CSC 1024

PROGRAMMING PRINCIPLES

TITLE OF REPORT:

A PYTHON-BASED INVENTORY MANAGEMENT SYSTEM

Prepared by:

Sian Yi Xuan (23099609)

Tan Wei Ting (23094709)

Tan Wen Xi (23093495)

Teoh En Yi (23051105)

Tiffany Fam Kar Ying (23052301)

Prepared for:

ASSOC. PROF. DR ANWAR P.P. ABDUL MAJEED

Date of Submission:

29 December 2024

Table of Contents

Video Presentation Link:

1.0 Explanation of System Design and Architecture

The Inventory Management System (IMS) is a system designed to efficiently handle the important inventory operations of product management, stock tracking, order processing, and reporting. The system is written in Python, a programming language that makes use of user-friendly, modular and can handle real-life scenarios of inventory control through modules. The system ensures that information will be available in subsequent sessions by using text files for storing the data.

The system incorporates three primary datasets: suppliers, orders, and products. These are stored in respective text files: Each file contains *products.txt* and *suppliers.txt* with structured data entries and *orders.txt* with product and supplier data entries. For instance, in products.txt we have product ID, name, description and price while in suppliers.txt we have supplier IDs, names and contact details. Having this design, the data can be retrieved and updated seamlessly, without losing their logical segregation information.

An important feature of the system is that it has a menu-driven interface from which the user can select and operate various functions. The included are adding new products or suppliers, updating product details, placing orders, looking at inventory, creating reports and exiting the program. Each function is implemented through Python user-defined functions which increase modularity and reuse code. For instance, once we have added a new product, we need to verify input formats, make sure the new product ID is unique and append it to the products.txt file. Just like placing an order validates that the product is available and updates stock levels.

The system is robust due to error handling. For example, cases that handle a file input or output error or that handle invalid user input gracefully are handled with techniques like try-except blocks. Using logical operators and loops makes sure repetitive types of tasks like duplicate checking or displaying inventory data work efficiently. In addition, conditional statements are used in the system to handle decision-making processes, including verification of stock levels prior to the processing of orders or the preparation of low stock reports.

The IMS consists of robust file handling, effective user interaction mechanism and modular programming and overall, it provides a comprehensive solution to the inventory control. To future enhance the architecture support of such features as a graphical user interface as well as integration with external systems for real-time updates, architecture has been designed to be extendable.

1.1 Add a New Product Function

The “Add new Product” feature is one of the core components of the IMS, which enables the user to enter actual data about a product into the IMS’s database. This function is very important in routinely updating the records and creates how the system can generate new tables as products are added. When the user select option Add a New Product, the application automatically asks for the Product ID, Product Name, Product Description, Price, and Stock (Quantity). All these inputs are important in ensuring that the record of the product in the system is as accurate as is possible.

First, the Product ID column must be of a certain format for example “P001” to mean the ID starts with P and then has three digits. It also has an added advantage of giving each product a number which would eliminate any confusion that may prevail in the records of the stores. With reference to the specified structure, the system will verify if the entered Product ID conforms to this format. If the input entered is in the wrong format, then they will receive an error message, then prompt them to re-enter the Product ID.

Then the user is asked to input Product Name and Description of the product that he wants to add online. This makes certain that these fields are not empty as a product may not be named or described for a good purpose in stock taking. The program also trims the inputs by removing any zero or more white spaces on left or right side of the string. If they have left the name or description empty, then the system will ask them to provide valid text.

Other categorizing that should be given include the Quantity of the product. For the quantity, the system insists on the value to be an integer greater or equal to 1 meaning there must be a stock whose management is hold by the system. Likewise, to store the Price of a product, it needs to be a valid number, so that the price data that is used when making transactions and preparing reports is correct.

When the product details are captured the system searches for similar products entered earlier. It scans the textual file containing current list of products, for example *products.txt*, to check the new Product ID to avoid duplication. If the product with the same ID is found, the user is informed, and the function will not allow the addition of such product. If there is no duplication, the product data is written to the file to keep updating the inventory with the new product information.

Through this function, IMS can have a current database of products avoiding fruits repeating itself by entering the wrong data many times hence providing only new products to the database. Moreover, the data validation mechanisms that accompany storages facilitate the data integrity and when using inventory as a storage, the user does not have to face some errors.

1.2 Update Product Details

The "Update Product Details" feature serves as an important part of the IMS. It is a feature that will allow the user to change the details of an existing product in the database. This function is what keeps the product records in the system clean and current. This includes taking users

through a series of processes to search and then update the product data they want to change. Users will select the “Update Product Details” option.
Immediately after the user selects “Update Product Details”, the application begins to read that ‘products.txt’ file, which is the inventory’s database. The file here provides some basic details about each of the products such as Product ID, Name, Description, Quantity and Price. Verification then is the first step to see if the file is there and that it’s not empty.

The function notifies the user of missing or incomplete files and aborts any operation on incomplete or missing records. The system will then display all the existing products in a clear and friendly format so that users can choose which product they would like to update. It shows each product with its ID, Name, Description, Quantity, and Price. They were ordered into a dictionary using Product ID to allow easy access. Additionally, the system shows the products and asks user to input the Product ID of the item that wants to update. If the entered Product ID does not exist in the inventory, the system will then alert the user and terminate the process. If the Product ID exists, the system will get the current details of the selected product.

The user is then asked to enter the product’s details: product Name, Description, Quantity, and Price. After the user prompts all these data, the system will ensure all the inputs are valid to ensure data integrity. For instance, the Name and Description fields cannot be left blank, the Quantity and Price of the item cannot be a negative number. In addition, if users wish to maintain the current values for those fields, they are allowed to leave blank for the specific field that does not want to change. This will keep the current values and keep this from overwriting data. The inputs will also update product information in the dictionary and back to the ‘products.txt’ file after the inputs are valid. All the changes made are saved and ready to be used again. Once the product details are updated it will confirm the user that the product details have been updated successfully. The system error handling mechanisms act as a robust function to work out problem handlers. If the ‘products.txt’ file is missing an error message will appear which informs the user. If validation errors occur, such as negative quantities or invalid prices, clear feedback will be provided to the user, in order for the application to run as smoothly and be user friendly.

In short, this feature was designed to facilitate update of product information in the inventory with accuracy, maintain data integrity and support system reliability. It might prevent errors and inconsistencies in your database. It is a pivotal function of IMS, it is responsible for the updating and keeping inventory data up to date.

1.3 Add a New Supplier

Firstly, this program starts with the function get_supplier() to read and parse existing supplier details into a list of dictionaries to check for duplicated entries. After opening the supplier.txt file in read mode, white spaces in each line will be removed, and each line will be split into a list of words based on white space. Based on the Python index, supplier_id is 0, name is 1, and contact is 2. Then, the function add_supplier will catch if the supplier details are duplicated.

To add a new supplier, users are required to enter the supplier ID, supplier name, and contact. However, each of these has specific requirements to limit what the users enter in order to let the supplier list be more organized and tidy. For supplier ID, it’s required to start with a letter (A-Z) and be followed by three digits (0-9). If the user enters more than four letters or digits, does not start with a letter, or is not followed by three digits after a letter, “Wrong supplier ID format, please try again” will be printed. For the supplier name, users are required to enter a name that only contains letters and has no spaces between letters; else, “Wrong name format. Please enter a name containing only letters and no spaces” will be printed. For supplier contact, the program only allowed 10-digit contact numbers for users to enter. If the users enter a contact number with more than 10 digits or there are other non-digit characters in the contact number, “Wrong contact format, please try again” will be printed. After successfully adding the supplier ID, supplier name, and contact, the function add_supplier(supplier_id, name, contact) will check if the information added is duplicated or not before printing out the new supplier details. Then, the program will ask the users whether they want to continue adding suppliers. If the users enter ‘No’ or ‘no,’ the latest supplier detail will be printed before the program ends. If the users enter ‘Yes’ or ‘yes,’ the function add_supplier_interactive() will be run again to let the users enter more supplier’s data.

1.4 Place an Order

This function is essential which enables user to order for desired products while ensuring inventory stock levels are updated and order records are maintained.

Firstly, the program will load product details from the products.txt file. The program will verify the availability of product in the file. If the file is empty, the program will notify user with an error message and terminate the process. The process will continue to process if products are available by displaying a clear overview of the inventory with extract details such as Product ID, Name, Description, Price, Stock in a formatted table. Next, the program will prompt user to enter desired product ID and quantity. The input of quantity will be validated to ensure it is a valid number and positive integer. Besides, the program will check whether the entered Product ID exists in the inventory. The requested quantity will be also checked against the available stock to ensure sufficient inventory is present. If the Product ID is invalid or the quantity exceeds available stock, the system will display an error message and stop the process. Once the inputs are validated, the system proceeds to update the inventory. It deducts the requested quantity from the current stock of the selected product. The updated product details are then written back to the products.txt file. Simultaneously, a unique Order ID is generated to track the transaction. This ID, along with the Product ID, Quantity, and Order Date, is recorded in the orders.txt file for future reference and reporting purposes. Lastly, a message will be displayed to inform user that the order is placed successfully.

1.5 View Inventory

The primary goal of the "View Inventory" functionality is to present users with the most up-to-date information about products available in the inventory. The program begins by reading data from the PRODUCTS_FILE which contains product details. Using the readlines() method, the function retrieves all entries, where each line represents a distinct product. If the file is empty, the program will display an error message and exit the process.

To ensure a clear and organized inventory display, the program formats the data in a tabular layout. A header row comprising column names like Product ID, Name, Description, Price, and Stock is followed by a separator line of dashes. Each product's details are extracted by splitting the corresponding file line based on commas. The length of the descriptions which exceeding 27 characters will be truncated and appended with an ellipsis.

Therefore, the "View Inventory" functionality effectively achieves its objective of presenting inventory data in a clear and accessible format.

1.6 Generate Reports

The 'Generate Reports' is an important part of the IMS which generates detailed information about the inventory. This function generates three types of reports: product sales, low stock items, and supplier orders. The IMS offers these reports to enable users to have access to critical information required in managing and making inventory decisions.

It starts by generating a report on low stock items. This reads product data from `products.txt` which contains a few important things about each product such as Product ID, Name,

Description, Quantity and Price. This data is processed by the system line by line and products are identified whose stock quantity is lower than a predefined threshold for example, below 5 units. They are compiled into a list of low stock items, and displayed to the user. The system informs the user if there are no low stock items. The inventory control is essential for timely restocking of products and this report is used for this purpose.
Then, the function is used to generate a report on the product sales from the `orders.txt` file. This file has Order ID, Product ID, Quantity and Order Date details about orders. Then the dictionary is used to aggregate the sales data to calculate the total quantity of sold for each product by the system, given each order. This is displayed as an aggregate of total sales of each product. However, if no sales data is found, the user is told. This product performance report will give you an idea of which items are selling the best, and how to stock them.

The ‘suppliers.txt’ file is used to derive the supplier orders and the final report concentrates on these. This file consists of supplier information including Supplier ID, Name and Contact Details. This data is read by the system and a list of all suppliers is displayed for users to access supplier information. Maintaining strong relationships with suppliers and a reliable supply chain are further assisted by this report.

It has robust error handling mechanism to make it reliable function. If any of the required files such as ‘products.txt’, ‘orders.txt’ or ‘suppliers.txt’ is missing, the system will give the user an appropriate error message. The function also allows for unexpected errors to be handled gracefully ensuring the application is stable, and friendly for the user. The “Generate Reports” feature in a nutshell is a great tool for getting an all around picture of inventory levels, sales performance, and supplier relationships. The IMS requires that it be maintained in efficient operations, informed decisions, and all success.

1.7 Exit

Finally, last feature in the IMS is the “Exit” function that also has a clearly understandable interface and allows the user to comfortably end the functioning of a program. The system utilizes a simple menu-based interface that constantly presents options of the different functions available at any one time, for example; entering a new product, modifying details of any product being sold or checking on the stock. Therefore, the Exit function is provided to give the users the full control of the session and it is as intuitive as possible.

However, when the client chooses the “Exit” choice from the menu they are asked whether they really want to exit for instance, “Are you sure you want to EXIT (Y/N)?”. This confirmation step is necessary so that, in the case when the user has selected this option by mistake, he or she can cancel it by pressing on the cancel button. If the user has inserted “y” (yes) the program will show a polite message like “Thank you for using the Inventory Management System” then exit the execution safely. When the user enters an “n” (no), the program goes back to the main control menu to enable the user to continue using the system.

As part of the utilization of this function, the user must enter either ‘y’ or ‘n’, specifying the kind of result he needs. If the user enters an invalid response, the system will ask the user again to enter the response which makes the system turn into a loop until a correct value is inserted. This approach has the added advantage of improving the general user experience, as well as the reduction of confusion and of errors. When incorporated in this way, the “Exit” function of the Inventory Management System guarantees the user a safe and civilized way to pull the plug on the show, while not compromising on ease or efficiency.

2.0 Discussion of Implementation Challenges and Solutions

1. The Problem of File Handling

Challenge:

Writing to files or reading from the files cause various errors. For example, data corruption or missing files.

Solution:

To solve this problem, error handling allows the code to employ try-except blocks. This will then allow it to prevent crashes and manage the file-related exception. Besides, the code predicts the absence of files and provide accurate messages. For instance, ("Note: products.txt file is not found. New file is creating.") Lastly, the program has implemented the function of atomic updates to read all file content into memory, thereby updating it. The risk of corruption will then be minimised.

2. The Challenge of Data Validation

Challenge:

The data such as names, product IDs, prices, supplier information and quantities inputted by the users that accommodate the required format can be an error pane.

Solution:

Input validation is implemented to ensure the code uses clear validation checks for inputs. For example, inputs such as verify names are non-empty, quantities are in positive integers, and the product IDs are ensured to follow a specific format. Besides, the feedback mechanism will also help users to correct their input as there are error messages to guide the users. This will them improve the usability. Error messages such as ("Error: Supplier name must only contain letters.") are used.

3.The Usability Challenge

Challenge:

Some users might think that the system is difficult to understand or navigate without the proper guidance.

Solution:

To solve this problem, instructional prompts allow the program to provide each operation with clear instructions. In addition, immediate feedback will be provided when the operation is successful or has errors. For example, it will display confirmation messages such as ("The product is added successfully.") when the operation is successful. However, if it is an error, an explanation will be provided.

4.The Challenge of Scalability and Maintenance

Challenge:

Extending functionality and managing data become complex when the system grows.

Solution:

Clear documentation such as comments that describe the logic and purpose of each function is provided and aids future developers. The modular design is modified which allows the functions to focus on specific tasks and become well-defined. This will then allow the code to be modified or extended.

5.Duplicate Entries Challenge

Challenge:

Duplicate entries lead to inconsistencies. For example, duplicate entries such as supplier details or product IDs.

Solution:

The system will undergo duplicate checks before adding a new supplier or product which checks the existing records for duplicate. This will be handled using conditions and loops to compare the new entries with existing ones. Furthermore, feedback on duplicates will allow the system to inform the user and prevent some operations. For instance, ("Duplicate entry is detected. Please try again with other special details.") and ("Error: The product ID already existed.")

3.0 Analysis of the System’s Strengths and Limitations

3.1 Strengths

The System is User-Friendly

The program provides error messages and clear prompts to make it easier for users to correct and understand their inputs. Besides, the main menu of the program provides an instinctive interface for users. This will then allow users to interact with different functionalities.

The System Consists of Data Validation

The data integrity is ensured when the input validation is implemented. For example, the checks are performed for positive integers for quantity, product ID format, and pieces that are in a valid numeric input. Moreover, the contact information and supplier IDs will also undergo challenging validation to minimise errors.

The System Provide Real-Time Updates

Some functions such as (place_order) and (update_product_details) will ensure the inventory is updated in real-time. As a result, this will help in reflecting the accurate stock levels.

The System Construct Reports Generation

Basic report generation is included in the system. For example, viewing the supplier details and identifying the low-stock items. Therefore, these allow the users to monitor the inventory effectively.

The System is Functionally Coverage

The features in the system are logically organised. Besides, the main program in the system will ensure that easy navigation is conducted through the menu options. Furthermore, core functionalities such as managing suppliers, viewing inventory, updating products, generating reports, and placing orders for inventory management are implemented by the system.

3.2 Limitations

The Storage is File-Based

The system relies on plain text files that result in the limitation of performance and scalability. This will happen especially when the program has larger datasets. Besides, the example of the plain text files are (product.txt and suppliers.txt). The structured or indexing querying like a database is not included in the system. Thus, it is inefficient for users to update or search data in large files.

The System's Lack of Security

The sensitive operations such as modifying inventory in the system are not secured. This will then cause the system to tend to spiteful or accidental misuses. Besides, the system does not consist of role-based access control and user authentication.

Weak UI or UX Design

The appeal and usability are limited compared to the graphical interfaces as the system is text-based. Apart from that, the system also doesn't support advanced navigation. This will then cause the users unable to go back to the previous menu.

4.0 Suggestions for Future Improvements and Enhancements

Add Filter and Search Options

The program should apply a search feature to improve. This is because it will allow the user to locate specific suppliers and products. It also allows the user to locate orders referring to criteria like supplier name, product ID, and order date. As a result, users will have the ability to find the relevant data quickly especially if the inventory goes large.

Enhance Editing and Data Update

In this program, functions such as updating the product details exist, but it should be extended to other modules to improve. For instance, updating order details and suppliers. This is to ensure that all information stored by the users can be modified instantly through the program. Hence, this will then help to reduce the need for manually editing the files.

Automate Data Validation Among All Functions

At this state, the input validation has already been applied individually for each function. Thus, a centralised validation system should be introduced to improve the program. This is to improve the consistency and reduce redundancy. This will then allow the users to reuse the functions such as validate quantities, prices and IDs.

Integrate Help and Documentation

The program should add a devoted help menu system to improve. This will guide the users through various functionalities. As a result, the system in the program will become more accessible to the users, especially new users.

Support Undo Function

An undo function should be implemented in the program to improve. The undo feature is available for important operations such as deleting or updating products. This is to make sure all accidental changes made by the users can be reversed. Therefore, this will then improve user confidence and data integrity.