# INVESTIGATORY PROJECT

## Super Market Management System

# TABLE OF CONTENTS:


| S NO. | TOPIC | PAGE NO. |
|---|---|---|
| 1 | INTRODUCTION | 5 |
| 2 | PROBLEM STATEMENT | 6 |
| 3 | FEASIBILITY | 7 |
| 4 | SCOPE | 9 |
| 5 | DESIGN | 10 |
| 6 | PAGE STRUCTURE | 11 |
| 7 | IMPLEMENTATION | 12 |
| 8 | SOFTWARE AND HARDWARE REQUIREMENTS | 13 |
| 9 | PROGRAM AND OUTPUT | **14** |
| 10 | TESTING | 18 |
| 11 | CONCLUSION AND BIBILIOGRAPHY | 19 |

# INTRODUCTION TO THE PROJECT:

The objective of this python project is to design a GUI for the Grocery store Management System which incorporates details of the Employees, the Manager, the Designation of the employees, the categories of the products, the details of the Customer, and a list of available commodities, and location information of the grocery stores.

Suppliers and details of commodities which shows which items are going to be out of stock for the store which has various branches situated in various areas with different Managers taking care of that data set.

This database is efficacious in running the grocery stores. The users of the database will be the store managers.

- Grocery Store Management System is designed to provide the grocery stores with the benefit of having everything online, from products data to customers data.
- It helps the store managers to perform various functions like checking the products stock, suppliers information, customers information and also allows them to check if a particular product is available in any other branch.
- It also helps to keep track of the store employees.
- Provides a user-friendly interface where everything can be accessed with just a button click

# PROBLEM STATEMENT:

Make a Grocery List for super market shopping with name, price and quantity; if the list already contains an item then only update the price and quantity it should not append the item name again. Ask user his/her budget initially and minus the budget after adding a new item in the list. If budgets go zero/0 then no more item could be bought and if some money left and user add item greater than budget left then inform “over price” or any other message. After the list is made any money left in the budget it should show an item within the budget from the list made. VALIDATION is a must.

# FEASABILITY OF THE PROJECT:

After doing the project Grocery Shop Management System, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

A. Economical Feasibility

B. Technical Feasibility

C.Operational Feasibility

# SCOPE OF THE PROJECT:

It may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to Grocery Shop Management System. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

Our project aims at Business process automation, i.e. we have tried to computerize various processes of Grocery Shop Management System.

- In computer system the person has to fill the various forms & number of copies of the forms can be easily generated at a time.
- In computer system, it is not necessary to create the manifest but we can directly print it, which saves our time.
- To assist the staff in capturing the effort spent on their respective working areas.
- To utilize resources in an efficient manner by increasing their productivity through automation.
- The system generates types of information that can be used for various purposes.
- It satisfy the user requirement
- Be easy to understand by the user and operator
- Be easy to operate

- Have a good user interface
- Be expandable
- Delivered on schedule within the budget.

# DESIGN OF THE PROJECT:

## Interface:

1. **Billing System:**
   - Allows the biller to input product details, prices, and quantities.
   - Calculates the total amount and applies discounts if applicable.
   - Provides different payment modes such as cash, card, and online payments.
   - Generates a detailed invoice for the customer.
2. **Product Management:**
   - Displays product details, including name, price, and available stock.
   - Allows for adding new products and updating existing stock information.
   - Provides a search function to quickly locate products.
3. **Customer Details Management:**
   - Stores customer contact information for billing and future reference.
   - Allows tracking of past purchases to offer personalized services.
4. **User-Friendly Navigation:**
   - Menu-driven structure with easy-to-follow prompts.
   - Clear error messages and validation checks to ensure accurate input.
   - Real-time updates on stock levels and transaction summaries.

The interface is structured to be accessible for both experienced and new users, ensuring smooth operation of the supermarket management process.

# PAGE STRUCTURE:

**The Super Market Management System follows a structured approach to efficiently manage billing, inventory, and customer transactions. The interface consists of various sections that enable smooth navigation and operation.**

**1. Home Page**

- **Displays a welcome message and main menu options.**
- **Provides the following choices:**
  - **View Items – Check the available products in stock.**
  - **Add Items – Input new products into the system.**
  - **Purchase Items – Process customer purchases and generate bills.**
  - **Search Items – Locate specific products in the inventory.**
  - **Edit Items – Update product details such as price and stock.**
  - **Exit – Close the program.**

**2. Product Management Page**

- **Lists all available products with details including name, price, and stock.**
- **Allows the addition, modification, and removal of products.**
- **Provides a search functionality to quickly find products.**

**3. Billing Page**

- **Captures customer purchase details.**
- **Calculates the total price, including applicable discounts and taxes.**
- **Displays the payment options (cash, card, online).**
- **Generates an invoice for the customer.**

**4. Customer Information Page**

- **Stores and manages customer details such as name, contact number, and purchase history.**
- **Helps in tracking customer preferences for better service.**

**5. Inventory Management Page**

- **Monitors stock levels of all products.**
- **Sends alerts when stock is running low.**
- **Enables updates to product availability.**

**6. Exit Page**

- **Ensures proper program termination.**
- **Displays a thank-you message before closing.**

**The structured layout of these pages ensures a seamless and efficient user experience while managing supermarket operations.**

# IMPLEMENTATION:

The **Super Market Management System** is implemented using Python, focusing on efficient inventory management, billing, and customer transactions. The system follows a modular approach to ensure flexibility and ease of use.

**1. System Flow**

The implementation consists of the following major steps:

1. **User Input Handling:** The program interacts with the user through a menu-driven interface.
2. **Data Processing:** Product details, prices, and quantities are stored and updated dynamically.
3. **Transaction Management:** The system tracks purchases and updates stock levels accordingly.
4. **Validation & Error Handling:** Ensures that invalid inputs are not accepted (e.g., non-numeric values for prices and quantities).
5. **Output Display:** Displays bills, stock details, and customer purchase summaries.

**2. Code Modules**

The system is implemented in different modules for better organization:

- **Main Menu Module:** Displays the main menu and handles user selection.
- **Product Management Module:** Manages the addition, deletion, and modification of products.
- **Billing Module:** Calculates total cost, generates bills, and processes payments.
- **Search & Edit Module:** Allows users to search for products and update details.

**3. Database Structure**

The system stores data in a simple list-based structure within Python. Each product is stored as a dictionary with the following attributes:

python
CopyEdit

```
item = {
    "name": "Product Name",
    "quantity": 10,
    "price": 50
}
```

All items are stored in a list for easy retrieval and updating.

**4. Functional Flow Example**

1. **User selects an option from the menu.**

2. **For purchases:**
    - The system checks product availability.
    - The price is calculated, and stock is updated.
    - A bill is generated.
3. **For adding or editing products:**
    - The system takes input for new products or updates existing ones.
    - Validations ensure accurate data entry.
4. **For inventory checks:**
    - The system displays available products and their stock levels.

**5. Advantages of the Implementation**

☑ **Efficient Management** – Reduces manual work in tracking stock and processing purchases.

☑ **User-Friendly Interface** – Easy-to-navigate menu-based system.

☑ **Scalable Structure** – Can be extended to include a database for better data management.

☑ **Error Handling & Validation** – Prevents incorrect data entry and ensures smooth operations.

This structured implementation ensures smooth operation, accurate inventory tracking, and efficient customer transactions.

# SOFTWARE AND HARDWARE REQUIREMENTS:

## Hardware requirements:

•HP laptop (Windows 11 version)

•Wi-Fi

## Software requirements:

•Python IDLE

•Word windows 10

•Google Chrome

•Notepad

# PROGRAM AND OUTPUT

**PROGRAM:**

```
#----------------SUPERMARKET MANAGEMENT SYSTEM----------
---------#
items = []
while True:
    display = input('Press enter to continue.')
    print('-----------------Welcome to the supermarket-----------------')
    print('1. View items\n2. Add items for sale\n3. Purchase items\n4.
Search items \n5. Edit items\n6. Exit')
    choice = input('Enter the number of your choice : ')

    if choice == '1' :
        print('-----------------View Items-----------------')
        print('The number of items in the inventory are : ',len(items))
        while len(items) != 0:
            print('Here are all the items available in the supermarket.')
            for item in items:
                for key, value in item.items():
                    print(key, ':', value)
            break

    elif choice == '2' :
        print('-----------------Add items-----------------')
        print('To add an item fill in the form')
        item = {}
        item['name'] = input('Item name : ')
        while True:
            try:
                item['quantity'] = int(input('Item quantity : '))
                break
            except ValueError:
                print('Quantity should only be in digits')
        while True:
            try:
```

```
            item['price'] = int(input('Price $ : '))
            break
        except ValueError:
            print('Price should only be in digits')
    print('Item has been successfully added.')
    items.append(item)

elif choice == '3' :
    print('-----------------purchase items-----------------')
    print(items)
    purchase_item = input('which item do you want to purchase? Enter
name : ')
    for item in items:
        if purchase_item.lower() == item['name'].lower() :
            if item['quantity'] != 0 :
                print('Pay ', item['price'] , 'at checkout counter.')
                item['quantity'] -= 1
            else:
                print('item out of stock.')

elif choice == '4' :
    print('-----------------search items-----------------')
    find_item = input('Enter the item\'s name to search in inventory : ')
    for item in items:
        if item['name'].lower() == find_item.lower():
            print('The item named ' + find_item + ' is displayed below
with its details')
            print(item)
        else:
            print('item not found.')

elif choice == '5' :
    print('-----------------edit items-----------------')
    item_name = input('Enter the name of the item that you want to
edit : ')
    for item in items:
        if item_name.lower() == item['name'].lower():
            print('Here are the current details of ' + item_name)
```

```
            print(item)
            item['name'] = input('Item name : ')
            while True:
                try:
                    item['quantity'] = int(input('Item quantity : '))
                    break
                except ValueError:
                    print('Quantity should only be in digits')
            while True:
                try:
                    item['price'] = int(input('Price $ : '))
                    break
                except ValueError:
                    print('Price should only be in digits')
            print('Item has been successfully updated.')
            print(item)
        else:
            print('Item not found')

    elif choice == '6' :
        print('-----------------exited-----------------')
        break

    else:
        print('You entered an invalid option')
```

**output:**

```
------------------Welcome to the supermarket------------------
1. View items
2. Add items for sale
3. Purchase items
4. Search items
5. Edit items
6. Exit
Enter the number of your choice :
```

# TESTING:

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate.

Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics.

Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

# CONCLUSION:

business opportunities increasing as never before, companies are in dire need of efficient management. One such key area is to maintain a methodical way of managing large databases, especially in the Retail sector. DBMS is a vital tool for future growth of business organizations. It offers a simple, efficient and reliable way of storing, managing and accessing data. The features offered by DBMS are :Query ability, Backup, Security and Computation which are the needs of a fast-paced corporate system.

To churn profits, companies need to have a good plan along with affective DBMS.

# BIBLIOGRAPHY:


- https://www.python.org/Cbse text book
- https://kandi.openweaver.com/?landingpage=python_all_projects&landingpage=python&utm_campaign=paid_search_productcapability_search&utm_medium=cpc&utm_source=google&utm_term=python_all_projects&utm_content=code&gclid=Cj0KCQjwwfiaBhC7ARIsAGvcPe4HfVzrHrHo38rVOQLNhKgz9_2w14nq8OyISWGTcC8_RaCbDn_2LLEaAtt9EALw_wcB