":
                    del2e()
                    logger.write("\n\n------------------------------------- \n")
                    logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
                    logger.write("Customer data exported to CSV! ")
                    with HiddenPrints():
                        try:
                            sender = telegram_bot_sendtext(dt_string + "\n" + "Sales data exported to CSV- REDFLAG Urgent Security Notice!")
                            print(sender)
                        except Exception:
                            pass
                    logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
                    logger.write("------------------------------------- \n\n\n")
                else:
                    print("This function is restricted on your account.")

            elif selected not in ("a", "b", "c", "d", "e", "A", "B", "C", "D", "E"):
                print("Please chose a valid option!")
                time.sleep(1)

        #Store Options:
        elif decfac == "3":
            from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
            init(convert = True)
            print(Fore.MAGENTA+'''
==============================================================================
Options:                                                  Client >>> Store Options
==============================================================================
    a: Manage Stock
    b: DBFA Stock Master
    c: Manage Vouchers
    d: Product Listing
    e: Sales Log
    f: Export Sales Data as CSV
    g: Invoice Deep Archive

What would you like to do?                '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
========================================  '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
'''+Fore.CYAN+'''Store Options ~'''+Fore.MAGENTA+'''
==============================================================================''
'+Fore.WHITE)
            storeselected = input("What would you like to do? ")
            print("\n")
```

```python

            if storeselected in ("a", "A"):
                del3a()

            elif storeselected in ("b", "B"):
                del3b()

            elif storeselected in ("c", "C"):
                if str(settingscommondumpfetch(10)) == "Administrator":
                    del3c()
                else:
                    print("This function is restricted on your account.")

            elif storeselected in ("d", "D"):
                del3d()

            elif storeselected in ("e", "E"):
                if str(settingscommondumpfetch(10)) == "Administrator":
                    del3e()
                    logger.write("\n------------------------------------- \n")
                    logger.write("Sales log accessed! ")
                else:
                    print("This function is restricted on your account.")

            elif storeselected in ("f", "F"):
                if str(settingscommondumpfetch(10)) == "Administrator":
                    del3f()
                    logger.write("\n\n------------------------------------- \n")
                    logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
                    with HiddenPrints():
                        try:
                            sender = telegram_bot_sendtext(dt_string + "\n" + "Customer data exported to CSV- REDFLAG Urgent Security Notice!")
                            print(sender)
                        except Exception:
                            pass
                    logger.write("Sales data exported to CSV! ")
                    logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
                    logger.write("------------------------------------- \n\n\n")
                else:
                    print("This function is restricted on your account.")

            elif storeselected in ("g", "G"):
                if str(settingscommondumpfetch(10)) == "Administrator":
                    print("Deep Archival Engine")
                    del3g()
                else:
                    print("This function is restricted on your account.")

            elif storeselected not in ("a", "b", "c", "d", "e", "f", "g", "A", "B", "C", "D", "E", "F", "G"):
                print("Please select a valid option! ")
                time.sleep(1)
                mainmenu()



        #Reports
        elif decfac == "4":
            if str(settingscommondumpfetch(10)) == "Administrator":
```

```
                command = "cls"
                os.system(command)
                print("-- Generating store report --")
                repstockfetch()
                print("Please wait...")
                repdatafetch()
                findMaximum = "select max(prodsales) from recmasterx"
                recx.execute(findMaximum)
                # Print the maximum score
                netr = recx.fetchone()[0]
                findin = "select prodid, prodname, prodprofit, prodsales, netprof from recmasterx WHERE prodsales = ?"
                arterx = (str(int(netr)),)
                recx.execute(findin, arterx)
                arterxout = recx.fetchall()

                findMaximumProf = "select max(netprof) from recmasterx"
                recx.execute(findMaximumProf)
                xnetr = recx.fetchone()[0]
                findin = "select prodid, prodname, prodprofit, prodsales, netprof from recmasterx WHERE netprof = ?"
                xarterx = str(int(xnetr))
                xxarterx = (xarterx,)
                recx.execute(findin, xxarterx)
                xarterxout = recx.fetchall()

                isol = sqlite3.connect(r'DBFA_vend.db')
                isolx = isol.cursor()
                isolx = isol.cursor()
                isolx.execute(("SELECT prodid, prodname, ordqty, delstat, vendor from stock WHERE delstat = ?"), ("TBD", ))
                rowsrec = isolx.fetchall()
                col_labels = [("P. ID", "P. Name", "Qty. to be delivered", "Status", "Vendor")]
                rowsxtb = col_labels + rowsrec

                import os, time
                currdir = str(os.getcwd())
                from pathlib import Path
                path = Path(os.getcwd())
                parentdir = str(Path(path.parent))
                userdir = os.path.expanduser('~')
                url = "https://raw.githubusercontent.com/deltaonealpha/DBFA_UpdateHandler/master/updates.txt"
                r = requests.get(url)
                dbfaver = ((str(r.content.decode('utf-8'))))[4:]
                xdbfaver = ((str(r.content.decode('utf-8'))))
                with open(parentdir+'\\updates.txt', 'r+') as upread:
                    upread = (str(upread.read())).strip()
                spass1 = []
                spass2 = []
                for i in dbfaver:
                    spass1.append(i)
                for j in upread[4: ]:
                    spass2.append(j)
                if float(upread[4: ]) > float(dbfaver):
                    pass
                else:
                    if xdbfaver == upread:
```

```
                        repvalx = 'repval1.png'
                    elif spass1 != spass2:
                        repvalx = 'repval2.png'

                def add_page_number(canvas, doc):
                    canvas.saveState()
                    canvas.setFont('Times-Roman', 10)
                    page_number_text = "%d" % (doc.page)
                    canvas.drawCentredString(
                        0.75 * inch,
                        0.75 * inch,
                        page_number_text
                    )
                    canvas.restoreState()

                import sqlite3 as sql
                csvexx=sql.connect(currdir+'\\DBFA_CUSTCC.db')
                print("Fetching data from database - II...")
                cursorx = csvexx.cursor()
                axct = cursorx.execute("select custid, custname, purchasecount, printf('%.2f', ptotalx), printf('%.2f', points) from custcc")
                axctx = []
                for i in axct:
                    axctx.append(i)
                axctx = [("C. ID","C. Name","Purchases Made","Total Amount","Loyalty Points")] + axctx
                doc = SimpleDocTemplate("dbfastorerep.pdf", pagesize=A4,
 rightMargin=2*cm,leftMargin=1.5*cm,
 topMargin=1*cm,bottomMargin=2*cm)

                delcount = 0
                filedel = open('./DBFAdeliveries.txt', 'r+')
                for line in filedel:
                    if line not in ("", " ", (), [], None, "\n"):
                        delcount+=1
                filedel.close()
                if delcount != 0:
                    delcount = "<b>Number of pending deliveries: </b>"+str(delcount)
                else:
                    delcount = "<b>No deliveries pending ~ </b>"

                import time, os, requests, datetime
                from datetime import datetime, date
                empmas = sqlite3.connect(r'dbfaempmaster.db')
                empmascur = empmas.cursor()
                empmascur.execute("SELECT DISTINCT * FROM emp")
                dump = empmascur.fetchall()
                Oiddump = []
                for row in dump:
                    Oiddump.append(row[0])
                Oiddump = "<br /><b>Employee IDs registered: </b>" + str(Oiddump) + "<br /><br /><b>Attendance Records & Metrics: </b><br /><br />"
                now = datetime.now()
                dt_string = now.strftime("%Y/%m/%d")  #datetime object containing current date and time
                month = datetime.now().month - 1
                if month < 1:
```

```
                    month = 12 + month  # At this point month is 0 or a negative number so we add
                if len(str(month)) == 1:
                    month = "0"+str(month)
                dt1mb = ('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
                empmascur.execute("SELECT * FROM attendance WHERE Date BETWEEN ? AND ? ORDER BY Date ASC", (dt1mb, dt_string))
                returned = empmascur.fetchall()
                attendreturned = [('Date', 'Employee OID', 'Time', 'Marked', 'In/Out')] + list(returned)
                from DBFADeepArchivalEngine import deepfetch_deeparchival, encoder_deeparchival
                from DBFADeepArchivalEngine import alphadecoder, dttdecoder, decoder_deeparchival, deepfetch_deeplisting
                ddareport = deepfetch_deeplisting()
                empmas = sqlite3.connect(r'dbfasales.db')
                empmascur = empmas.cursor()
                now = datetime.now()
                dt_string = now.strftime("%Y/%m/%d")  #datetime object containing current date and time
                month = datetime.now().month - 1
                if month < 1:
                    month = 12 + month  # At this point month is 0 or a negative number so we add
                if len(str(month)) == 1:
                    month = "0"+str(month)
                dt1mb = ('%s'%now.strftime("%Y")+'%s'%"-"+'%s'%month+'%s'%"-"+now.strftime("%d"))
                empmascur.execute("SELECT * FROM sales WHERE date BETWEEN ? AND ?", (dt1mb, dt_string))
                rtreturned = empmascur.fetchall()
                rtreturned = [('Sales No. ', 'Cust. ID', 'Product IDs', 'Total Amount', 'Profit Earned', 'Date')] + list(rtreturned)

                # container for the 'Flowable' objects
                elements = []
                telegram_bot_sendtext("Access alert: Store Report")
                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 />")
                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 />")
                t6dot = ("Report generated on: " + dt_string + "<br /><br />")
                t20dot = ("<b>Installation Version:</b> DBFA Inception 1.0<br /><br />")
                t21dot = ("<br /><b>Vouchers In-Circulation:</b><br />")
                t22dot = ("<br />"+delcount+"<br />")
                t26dot = ("<br /><br /><b>Sales records for this month:</b><br /><br />")
                t27dot = ("<bt /><br /><br /><b>DBFA Deep Archival Engine Data (past month): </b><br /><br />")
                t3dot = ("<br /><br /><b>Most sold listing: </b><br />")
                t4dot = ("<br /><br /><b>Total profit per listing: </b><br /><br />")
                t5dot = ("<br /><br /><b>Most profit making listing: </b><br /><br />")
                t8dot = ("<br /><br /><b>Customer purchases: </b><br /><br />")
```

```
                t10dot = ("<br /><br /><br /><br /><br /><br /><b>DBFA Stock Orders Report: </b><br />Product stock yet to be recieved: <br /><br />")
                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 />")
                colas = (30, 300, 60, 50, 50)

                netprof = sqlite3.connect('recmaster.db')
                netprofx = netprof.cursor()
                netprofx.execute("SELECT * FROM recmasterx")
                dataout = netprofx.fetchall()
                rowheights = ()
                leng = len(dataout)
                for i in range(1, leng+2):
                    rowheights += (20,)
                import sqlite3
                isol = sqlite3.connect(r'cponmgmtsys.db')
                isolx = isol.cursor()
                isolx.execute("SELECT DISTINCT cponid, cponlim FROM cponmaster")
                voucherrows = isolx.fetchall()
                voucherrows = [('Voucher Identifier', 'Usage Left')] + list(voucherrows)
                text1=Paragraph(t1dot)
                text2=Paragraph(t2dot)
                text3=Paragraph(t3dot)
                text4=Paragraph(t4dot)
                text5=Paragraph(t5dot)
                text6=Paragraph(t6dot)
                text8=Paragraph(t8dot)
                text10=Paragraph(t10dot)
                text11=Paragraph(t11dot)
                text20=Paragraph(t20dot)
                text21=Paragraph(t21dot)
                text22=Paragraph(t22dot)
                text23=Paragraph(Oiddump)
                text26=Paragraph(t26dot)
                text27=Paragraph(t27dot)
                textfiller=Paragraph("<br />")
                x=Table(rows, colas, rowheights)
                t=Table(arterxout)
                t2=Table(xarterxout)
                t4=Table(axctx)
                t5=Table(rowsxtb)
                t6=Table(voucherrows)
                t7=Table(attendreturned)
                t8=Table(rtreturned, colWidths=[80,80,140,120,70,70])
                t9=Table(ddareport, colWidths=[50,110,250])
                if tabarter == ['--']:
                    t7dot = ("<br /><br /><b>All listings currently in stock!</b><br /><br />")
                else:
                    t7dot = ("<br /><br /><b>Products running low on stock: </b><br /><br />")


                tblStyle = TableStyle([('TEXTCOLOR',(0,0),(-1,-1),colors.black),
                                       ('VALIGN',(0,0),(-1,-1),'TOP'),
                                       ('LINEBELOW',(0,0),(-1,-1),1,colors.black),
                                       ('BOX',(0,0),(-1,-1),1,colors.black),
                                       ('BOX',(0,0),(0,-1),1,colors.black)])
```

```
                tblStyle.add('BACKGROUND',(0,0),(1,0),colors.lightblue)
                tblStyle.add('BACKGROUND',(0,1),(-1,-1),colors.lightblue)
                GRID_STYLE = TableStyle(
                    [('GRID', (0,0), (-1,-1), 0.25, colors.black),
                    ('ALIGN', (1,1), (-1,-1), 'LEFT')]
                    )
                text7=Paragraph(t7dot)
                t3=Table(tabarter)
                t.setStyle(GRID_STYLE)
                t2.setStyle(GRID_STYLE)
                t3.setStyle(GRID_STYLE)
                t4.setStyle(GRID_STYLE)
                t5.setStyle(GRID_STYLE)
                t6.setStyle(GRID_STYLE)
                t7.setStyle(GRID_STYLE)
                t8.setStyle(GRID_STYLE)
                t9.setStyle(GRID_STYLE)
                x.setStyle(GRID_STYLE)
                # ----------------------------------------------------------------
-------------
                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.savefig('DBFAplot.png', dpi=300, bbox_inches='tight')
                # ----------------------------------------------------------------
-------------
                secck = Image(repvalx)
                secck.drawHeight =  1.5*inch
                secck.drawWidth = 4.04*inch
                delI = Image('DBFAplot.png')
                delI.drawHeight =  4.2*inch
                delI.drawWidth = 5.5*inch
                elements.append(text1)
                elements.append(text2)
                elements.append(text6)
```

```
                elements.append(text20)
                elements.append(secck)
                elements.append(text21)
                elements.append(t6)
                elements.append(text22)
                elements.append(text23)
                elements.append(t7)
                elements.append(text3)
                elements.append(t)
                elements.append(text5)
                elements.append(t2)
                elements.append(text7)
                elements.append(t3)
                elements.append(text4)
                elements.append(x)
                elements.append(text8)
                elements.append(t4)
                elements.append(text10)
                elements.append(t5)
                elements.append(text11)
                elements.append(delI)
                elements.append(textfiller)
                elements.append(text26)
                elements.append(t8)
                elements.append(text27)
                elements.append(t9)
                # write the document to disk
                doc.build(elements,
                    onFirstPage=add_page_number,
                    onLaterPages=add_page_number,)
                for i in tqdm (range (100), desc="Analysing Data    : "):
                    time.sleep(0.00001)
                for i in tqdm (range (100), desc="Fetching Logs     : "):
                    time.sleep(0.00001)
                for i in tqdm (range (100), desc="Publishing Report : "):
                    time.sleep(0.00001)
                print("Opening store report now")
                os.startfile('dbfastorerep.pdf')
                time.sleep(2)
            else:
                    print("This function is restricted on your account.")


        #DBFA Backup&Switch
        elif decfac == "6":
            if str(settingscommondumpfetch(10)) == "Administrator":
                print('''DBFA Backup & Switch v2.0

a. ) Create a backup
b. ) Restore from backup file (Switch/ Restore)

c. ) 'Enter' to return to main menu
                ''')
                basfac = input("What would you like to do?: ")
                if basfac in ("A", "a"):
                    os.startfile(r'delauth.py')
                    time.sleep(0.3)
                    os._exit(0)
                elif basfac in ("B", "b"):
                    os.startfile(r'dbfarestoration.py')
```

```
                    time.sleep(0.3)
                    os._exit(0)
                else:
                    pass
            else:
                    print("This function is restricted on your account.")



        #Stock Ordering Option
        elif decfac == "5":
            from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
            init(convert = True)
            time.sleep(2)
            print(Fore.MAGENTA+'''
==============================================================================
Options:                                          Client >>> Delivery Handler
==============================================================================
For issuing new delivery orders, use the invoicing option
-------------------------------------------------------
a: View existing deliveries
b: Show delivery count
c: Confirm a delivery

What would you like to do?            '''+Fore.MAGENTA+'''
===================================  '''+Fore.MAGENTA+'''
'''+Fore.CYAN+'''DBFA Delivery Handler ~'''+Fore.MAGENTA+'''
=============================================================================='''+Fore.WHITE)
            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:
                    if line not in ("", " ", (), [], None, "\n"):
                        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 == "8":
            print("\nAbout DBFA: ")
            print("DBFA Inception 1.0 by deltaonealpha")
            print("Fetching latest licensing information.......")
            logoprintxrt()
            time.sleep(0.5)
            print(" ")
            print("_______ Licensing _______")
            print("           GNU PUBLIC LICENSE - TERMS AND CONDITIONS")
            print("    <deltaBillingFramework>  Copyright (C) 2020 Pranav Balaji and Sushant Gupta")
            print("    This program comes with ABSOLUTELY NO WARRANTY; for details type *show w*.")
            print("    This is free software, and you are welcome to redistribute it")
            print("    under certain conditions; type *show c* for details. ")
            toaster.show_toast("DFBA Framework Runtime Broker", "©2020: DBFA by Pranav Balaji and Sushant Gupta", duration = 1.5)
            print(" ")
            print("Visit: www.github.com/deltaonealpha/deltaBillingFramework for complete licensing terms. ")
            print(" ")
            print(" ")
            webbrowser.open('https://telegra.ph/DBFA-Licensing-Information-08-16')
            print("--------------------------------------------------")
            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 == "10":
            if str(settingscommondumpfetch(10)) == "Administrator":
                def transitionprogress():
                    from colorama import init, Fore, Back, Style
```

```
                    os.system("cls")
                    time.sleep(1)
                    print(Fore.WHITE+'|'+Fore.RED+'▀▀▀▀ OFF |')
                    time.sleep(0.3)
                    print(Fore.WHITE+'|'+Fore.RED+'▀▀▀      '+Fore.GREEN+'▀|')
                    time.sleep(0.3)
                    print(Fore.WHITE+'|'+Fore.RED+'▀▀      '+Fore.GREEN+'▀▀|')
                    time.sleep(0.3)
                    print(Fore.WHITE+'|'+Fore.RED+'▀     '+Fore.GREEN+'▀▀▀|')
                    time.sleep(0.3)
                    print(Fore.WHITE+'|'+Fore.GREEN+ ' ON  ▀▀▀▀|'+Fore.WHITE)
                    time.sleep(1.24)
                    os.system("cls")


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

                os.system("cls")

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

print("▀▀▀▀
```

```
                            print(" 3:    Enable DBFA Music Controls (beta):", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      '))
                        if (settingscommonfetch(4)) == 1:
                            print(" 4:    Open CSV when exported:", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      ')
                        else:
                            print(" 4:    Open CSV when exported:", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      '))
                        if (settingscommonfetch(7)) == 1:
                            print(" 7:    Use new DBFA Menu style:", '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|      '+Fore.RED)
                            print(" ")
                        else:
                            print(" 7:    Use new DBFA Menu style:", ('|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|      ')+Fore.RED)
                            print(" ")

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

                        print("Go to 'DBFA Security' option for more. ")
                        print(Fore.MAGENTA+" 11:   Check for updates:"+' |'+"[redacted] Proceed > "+Fore.WHITE+"|  " )
                        print("")
                        print(Fore.RED+" 12:   Return to Main Menu:"+' |'+"[redacted] Proceed > "+Fore.WHITE+"| " )
                        print("")

 #print("[redacted]")
                        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+'[redacted]'+Fore.WHITE+'|')
                            settfac1x = input(("n:     "+'|'+Fore.RED+'[redacted]'+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+'[redacted]'+Fore.WHITE+'|')
                            settfac1x = input(("n:     "+'|'+Fore.RED+'[redacted]'+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("")
                                time.sleep(1)
                                settingsmenu()

                        elif settfac == "3":
                            print('''In our mission of making DBFA the ultimate space to control your entire store and its functioning,
                            we keep adding tiny tid-bits to make that process even easier.
                            DBFA Music Controls is one such feature introduced in DBFA 8 RC3x (IB3).
                            When you disable this functionality:
                                    - The currently-playing track will no longer be displayed.
                                    - DBFA Music Controls, including but not limited to pause/play, prev and next will be restricted. ''')
                            print(" ")
                            print("Enable DBFA Music Controls? ")
                            print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
                            settfac1x = input(("n:     "+'|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|: '))
```

```
                            if settfac1x == "y":
                                settingsmodifier(3, 1)
                                transitionprogress()
                                print("DBFA Music Controls Service will be started with the next menu-cycle. ")
                                print("")
                                time.sleep(1)
                                settingsmenu()
                            elif settfac1x == "n":
                                settingsmodifier(3, 0)
                                transitionprogressneg()
                                print("DBFA Music Controls Service will be restricted from the next menu-cycle.")
                                print("")
                                time.sleep(1)
                                settingsmenu()
                            else:
                                print("That's an invalid input... ")
                                print("")
                                time.sleep(1)
                                settingsmenu()


                        elif settfac == "4":
                            print("CSV files once generated are auto-opened in your default worksheet app")
                            print("Example: Microsoft Excel, LibreOffice Calc, Google Docs, et cetera.")
                            print(" ")
                            print("Open file after export? ")
                            print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
                            settfac1x = input(("n:     "+'|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|: '))
                            if settfac1x == "y":
                                settingsmodifier(4, 1)
                                transitionprogress()
                                print("DBFA will now open CSV files when exported on request. ")
                                print("")
                                time.sleep(1)
                                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 == "7":
                            print("This option let's you switch between the older DBFA menu-style")
                            print("and the newer one as introduced with DBFA 8.12")
```

```
                            print("\nFor the best visual experience with DBFA, we recommend you to use the newer design.\n\n")
                            print("DBFA Menu-Style: ")
                            print("1: Use new style (recommended)")
                            print("2: Use old style")
                            msdsfac = input("Please make a choice: ")
                            if msdsfac == "1":
                                settingsmodifier(7, 1)
                                transitionprogress()
                                print("New menu style applied! ")
                            if msdsfac == "2":
                                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 = currdir+'\\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 == "11":
                            print("Please use DBFA Updater from the main menu (option 11)")
                            settingsmenu()


                        elif settfac == "11":
                            break
                            break
                            break

                        else:
                            print("That's an invalid input... ")
                            print("")
                            time.sleep(1)
                            break
                            break
                            break

                    settingsmenu()
                else:
                        print("This function is restricted on your account.")
```

```
        #Profit Graph Plotter
        elif decfac == "7":
            if str(settingscommondumpfetch(10)) == "Administrator":
                time.sleep(0.5)
                print("---- DBFA Sales Analyzer Engine v1 ----")
                time.sleep(0.5)
                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")
                time.sleep(1)
                print("Please wait while we analyze store sales..\n\n")
                time.sleep(2)
                print("A new window will be shortly opened. ")
                print("You're requested to close the same when you want to return to DBFA's main menu.\n")
                time.sleep(1.7)
                os.startfile(r'plotter.pyw')
            else:
                print("This function is restricted on your account.")


        #DBFA Updater
        elif decfac == "9":
            if str(settingscommondumpfetch(10)) == "Administrator":
                import requests, os, time, shutil, oschmod
                os.system('cls')
                print('''---------------------------------
delta Update Deployment Service
---------------------------------
Checking for updates. . .
---------------------------------''')
                time.sleep(1)

                import os, time
                currdir = str(os.getcwd())
                from pathlib import Path
                path = Path(os.getcwd())
                parentdir = str(Path(path.parent))
                #print(parentdir)
                #print(parentdir+'\\updates.txt')
                userdir = os.path.expanduser('~')
                url = "https://raw.githubusercontent.com/deltaonealpha/DBFA_UpdateHandler/master/updates.txt"
                r = requests.get(url)
                dbfaver = ((str(r.content.decode('utf-8'))))[4:]
                xdbfaver = ((str(r.content.decode('utf-8'))))
                with open(parentdir+'\\updates.txt', 'r+') as upread:
                    upread = (str(upread.read())).strip()

                print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
                time.sleep(1)
                spass1 = []
                spass2 = []
                for i in dbfaver:
                    spass1.append(i)
                for j in upread[4: ]:
                    spass2.append(j)
                if float(upread[4: ]) > float(dbfaver):
                    pass

```

```
                else:
                    if xdbfaver == upread:
                        print("This installation of DBFA is already up-to date ~\n")

                    elif spass1 != spass2:
                        print("Please wait...")
                        time.sleep(2)
                        os.startfile(r'updateDBFA.py')
                        print("Updater opened in a different window")
                        time.sleep(2)
                        os._exit(0)
            else:
                print("This function is restricted on your account.")

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

        #DBFA EMPLOYEE MANAGER
        elif decfac in ("emp", "EMP", "EMPLOYEE", "employee", "manager", "MANAGER", "empm", "EMPM"):
            print("Continuing to DBFA Employee Manager - -")
            time.sleep(2)
            print("\n-------------------------------------------------")
            print("This action needs administrator authorization.")
            print("This will provide access to: ")
            print("     - All employee data on your DBFA installation")
            print("\n-------------------------------------------------")
            time.sleep(1)
            import getpass
            passrt = getpass.getpass(prompt='Enter password: ', stream=None)
            if passrt == str(securedatafetch(6)):
                with HiddenPrints():
                    try:
                        sender = telegram_bot_sendtext(dt_string + "\n" + "DBFA Employee Manager accessed! \n\n - DBFA Security")
                        print(sender)
                    except Exception:
                        pass

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

```
                print("                    ( ◕_◕ )")
                time.sleep(1)
                os.system('cls')

                def empmenu():
                    from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
                    init(convert = True)

 print(Fore.CYAN+'''\n------------------------------------------------------------------------------------------\nOptions:
1  - Hire an employee                    5  - Scheduler & Leave Application
2  - View employee records
3  - Update employee details             6  - Mark attendance
4  - Fire an employee ( ●'◡'● )         7  - Work records - All
                                        8  - Work records - oID-specf.
11 - Pay salary                         9  - Work records - All (past 30)
12 - <<< Back to DBFA menu              10 - Work records - oID-specf. (past 30)
                                        13 - View Salary Payment Records

What would you like to do?                  '''+Fore.WHITE+'''▄▀▄ ▄▀ ▀▄▀ ▄▀ ▄▀ Employee'''+Fore.CYAN+'''
------------------------------------------  '''+Fore.WHITE+'''▀▄▀ ▄▀▄ ▀▄ ▄▀ ▄▀ Manager 2.12'''+Fore.CYAN+'''
'''+Fore.MAGENTA+'''The Ultimate Employee Buster ~'''+Fore.CYAN+'''
------------------------------------------------------------------------------------------''' +Fore.WHITE)

                while (1):
                    empmenu()
                    empfac = input("What would you like to do? ")
                    if empfac == "13":
                            import sqlite3
                            empmas = sqlite3.connect(r'dbfaempmaster.db')
                            empmascur = empmas.cursor()
                            empmascur.execute("SELECT * FROM salpay")
                            datastream = empmascur.fetchall()
                            from tabulate import tabulate
                            print("Payment records: ")
                            print(tabulate(datastream, headers=['Employee ID:', "DATE of payment:", 'Amount paid:'], tablefmt='fancy_grid'))

                    if empfac == "1":
                        print("DBFA will now be opening a seperate window due to GUI-restrictions.")
                        time.sleep(2)
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Hire Employee - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                        os.startfile(r'dbfaempman.py')


                    if empfac == "2":
                        print("\n2. View employee records\n------------------------------------------------------------")
                        empmas = sqlite3.connect(r'dbfaempmaster.db')
```

```
                        empmascur = empmas.cursor()
                        print("Employee records as maintained by DBFA: ")
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Records - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                        empmascur.execute("SELECT * FROM emp")
                        emprows = empmascur.fetchall()
                        time.sleep(1)
                        for emprow in emprows:
                            print(emprow, "\n")
                        print("\n\n-----------------------------------------------")
                        time.sleep(3)

                    if empfac == "3":
                        print("3. Change employee details")
                        empmas = sqlite3.connect(r'dbfaempmaster.db')
                        empmascur = empmas.cursor()
                        empmascur.execute("SELECT * FROM emp")
                        emprows = empmascur.fetchall()
                        time.sleep(1)
                        for emprow in emprows:
                            print(emprow, "\n")
                        print("\n")
                        emppay = str(input("Enter the Oid (Employee ID) to change details for: "))
                        empmascur.execute("SELECT Name FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                        firename = str(empmascur.fetchall()[0][0])
                        confofac = input("CONFIRM: Change details for "+firename+"? (y/n): ")
                        if confofac == "y":
                            from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
                            init(convert = True)
                            print(Fore.MAGENTA+'''

==================================================================================
    Options:                            Employee Manager >>> Employee Details Modifier

==================================================================================
        1. Change Name
        2. Change Email
        3. Change Mobile Contact

        4. Change Residential Address
        5. Change UPI Payments ID

        6. Change Department
        7. Change Designation (POST)
        8. Change Salary

    What would you like to do?                '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
    ========================================  '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
    '''+Fore.CYAN+'''Employee Details Modifier ~'''+Fore.MAGENTA+'''
```

```
==========''+Fore.WHITE)
                            subfac = input("What would you like to do? ")
                            if subfac == "1":
                                print("1. Change Name")
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                newmodif = input("Enter the changed name: ")
                                time.sleep(1)
                                empmascur.execute("UPDATE emp SET Name = ? WHERE Oid = ?", (newmodif, emppay))
                                empmas.commit()
                                print("Name changed for OID", emppay, " from ", firename, "to ", newmodif)
                                time.sleep(1)
                                with HiddenPrints():
                                    try:
                                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Details Changed - deltaDBFA")
                                        print(sender)
                                    except Exception:
                                        pass
                                print("--------")
                                time.sleep(1)

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

                            if subfac == "3":
                                print("3. Change Mobile Contact")
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT Mobile FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                                firename = str(empmascur.fetchall()[0][0])
                                newmodif = input("Enter the changed mobile contact: ")
```

```
                                time.sleep(1)
                                empmascur.execute("UPDATE emp SET Mobile = ? WHERE Oid = ?", (newmodif, emppay))
                                empmas.commit()
                                with HiddenPrints():
                                    try:
                                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Details Changed - deltaDBFA")
                                        print(sender)
                                    except Exception:
                                        pass
                                print("Mobile contact changed for OID", emppay, " from ", firename, "to ", newmodif)
                                time.sleep(1)
                                print("--------")
                                time.sleep(1)

                            if subfac == "4":
                                print("4. Change Residential Address")
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT Address FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                                firename = str(empmascur.fetchall()[0][0])
                                newmodif = input("Enter the changed address (in one-line): ")
                                time.sleep(1)
                                empmascur.execute("UPDATE emp SET Address = ? WHERE Oid = ?", (newmodif, emppay))
                                empmas.commit()
                                with HiddenPrints():
                                    try:
                                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Details Changed - deltaDBFA")
                                        print(sender)
                                    except Exception:
                                        pass
                                print("Address changed for OID", emppay, " from ", firename, "to ", newmodif)
                                time.sleep(1)
                                print("--------")
                                time.sleep(1)

                            if subfac == "5":
                                print("5. Change UPI Payments ID")
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT UPI FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                                firename = str(empmascur.fetchall()[0][0])
                                newmodif = input("Enter the changed UPI ID: ")
                                time.sleep(1)
                                empmascur.execute("UPDATE emp SET UPI = ? WHERE Oid = ?", (newmodif, emppay))
                                empmas.commit()
                                with HiddenPrints():
                                    try:
                                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Details Changed - deltaDBFA")
                                        print(sender)
```

```
                                    except Exception:
                                        pass
                                print("UPI ID changed for OID", emppay, " from ", firename, "to ", newmodif)
                                time.sleep(1)
                                print("--------")
                                time.sleep(1)

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

                            if subfac == "7":
                                print("7. Change Designation (POST)")
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT Post FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                                firename = str(empmascur.fetchall()[0][0])
                                newmodif = input("Enter the new designation: ")
                                time.sleep(1)
                                empmascur.execute("UPDATE emp SET Post = ? WHERE Oid = ?", (newmodif, emppay))
                                empmas.commit()
                                with HiddenPrints():
                                    try:
                                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Details Changed - deltaDBFA")
```

```
                                    print(sender)
                                except Exception:
                                    pass
                                print("Designation (Post) changed for OID", emppay, " from ", firename, "to ", newmodif)
                                time.sleep(1)
                                print("--------")
                                time.sleep(1)

                            if subfac == "8":
                                print("8. Change Salary")
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT Salary FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                                firename = str(empmascur.fetchall()[0][0])
                                newmodif = input("Enter the new salary (net; not incremental): ")
                                time.sleep(1)
                                empmascur.execute("UPDATE emp SET Salary = ? WHERE Oid = ?", (newmodif, emppay))
                                empmas.commit()
                                with HiddenPrints():
                                    try:
                                        sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Details Changed - deltaDBFA")
                                        print(sender)
                                    except Exception:
                                        pass
                                print("Salary changed for OID", emppay, " from ", firename, "to ", newmodif)
                                time.sleep(1)
                                print("--------")
                                time.sleep(1)
                            else:
                                print("Invalid option! \n\n")
                                time.sleep(1)
                        else:
                            print("Invalid option! \n\n")
                            time.sleep(1)


                    if empfac == "4":
                        empmas = sqlite3.connect(r'dbfaempmaster.db')
                        empmascur = empmas.cursor()
                        print("4. Fire an employee ┌●'◡'●└┐ \n")
                        empmascur.execute("SELECT * FROM emp")
                        emprows = empmascur.fetchall()
                        time.sleep(1)
                        for emprow in emprows:
                            print(emprow, "\n")
                        print("\n")
                        emppay = str(input("Enter the Oid (Employee ID) for the employee to fire: "))
                        empmascur.execute("SELECT Name FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                        firename = str(empmascur.fetchall()[0][0])
                        confofac = input("CONFIRM: Fire "+firename+"? (y/n): ")
                        if confofac == "y":
                            reasonfire = input("Enter the reason for firing: ")
```

```
                            print("FIRING EMPLOYEE! ")
                            empmascur.execute("DELETE FROM emp WHERE Oid = ?", ("%"+emppay+"%", ))
                            empmas.commit()
                            telethon = ""
                            print("\n\n")
                            print("---------------------------------------------")
                            telethon += ("----------------------------\n")
                            print("DBFA Employee Firing Record        deltaDBFA")
                            telethon += ("DBFA Employee Firing Record\n")
                            print("- - - - - - - - - - - - - - - - - - - - - - - -")
                            telethon += ("- - - - - - - - - - - - - - - - -\n")
                            print("Name  :", '%s'%firename)
                            telethon += ("Name  : "+ '%s'%firename)
                            print("Reason:", '%s'%reasonfire)
                            telethon += ("\nReason: "+ '%s'%reasonfire)
                            print("- - - - - - - - - - - - - - - - - - - - - - - -")
                            telethon += ("\n- - - - - - - - - - - - - - - - - -\n")
                            telethon += ("~ deltaDBFA\n")
                            telethon += ("----------------------------\n\n")
                            telegram_bot_sendtext(telethon)
                            print("~ Event logged in Infinity Logger & Telegram.")
                            print("---------------------------------------------\n\n")

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

                    if empfac == "5":
                        print("\nDBFA Scheduler handles duty scheduling and shifts for 24x7 services.\nDBFA can automatically schedule shifts and handle leave applications.")
                        from colorama import init, Fore, Back, Style #color-settings for the partner/sponsor adverts
                        init(convert = True)
                        print(Fore.MAGENTA+'''

==================================================================================
    Options:                                        Employee Manager >>> deltaScheduler

==================================================================================
        a. View tomorrow's schedule
        b. Apply for a leave (effective tomorrow only)
        c. Return to main menu ~

    What would you like to do?                '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
    ========================================  '''+Fore.WHITE+''''''+Fore.MAGENTA+'''
    '''+Fore.CYAN+'''Shift Scheduler ~'''+Fore.MAGENTA+'''

==================================================================================''' +Fore.WHITE)
                        submen = input("What would you like to do?: ")
                        if submen in ("A", "a"):
                            with open('lastsched.txt', 'r+', encoding="utf-8") as file:
                                print("Latest schedule as sent to employees:\n", file.read())
                                file.close()
```

```
                            contfac = input("Press 'enter' to continue: (workspace will be cleared!)")
                        if submen in ("B", "b"):
                            import sqlite3
                            print("Apply for leave ~") #UDRN
                            OiD = input("Enter your Employee ID (O-ID): ")
                            #make db confo here
                            from datetime import datetime, timedelta
                            now = datetime.now()
                            now = now + timedelta(days=1)
                            dt_string = now.strftime("%Y-%m-%d")
                            empmas = sqlite3.connect(r'dbfaempmaster.db')
                            empmascur = empmas.cursor()
                            empmascur.execute("SELECT Name, Oid FROM emp WHERE Oid = ?", ('%s'%OiD,))
                            datastream = empmascur.fetchall()
                            print(datastream[0][1])
                            if int(OiD) == int(datastream[0][1]):
                                confo = input(datastream[0][0] + OiD + ": Confirm leave application? for " + '%s'%dt_string + "(y/n): ")
                                if confo in ("y", "Y"):
                                    print("Leave application recieved ~ Enjoy your day tomorrow :)")
                                    empmas = sqlite3.connect(r'dbfaempmaster.db')
                                    empmascur = empmas.cursor()
                                    empmascur.execute("INSERT INTO leave(Oid, Date) VALUES (?, ?)", ('%s'%OiD, '%s'%dt_string, ))
                                    empmas.commit()
                                else:
                                    print("Oof. Now get to work ;) ")
                            else:
                                print("Invalid OiD identifier. Please try again ~")

                        if submen in ("C", "c"):
                            pass


                    if empfac == "6":
                        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()
                                dt_string = now.strftime("%Y/%m/%d")  #datetime object containing current date and time
                                tm_string = now.strftime("%H:%M:%S")  #datetime object containing current date and time
                                empmascur.execute("SELECT count(*) FROM attendance WHERE Date = ? AND OiD = ?", (dt_string, Oid,))
                                data = empmascur.fetchone()[0]
                                if data==0:
                                    #print('No record on %s'%dt_string+' for Employee %s'%Oid)
                                    #print("insert into attendance(Date, OiD, Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
                                    empmascur.execute("insert into attendance(Date, OiD, Time, YN, IO) values(?, ?, ?, ?, ?)", (dt_string, Oid, tm_string, 'Y', 'I'))
                                    #empmascur.execute("UPDATE attendance SET Oid = 'Y' WHERE DATE = ?", (dt_string))
                                    empmas.commit()
                                    print("\n----------------DBFA-----------------")
                                    print("C1 ENTRY: Marked Attendance! OiD: ", Oid)
                                    with HiddenPrints():
                                        try:
                                            sender = telegram_bot_sendtext(dt_string + "\n" + "C1 ENTRY: Marked attendance - OiD"+'%s'%Oid)
                                            print(sender)
                                        except Exception:
                                            pass
                                    print("-------------------------------------\n")
                                elif data==1:
                                    #print('Component %s found in %s row(s)'%(dt_string, data))
                                    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)
                                    with HiddenPrints():
                                        try:
                                            sender = telegram_bot_sendtext(dt_string + "\n" + "C2 DAY END: Marked attendance - OiD"+'%s'%Oid)
                                            print(sender)
                                        except Exception:
                                            pass
                                    print("-------------------------------------\n")
                                elif data > 1:
                                    print("You can only mark in/out attendance once a day! \n")
                            else:
                                print("OiD not found! \n")

                    if empfac == "7":
                        print("7. Attendance records - All")
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Attendance Records - deltaDBFA")
                                print(sender)
```

```
                            except Exception:
                                pass
                        import sqlite3, time, os, requests, datetime
                        from datetime import datetime, date

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

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

                        print("OiDs registered: ", Oiddump)


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

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


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


                    if empfac == "8":
                        print("8. Attendance records - OiD-specific")
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Attendance Records - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                        import sqlite3, time, os, requests
                        from datetime import datetime  #for reporting the billing time and date

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

                        empmascur.execute("SELECT DISTINCT * FROM emp")
                        dump = empmascur.fetchall()
                        Oiddump = []
```

```
                        for row in dump:
                            Oiddump.append(row[0])

                        print("OiDs registered: ", Oiddump)

                        try:
                            Oid = int(input("DBFA ATTENDANCE RECORDS- Enter the 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)
                                empmascur.execute("SELECT * FROM attendance WHERE Oid = ? ORDER BY Date ASC, Time, IO", ('%s'%Oid))
                                returned = empmascur.fetchall()
                                for row in returned:
                                    print(row)


                    if empfac == "9":
                        print("9. Attendance records - All (THIS MONTH)")
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Attendance Records - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                        import sqlite3, time, os, requests, datetime
                        from datetime import datetime, date

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

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

                        print("OiDs registered: ", Oiddump)


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

                        month = datetime.now().month - 1
                        if month < 1:
                            month = 12 + month  # At this point month is 0 or a negative number so we add
                        if len(str(month)) == 1:
                            month = "0"+str(month)
```

```
                        dt1mb = ('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))


                        time.sleep(0.5)
                        print("\nLoading data for days between "+'%s'%dt1mb+" and "+'%s'%dt_string+"\n")
                        empmascur.execute("SELECT * FROM attendance WHERE Date BETWEEN ? AND ? ORDER BY Date ASC", (dt1mb, dt_string))
                        returned = empmascur.fetchall()
                        for row in returned:
                            print(row)


                    if empfac == "10":
                        print("10. Attendance records - OiD-specific (THIS MONTH)")
                        with HiddenPrints():
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" + "Accessed: Employee Attendance Records - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                        import sqlite3, time, os, requests, datetime
                        from datetime import datetime, date

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

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

                        print("OiDs registered: ", Oiddump)

                        try:
                            Oid = int(input("DBFA ATTENDANCE RECORDS- Enter the OiD: "))
                            trypass = 1
                        except:
                            print("OiDs are integer-only. Please retry using valid credentials! \n")
                            trypass = 0

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

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

                        if trypass == 1:
```

```
                            if Oid in Oiddump:
                                print("OiD found ")
                                time.sleep(0.5)
                                print("\nLoading data for days between "+'%s'%dt1mb+" and "+'%s'%dt_string+"\n")
                                empmascur.execute("SELECT * FROM attendance WHERE OiD = ? AND Date BETWEEN ? AND ? ORDER BY Date ASC", ('%s'%Oid, dt1mb, dt_string))
                                returned = empmascur.fetchall()
                                for row in returned:
                                    print(row)


                    if empfac == "11":
                        from datetime import datetime, date
                        now = datetime.now()
                        dt_string = now.strftime("%d")  #datetime object containing current date and time
                        #print(dt_string)
                        if dt_string in ("01", "02", "03", "04", "05"):
                            def errorout():
                                print("Invalid parameters, please retry")
                                exit

                            import sqlite3, os, datetime
                            from datetime import datetime, date

                            def getpost(Oid):
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT Post FROM emp WHERE OiD = ?", ('%s'%Oid,))
                                daysrt = empmascur.fetchall()
                                return daysrt[0][0]


                            def getdays(Oid):
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                now = datetime.now()
                                dt_string = now.strftime("%Y/%m/%d")  #datetime object containing current date and time
                                month = datetime.now().month - 1
                                if month < 1:
                                    month = 12 + month  # At this point month is 0 or a negative number so we add
                                if len(str(month)) == 1:
                                    month = "0"+str(month)
                                dt1mb = ('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
                                empmascur.execute("SELECT * FROM attendance WHERE OiD = ? AND Date BETWEEN ? AND ? ORDER BY Date ASC", ('%s'%Oid, dt1mb, dt_string))
                                returned = empmascur.fetchall()
                                return len(returned)


                            def gethoursavg(Oid):
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT * FROM attendance WHERE OiD = ? ORDER BY Date ASC, Time", ('%s'%Oid,))
```

```
                                count = 0
                                rows = empmascur.fetchall()
                                datelist = []
                                for i in rows:
                                    datelist.append(i[0])
                                a = [i for i in rows if datelist.count(i[0])>1]
                                netr = (len(datelist) - len(a)) * 8
                                ini_list = [i[2] for i in a]
                                from datetime import timedelta
                                diff_list = []
                                for x, y in zip(ini_list[0::], ini_list[1::]):
                                    t1 = str((timedelta(hours=int(y.split(':')[0]), minutes=int(y.split(':')[1])) - timedelta(hours=int(x.split(':')[0]), minutes=int(x.split(':')[1]))).split(':')[0]
                                    diff_list.append(t1)
                                del diff_list[1::2]
                                for i in range(0, len(diff_list)):
                                    diff_list[i] = int(diff_list[i])
                                if int(netr) > 0:
                                    diff_list.append(netr)
                                return (sum(diff_list)/len(diff_list))


                            def getovertime(Oid):
                                from datetime import datetime, date
                                now = datetime.now()
                                dt_string = now.strftime("%Y/%m/%d")  #datetime object containing current date and time
                                month = datetime.now().month - 1
                                if month < 1:
                                    month = 12 + month  # At this point month is 0 or a negative number so we add
                                if len(str(month)) == 1:
                                    month = "0"+str(month)
                                dt1mb = ('%s'%now.strftime("%Y")+'%s'%"/"+'%s'%month+'%s'%"/"+now.strftime("%d"))
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("SELECT * FROM attendance WHERE OiD = ? AND Date BETWEEN ? AND ? ORDER BY Date ASC", ('%s'%Oid, dt1mb, dt_string, ))
                                count = 0
                                rows = empmascur.fetchall()
                                datelist = []
                                for i in rows:
                                    datelist.append(i[0])
                                a = [i for i in rows if datelist.count(i[0])>1]
                                netr = (len(datelist) - len(a)) * 8
                                ini_list = [i[2] for i in a]
                                from datetime import timedelta
                                diff_list = []
                                for x, y in zip(ini_list[0::], ini_list[1::]):
                                    t1 = str((timedelta(hours=int(y.split(':')[0]), minutes=int(y.split(':')[1])) - timedelta(hours=int(x.split(':')[0]), minutes=int(x.split(':')[1]))).split(':')[0]
                                    diff_list.append(t1)
                                del diff_list[1::2]
                                for i in range(0, len(diff_list)):
                                    diff_list[i] = int(diff_list[i])
                                finlist = []
                                for i in range(0, len(diff_list)):
```

```
                                if diff_list[i] > 8:
                                    finlist.append(diff_list[i])
                            return sum(finlist)



                        def calcengine(Oid, bonus):
                            postnet = ("ceo", "cto", "admin", "it", "sales", "maintanence", "logistics")
                            salnet = (97000, 84000, 42000, 35000, 32000, 22000, 21000)
                            days = 30
                            leave_days = 30-int(days)
                            post = getpost(Oid)
                            avg_hours_week = gethoursavg(Oid)
                            hours_overtime = getovertime(Oid)
                            paynet = int(salnet[postnet.index(post.lower())])
                            if int(avg_hours_week) < 8:
                                paynet = paynet - ((8-int(avg_hours_week))*1000)
                            if int(hours_overtime) > 0:
                                paynet += (int(hours_overtime)*1500)
                            if int(leave_days) > 3:
                                paynet = paynet - ((int(leave_days)-3)*750)
                            if int(bonus) > 0:
                                paynet += bonus
                            print("----------------------------")
                            print("Average hours worked     : ", avg_hours_week)
                            print("Overtime hours worked    : ", hours_overtime)
                            print("Leave(s) taken           : ", leave_days)
                            print("Bonus added              : ", bonus)
                            print("----------------------------\n")
                            return paynet


                        import time
                        import pyqrcode, png, os
                        from pyqrcode import QRCode

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

                        empmascur.execute("SELECT Oid, Name, Mobile, UPI, Dept, Post, Salary FROM emp")
                        emprows = empmascur.fetchall()
                        from tabulate import tabulate
                        print(tabulate(emprows, headers=['O-ID', 'Name', 'Mobile', 'UPI', 'Dept', "Post", 'Salary'], tablefmt='fancy_grid'))
                        #for emprow in emprows:
                            #print(emprow)

                        time.sleep(2)

                        emppay = str(input("Enter the Oid (Employee ID) to pay salary for: "))
                        empmascur.execute("SELECT * FROM emp WHERE Oid LIKE ?", ("%"+emppay+"%", ))
                        print(tabulate(empmascur.fetchall(), headers=['O-ID', 'Name', 'DOB', 'Email', 'Mobile', 'Address', 'UPI', 'Dept', "Post", 'Salary'], tablefmt='fancy_grid'))
                        print('\n')
```

```
                            print("Please pay the employee: ₹", calcengine(1, 0))
                            time.sleep(2)
                            emppayconfox = input("\n\nPay salary? (y/n): ")
                            if emppayconfox == "y":
                                #emarpay = str("%"+'%s'%emppay+"%")
                                empmascur.execute("SELECT Name, UPI FROM emp WHERE Oid LIKE ?", (emppay,) )
                                tempemppay = empmascur.fetchall()
                                print("Paying ", list(tempemppay[0])[0], "at ", list(tempemppay[0])[1])
                                name = list(tempemppay[0])[0]
                                upid = list(tempemppay[0])[1]
                            else:
                                empmenu()
                                exit
                                print("Aaaa")

                            #upid = '9810141714@upi'
                            #name = 'KPBalaji'

                            s = "upi://pay?pa="+'%s'%upid+"&pn="+'%s'%name+"&cu=INR"

                            # Generate QR code
                            url = pyqrcode.create(s)

                            url.png('payqr.png', scale = 6)
                            from PIL import Image, ImageDraw, ImageFont
                            image = Image.open('payqr.png')
                            try:
                                sender = telegram_bot_sendtext(dt_string + "\n" + "Started Process: Issue Salary - deltaDBFA")
                                print(sender)
                            except Exception:
                                pass
                            draw = ImageDraw.Draw(image)
                            font = ImageFont.truetype(fontsdir, size=200)
                            (x, y) = (5, 250)
                            xname = 'Scan to pay with UPI              deltaDBFA'
                            draw.text((x, y), xname) #, fill=color)
                            (x, y) = (5, 5)

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

                            print("Please pay the employee: ₹", calcengine(1, 0))
                            print("-------------------------------------")
                            time.sleep(1)
                            print("Scan the code in a UPI app to pay")
                            time.sleep(1)

                            os.system(currdir+'\\payqr.png')

                            time.sleep(5)
                            paycheck = input("Mark salary as 'PAID'? (y/n): ")
                            if paycheck == "y":
                                print("Salary paid!")
                                from datetime import datetime, date
                                now = datetime.now()
```

```
                                dt_string = now.strftime("%d")  #datetime object containing current date and time
                                empmas = sqlite3.connect(r'dbfaempmaster.db')
                                empmascur = empmas.cursor()
                                empmascur.execute("insert into salpay(Oid, Date, Amount) values(?, ?, ?)", (emppay, dt_string, calcengine(1, 0)))
                                empmas.commit()
                            else:
                                print("Not paid. ")


                        else:
                            print("Salary payments can only be made between 01st - 05th of each month. ~")



                    if empfac == "12":
                        print("Returning to DBFA Main.. ")
                        time.sleep(1)
                        break

                    #else:
                        #import os
                        #os.system('cls')

            else:
                print("Invalid password! ")


        #CIT
        elif decfac == "11":
            if str(settingscommondumpfetch(10)) == "Administrator":
                import time, os, requests

                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

                currdir = str(os.getcwd())
                from pathlib import Path
                path = Path(os.getcwd())
                parentdir = str(Path(path.parent))
                #print(parentdir)
                #print(parentdir+'\\updates.txt')
                userdir = os.path.expanduser('~')
                url = "https://raw.githubusercontent.com/deltaonealpha/DBFA_UpdateHandler/master/updates.txt"
                try:
                    r = requests.get(url)
                    dbfaver = ((str(r.content.decode('utf-8'))))[4:]
                    xdbfaver = ((str(r.content.decode('utf-8'))))
                    with open(parentdir+'\\updates.txt', 'r+') as upread:
                        upread = (str(upread.read())).strip()
```

```
                    spass1 = []
                    spass2 = []
                    for i in dbfaver:
                        spass1.append(i)
                    for j in upread[4: ]:
                        spass2.append(j)
                    if float(upread[4: ]) > float(dbfaver):
                        pass
                    else:
                        if xdbfaver == upread:
                            uptd = 1
                        elif spass1 != spass2:
                            uptd = 2
                    netcon = 1
                except (requests.ConnectionError, requests.Timeout) as exception:
                    netcon = 0


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

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

                os.system('cls')
                from colorama import init, Fore, Back, Style #color-settings
                print(Fore.RED+"DBFA Security Options are designed to execute functions which may render your DBFA installation completely unusable/ delete all data/ crash. Proceed with caution.")
                time.sleep(3)
                os.system('cls')
                init(convert = True)
                print(Fore.WHITE+'''

```

```
==========================================================================================
DBFA Security Dashboard
deltasoftware
==========================================================================================
'''+Fore.RED+'''Administrator Access Granted
==========================================================================================''''''+Fore.WHITE)
                if netcon == 1:
                    if uptd == 1:
                        print(Fore.GREEN+'''
                                                    Security Status  : DBFA is up-to-date ~
                                                    File Integrity   : Acceptable
                                                    Storage Integrity: Strong
                                                    Webhook Health   : Strong (INTERNET CONNECTED)
                    '''+Fore.CYAN)
                    else:
                            print(Fore.RED+'''
                                                    Security Status  : DBFA is running on an OUTDATED build!
                                                    File Integrity   : Acceptable
                                                    Storage Integrity: Strong
                                                    Webhook Health   : Strong (INTERNET CONNECTED)
                    '''+Fore.CYAN)
                elif netcon == 0:
                    print(Fore.RED+'''
                                                    Security Status  : Connection to server unavailable!
                                                    File Integrity   : Acceptable
                                                    Storage Integrity: Strong
                                                    Webhook Health   : DISCONNECTED (NO INTERNET)
                    '''+Fore.CYAN)
                print(Fore.WHITE+'''
a - '''+Fore.RED+'''DELETE ALL'''+Fore.WHITE+' customer records
                    :'+Fore.WHITE, '|'+Fore.RED+"▄▄ Proceed > "+Fore.RED+'''|'''+Fore.CYAN+'''
b - '''+Fore.RED+'''DELETE ALL'''+Fore.WHITE+' store records
                   :'+Fore.WHITE, '|'+Fore.RED+"▄▄ Proceed > | "+Fore.CYAN)
                print('''
c - Change '''+Fore.RED+'''ADMINISTRATOR'''+Fore.CYAN+''' password
d - Change password for employee accounts >>
e - Create more login accounts (UAC - sls-level-permset ) >>\n''')
                #if (settingscommonfetch(5)) == 0:
                print(Fore.MAGENTA+"f - Enable database encryption
                       : UNDER DEVELOPMENT "+Fore.CYAN+'      ')
                #else:
                    #print("e - Enable database encryption                        :", (Fore.RED+'|▄▄▄▄'+Fore.RED+' OFF|      '+Fore.WHITE))
                if (settingscommonfetch(6)) == 1:
```

```
                    print(Fore.CYAN+"g - Enable DBFA Secure Two-Factor-Authenication:", Fore.GREEN+'| ON '+Fore.GREEN+'[redacted]|'+Fore.CYAN+')
                else:
                    print(Fore.CYAN+"g - Enable DBFA Secure Two-Factor-Authenication:", (Fore.RED+'|[redacted]'+Fore.RED+' OFF|      '+Fore.CYAN))

                print('''\nh - <<< Return to main menu '''+Fore.RED+'''==========================================================================================''')
                dsoa = input("What would you like to do?: "+Fore.WHITE)
                if dsoa in ("a", "A"):
                    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("Confirm operation - DELETE all customer records? ")
                    print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
                    settfac1x = input(("n:     "+ '|'+Fore.RED+'[redacted]'+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(0.5)
                        os._exit(0)
                    elif settfac1x == "n":
                        print("Cancelled ~")
                    else:
                        print("That's an invalid input... ")

                if dsoa in ("b", "B"):
                    print('''This option PERMANENTLY CLEARS ALL DBFA VOUCHERS/ COUPONS
                All current vouchers/ coupons WILL BE LOST.
                Vouchers already issued will become redundant unless manually re-added again.
                Validity and usage limits will be lost for all voucher/ coupon instanced recorded by DBFA.

                However, DBFA's logged voucher/ coupon usage will continue to exist in memory and will not be erased.

                This execution can NOT BE REVERSED.
                DATA INTEGRITY MAY BE LOST during this process.
                Proceed with caution! ''')
                    print(" ")
                    print("Confirm operation - DELETE all store records? ")
                    print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
                    settfac1x = input(("n:     "+ '|'+Fore.RED+'[redacted]'+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'securepackxvc.py')
                        time.sleep(0.5)
                        os._exit(0)
                    elif settfac1x == "n":
                        print("Cancelled ~")
                    else:

                        print("That's an invalid input... ")

                    print()
                if dsoa in ("c", "C"):
                    print("Change password for ADMINISTRATOR account ~ ")
                    time.sleep(0.5)
                    import sqlite3
                    settings = sqlite3.connect(r'dbfasettings.db')
                    settingsx = settings.cursor()
                    password = input("Enter new password for *ADMINISTRATOR ACCOUNT*: ")
                    passworfconf = input("Confirm password: ")
                    if password == passworfconf:
                        confac = input("Change password for the administrator account? (y/n): ")
                        if confac in ("y", "Y"):
                            settingsx.execute(("UPDATE passkeyhandler SET Col1 = ? WHERE Sno = ?"), (passworfconf, 6))
                            settings.commit()
                        else:
                            print("Password change cancelled ~ ")
                    else:
                        print("Password confirmation invalid! ~ ")
                if dsoa in ("d", "D"):
                    print("Option under development! ")
                if dsoa in ("e", "E"):
                    print("User Account Creation Wizard ")
                    import sqlite3
                    settings = sqlite3.connect(r'dbfasettings.db')
                    settingsx = settings.cursor()
                    settingsx.execute("SELECT MAX(Sno) FROM passkeyhandler")
                    sno = (int(settingsx.fetchall()[0][0]) + 1)
                    uid = input("Enter username for new account: ")
                    kpass = input("Enter password for:" + str(uid) + ": ")
                    confkpass = input("Confirm password: ")
                    if kpass == confkpass:
                        print("Alloting permission set 'sales_group' to ", uid)
                        settingsx.execute(("INSERT INTO passkeyhandler(Sno, Col1, Col2) VALUES (?, ?, ?)"), (sno, uid, kpass))
                        settings.commit()
                        time.sleep(2)
                        print("User account registered. Reboot DBFA Client from login to view changes.")
                    else:
                        print("Wrong re-entry of password. Retry from Employee Manager.")

                if dsoa in ("f", "F"):
```

```
                    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.\n''')

                if dsoa in ("g", "G"):
                    print("----DBFA 2FA MANAGER----")
                    print("DBFA allows you to use 2FA to safeguard your account if your password gets compromised.\n")
                    print("Please do note that enabling/ disabling 2FA will reboot DBFA Store Manager!~\n")
                    import os
                    print(Fore.WHITE+"Connecting to your delta account. . ."+Fore.WHITE)
                    time.sleep(2)
                    print("Changing 2FA settings requires 2FA to proceed: ")
                    time.sleep(1)
                    os.startfile('modif2fa.py')
                    time.sleep(0.5)
                    os._exit(0)
            else:
                print("This option is restricted on your account! ~ ")



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

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

```

```
        elif decfac in ("2c", "2C", "2 c", "2 C"):
            del2c()
            logger.write("-------------------------------------- \n")
            logger.write("\nDBFA Customer Purchase Records accessed! \n")

        elif decfac in ("2d", "2D", "2 d", "2 D"):
            del2d()
            logger.write("-------------------------------------- \n")
            logger.write("\nCustomer search used. \n")

        elif decfac in ("2e", "2E", "2 e", "2 E"):
            if str(settingscommondumpfetch(10)) == "Administrator":
                del2e()
                logger.write("\n\n-------------------------------------- \n")
                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
                logger.write("Customer data exported to CSV! ")
                with HiddenPrints():
                    try:
                        sender = telegram_bot_sendtext(dt_string + "\n" + "Sales data exported to CSV- REDFLAG Urgent Security Notice!")
                        print(sender)
                    except Exception:
                        pass
                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
                logger.write("-------------------------------------- \n\n\n")
            else:
                print("This function is restricted on your account ")



        # Direct Calls Section - 3
        elif decfac in ("3a", "3A", "3 a", "3 A"):
            del3a()

        elif decfac in ("3b", "3B", "3 b", "3 B"):
            del3b()

        elif decfac in ("3c", "3C", "3 c", "3 C"):
            if str(settingscommondumpfetch(10)) == "Administrator":
                del3c()
            else:
                print("This function is restricted on your account.")

        elif decfac in ("3d", "3D", "3 d", "3 D"):
            del3d()

        elif decfac in ("3e", "3E", "3 e", "3 E"):
            if str(settingscommondumpfetch(10)) == "Administrator":
                del3e()
                logger.write("\n-------------------------------------- \n")
                logger.write("Sales log accessed! ")
            else:
                print("This function is restricted on your account.")


        elif decfac in ("3f", "3F", "3 f", "3 F"):
            if str(settingscommondumpfetch(10)) == "Administrator":
                del3f()
                logger.write("\n\n-------------------------------------- \n")
                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
```

```
                with HiddenPrints():
                    try:
                        sender = telegram_bot_sendtext(dt_string + "\n" + "Customer data exported to CSV- REDFLAG Urgent Security Notice!")
                        print(sender)
                    except Exception:
                        pass
                logger.write("Sales data exported to CSV! ")
                logger.write("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! \n")
                logger.write("-------------------------------------- \n\n\n")
            else:
                    print("This function is restricted on your account.")

        elif decfac in ("3g", "3G", "3 f", "3 F"):
            if str(settingscommondumpfetch(10)) == "Administrator":
                print("Deep Archival Engine")
                del3g()
            else:
                    print("This function is restricted on your account.")

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

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

except (requests.ConnectionError, requests.Timeout) as exception:
    import os, time
    os.system('cls')
    from colorama import init, Fore, Back, Style #color-settings
    print(Fore.RED+"We at DBFA take pride in our security. \nWhile we want to provide everyone with the best DBFA experience ever, we cannot cut on security.")
    print("Due to this commitment of ours, DBFA cannot function without an active internet connection.")
    init(convert = True)
    print(Fore.RED+'''
            ▄▄▄▄▄▄▄▄
        ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄                    Security Status  : Connection to server unavailable!
        ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄                    File Integrity   : Acceptable
        ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄                    Storage Integrity: Strong
        ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄                    Webhook Health   : DISCONNECTED (NO INTERNET)
            ▄▄▄▄▄▄▄▄
            ▄▄▄▄▄▄▄▄''')

    print("\nPlease arrange the same and re-run DBFA Client."+Fore.WHITE+"\n\nDBFA Error Code: del1012rt-naincon\n\nYou may use this error code for further reference/support. Please note it down. DBFA will automatically exit in 200 seconds.")
    time.sleep(200)
except:
    try:
        import os, requests, time
        os.system('cls')
        error_message = traceback.format_exc()
        print("DBFA has crashed due to unexpected reasons. ")
        print("DBFA will now automatically check its installation for integrity issues:: ")
        time.sleep(1)
        print("Error recieved from interpreter follows:")
```

```
        print("    ", error_message)
        time.sleep(3)
        time.sleep(2)
        from md5checker import make_hash

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

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

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

        print("Fetching logged MD5.")
        try:
            localdump = str(open(parentdir+'\\md5').read())
            sorteddump = (localdump.split())
            print("Arranging hashes ###")
            local_md5 = []
            local_md5.append(sorteddump[0].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[1].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[2].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[3].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[4].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[5].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[6].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[7].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[8].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[9].replace(" ", "").replace("\n", ""))
            local_md5.append(sorteddump[10].replace(" ", "").replace("\n", ""))
        except:
            print("DBFA's code has been tampered with! Please rectify this to avoid such program crashes!")
            print("    rrtt - Master Copy Error: dta=intl.err_instldir?imp=md5lognotfound")


        print("Hashing MD5 from deployed code.")
        live_md5 = []
```

```
        live_md5.append(make_hash('bleading_edge.py', algo='md5'))
        live_md5.append(make_hash('modif2fa.py', algo='md5'))
        live_md5.append(make_hash('dbfaempman.py', algo='md5'))
        live_md5.append(make_hash('plotter.pyw', algo='md5'))
        live_md5.append(make_hash('delauth.py', algo='md5'))
        live_md5.append(make_hash('dbfabackupper.py', algo='md5'))
        live_md5.append(make_hash('authtimeout.pyw', algo='md5'))
        live_md5.append(make_hash('securepack.py', algo='md5'))
        live_md5.append(make_hash('securepackxvc.py', algo='md5'))
        live_md5.append(make_hash('wrelogin.pyw', algo='md5'))
        live_md5.append(make_hash('run_DBFA.pyw', algo='md5'))
        print("Arranging hashes ###")


        alphas = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

        for i in range(len(listing)):
            if live_md5[i] != local_md5[i]:
                print("! - Misaligned MD5 at")
                print(live_md5[i], local_md5[i], "////\n")
        print("Alignment check done p1x")

        for i in server_md5:
            arter = ""
            for j in i.replace(" ", "").replace("\n", ""):
                if str(j).isalpha() is True:
                    arter += '%d'%(alphas.index(j.lower()))
                else:
                    arter += j
            server_md5[server_md5.index(i)] = int(arter)

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

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

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

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

```
        elif live_md5 == local_md5:
            marker = 2
        else:
            marker = 2

        time.sleep(3)

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

    except IndexError:
        print("---------------------------------------------")
        print("dnss internal error: (error code - rep.pvt-01)")
        time.sleep(10)


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

<Deep Archival Services Engine>

// The Code 2/12

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

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

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

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

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

    ordernet = orders.split('.')

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

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


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

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

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

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

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

    from tabulate import tabulate

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

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

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

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

// The Code 3/12

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

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

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

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

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

        CAPSLOCK = CAPSLOCK_STATE()

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

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

        else:
            layout = [  [sgx.Text('Yo we know security...')],
                        [sgx.Text('Username: '), sgx.InputText()],
                        [sgx.Text('Password: '), sgx.InputText(password_char='*')],
                        [sgx.Button('Login'), sgx.Button('Cancel')]]
        # Create the Window
        window = sgx.Window('DNSS Authenication Service', layout)
        # Event Loop to process "events" and get the "values" of the inputs
    while True:
        event, values = window.read()
        if event in (None, 'Cancel'): # if user closes window or clicks cancel
            window.close
            break
        window.close()
        window.close()
        if values[0] == 'ed' and values[1] == 'edd':
            #os.close(r'DDD.py')
            window.close()
            userblock = open(r"userblock.txt","a+") #Opening / creating (if it doesn't exist already) the .txt record file
            userblock.write('ed')
            #time.sleep(2)
            OAuthset()
            userblock.close()
            print("logging success")
            os.startfile('bleading_edge.py')
            window.close()
            window.close()
            exit
            exit
            exit
            break
        else:
 os.startfile(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\wrelogin.pyw')
        #window.close
        #erraise()
import PySimpleGUI as sg
if os.path.exists(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.txt'):
    os.remove(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.txt')
if os.path.exists(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.zconf'):
 os.remove(r'C:\Users\balaj\OneDrive\Documents\GitHub\DBFA\master\userblock.zconf')
sg.theme('DarkTeal9') # Add a touch of color
# All the stuff inside your window.
layout = [  [sg.Text("DBFA Security")],
            [sg.Text("This program requires you to login.")],
            [sg.Button('Login'), sg.Button('Exit')] ]
# Create the Window
```

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

```

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

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

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

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

    CAPSLOCK = CAPSLOCK_STATE()

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

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

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

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

Login()

```

<Database Deletion Services>

// The Code 4/12

```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")
```

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

x = "cponmgmtsys.db"

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

```

<Backup Authentication Provider Service (1/2)>

// The Code 5/12

```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("")
```

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

                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"
                sender = telegram_bot_sendtext(texter)
                main()
        else:
            sgx.theme('DarkRed')
            Login()


def get_updates(offset=None):
    global updates
    url = URL + "getUpdates"
    if offset:
        url += "?offset={}".format(offset)
    js = get_json_from_url(url)
    updates = js
    return updates


def get_last_update_id(updates):
    global last_update_id, valn
    update_ids = []
    for update in updates["result"]:
        update_ids.append(int(update["update_id"]))
    try:
        last_update_id = max(update_ids)
        lns = open(r"lastupdateid.txt", "r+")  #Opening / creating (if it doesn't exist already) the .txt record file
        lns.truncate(0)
        lns.write('%d'%last_update_id)
    except ValueError:
        lns = open(r"lastupdateid.txt", "r+")  #Opening / creating (if it doesn't exist already) the .txt record file
        last_update_id = int(lns.read())+1




def echo_all(updates):
    global last_update_id
    for update in updates["result"]:
        text = update["message"]["text"]
        chat = update["message"]["chat"]["id"]
        global valn
        #print(valn)
        if valn == 0:
            if text == "ed":
                valn = 1
                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)
                ins = open(r"delauth.txt", "a+")  #Opening / creating (if it doesn't exist already) the .txt record file
```

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

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



def get_last_chat_id_and_text(updates):
    global last_update_id

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


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



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

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


Login()

```

<Backup Creation Service (2/2)>

// The Code 6/12

```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\\DBFATempc'):
 shutil.rmtree(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATempc', ignore_errors=True)
        if os.path.exists(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Backup&Switchc'):
 shutil.rmtree(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFA_Backup&Switchc', ignore_errors=True)
        if not os.path.exists(r'C:\\Users\\balaj\\OneDrive\\Documents\\GitHub\\DBFA\\master\\DBFATempc'):
 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_Backup&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.")

```

<Update Deployment Services>

// The Code 7/12

```python
import requests, os, time, shutil, oschmod
os.system('cls')
print('''---------------------------------
 delta Update Deployment Service
---------------------------------
Checking for updates. . .
---------------------------------''')
time.sleep(1)

import os, time
global curdir, parentdir, fontsdir
currdir = str(os.getcwd())
from pathlib import Path
path = Path(os.getcwd())
parentdir = str(Path(path.parent))
#print(parentdir)
#print(parentdir+'\\updates.txt')
userdir = os.path.expanduser('~')
url = "https://raw.githubusercontent.com/deltaonealpha/DBFA_UpdateHandler/master/updates.txt"
r = requests.get(url)
dbfaver = ((str(r.content.decode('utf-8'))))[4:]
xdbfaver = ((str(r.content.decode('utf-8'))))
with open(parentdir+'\\updates.txt', 'r+') as upread:
    upread = (str(upread.read())).strip()

print("Server: ", dbfaver, "\nLocal: ", upread[4: ])
time.sleep(1)
spass1 = []
spass2 = []
for i in dbfaver:
    spass1.append(i)
for j in upread[4: ]:
    spass2.append(j)
if float(upread[4: ]) > float(dbfaver):
    pass

else:
    if xdbfaver == upread:
        print("This installation of DBFA is already up-to date ~ \n")
        dddu_ddu = input("Press 'enter' to restart DBFA: ")
        os.startfile('run_DBFA.pyw')

    elif spass1 != spass2:
        time.sleep(1)
        print("A new DBFA update is available: DBFA", dbfaver, "\nUpdate Notes: \n")
        clog = requests.get('https://raw.githubusercontent.com/deltaonealpha/DBFA_UpdateHandler/master/updatechangelog.txt')
        print(str(clog.content.decode('utf-8')), "\n\n")
        time.sleep(2)
        updateconfo =  input("Update DBFA now? (y/n): ")
        if updateconfo == "y":
            try:
                os.remove(currdir+'\\updatertemp')
                shutil.rmtree(currdir+'\\updatertemp', ignore_errors=True)
                time.sleep(0.000001)
                os.remove(currdir+'\\DBFA_UpdateHandler')
                shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
```

```
            except:
                pass
            if os.path.exists(currdir+'\\DBFA_UpdateHandler'):
                shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
                shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
            if os.path.exists(currdir+'\\DBFA_UpdateHandler'):
                os.startfile(currdir+'\\DBFA_UpdateHandler')
                dddu_ddu = input("Please go to DBFA's installation folder and delete the folder 'DBFA_UpdateHandler'. Press enter once done: ")
            try:
                oschmod.set_mode(currdir+'\\DBFA_UpdateHandler', "777")
            except:
                pass
            shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
            try:
                os.rmdir(currdir+'\\DBFA_UpdateHandler')
            except:
                pass
            if os.path.isdir(currdir+'\\DBFA_UpdateHandler') == True:
                shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
                print("Cleaning-up previous update packages... ")
                shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
                try:
                    os.rmdir(currdir+'\\DBFA_UpdateHandler')
                except:
                    pass
            else:
                pass
            if os.path.exists(currdir+'\\DBFA_UpdateHandler'):
                shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
            try:
                os.system('git clone https://github.com/deltaonealpha/DBFA_UpdateHandler')
            except:
                print("The directory 'DBFA_UpdateHandler' already exists.")
                print("Please delete it from DBFA's installation location and re-run the updater.")
            #os.system('git log')
            print("Commit: ")
            os.system(r'git rev-parse HEAD')
            file_path = currdir+'\\DBFA_UpdateHandler\\updatepackage.zip'
            import os, shutil, time
            if os.path.exists(currdir+'\\updatertemp'):
                shutil.rmtree(currdir+'\\updatertemp', ignore_errors=True)
            os.makedirs(currdir+'\\updatertemp', mode = 0o777)
            import zipfile
            with zipfile.ZipFile(file_path, 'r') as zip_ref:
                zip_ref.extractall(currdir+'\\updatertemp')
            from tqdm import tqdm
            import shutil, os, time, sys
            slave = currdir
            for i in tqdm (range (100), desc="Updating DBFA: "):
                for j in os.scandir(currdir+'\\updatertemp'):
                    master = currdir+'\\updatertemp\\'+str(j).replace("<DirEntry '", "").replace("'>", "")
                    shutil.copy(master, slave)
                    time.sleep(0.00001)
                master = currdir+'\\DBFA_UpdateHandler\\updates.txt'
                shutil.copy(master, parentdir)
```

```
            for i in tqdm (range (100), desc="Cleaning up: "):
                shutil.rmtree(currdir+'\\updatertemp', ignore_errors=True)
                time.sleep(0.000001)
                shutil.rmtree(currdir+'\\DBFA_UpdateHandler', ignore_errors=True)
            print("Your DBFA installation has been succesfully updated to DBFA", dbfaver, ".")
            dddu_ddu = input("Press 'enter' to restart DBFA: ")
            os.startfile('run_DBFA.pyw')

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

<Hire-Employee GUI Form Script>

// The Code 8/12

```python
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-')]]

```

```python

layout =   [[sg.Text('Hire an employee')],
            #### Section 1 part ####
            [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')],
            [collapse(section1, '-SEC1-')],
            #### Section 2 part ####
            [sg.T(SYMBOL_DOWN, enable_events=True, k='-OPEN SEC2-', text_color='white'),
            sg.T('Personal Details', enable_events=True, text_color='white', k='-OPEN SEC2-TEXT')],
            [collapse(section2, '-SEC2-')],
            #### Buttons at bottom ####
            [sg.Button('Proceed'), sg.Button('Exit')]]


layout3 = [[sg.Text('Hire an employee')],
            [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')],
            [collapse(section3, '-SEC3-')],
            [sg.Button('Complete Form >>>'), sg.Button('Exit')]]

window = sg.Window('deltaDBFA 8.2 - Add New Employee', layout)
arter = 0
opened1, opened2 = True, True

while True:             # Event Loop
    event, values = window.read()
    eventx, valuesx = event, values
    #print(event, values)
    if event == sg.WIN_CLOSED or event == 'Exit':
        break

    if event.startswith('-OPEN SEC1-'):
        opened1 = not opened1
        window['-OPEN SEC1-'].update(SYMBOL_DOWN if opened1 else SYMBOL_UP)
        window['-SEC1-'].update(visible=opened1)

    if event.startswith('-OPEN SEC2-'):
        opened2 = not opened2
        window['-OPEN SEC2-'].update(SYMBOL_DOWN if opened2 else SYMBOL_UP)
        window['-OPEN SEC2-CHECKBOX'].update(not opened2)
        window['-SEC2-'].update(visible=opened2)

    if  event.startswith('Proceed'):
        axt = open(r"dbfaempre.txt", "w+")
        axt.write(str(eventx))
        axt.write(str(valuesx))
        #print(eventx, valuesx)
        global ds1
        ds1 = (eventx, valuesx)
        window.close()
        window = sg.Window('deltaDBFA 8.2 - Add New Employee', layout3)
        opened1, opened2 = True, True
        while True:             # Event Loop
            event, values = window.read()
            eventr, valuesr = event, values
            global ds2
```

```python
            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)

```

```python
    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 9/12

```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")


```

```python
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()

def getOTP():
    digits = "0123456789"
    otp = ""
    otp += (digits[math.floor(random.random() * 10)])
    otp += (digits[math.floor(random.random() * 10)])
    otp += (digits[math.floor(random.random() * 10)])
    otp += (digits[math.floor(random.random() * 10)])
    otp += (digits[math.floor(random.random() * 10)])
    otp += (digits[math.floor(random.random() * 10)])
    return otp

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

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

    # Start the Bot
    updater.start_polling()

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

def 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
            print("DBFA 2FA Modification cancelled!! ")
            time.sleep(5)
            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("delta2FAAuthenicationAPI-lite")
            print("As you're accessing sensitive information, this request needs to be validated.")
            time.sleep(2)
```

```
            print("")
            print("A one-time-password (OTP) has been sent to your Telegram account to validate this request")
            time.sleep(0.5)
            for i in tqdm (range (10), desc="Generating and sending OTP"):
                time.sleep(0.00001)
            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"
            sender = telegram_bot_sendtext(texter)
            window.close()
            window.close()
            mainprocess()

        else:
            sgx.theme('DarkRed')
            Login()

def mainprocess():
    time.sleep(1)
    tries = 0
    while(1):
        passkeyinput = input("Enter the validation key recieved: ")
        if tries in (0, 1, 2, 3, 4):
            if passkeyinput == delsecox:
                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")
                tries += 1
                time.sleep(1)
                from colorama import init, Fore, Back, Style
                print("\n\nEnable DBFA two-factor-authenication? ")
                print("y:    ",  '| ON '+Fore.GREEN+'[redacted]'+Fore.WHITE+'|')
                settfac1x = input(("n:     "+ '|'+Fore.RED+'[redacted]'+Fore.WHITE+' OFF|: '))
                if settfac1x == "y":
                    if (settingscommonfetch(6)) != 1:
                        settingsmodifier(6, 1)
                        transitionprogress()
                        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)

                if settfac1x == "n":
                    if (settingscommonfetch(6)) != 0:
                        settingsmodifier(6, 0)
                        transitionprogressneg()
                        time.sleep(3)
```

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

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

delsecox = getOTP()
Login()
```

<Sales Analysis Provider Service>

```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()
```

<First Time Configuration Service>

// The Code 11/12

```python
# ------------------------------------------- #
# DBFA First Time Automatic Setup             #
# deltasystems deltasoftware DBFA Client Manager #
# Licensed under the GNUAPL License           #
# ------------------------------------------- #

import sqlite3, os, time, sys
os.system('cls')
time.sleep(1)

global curdir, parentdir, fontsdir
currdir = str(os.getcwd())
file_path = currdir+'\\initsetup.zip'

def mover():
    from tqdm import tqdm
    import shutil, os, time, sys
    slave = currdir
    for i in tqdm (range (100), desc="Restoring Databases                        "):
        master = currdir+'\\TempRestoration\\DBFATempc\\DBFA_vend.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\cponmgmtsys.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\DBFA.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\DBFA_CUSTCC.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\DBFA_handler.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\recmaster.db'
        shutil.copy(master, slave)
        time.sleep(0.0000000001)
        master = currdir+'\\TempRestoration\\DBFATempc\\invoicemaster.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\dbfaempmaster.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\dbfasales.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\dbfasettings.db'
        if os.path.exists(currdir+'\\delauth.txt'):
            os.remove(currdir+'\\delauth.txt')
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\DeepArchivalVault.db'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\recmaster.db'
        shutil.copy(master, slave)

    for i in tqdm (range (100), desc="Restoring Session Keys                     "):
        master = currdir+'\\TempRestoration\\DBFATempc\\backupconfig.txt'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\lastupdateid.txt'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\lastupdateid2.txt'
        shutil.copy(master, slave)
        if os.path.exists(currdir+'\\delauth.txt'):
            os.remove(currdir+'\\delauth.txt')
        master = currdir+'\\TempRestoration\\DBFATempc\\lastsched.txt'
        shutil.copy(master, slave)
        time.sleep(0.0000000001)

```

```python
    for i in tqdm (range (100), desc="Restoring Authorization Keys              "):
        master = currdir+'\\TempRestoration\\DBFATempc\\tempfile.temp'
        shutil.copy(master, slave)
        if os.path.exists(currdir+'\\delauth.txt'):
            os.remove(currdir+'\\delauth.txt')
        master = currdir+'\\TempRestoration\\DBFATempc\\client_secrets.json'
        shutil.copy(master, slave)
        time.sleep(0.0000000001)

    for i in tqdm (range (100), desc="Restoring Payment Keys                    "):
        master = currdir+'\\TempRestoration\\DBFATempc\\DBFAdeliveries.txt'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\payqr.png'
        if os.path.exists(currdir+'\\delauth.txt'):
            os.remove(currdir+'\\delauth.txt')
        shutil.copy(master, slave)
        time.sleep(0.0000000001)

    for i in tqdm (range (100), desc="Restoring Registery & Logs Files          "):
        master = currdir+'\\TempRestoration\\DBFATempc\\log.txt'
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\devclog.txt'
        if os.path.exists(currdir+'\\delauth.txt'):
            os.remove(currdir+'\\delauth.txt')
        shutil.copy(master, slave)
        master = currdir+'\\TempRestoration\\DBFATempc\\registry.txt'
        shutil.copy(master, slave)
        time.sleep(0.0000000001)
        time.sleep(0.0000000001)
    if os.path.exists(currdir+'\\TempRestoration'):
        shutil.rmtree(currdir+'\\TempRestoration', ignore_errors=True)
    print("DBFA Backup Restoration Completed! ~")

import os, shutil, time
if os.path.exists(currdir+'\\TempRestoration'):
        shutil.rmtree(currdir+'\\TempRestoration', ignore_errors=True)
os.makedirs(currdir+'\\TempRestoration', mode = 0o777)
import zipfile
with zipfile.ZipFile(file_path, 'r') as zip_ref:
        zip_ref.extractall(currdir+'\\TempRestoration')

print('''----------------------------------------
| DBFA Initialization Package detected! |
----------------------------------------''')
print("Getting ready to initialize DBFA Manager ~")
time.sleep(2)
mover()
print("DBFA files loaded succesfully ~")
time.sleep(2)

print('''
CLI
Store
Manager

DBFA First-Time Setup Wizard
(WARNING: DO NOT SKIP THIS!)''')
time.sleep(1)

def settingsmodifier(SettingsType, NewValue):
```

```
    import sqlite3
    settings = sqlite3.connect(r'dbfasettings.db')
    settingsx = settings.cursor()
    settingsx.execute(("UPDATE passkeyhandler SET Col1 = ? WHERE Sno = ?"), (NewValue, SettingsType))
    settings.commit()


time.sleep(1)
print("Welcome to DBFA, the most feature packed* open-source store manager out there!")
time.sleep(1)
print("With DBFA, we at deltasoftware aim to provide a smooth and open source experience with a focus on security, packaged along with some quirky features!")
time.sleep(1)
print("\nThis setup will guide you through getting some user-dependant settings ready for you to use DBFA.")
time.sleep(1)
print("Loading setup...")
time.sleep(5)
os.system('cls')
print("DBFA emails customers their invoice and OTPs for loyalty point usage. For this, we will need you to create a special *but free* email account.")
time.sleep(1)
time.sleep(1)
print('''Please follow these steps:
        1. Create a gmail account (different from your personal account; DO NOT use your personal password; this email will be exposed to customers)''')
initset_email = input("Enter the email ID: ")
initset_emailpass = input("Enter the email ID's password: ")
time.sleep(1)
settingsmodifier(5, str(initset_email))
settingsmodifier(1, str(initset_emailpass))
###############SAVING

print("DBFA sends access notifications, purchase records and OTPs for 2FA to the store owner on Telegram for security purposes.")
time.sleep(1)
print("We will guide you through getting this set-up.")
time.sleep(1)
print('''        1. Install Telegram on your mobile device (Android/ iOS/ HarmonyOS")
        2. Create a Telegram account.
        3. Now search for 'BotFather' on Telegram. (LINK: https://t.me/BotFather)
        4. Send '/newbot' to BotFather as a message.
        5. Now provide BotFather with the bot's name, (and optionally about and description)

        DBFA will use this bot to communicate with you.

        BotFather will now provide you with a 'HTTP API CODE' (EXAMPLE: will look like 123124gk:19846138469369468136496134134613 4)
        ''')
time.sleep(1)
initset_btoken = input("Enter the HTTP-ACCESS-TOKEN exactly as given including ':' here: ")
settingsmodifier(2, initset_btoken)
time.sleep(1)
###############SAVING
print("DBFA also maintains a Telegram group in which the bot sends shift schedules. Please follow instructions as given below for the same:")
```

```python
time.sleep(1)
print('''        1. Create a group in Telegram.
        2. Add the BOT you just created to that group.
        3. Now tap on the group's name, select 'Add Members' and search for 'Rose'. (LINK: https://t.me/MissRose_bot)
        4. Add this "Rose" bot to your group.
        5. After adding Rose, send '/id' in the group as a message. Rose will respond with a value to this.

        Please enter that value WITH THE NEGATIVE SIGN below:
        (NOTE: You may now remove Rose from the group if you don't want automatic group moderation.)
''')
time.sleep(1)
initset_gchatid = input("Paste the ID exactly as returned by Rose including the negative sign: ")
settingsmodifier(4, initset_gchatid)
time.sleep(1)
###############SAVING
print("For the DBFA bot to send you messages, search for the bot 'Get ID', and send it any text. It will respond with your 'Chat ID'. Please submit the same below: ")
initset_schatid = input("Enter your 'Chat ID' as recieved from the 'Get ID' bot: ")
settingsmodifier(3, initset_schatid)
time.sleep(1)
###############SAVING
print("Now please visit the bot you just created and click on start. You may now exit Telegram.")
time.sleep(1)
print("\n\nA few more things: ")
initset_adminpass = input("Enter an Administrator password (DO NOT REVEAL THIS TO YOUR EMPLOYEES): ")
settingsmodifier(6, initset_adminpass)
###############SAVING
time.sleep(1)
print("DBFA will now ask you for a username and password for your cashiers. They will use these to log-into DBFA.")
initset_cshruname = input("Enter a username for your cashiers: ")
settingsmodifier(7, initset_cshruname)
###############SAVING
initset_cshrpass = input("Enter a password for your cashier account: ")
settingsmodifier(8, initset_cshrpass)
###############SAVING
print("That's it! Please give us a few seconds to prepare your DBFA installation.")
time.sleep(2)
settingsmodifier(9, "1")
print("DBFA has been readied! Please use the shortcut on your desktop to start DBFA Client.")
```

<Process Handler GUI Prompts>

// The Code 12/12

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


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




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

Login()

```

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




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

Login()
```

```
59
60
61
```

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

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

```python


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()



```

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




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

```python

Login()


```

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

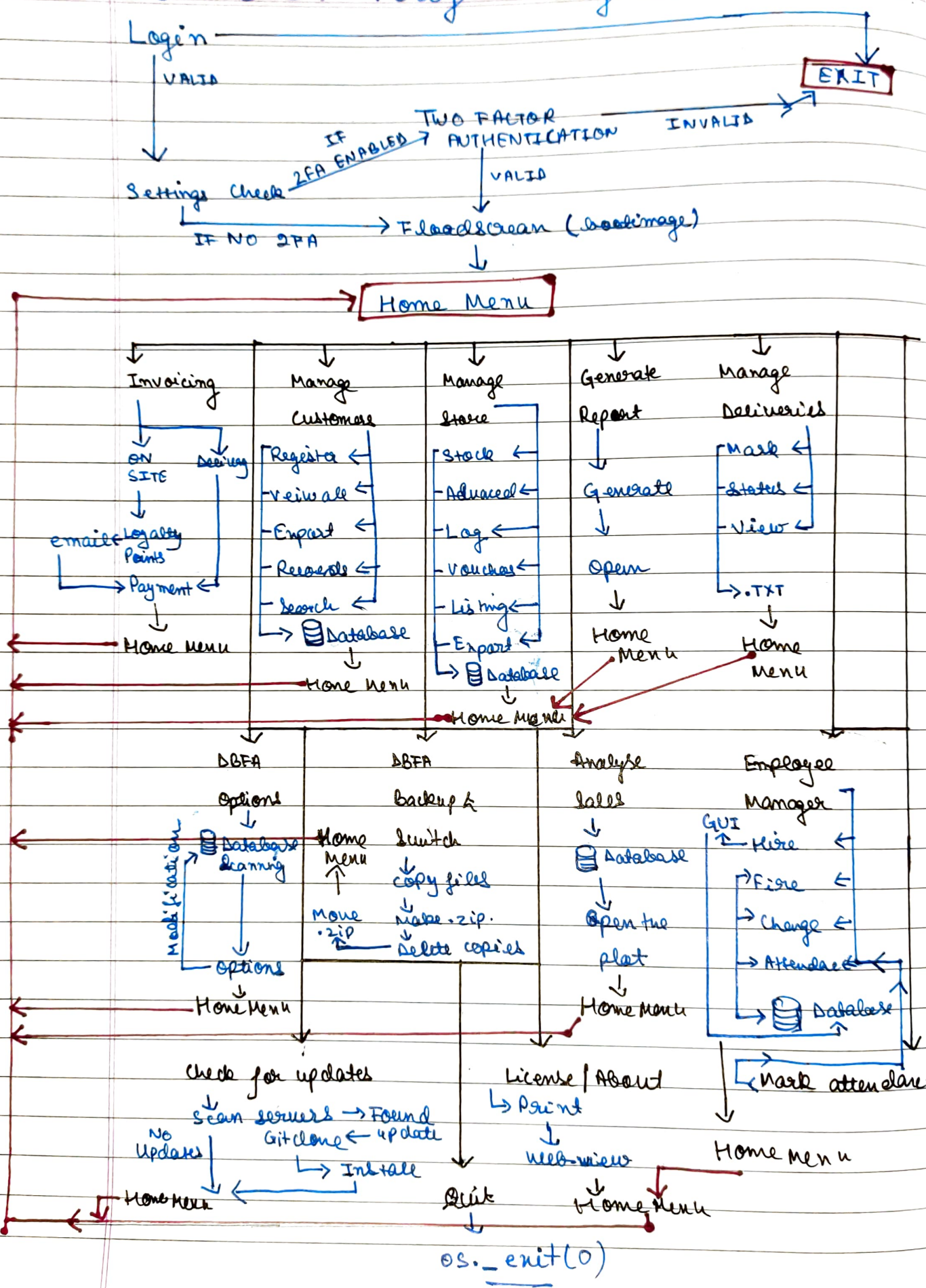
```
    os.remove(r'userblock.txt')
if os.path.exists(r'userblock.zconf'):
    os.remove(r'userblock.zconf')

Login()
```

(Please Turn Over)

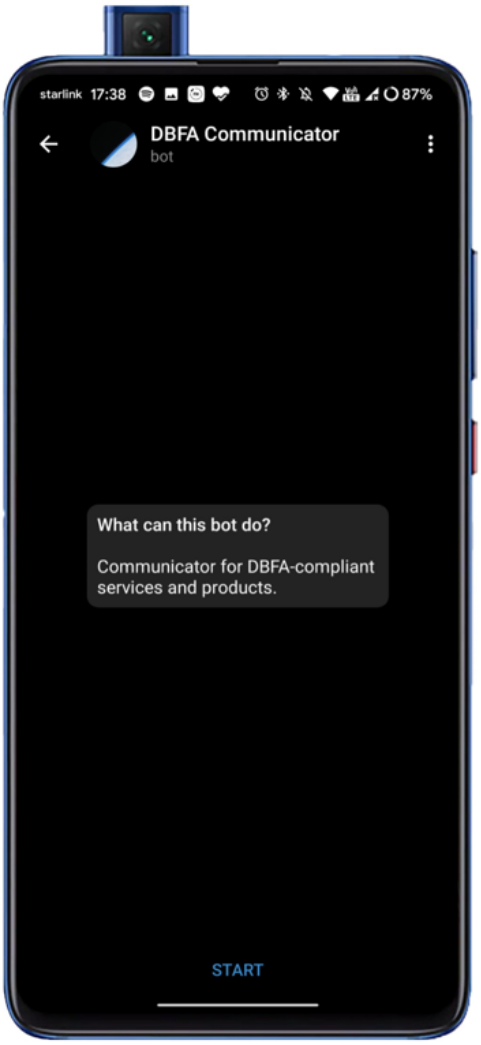
// Logic Chart

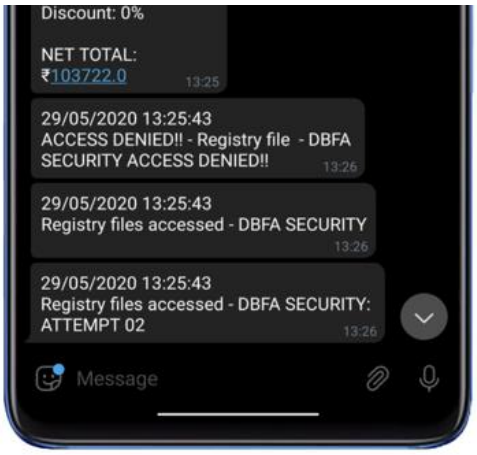
DBFA 8.34 Program Logic

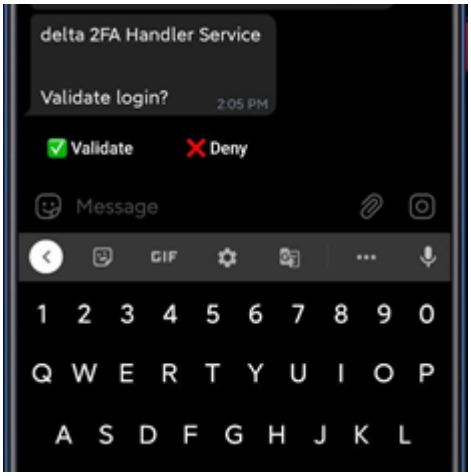


(Please Turn Over)

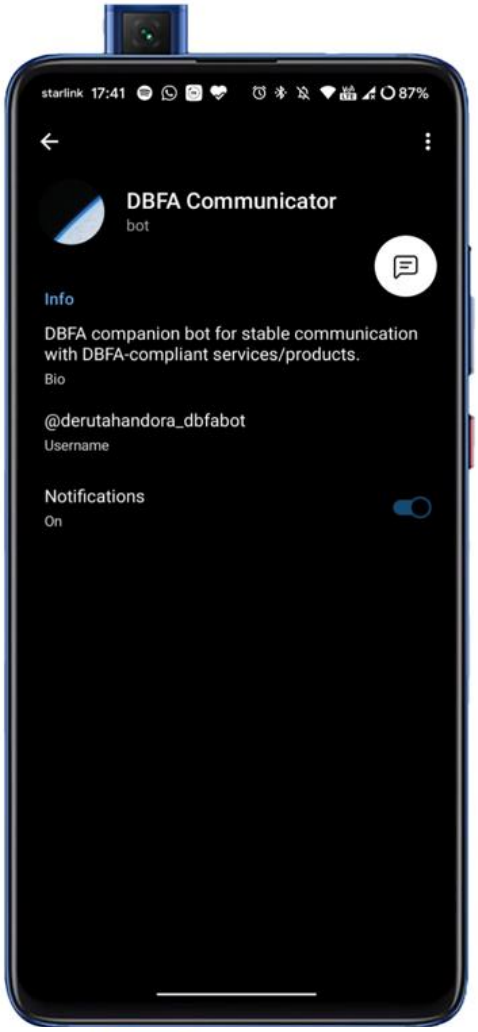
// Telegram Integration

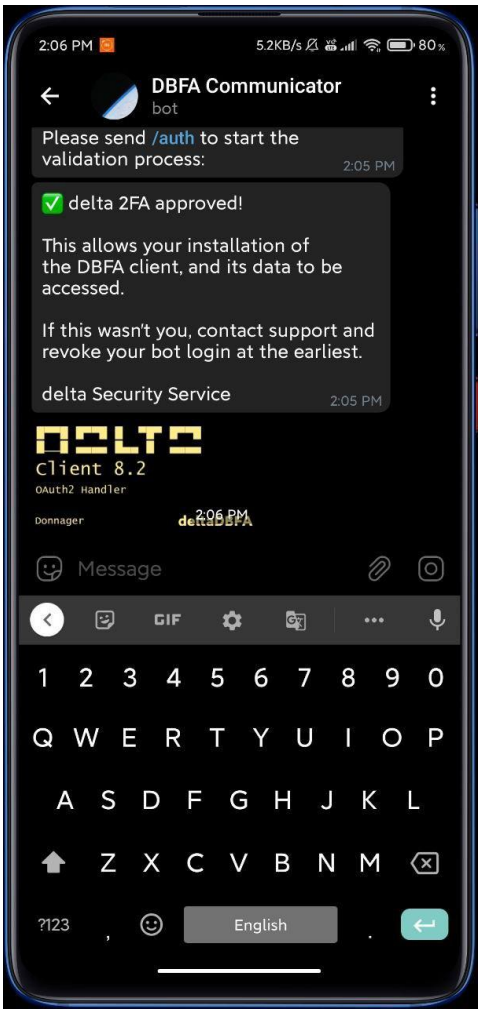

starlink 17:38
87%
DBFA Communicator
bot
What can this bot do?
Communicator for DBFA-compliant services and products.
START

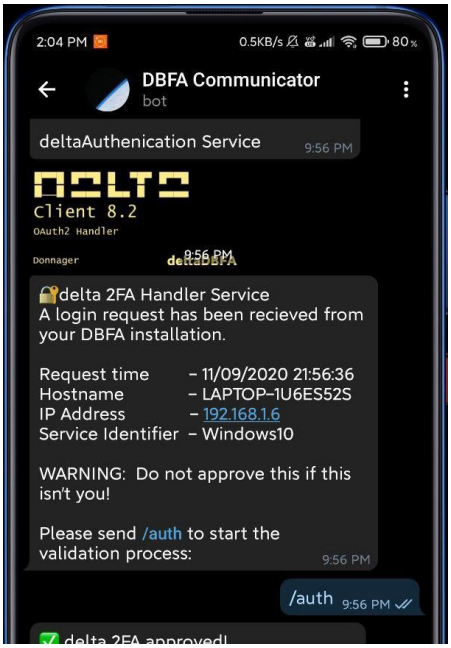

Discount: 0%
NET TOTAL:
₹103722.0
13:25
29/05/2020 13:25:43
ACCESS DENIED!! - Registry file - DBFA SECURITY ACCESS DENIED!!
13:26
29/05/2020 13:25:43
Registry files accessed - DBFA SECURITY
13:26
29/05/2020 13:25:43
Registry files accessed - DBFA SECURITY: ATTEMPT 02
13:26
Message

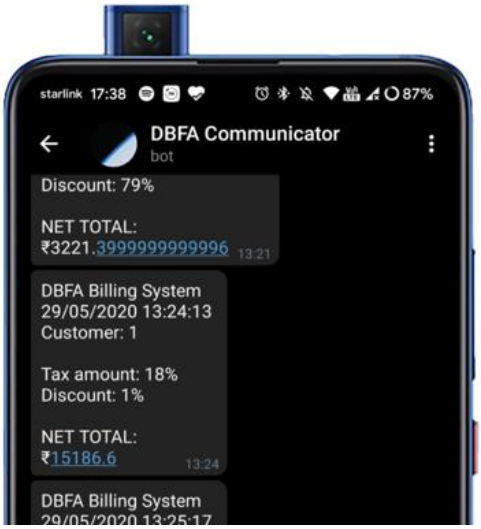

delta 2FA Handler Service
Validate login?
2:05 PM
Validate
Deny
Message
GIF

2:06 PM
5.2KB/s
80%
DBFA Communicator
bot
2:04 PM
Client 8.2
OAuth2 Handler
Donnager
2:06 PM
Message
GIF


starlink 17:41
87%
DBFA Communicator
bot
Info
DBFA companion bot for stable communication with DBFA-compliant services/products.
Bio
@derutahandora_dbfabot
Username
Notifications
On


2:06 PM
5.2KB/s
80%
DBFA Communicator
bot
Please send /auth to start the validation process:
2:05 PM
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
2:05 PM
Client 8.2
OAuth2 Handler
Donnager
2:06 PM
Message
GIF


2:04 PM
0.5KB/s
80%
DBFA Communicator
bot
deltaAuthenication Service
9:56 PM
Client 8.2
OAuth2 Handler
Donnager
9:56 PM
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:
9:56 PM
/auth 9:56 PM


starlink 17:38
87%
DBFA Communicator
bot
Discount: 79%
NET TOTAL:
₹3221.3999999999996
13:21
DBFA Billing System
29/05/2020 13:24:13
Customer: 1
Tax amount: 18%
Discount: 1%
NET TOTAL:
₹15186.6
13:24
DBFA Billing System

(Please Turn Over)

// The Future

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

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

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

DBFAx 0.01

(Please Turn Over)

// Bibliography

WORKS CITED FROM:

- Core Python Programming by R. Nageshwar Rao
- Python 3 Documentation

- Google – google.com
- StackOverflow – stackoverflow.com
- Programwiz - programiz.com

- Python for Class – XII by Preeti Arora

Find sample files here including store report, exported CSVs & store logs as generated by DBFA.

// Samples

DBFA Automatic Store Report:
This report has been automatically generated. This lists the profit earned, stock analytics and customer records as logged by DBFA.

DBFA synchronously updates its database alongwith algorithmic data interpretation to deliver these reports. This report contains information from the start of using DBFA on this system.

Report generated on: 2020/11/04

Installation Version: DBFA Inception 1.0

Security Check
DBFA Manager
Inception 1.0

Security Status : DBFA is up-to-date ~
File Integrity : Acceptable
Storage Integrity: Strong
Webhook health : Strong (connected)

Vouchers In-Circulation:

Voucher Identifier	Usage Left
A12R	9

Number of pending deliveries: 1

Employee IDs registered: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Attendance Records & Metrics:

Date	Employee OID	Time	Marked	In/Out
2020/10/04	1	07:18:00	Y	I
2020/10/04	1	19:18:11	Y	O
2020/10/06	1	11:04:14	Y	I
2020/10/06	1	21:04:22	Y	O
2020/10/22	1	21:22:16	Y	I
2020/10/22	1	21:22:21	Y	O

Most sold listing:

1	TV 4K OLED 50	2000	66	132000

Most profit making listing:

2	TV FHD OLED 50	4500	50	225000

Products running low on stock:

30	Polowski Tactical Flashlight	Stock Remaining:	3
41	KZ ZST X IEMs	Stock Remaining:	0
42	KZ ZSN X PRO IEMs	Stock Remaining:	0

Total profit per listing:

P.ID	Prod. Name	Profit P.U.	Qty. Sold	Net Profit
1	TV 4K OLED 50	2000	66	132000
2	TV FHD OLED 50	4500	50	225000
3	8K QLED 80	5700	27	153900
4	Redmi K20 PRO	2000	38	76000
5	Redmi K20	2100	52	109200
6	Redmi Note 9 PRO	1470	14	20580
7	POCOPHONE F	300	25	7500
8	Mi MIX ALPHA	11000	20	220000
9	Wireless Headphones	400	20	8000
10	Noise-Cancelling Wireless Headphones	2000	21	42000
11	Essentials Headphones	100	20	2000
12	Gaming Headphones	370	49	18130
13	Truly-Wireless Eadphones	450	2	900
14	Neckband-Style Wireless Earphones	120	37	4440
15	Essentials Earphones	50	8	400
16	Gaming Earphones	275	1	275
17	30W Bluetooth Speakers	649	28	18172
18	20W Bluetooth Speakers	140	14	1960
19	9Essentials Bluetooth Speaker	50	7	350
20	BOSE QC35	1050	3	3150
21	Essentials Home Theatre	978	15	14670
22	Wired Speaker - 5.	150	12	1800
23	Essentials Wired Speaker - STEREO	100	5	500
24	Tactical Power Bank 30000mah	320	12	3840
25	5Essentials Power Bank 0000mah	98	14	1372
26	Essentials Mouse	75	9	675
27	Logitech G604 LightSpeed Wireless	170	23	3910
28	Tactical Essentials Keyboard	60	17	1020
29	DROP GS21k RGB Gaming Keyboard	275	19	5225
30	Polowski Tactical Flashlight	90	5	450
31	OneFiber Wi-Fi Router AX7	210	7	1470
32	Mijia Mesh Wi-Fi Router	780	5	3900
33	lapcare 0W Laptop Adapter	50	4	200

34	lapcare 60W Laptop Adapter	35	9	315
35	Spigen Phone Case(s)	50	3	150
36	Essentials Phone Charger 150W	30	2	60
37	HyperPower Type-C Gallium-Nitride Charger 100W	100	7	700
38	ASUS Zephyrus G4 Gaming Laptop	8000	3	24000
39	L XPS 5 Content Creator's Laptop	9000	4	36000
40	Hewlett-Packard Essential's Student's Laptop (Chromebook)	1790	8	14320
41	KZ ZST X IEMs	140	0	0
42	KZ ZSN X PRO IEMs	240	0	0

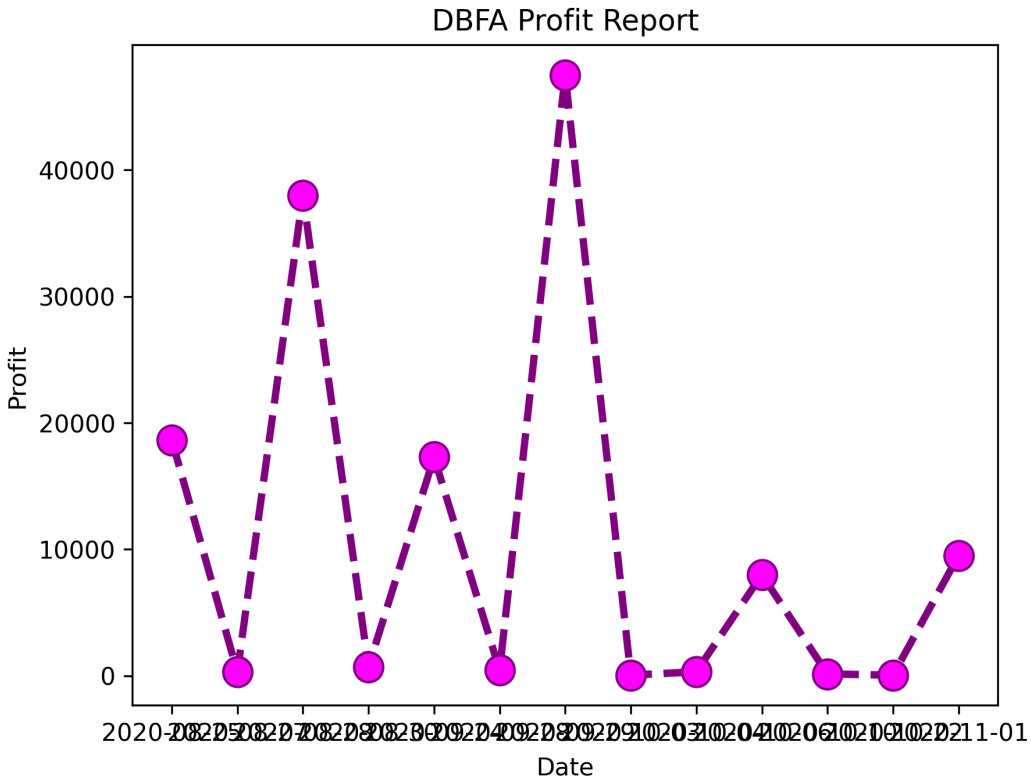
Customer purchases:

C. ID	C. Name	Purchases Made	Total Amount	Loyalty Points
0	Unregistered	11	283090.26	2803.87
1	Pranav	95	4474465.07	696.25
2	Sushant	5	147208.00	2428.25
3	Atharv	0	0.00	0.00
4	Pulkit	0	0.00	0.00
5	Binod	1	54280.00	460.00
6	Achintya	0	0.00	0.00
8	delta	0	0.00	0.00
9	Delta	0	0.00	0.00
10	Test	0	0.00	0.00
11	delta	0	0.00	0.00
12	delta	2	55932.00	474.00
13	delta	0	0.00	0.00

DBFA Stock Orders Report:
Product stock yet to be recieved:

P. ID	P. Name	Qty. to be delivered	Status	Vendor
2	TV FHD OLED 50	1	TBD	deltaSystems
21	Essentials Home Theatre	3	TBD	deltaSystems

DBFA Sales Analysis Plotter:
DBFA uses advanced data analysis algorithms to generate this plot. This is as per the latest data sets available.



Sales records for this month:

Sales No.	Cust. ID	Product IDs	Total Amount	Profit Earned	Date
36	1	15, 14,	3250.9	325	2020-10-04
37	1	37,	582.507	8000	2020-10-06
38	1	14,	1168.2	50	2020-10-10
39	1	14,	0	50	2020-10-10
40	1	14,	0	50	2020-10-10
41	1	14,	0	50	2020-10-22
42	12	1, 12,	51566	4950	2020-11-01
43	1	12, 14, 17, 19, 20, 21, 22, 22, 27, 28, 29, 30, 31, 32,	7828.52299999999	4573	2020-11-01
44	1	41,	849.5999999999999	0	2020-11-01

DBFA Deep Archival Engine Data (past month):

Inv. ID	Date	Key
204	22/10/2020 21:15:40	1400000004z4z5z0000010000022y1
127	28/09/2020 20:21:53	120020030033000004z5z12z20z1z0
179	29/09/2020 16:55:04	210000004z4z5z0000010000029y09
181	29/09/2020 16:59:29	120000004z4z5z0000010000029y09
184	29/09/2020 17:06:47	10000004z4z5z0000010000029y09y
186	29/09/2020 17:10:27	50000004z4z5z0000010000029y09y
188	29/09/2020 17:15:18	500700170027002800290000004z4z

custt	custname	email
0	Unregistered	unreg@dbfa.com
1	Pranav	balaji.pranav@outlook.in
2	Sushant	sush.gaming2012@gmail.com
3	Atharv	atharv@dbfa.com
4	Pulkit	pulkyeet@testdbfa.com
5	Binod	binod@dbfa.com
6	Achintya	achintya@dbfa.com
8	delta	admin@dbfa.com
9	Delta	delta@dbfa.com
10	Test	test@dbfa.com
11	delta	contactdelta@delta.com
12	delta	contactdelta@delta.com
13	delta	contactdelta@delta.com
custid		purchasecount
0	Unregistered	11
1	Pranav	93
2	Sushant	5
3	Atharv	0
4	Pulkit	0
5	Binod	1
6	Achintya	0
8	delta	0
9	Delta	0
10	Test	0
11	delta	0
12	delta	1
13	delta	0

Date and time: 09/08/2020 19:52:28

Date and time: 09/08/2020 19:52:28
Log file access attempt - Oauth failiure!!!

Date and time:
09/08/2020 19:52:28
Log file access attempt - AUTHORIZATION SUCCESS
/2020 22:26:32
Customer registry accessed!

DBFA Billing Framework by Pranav Balaji
Licensed under the GNU PUBLIC LICENSE
ed
Automated Store Registry:

Date and time: 27/05/2020 22:51:31
New customer registered: custname: Pranav Balaji custemail: test@dbfa.com
---

DBFA Billing Framework by Pranav Balaji
Licensed under the GNU PUBLIC LICENSE
ed
Automated Store Registry:

DBFA Billing Framework by Pranav Balaji
Licensed under the GNU PUBLIC LICENSE
ed
Automated Store Registry:

Date and time: 27/05/2020 22:52:34
Customer registry accessed!

DBFA Billing Framework by Pranav Balaji
Licensed under the GNU PUBLIC LICENSE
ed
Automated Store Registry:

Date and time: 27/05/2020 22:53:02
Customer registry accessed!

DBFA Billing Framework by Pranav Balaji
Licensed under the GNU PUBLIC LICENSE
ed
Automated Store Registry:

Appending product to order:
Essentials Earphones
Appending product to order:
Essentials Power Bank 10000mah
Appending product to order:
Redmi K20
Appending product to order:
Redmi K20
---
ed
Automated Store Registry:
---------- Cust. ID:
1
Date and time:
09/11/2020 19:44:30
---
ed
Automated Store Registry:
---------- Cust. ID:
1
Date and time:
09/11/2020 20:00:38
Appending product to order:
Gaming Headphones
Appending product to order:
Neckband-Style Wireless Earphones
Last cart item removed! ~ Total amount billed for:
4147.7
---------- Cust. ID:
1
Date and time:
09/11/2020 20:03:01
Appending product to order:
Gaming Headphones
Appending product to order:
Essentials Earphones
Appending product to order:
Essentials Earphones
---
ed
Automated Store Registry:
---------- Cust. ID:
1
Date and time:
09/11/2020 20:05:23
Appending product to order:
Gaming Headphones
Appending product to order:
Neckband-Style Wireless Earphones
Appending product to order:
30W Bluetooth Speakers
Appending product to order:
Wired Speaker - 5.1
Last cart item removed! ~ ---

```
def package_dbfadonager():
    /**
    * The program DBFAdonager implements an application that
    * houses a store management software with many integrations.
    *
    */
    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()
```

// dbfaINCEPTION1

Developed by

Python Board Project

Pranav Balaji and Sushant Gupta