APPLICATION DEVELOPMENT: SHOPPING MALL MANAGEMENT SYSTEM BY PYTHON

Table of Contents

Introduction:

Overview

A complete software program called the Shopping Mall Management System was created to improve and streamline the operations of shopping malls. By meeting the demands of both mall owners and customers, this Python-based system acts as a centralised platform to manage many areas of shopping mall administration.

Aim

The main goal of this study is to create a strong and effective data management system for shopping malls, solving the difficulties in managing a contemporary mall.

Objective

- To develop an easy to use and productive shopping mall management system utilizing Python to smooth out and automate key administrative processes.
- To improve client experience by giving highlights, like, online reservations, ongoing stock following, and customized advancements.
- To ensure robust security and data integrity to safeguard delicate data while improving functional productivity in shopping mall management.

Scope

The Shopping Mall Management System will envelop works, for example, occupant management, stock following, security observing, guest analytics, reservation systems, and revealing instruments (Valks *et al.* 2021). It will be an easy to use and adaptable arrangement, offering mall directors the adaptability to adjust it to their particular prerequisites. The system's extension additionally incorporates the joining of arising innovations like IoT devices, AI-driven analytics, and mobile applications to make a cutting edge and responsive mall management stage. Eventually, the objective is to work on the general proficiency, productivity, and consumer loyalty levels of shopping malls through creative Python-based application development.

Problem identification:

The creation of a program that can thoroughly manage all elements of a shopping mall has been cited as an issue or a necessity for the Shopping Mall Management System. In terms of successful consumer involvement, flawless sales operations, and efficient store and inventory management, modern shopping malls confront a number of obstacles. A centralised system that can handle these issues, improve mall operations, and give customers a better shopping experience is becoming increasingly necessary.

Functional Requirements:

a) Store Management: The system must enable mall managers to add, modify, and remove store data, including information on the name, address, and phone number of each business. This function makes sure that the mall directory is correct and up to date.

b) Inventory Administration: The system has to monitor and control store inventory, including goods, stock levels, and replenishment requirements. For preventing shortages and overload problems and to improve inventory control, it should give real-time data.

c) Sales Administration: The programme must enable sales transactions, provide receipts, and preserve a thorough history of sales (Okunogbe *et al.* 2023). This capability aids in the analysis of sales patterns and guarantees a simple checkout procedure.

Non-Functional Conditions:

a) User-Friendly Interface: For assurance that mall workers can quickly browse and operate the system without substantial training, the program should have an easy to use and accessible appearance.

b) Secured Storage of Data and Access: When it comes to sensitive data, such as customers and sales records, protection is of the utmost importance. To safeguard data from unauthorised access and security breaches, the system must use strong security measures.

User Stories/Use Cases:

- As a mall administrator, I want to add and update store information with the goal that the mall registry remains accurate and cutting-edge, facilitating better navigation for customers.
- As a store manager, I want to track my store's stock in real-time to avoid stockouts and overload situations, streamlining my store's performance.
- As a cashier, I want to deal with sales transactions rapidly and generate receipts for clients to guarantee a smooth and effective checkout process (Acosta *et al.* 2020).

- As a marketing manager, I want to maintain client records and analyze purchase history to carry out powerful loyalty programs and personalized marketing campaigns.

Design and Planning:

Use Case Diagram

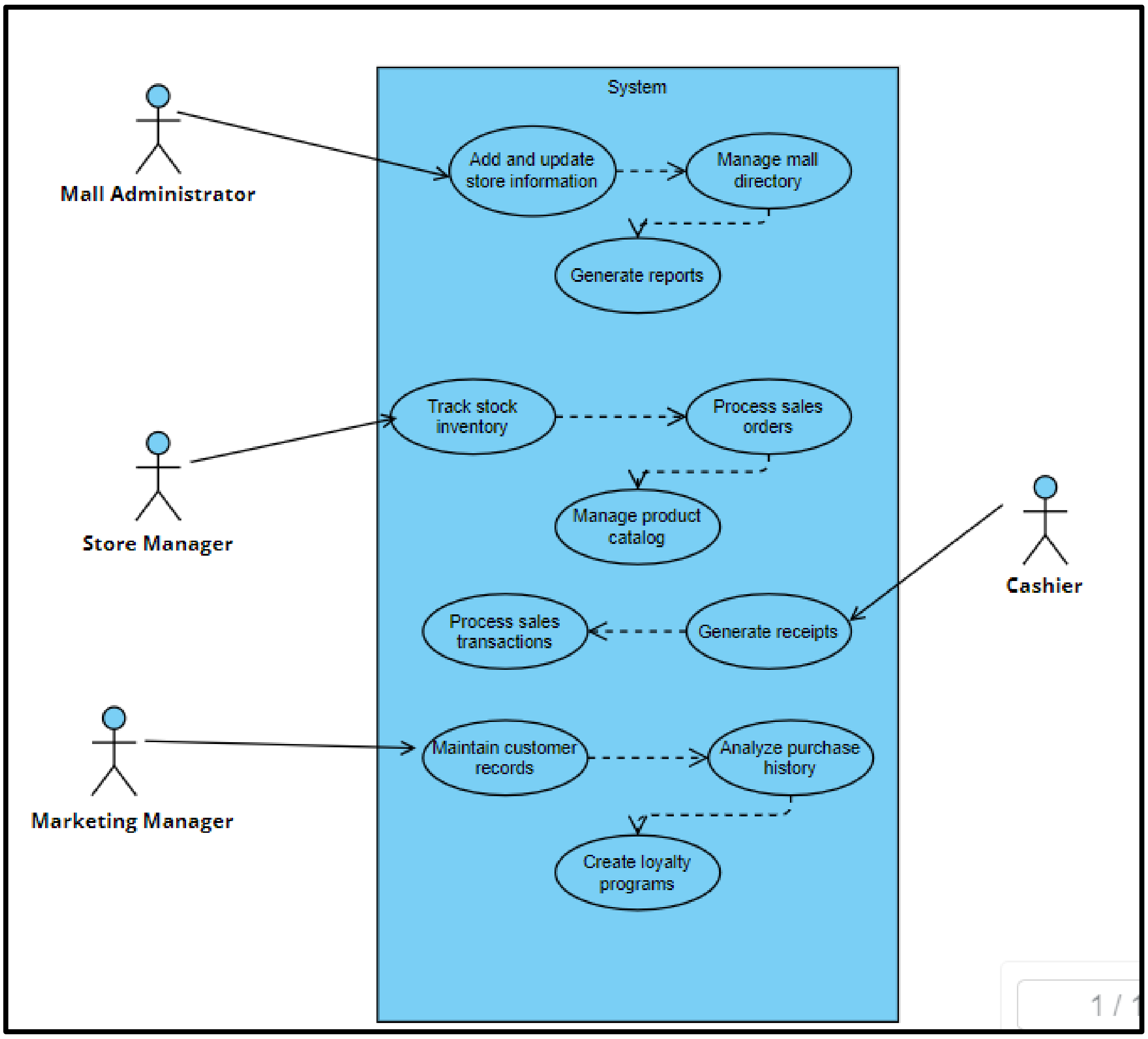


Figure 1: Use Case Diagram

(Source: Created by Visual Paradigm)

In the above figure, a use case diagram has been done. The mall administrator is a crucial role, who is in charge of managing the mall directory and updating store information. They are essential in producing the reports that support decision-making and mall management. Store managers are responsible for maintaining stock inventories, overseeing the product catalogue, and processing sales orders, all of which directly contribute to the efficient operation of specific businesses. At the point of sale, cashiers manage sales transactions and provide receipts to guarantee a smooth client experience. The marketing manager keeps track of consumer information, examines buying patterns, and uses this information to develop successful loyalty programs and marketing initiatives that improve client involvement and spur company expansion.

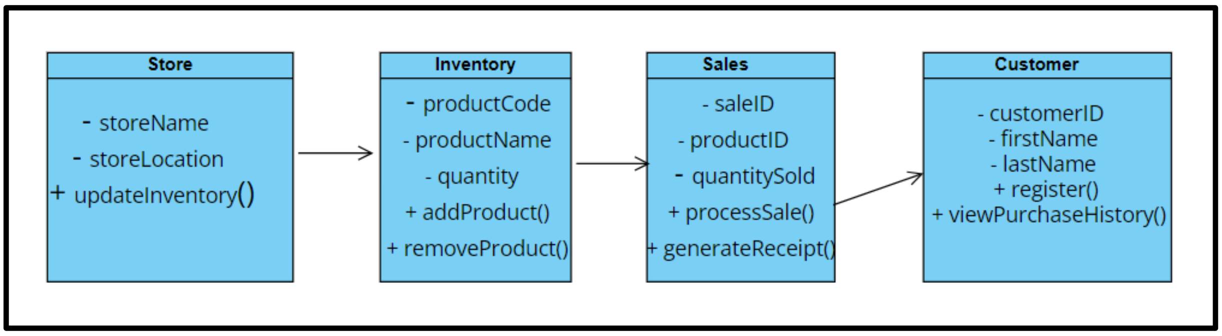


Figure 2: Class Diagram

(Source: Created by Visual Paradigm)

The class diagram introduced portrays the underlying parts of a worked on retail the executives framework, including four center classes: “Store”, “Inventory”, “Sales”, and “Customer”, each addressing vital elements inside the framework.

The “Store” class holds credits like “storeName” and “storeLocation” and can communicate with the “Inventory” class through a one-to-numerous affiliation, meaning that a store can have different inventory things. The “Store” class additionally includes a “updateInventory()” strategy, working with inventory on the board.

The “Inventory” class embodies item related data, for example, “productCode”, “productName”, and “amount” and gives strategies like “addProduct()” and “removeProduct()” for inventory control.

The “Sales” class tracks sales exchanges, with credits like “saleID” and “quantitySold”, and incorporates a “processSale()” strategy for taking care of sales tasks, as well as a “generateReceipt()” technique for receipt age.

The “Customer” class addresses customers and has characteristics, like, “customerID”, “firstName”, and “lastName”, alongside techniques like “register()” for customer enlistment and “viewPurchaseHistory()” for getting to buy history.

Implementation:

For the purpose to design and provide a complete solution for effective mall administration, the Shopping Mall Management System was implemented using Python. This system was created to optimise customer, sales, inventory, and store administration while using a number of well-known data structures, algorithms, and libraries to guarantee functionality and dependability.

Key capabilities created:

Store Administration:

- proprietors of shops can register their companies, providing information such as the name, kind, and location of the store.
- The structure of the system controls the distribution of retail spaces and gives distinctive identifying numbers.
- Owners of businesses can change their contact details and hours of operation.

Inventory Control:

- Owners of retail establishments have access to and control over their product inventory, including stock levels, costs, and descriptions.
- Overstocking or understocking problems are avoided with the aid of real-time inventory tracking (Mohammed, 2021).
- When supply levels are low, business owners receive notifications.

Management of sales:

- The system keeps track of sales transactions and issues clients with receipts.
- For making wise business decisions, store owners may analyse analytics and sales information.
- Secure and convenient transactions are facilitated through integration with payment gateways.

Management of clients:

- Customers get access to store directories, may register accounts, and can receive tailored promotions based on their choices.
- Customer profiles keep track of past purchases, which helps the system make product recommendations.
- It is possible to administer loyalty programmes and incentives to encourage repeat business.

Important Data Structures, Libraries, and Algorithms

Database management: For interacting with a relational database, Python's SQLAlchemy package was utilised, which made it possible to save and retrieve data quickly.

Inventory tracking: A powerful algorithm was put in place to keep track of current stock levels and deliver warnings when they cross a certain threshold.

Recommendation Engine: Collaborative filtering algorithms were employed as the recommendation engine to make product recommendations to clients based on their previous purchasing habits (Pratama *et al.* 2020).

Security: For the security of sensitive data, authentication and permission methods were put into place utilising libraries like Flask-Security.

Processing of payment: Integration with payment processors like Stripe or PayPal to manage financial transactions securely.

Testing of Python code:

A number of unit tests were developed and put into place to evaluate key functionality in order to guarantee the stability and dependability of this Shopping Mall Management System. These examinations addressed tasks like managing customers, maintaining inventory, processing sales, as well as adding or updating store information. Unit tests, for instance, confirmed that processing sales transactions, updating stock levels, and adding products to inventory led to the desired results (Forghani *et al.* 2022). The seamless connection between these separate components was subsequently tested through integration. This comprised scenarios in which retailers added items to their inventories, employees processed sales, and precise customer data was collected. The test report included a summary of the test cases, the findings, and any problems that came up.

Documentation:

User Manual

The Shopping Mall Management System's user manual informs mall administrators, store managers, cashiers, as well as marketing managers on how to operate it effectively. The mall directory is accurate thanks to the ability of mall managers to modify and update store information. Store managers keep an observation on the inventory in real time to avoid stockouts and overloads. The smooth way that cashiers handle sales transactions and produce receipts improves the checkout experience for customers. For customized loyalty programs and individualized marketing efforts, marketing managers keep track of customer information and examine purchase history. In order to optimize mall operations and consumer pleasure, the system maintains data security and combines payment platforms for secure transactions (Smith and Lorenz, 2021).

Installation Procedures

Users ought to carry out following actions in order to set up the Shopping Mall Management System. It is needed to make sure a Python environment is configured first, ideally using virtual environments. The repository for the application should then be copied from the specified source. Then, use pip to install the required dependencies, which are normally mentioned in a "requirements.txt" file, after navigating to the project directory. Next, define the database type as well as credentials within the application settings when configuring the database connection. Apply modifications to the database schema to begin with. Finally, launch the program utilizing any Python web server or as a command appropriate to the framework (Muhtarulloh *et al.* 2023).

Code Documentation

The shopping mall management system's Python source code is offered. It has a number of classes for handling retail locations, stock, sales, and clients. Users can execute a variety of operations, including adding or modifying stores, maintaining inventory, processing sales, including handling customer data, using the main_menu() method, which acts as the main menu for dealing with the system. The code includes functions for managing stores, such as add_store(), update_store(), and delete_store(), and for managing inventories, such as implementing products to inventory, maintaining stock levels, and removing products, respectively, use the manage_inventory() function.

Deployment:

Deploying the implementation of the system in the Python environment

Step	**Task Description**	**Actions**
1	Preparing the Python Environment	Assure Python is installed.
2	Cloning the Application Repository	Cloning the project repository from the source.
3	Installing Dependencies	Using pip to install needed dependencies.
4	Configuring Database	Setting up the database connection and also schema.
5	Running the Application	Executing the main Python script for launching the application.

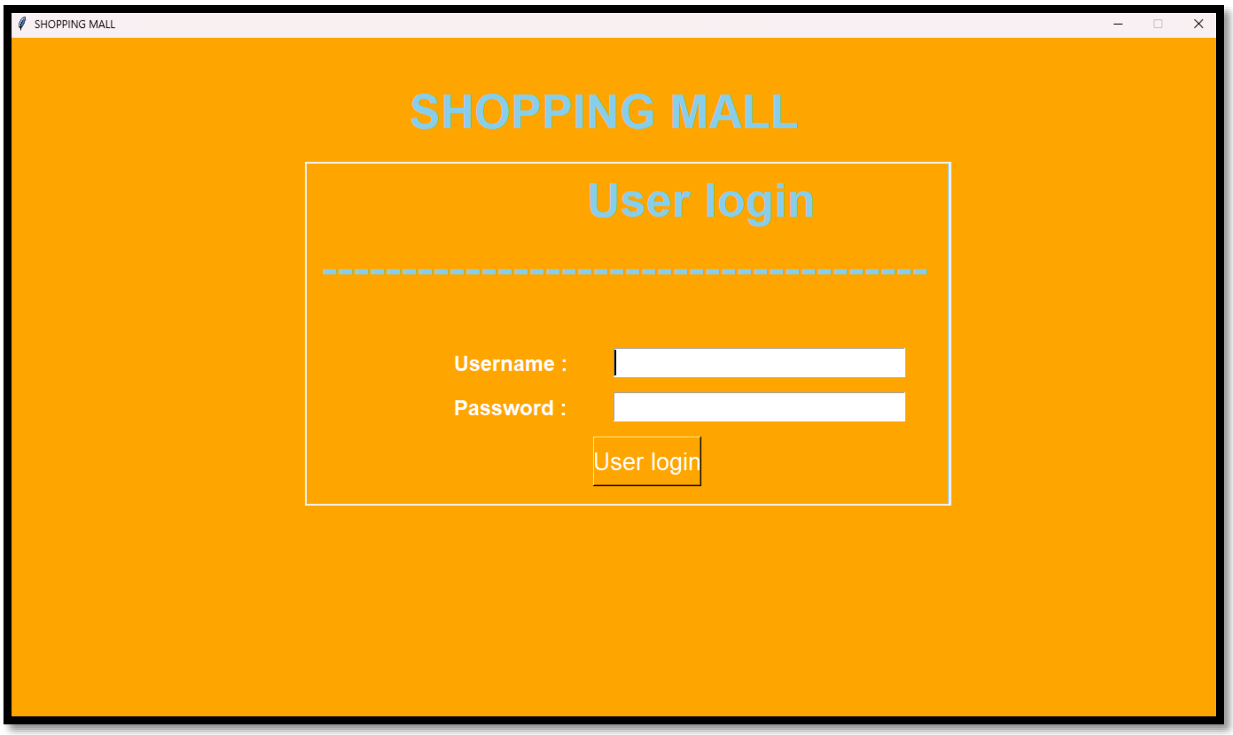


Figure 3: Home page of the Shopping Mall Management System

(Source: Obtained from Python environment)

The figure mentioned above displaying the Home page of the Shopping Mall Management System.

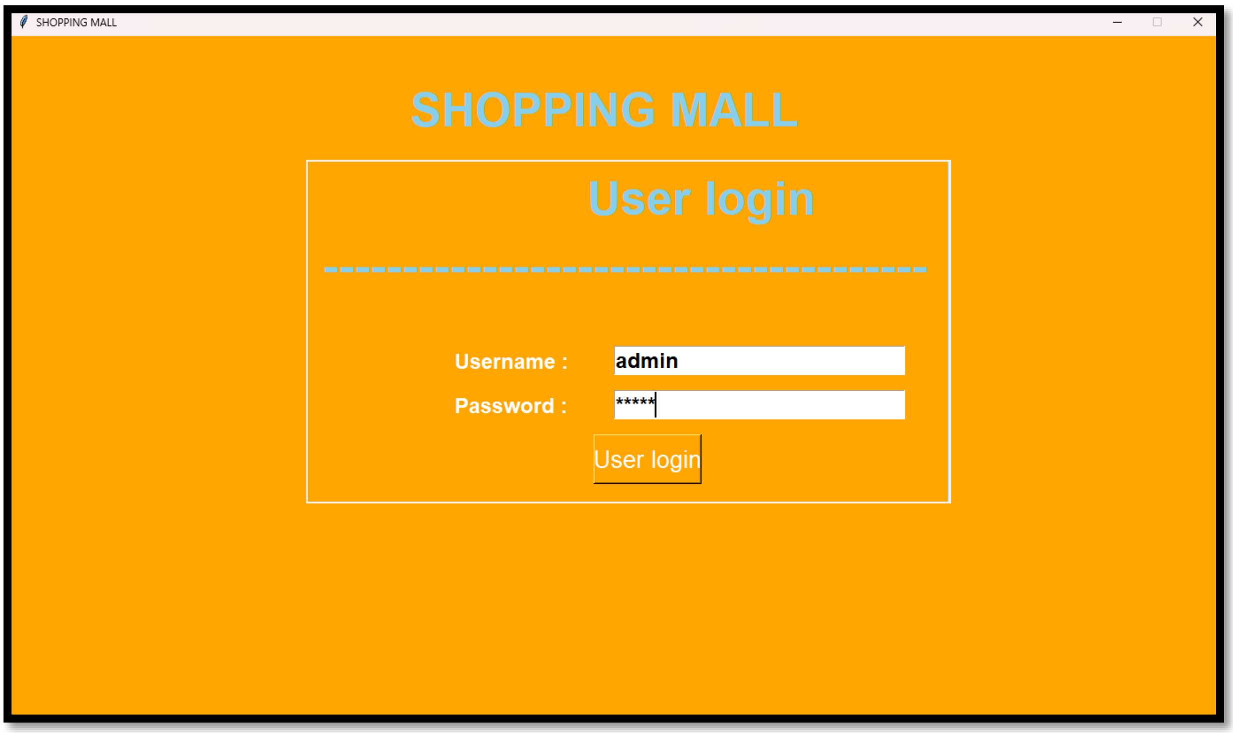


Figure 4: Administration login successful

(Source: Obtained from Python environment)

The picture depicts that admin login is successful.

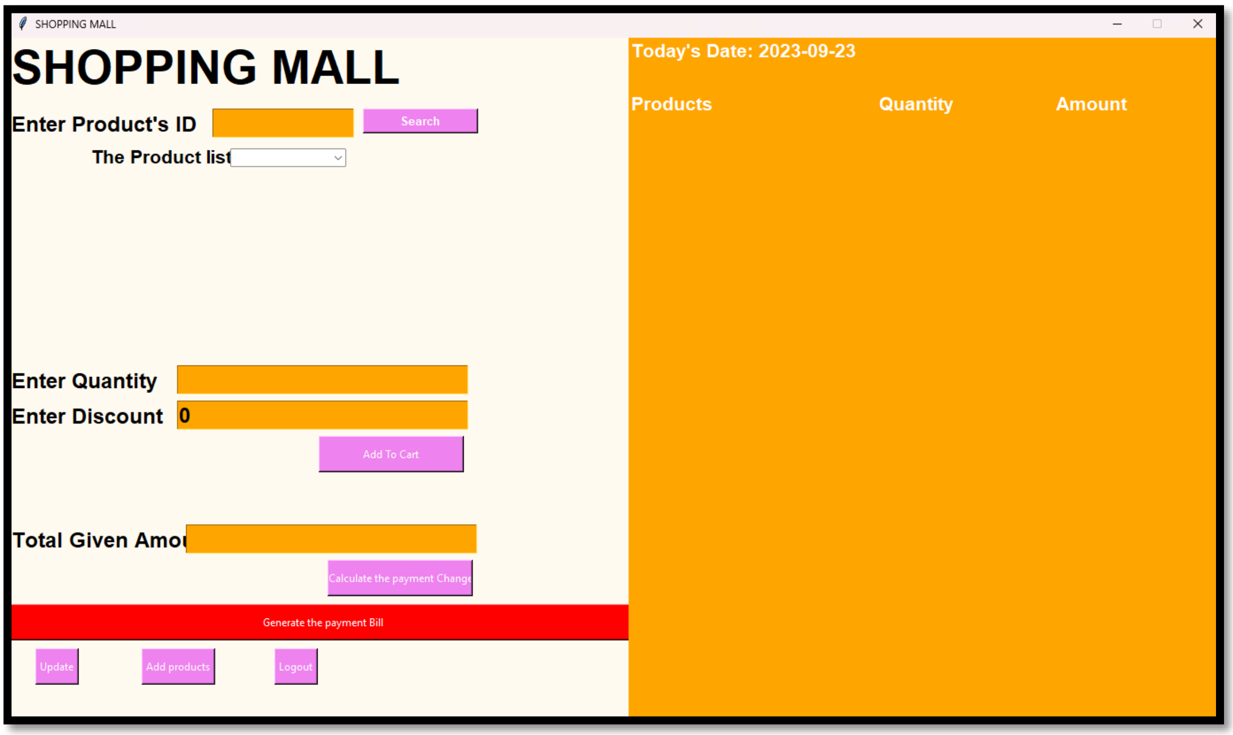


Figure 5: Displaying the home page

(Source: Obtained from Python environment)

The above image displays the home page of the shopping mall management system.

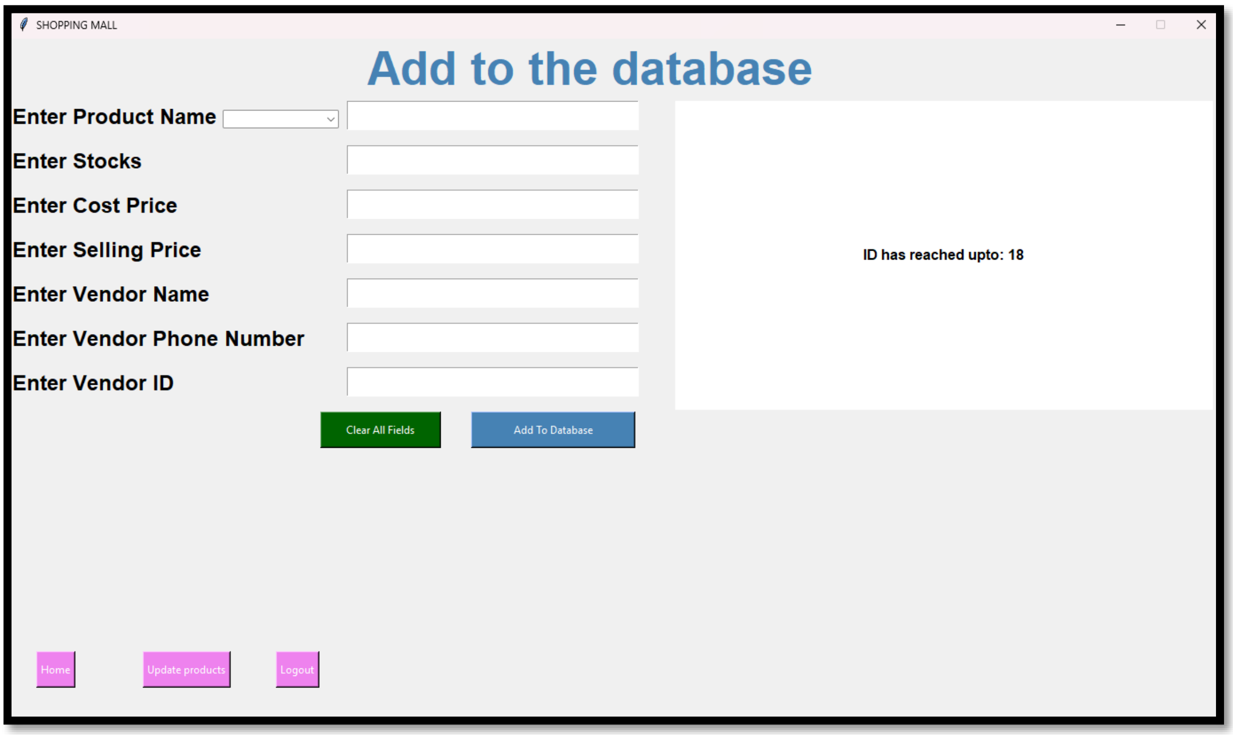
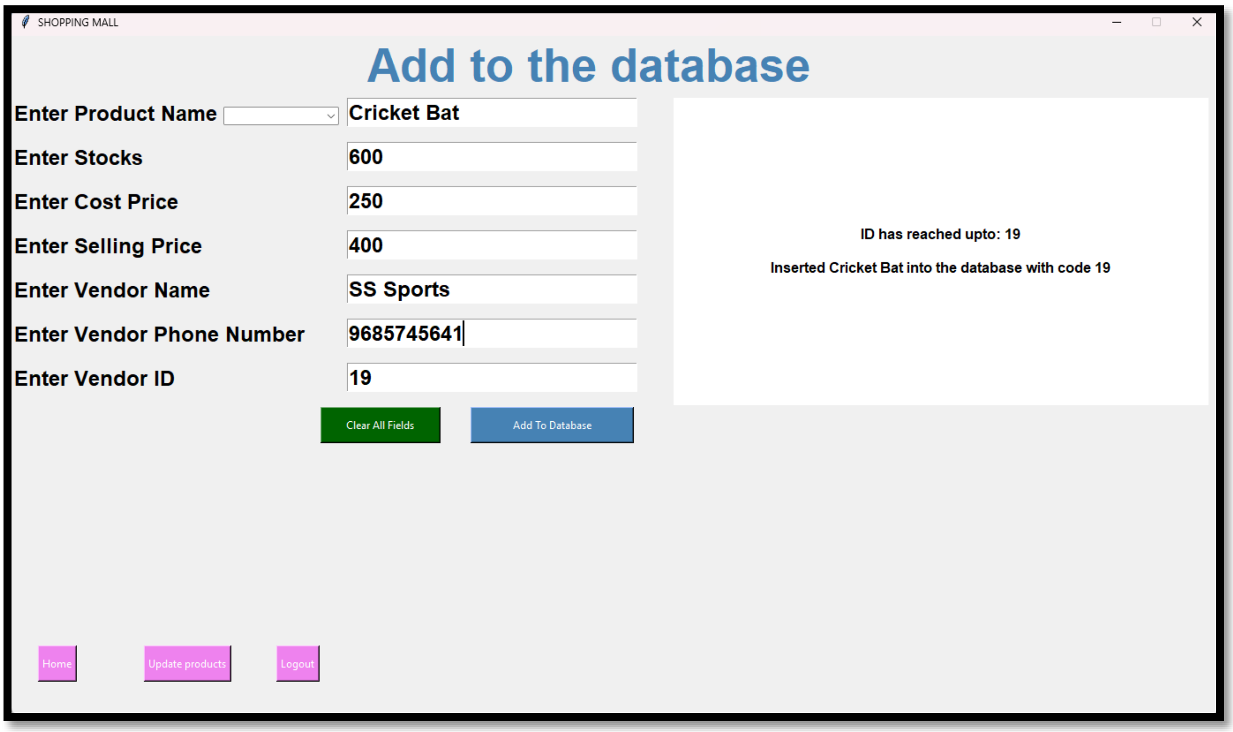


Figure 6: Adding item to database page

(Source: Obtained from Python environment)

The image elaborates that in this page, the user can add the items for the shopping mall.



Add to the database

Enter Product Name	Cricket Bat
Enter Stocks	600
Enter Cost Price	250
Enter Selling Price	400
Enter Vendor Name	SS Sports
Enter Vendor Phone Number	9685745641
Enter Vendor ID	19

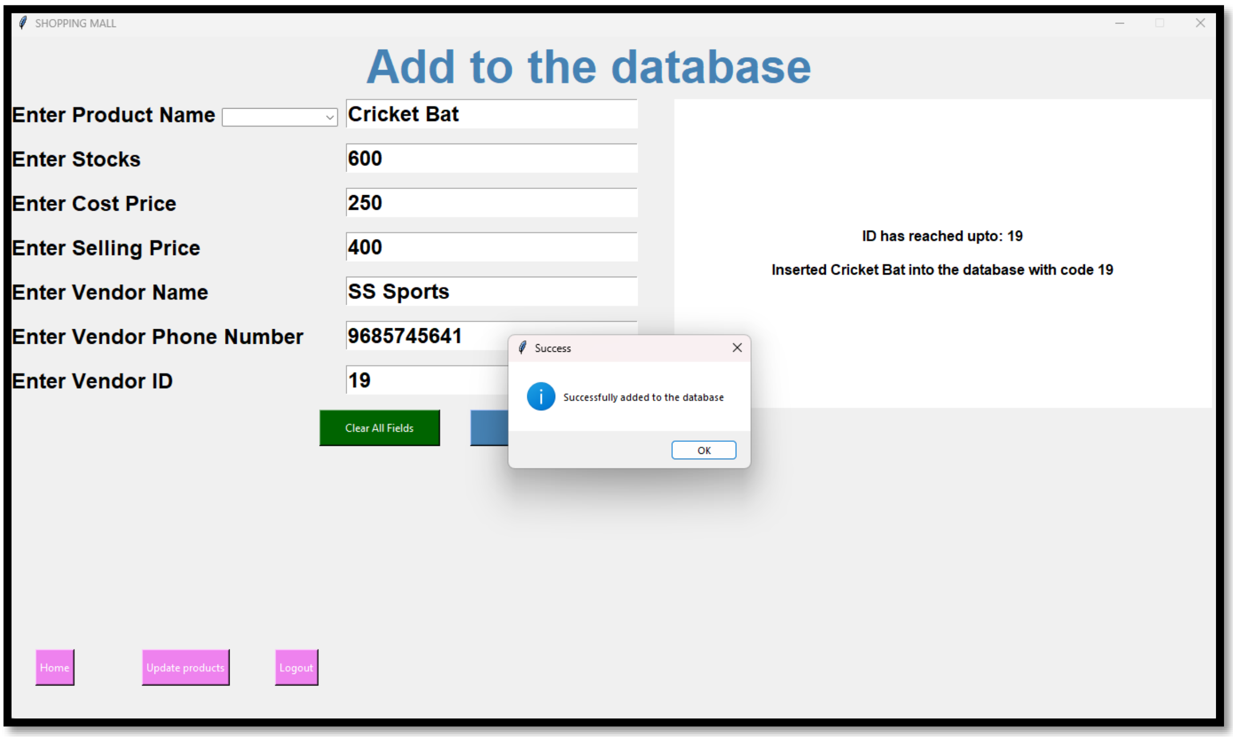
ID has reached upto: 19

Inserted Cricket Bat into the database with code 19

Figure 7: Adding item to database

(Source: Obtained from Python environment)

The picture discloses that the addition of item to the database by entering manually.



Add to the database

Enter Product Name	Cricket Bat
Enter Stocks	600
Enter Cost Price	250
Enter Selling Price	400
Enter Vendor Name	SS Sports
Enter Vendor Phone Number	9685745641
Enter Vendor ID	19

ID has reached upto: 19

Inserted Cricket Bat into the database with code 19

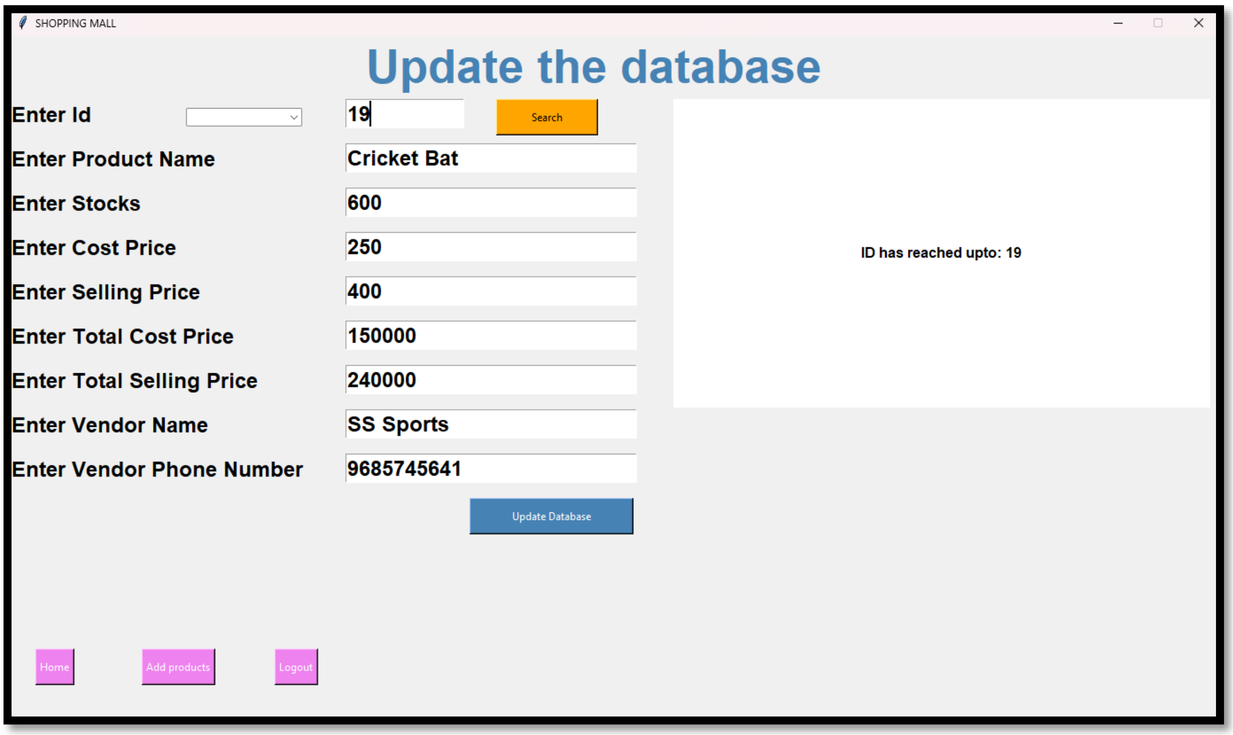
Success

Successfully added to the database

Figure 8: Item added successfully to the database

(Source: Obtained from Python environment)

The above figure describes that the item is added to the database successfully.



Update the database

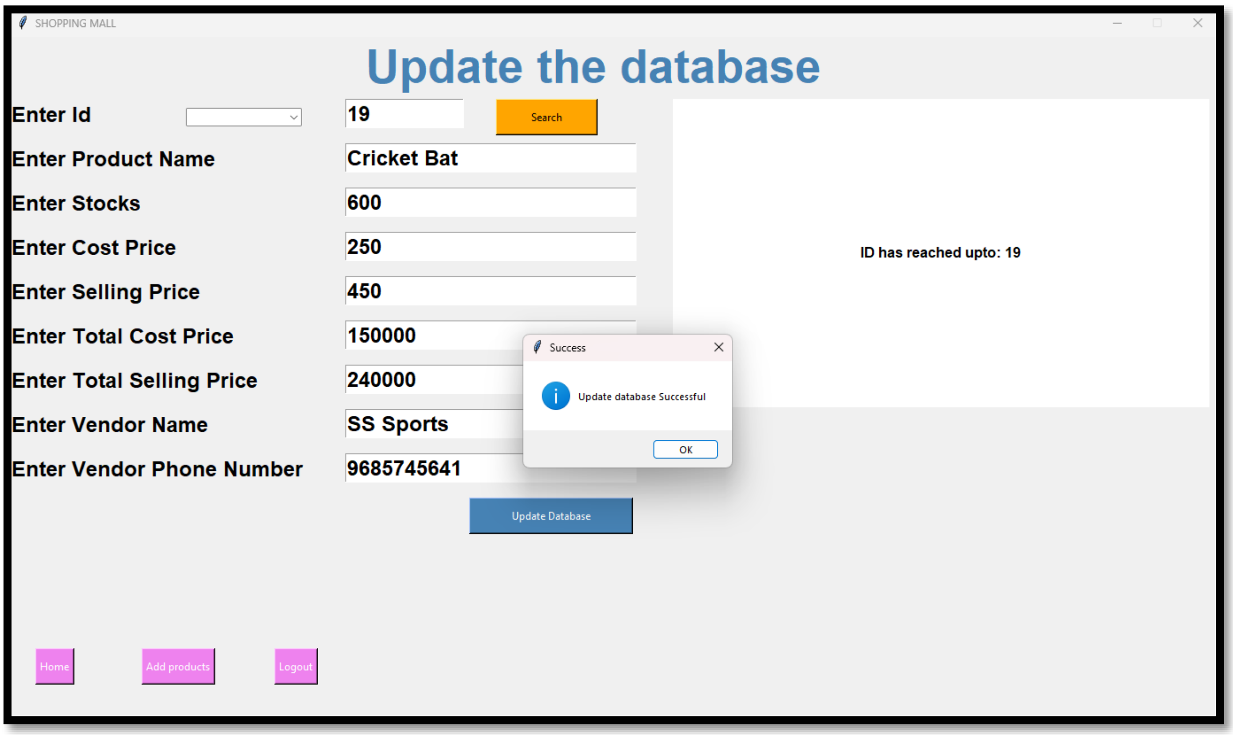
Enter Id	19
Enter Product Name	Cricket Bat
Enter Stocks	600
Enter Cost Price	250
Enter Selling Price	400
Enter Total Cost Price	150000
Enter Total Selling Price	240000
Enter Vendor Name	SS Sports
Enter Vendor Phone Number	9685745641

ID has reached upto: 19

Figure 9: Product update in the database

(Source: Obtained from Python environment)

The above figure derives that the admin wants to update some product details from their database.



Update the database

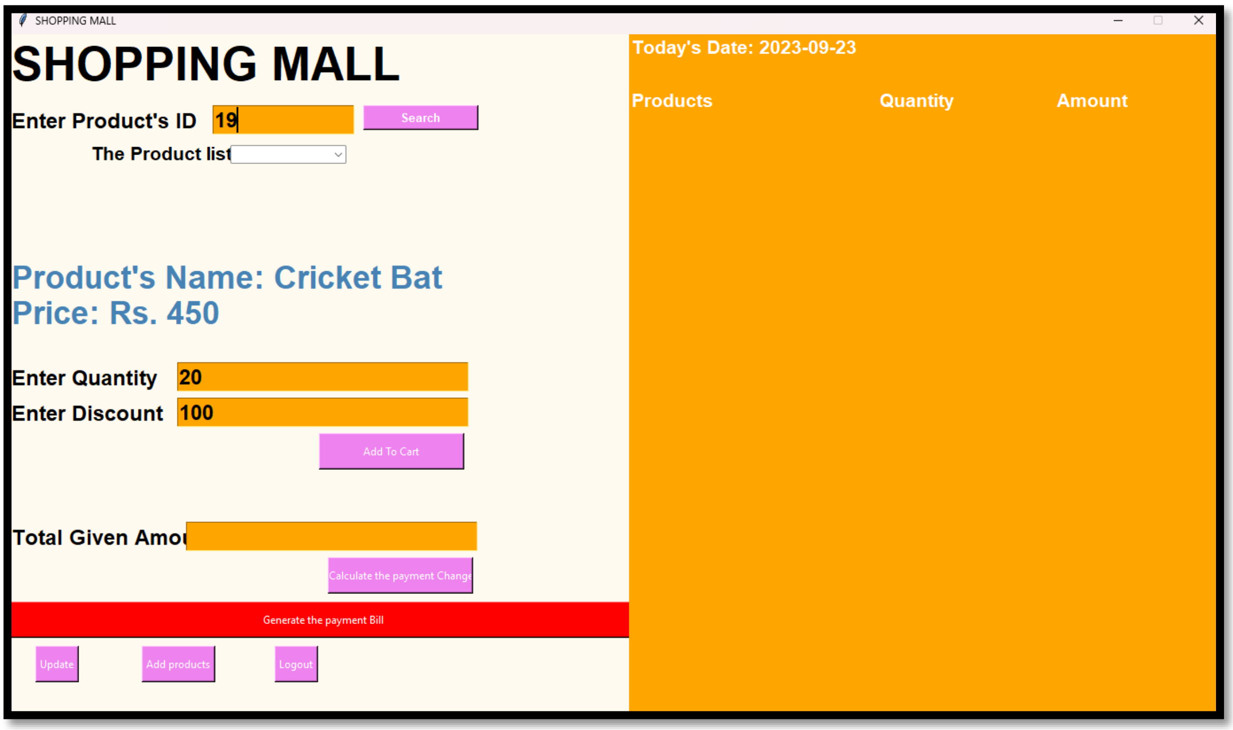
Enter Id	19
Enter Product Name	Cricket Bat
Enter Stocks	600
Enter Cost Price	250
Enter Selling Price	450
Enter Total Cost Price	150000
Enter Total Selling Price	240000
Enter Vendor Name	SS Sports
Enter Vendor Phone Number	9685745641

ID has reached upto: 19

Update database Successful

Figure 10: Updating of the item present in the database

In the above image, the selling price of the cricket bat has been updated successfully.


SHOPPING MALL

Today's Date: 2023-09-23

Enter Product's ID: 19

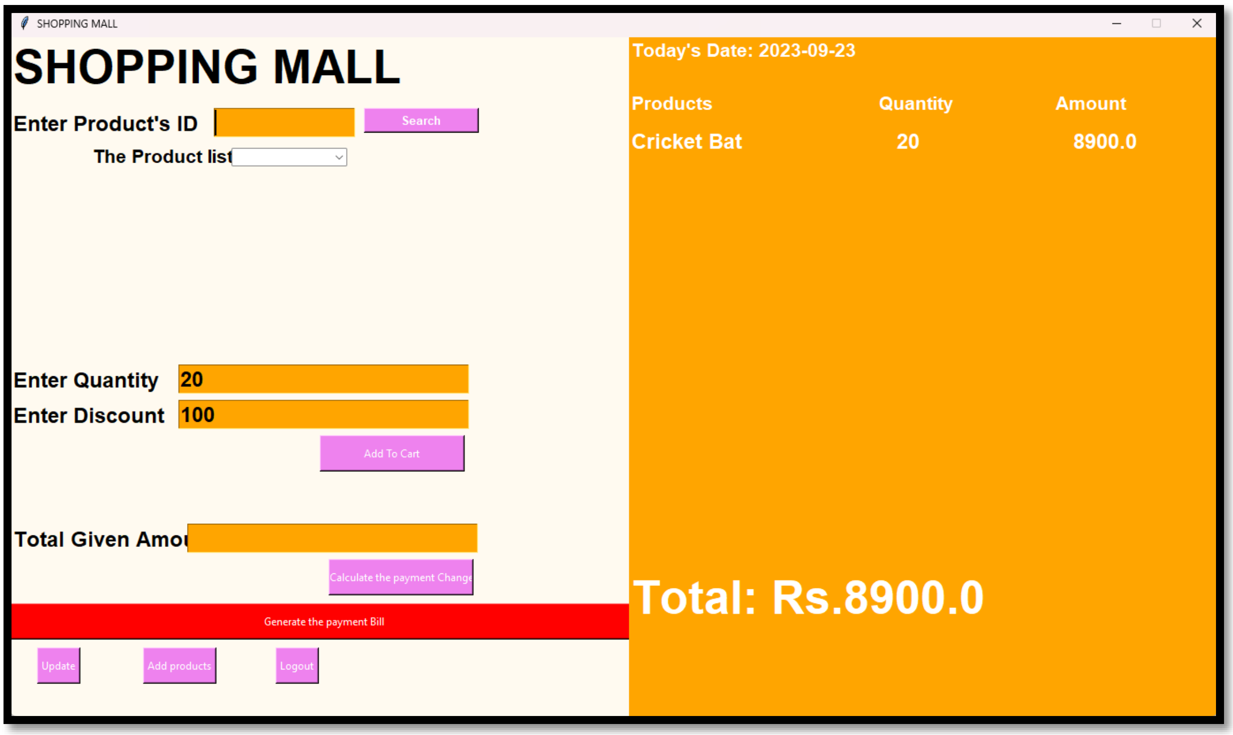
Products | Quantity | Amount

Product's Name: Cricket Bat
Price: Rs. 450

Enter Quantity: 20
Enter Discount: 100

Figure 11: Purchasing cricket bat from the shopping mall

In the above image, it is showing that 20 cricket bats have been issued for the purpose of selling


SHOPPING MALL

Today's Date: 2023-09-23

Products	Quantity	Amount
Cricket Bat	20	8900.0

Enter Quantity: 20
Enter Discount: 100

Total: Rs.8900.0

Figure 12: Billing page

In the above image, the total amount of bill has been generated for 20 cricket bats with a discount of 100.

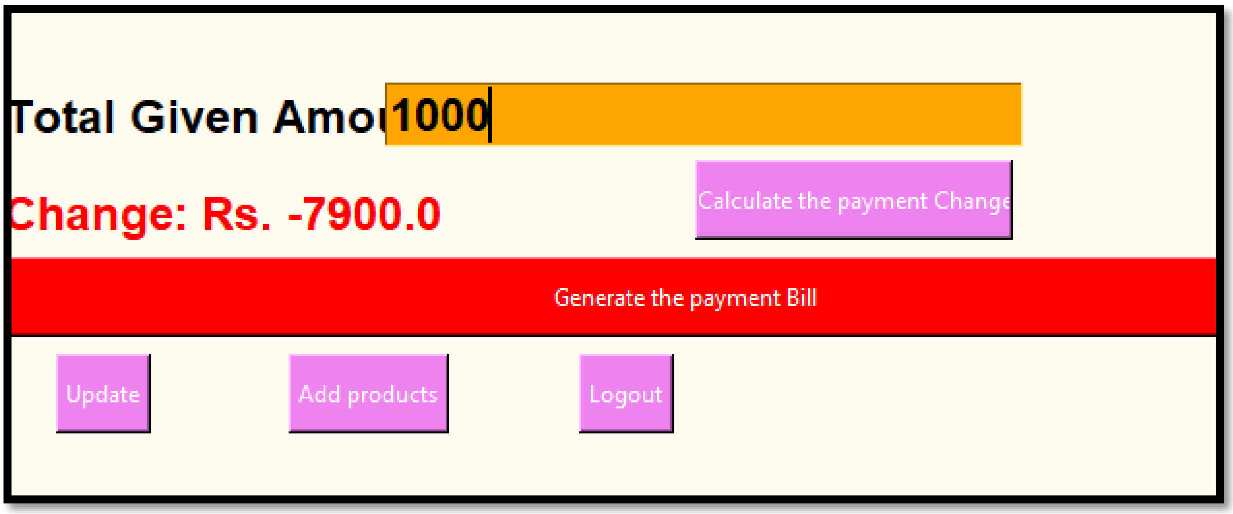


Figure 13: Calculating the payment change

In the above image, the total payment have been calculated based on the changes.

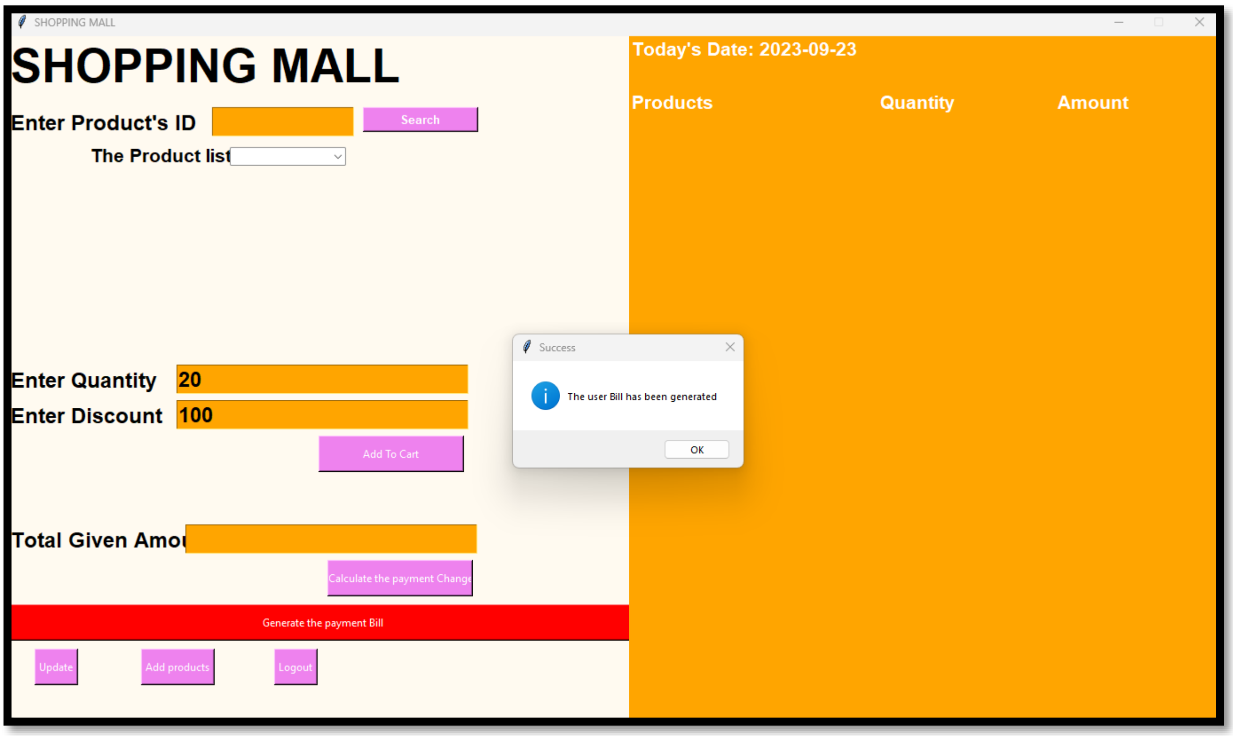


Figure 14: Generating the final bill

In the above image, the final bill have been generated successfully and the pop up has been shown.

Conclusion:

The critical accomplishments and results of the evaluation of the Shopping Mall Management System in the Python climate include:

Successful Implementation: The system was successfully executed, including a utilitarian landing page, administrator login, staff login, inventory management, employee management, receipt creation, and an "About Us" segment.

User Authentication: The system accomplished secure user authentication, with both administrator and staff logins being successful, guaranteeing information protection.

Inventory and Employee Management: The application really handles inventory and employee information, empowering the expansion of things and staff to the data set.

The necessity of a user-friendly interface and the significance of strong security measures to secure confidential information are only two lessons discovered throughout the development process. Debugging and assuring the seamless integration of multiple components were difficulties encountered.

Future improvements can include linking with other systems for more thorough administration of the retail mall's activities, introducing real-time updates, and expanding analytical and reporting capabilities. Additionally, for an improved seamless user experience, customer input should direct advancements in functionality and usability.

References:

Valks, B., Arkesteijn, M.H., Koutamanis, A. and den Heijer, A.C., 2021. Towards a smart campus: supporting campus decisions with Internet of Things applications. Building Research & Information, 49(1), pp.1-20. Available At: https://www.tandfonline.com/doi/pdf/10.1080/09613218.2020.1784702

Okunogbe, O. and Santoro, F., 2023. The Promise and Limitations of Information Technology for Tax Mobilization. The World Bank Research Observer, 38(2), pp.295-324. Available At: https://openknowledge.worldbank.org/bitstreams/669f218c-14f3-49b8-aa60-cc624321f9b0/download

Acosta, D., Alquizar, M.L., Junio, C.J., Talara, D.C. and Buladaco, M.V., 2020. Analysis and design of sales and inventory management system for yochang general merchandise. Dyrien Cris and Buladaco, Mark Van, Analysis and Design of Sales and Inventory Management System for Yochang General Merchandise (June 29, 2020). Available At: https://www.zbw.eu/econis-archiv/bitstream/11159/441663/1/EBP07647769X_0.pdf

Mohammed, F., 2021. AN ASSESSMENT OF INVENTORY MANAGEMENT SYSTEM THE CASE OF HABESHA CEMENT SHARE COMPANY (Doctoral dissertation, ST. MARY'S UNIVERSITY). Available At: http://repository.smuc.edu.et/bitstream/123456789/6657/1/feti%20last%20thesis-converted.pdf

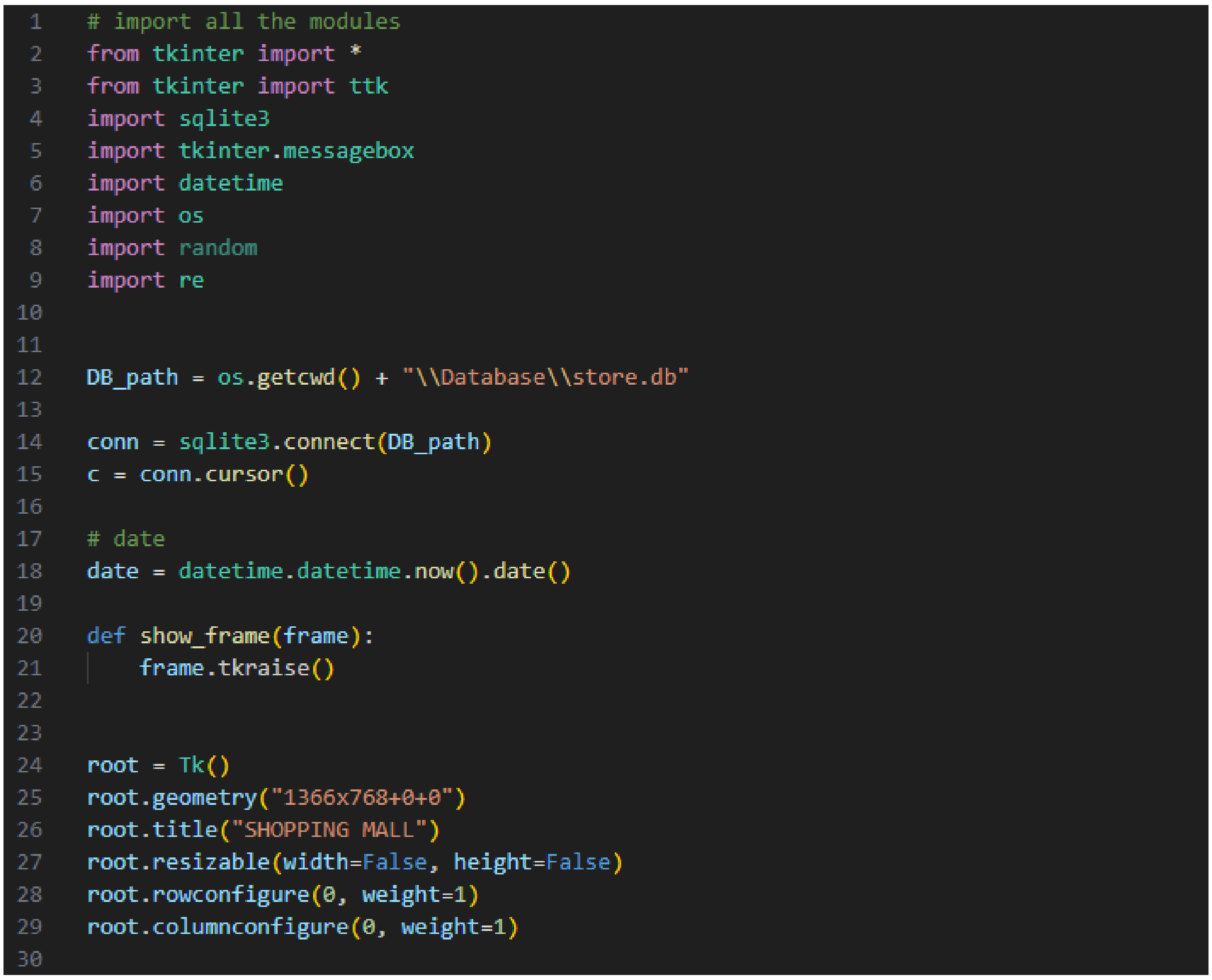
Pratama, B.Y., Budi, I. and Yuliawati, A., 2020. Product recommendation in offline retail industry by using collaborative filtering. International Journal of Advanced Computer Science and Applications, 11(9), pp.635-643. Available At: https://www.academia.edu/download/80781428/Paper_75-Product_Recommendation_in_Offline_Retail_Industry.pdf

Smith, P. and Lorenz, C.D., 2021. LiPyphilic: A Python toolkit for the analysis of lipid membrane simulations. Journal of chemical theory and computation, 17(9), pp.5907-5919. Available at: https://www.biorxiv.org/content/10.1101/2021.05.04.442445.full.pdf

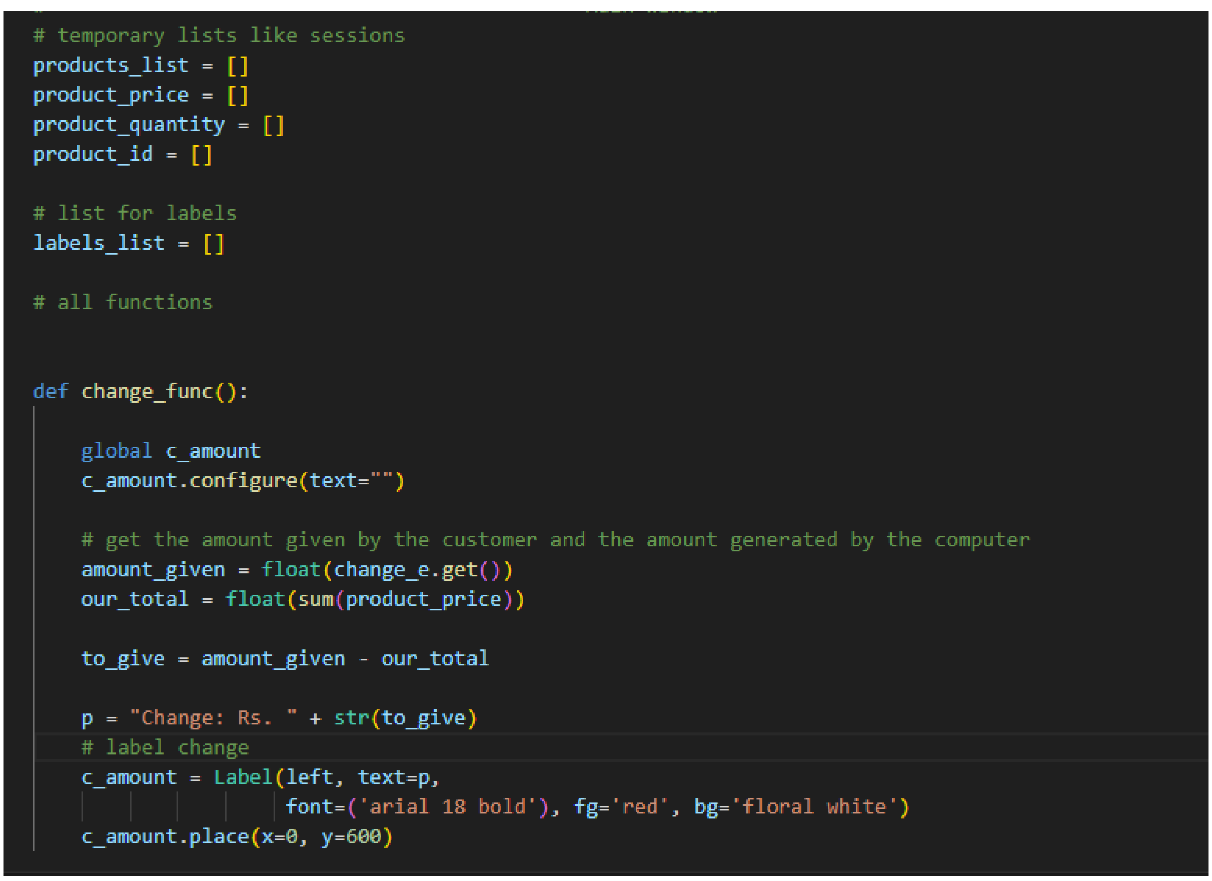
Muhtarulloh, F., Maulidina, A. and Huda, A.F., 2023. Solution Optimal Transportation Problems Using the Sirisha Viola Modification Method Python assisted programming. JINAV: Journal of Information and Visualization, 4(1), pp.117-125. Available at: https://jinav.my.id/index.php/jinav/article/download/1660/1204

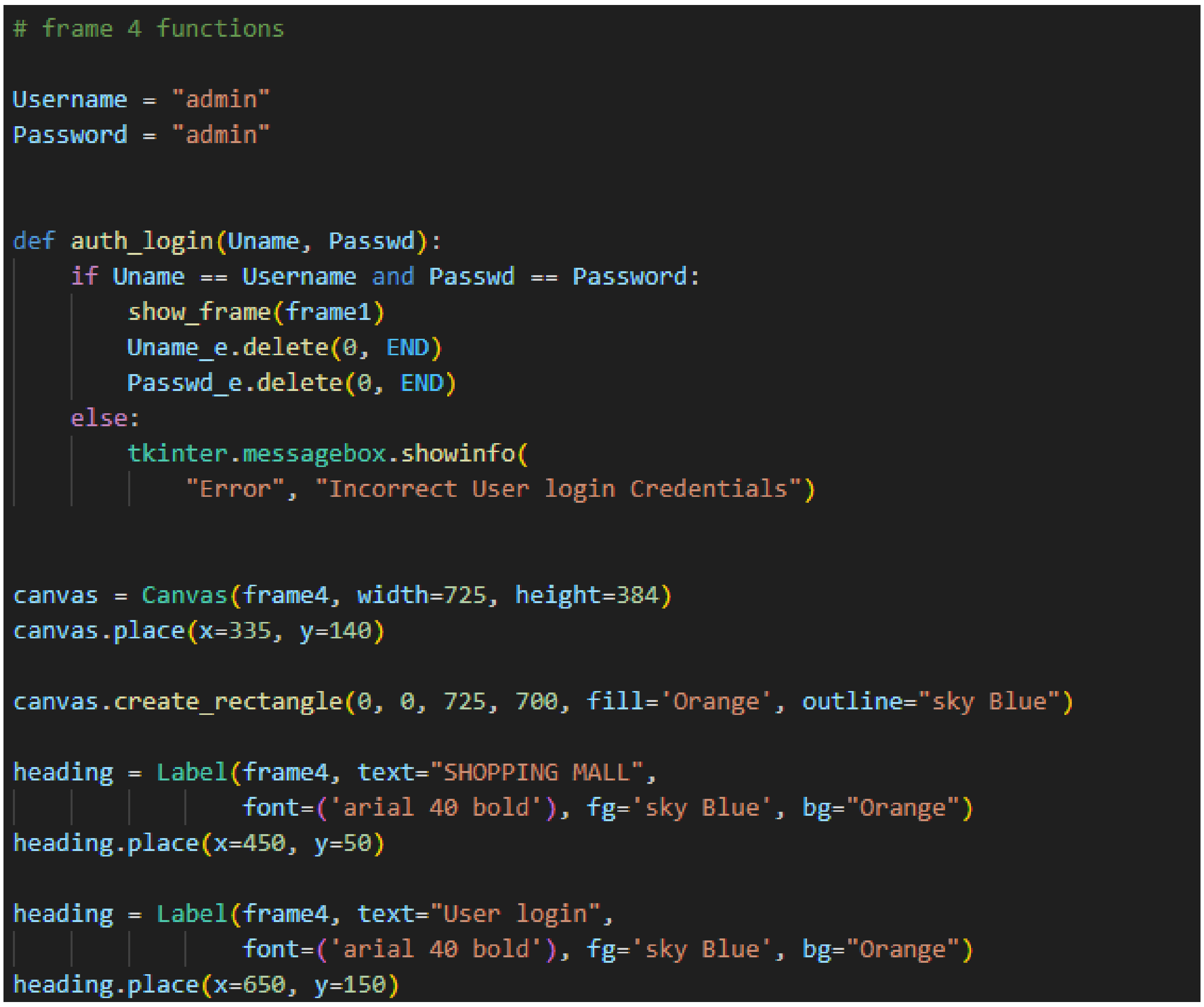
Forghani, E., Sheikh, R., Hosseini, S.M.H. and Sana, S.S., 2022. The impact of digital marketing strategies on customer's buying behavior in online shopping using the rough set theory. International journal of system assurance engineering and management, pp.1-16. Available at: https://www.researchgate.net/profile/Reza-Sheikh/publication/355172422_The_impact_of_digital_marketing_strategies_on_customer's_buying_behavior_in_online_shopping_using_the_rough_set_theory/links/648d71788de7ed28ba30d773/The-impact-of-digital-marketing-strategies-on-customers-buying-behavior-in-online-shopping-using-the-rough-set-theory.pdf

Appendices:



```python
# import all the modules
from tkinter import *
from tkinter import ttk
import sqlite3
import tkinter.messagebox
import datetime
import os
import random
import re


DB_path = os.getcwd() + "\\Database\\store.db"

conn = sqlite3.connect(DB_path)
c = conn.cursor()

# date
date = datetime.datetime.now().date()

def show_frame(frame):
    frame.tkraise()


root = Tk()
root.geometry("1366x768+0+0")
root.title("SHOPPING MALL")
root.resizable(width=False, height=False)
root.rowconfigure(0, weight=1)
root.columnconfigure(0, weight=1)

```


```
# temporary lists like sessions
products_list = []
product_price = []
product_quantity = []
product_id = []

# list for labels
labels_list = []

# all functions


def change_func():

    global c_amount
    c_amount.configure(text="")

    # get the amount given by the customer and the amount generated by the computer
    amount_given = float(change_e.get())
    our_total = float(sum(product_price))

    to_give = amount_given - our_total

    p = "Change: Rs. " + str(to_give)
    # label change
    c_amount = Label(left, text=p,
                     font=('arial 18 bold'), fg='red', bg='floral white')
    c_amount.place(x=0, y=600)
```



```python
# frame 4 functions

Username = "admin"
Password = "admin"


def auth_login(Uname, Passwd):
    if Uname == Username and Passwd == Password:
        show_frame(frame1)
        Uname_e.delete(0, END)
        Passwd_e.delete(0, END)
    else:
        tkinter.messagebox.showinfo(
            "Error", "Incorrect User login Credentials")


canvas = Canvas(frame4, width=725, height=384)
canvas.place(x=335, y=140)

canvas.create_rectangle(0, 0, 725, 700, fill='Orange', outline="sky Blue")

heading = Label(frame4, text="SHOPPING MALL",
                font=('arial 40 bold'), fg='sky Blue', bg="Orange")
heading.place(x=450, y=50)

heading = Label(frame4, text="User login",
                font=('arial 40 bold'), fg='sky Blue', bg="Orange")
heading.place(x=650, y=150)
```