**Nike Shoe Inventory Management System**

**Overview**

The Nike Shoe Inventory Management System is a Python-based application designed to manage an inventory of shoes efficiently. This system utilizes the **Tkinter** library for the graphical user interface (**GUI**) and **JSON** for data storage. It supports operations such as adding, searching, updating, and deleting shoe records. Additionally, it provides various reports on the inventory, including **total shoes**, **shoes by color**, **shoes by size**, and a **comprehensive report**.

**Company Background**

Nike, Inc. is a global leader in the design, manufacturing, and marketing of athletic footwear, apparel, and equipment. Founded on January 25, 1964, as Blue Ribbon Sports and officially becoming Nike, Inc. in 1971, the company has grown to become one of the most recognized brands worldwide.

Named after the Greek goddess of victory, Nike is renowned for its innovative products, cutting-edge technology, and marketing prowess. This project focuses on managing Nike's shoe inventory, aiming to streamline operations, improve inventory accuracy, and enhance decision-making processes.

**System Objective**

The Nike Shoe Inventory Management System is designed to provide an efficient and user-friendly interface for managing the inventory of Nike shoes. The system aims to achieve the following objectives:

1. Efficient Inventory Management: Streamline the process of adding, updating, searching, and deleting shoe inventory.
2. Accurate Reporting: Generate comprehensive reports on the total number of shoes, shoes by color, and shoes by size.
3. Enhanced User Interaction: Provide a clear and intuitive graphical user interface (GUI) for users to interact with the system.
4. Data Integrity: Ensure that all inventory data is accurately recorded, updated, and stored in a secure manner.
5. Improved Decision Making: Provide detailed and accurate reports to help management make informed decisions regarding inventory and sales strategies.

**Functions for User Input and Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **add\_shoe\_gui()** | Adds a new shoe to the inventory. | Text entries | MessageBox (Success/Error) |
| **search\_shoe\_gui()** | Searches for a shoe in the inventory by SKU. | Text entry | Custom MessageBox (Shoe Details) |
| **update\_shoe\_gui()** | Updates the details of an existing shoe in the inventory. | Text entries | MessageBox (Success/Error) |
| **delete\_shoe\_gui()** | Deletes a shoe from the inventory based on SKU. | Text entry | MessageBox (Success) |
| **generate\_reports()** | Generates different types of reports based on selected report type. | Radio buttons | Custom MessageBox (Report Details) |

**Main Components**

The system is divided into four main components:

* GUI (**main.py**)
* Inventory Management (**inventory.py**)
* Reporting (**report.py**)
* Shoe Class (**shoe.py**)
* GUI (**main.py**)

The GUI is built using the **Tkinter** library. It consists of multiple sections for different functionalities: **adding** shoes, **updating** shoes, **deleting** shoes, **searching** shoes by **SKU**, and generating reports. Here's a breakdown of each section:

**Adding a Shoe**

The add\_shoe\_gui() function allows users to add a new shoe to the inventory by entering the shoe's name, size, color, quantity, and SKU. This function validates the size and quantity inputs to ensure they are integers and then calls the add\_shoe() function from inventory.py to update the inventory. Upon successful addition, a success message is displayed using messagebox.showinfo(). If there's an error in the input, an error message is shown using messagebox.showerror().

**Searching for a Shoe**

The search\_shoe\_gui() function enables users to search for a shoe in the inventory by its SKU. It calls the search\_shoe() function from inventory.py and, if the shoe is found, displays the shoe details in a custom message box. If the shoe is not found, an error message is displayed.

**Updating a Shoe**

The update\_shoe\_gui() function allows users to update the details of an existing shoe in the inventory. Users can enter the new name, size, color, and quantity. The function validates the size and quantity inputs to ensure they are integers. It then calls the update\_shoe() function from inventory.py to update the shoe details. A success message is displayed upon successful update, or an error message if the shoe is not found.

**Deleting a Shoe**

The delete\_shoe\_gui() function lets users delete a shoe from the inventory by entering its SKU. It calls the delete\_shoe() function from inventory.py to remove the shoe from the inventory. A success message is displayed upon successful deletion.

**Generating Reports**

The generate\_reports() function generates different types of inventory reports based on the user's selection from the radio buttons. It can generate a total shoes report, shoes by color report, shoes by size report, or a comprehensive report. The selected report is displayed in a custom message box.

**Add Shoe**

Users can add new shoes to the inventory by entering the **name**, **size**, **color**, **quantity**, and **SKU**. The data is validated before being passed to the **add\_shoe** function.

def add\_shoe\_gui():  
 name = name\_entry.get()  
 size = size\_entry.get()  
 color = color\_entry.get()  
 quantity = quantity\_entry.get()  
 sku = sku\_entry.get()  
 try:  
 size = int(size)  
 quantity = int(quantity)  
 add\_shoe(name, size, color, quantity, sku)  
 messagebox.showinfo("Success", "Shoe added successfully.")  
 except ValueError:  
 messagebox.showerror("Error", "Size and Quantity must be integers.")

**Search Shoe**

Users can search for a shoe by its **SKU**. If found, the shoe's details are displayed in a custom message box.

def search\_shoe\_gui():  
 sku = search\_sku\_entry.get()  
 shoe = search\_shoe(sku)  
 if shoe:  
 shoe\_details = (  
 f"Shoe Details:\n"  
 f"Name: {shoe['name']}\n"  
 f"Size: {shoe['size']}\n"  
 f"Color: {shoe['color']}\n"  
 f"Quantity: {shoe['quantity']}\n"  
 f"SKU: {shoe['sku']}"  
 )  
 custom\_messagebox("Shoe Found", shoe\_details)  
 else:  
 custom\_messagebox("Error", "Shoe not found.")

**Update Shoe**

Users can update details of an existing shoe by providing its **SKU** and the new details.

def update\_shoe\_gui():  
 sku = update\_sku\_entry.get()  
 updates = {}  
 if update\_name\_entry.get():  
 updates['name'] = update\_name\_entry.get()  
 if update\_size\_entry.get():  
 try:  
 updates['size'] = int(update\_size\_entry.get())  
 except ValueError:  
 messagebox.showerror("Error", "Size must be an integer.")  
 return  
 if update\_color\_entry.get():  
 updates['color'] = update\_color\_entry.get()  
 if update\_quantity\_entry.get():  
 try:  
 updates['quantity'] = int(update\_quantity\_entry.get())  
 except ValueError:  
 messagebox.showerror("Error", "Quantity must be an integer.")  
 return  
 updated\_shoe = update\_shoe(sku, \*\*updates)  
 if updated\_shoe:  
 messagebox.showinfo("Success", "Shoe updated successfully.")  
 else:  
 messagebox.showerror("Error", "Shoe not found.")

**Delete Shoe**

Users can delete a shoe from the inventory by entering its **SKU**.

def delete\_shoe\_gui():  
 sku = delete\_sku\_entry.get()  
 delete\_shoe(sku)  
 messagebox.showinfo("Success", "Shoe deleted successfully.")

**Generate Reports**

Users can generate various reports based on the inventory data. The available reports include **total shoes**, **shoes by color**, **shoes by size**, and a **comprehensive report**.

def generate\_reports():  
 report\_type = report\_var.get()  
 if report\_type == 'Total Shoes':  
 custom\_messagebox("Total Shoes Report", generate\_total\_shoes\_report())  
 elif report\_type == 'Shoes by Color':  
 report = generate\_shoes\_by\_color\_report()  
 report\_str = "\n".join(f"{color}: {quantity}" for color, quantity in report.items())  
 custom\_messagebox("Shoes by Color Report", report\_str)  
 elif report\_type == 'Shoes by Size':  
 report = generate\_shoes\_by\_size\_report()  
 report\_str = "\n".join(f"Size {size}: {quantity}" for size, quantity in report.items())  
 custom\_messagebox("Shoes by Size Report", report\_str)  
 elif report\_type == 'Comprehensive Report':  
 custom\_messagebox("Comprehensive Report", generate\_comprehensive\_report())  
 else:  
 custom\_messagebox("Error", "Invalid report choice.")

1. **Inventory Management (inventory.py)**

This module handles the core operations of managing the inventory, including adding, searching, updating, and deleting shoe records. It uses a **JSON** file (**database.json**) to store the inventory data.

**Load and Save Inventory**

These functions handle reading from and writing to the **JSON** file.

def load\_inventory():  
 try:  
 with open(INVENTORY\_FILE, 'r') as file:  
 return json.load(file)  
 except FileNotFoundError:  
 return []  
  
def save\_inventory(inventory):  
 with open(INVENTORY\_FILE, 'w') as file:  
 json.dump(inventory, file)

**Add Shoe**

This function adds a new shoe to the inventory.

def add\_shoe(name, size, color, quantity, sku):  
 inventory = load\_inventory()  
 shoe = Shoe(name, size, color, quantity, sku).\_\_dict\_\_  
 inventory.append(shoe)  
 save\_inventory(inventory)

**Search Shoe**

This function searches for a shoe by its SKU.

def search\_shoe(sku):  
 inventory = load\_inventory()  
 for shoe in inventory:  
 if shoe['sku'] == sku:  
 return shoe  
 return None

**Update Shoe**

This function updates the details of an existing shoe.

def update\_shoe(sku, \*\*kwargs):  
 inventory = load\_inventory()  
 for shoe in inventory:  
 if shoe['sku'] == sku:  
 shoe.update(kwargs)  
 save\_inventory(inventory)  
 return shoe  
 return None

**Delete Shoe**

This function deletes a shoe from the inventory by its SKU.

def delete\_shoe(sku):  
 inventory = load\_inventory()  
 inventory = [shoe for shoe in inventory if shoe['sku'] != sku]  
 save\_inventory(inventory)

1. **Reporting (report.py)**

This module generates various reports based on the inventory data.

**Total Shoes Report**

This function generates a report of the total number of shoes in the inventory.

def generate\_total\_shoes\_report():  
 inventory = list\_shoes()  
 total\_shoes = sum(shoe['quantity'] for shoe in inventory)  
 return f"Total number of shoes: {total\_shoes}"

**Shoes by Color Report**

This function generates a report of the number of shoes by color.

def generate\_shoes\_by\_color\_report():  
 inventory = list\_shoes()  
 report = {}  
 for shoe in inventory:  
 color = shoe['color']  
 if color in report:  
 report[color] += shoe['quantity']  
 else:  
 report[color] = shoe['quantity']  
 return report

**Shoes by Size Report**

This function generates a report of the number of shoes by size.

def generate\_shoes\_by\_size\_report():  
 inventory = list\_shoes()  
 report = {}  
 for shoe in inventory:  
 size = shoe['size']  
 if size in report:  
 report[size] += shoe['quantity']  
 else:  
 report[size] = shoe['quantity']  
 return report

**Comprehensive Report**

This function generates a comprehensive report that includes the total number of shoes, shoes by color, and shoes by size.

def generate\_comprehensive\_report():  
 total\_shoes = generate\_total\_shoes\_report()  
 shoes\_by\_color = generate\_shoes\_by\_color\_report()  
 shoes\_by\_size = generate\_shoes\_by\_size\_report()  
  
 report = f"{total\_shoes}\n\nShoes by Color:\n"  
 for color, quantity in shoes\_by\_color.items():  
 report += f"{color}: {quantity}\n"  
  
 report += "\nShoes by Size:\n"  
 for size, quantity in shoes\_by\_size.items():  
 report += f"Size {size}: {quantity}\n"  
  
 return report

1. **Shoe Class (shoe.py)**

This class represents a shoe and its attributes. It is used to create shoe objects in the inventory.

class Shoe:  
 def \_\_init\_\_(self, name, size, color, quantity, sku):  
 self.name = name  
 self.size = size  
 self.color = color  
 self.quantity = quantity  
 self.sku = sku  
  
 def \_\_repr\_\_(self):  
 return (  
 f" name='{self.name}',\n"  
 f" size={self.size},\n"  
 f" color='{self.color}',\n"  
 f" quantity={self.quantity},\n"  
 f" sku='{self.sku}'\n"  
 )

**Flow of Data**

**Adding a Shoe:**

* User inputs the shoe details in the GUI.
* The **add\_shoe\_gui** function validates the input and calls the **add\_shoe** function.
* The **add\_shoe** function creates a Shoe object and adds it to the inventory, which is then saved to **database.json**.

**Searching for a Shoe:**

* User inputs the SKU in the GUI.
* The **search\_shoe\_gui** function calls the **search\_shoe** function.
* The **search\_shoe** function searches for the shoe in the inventory and returns the details if found.

**Updating a Shoe:**

* User inputs the SKU and new details in the GUI.
* The **update\_shoe\_gui** function validates the input and calls the **update\_shoe** function.
* The **update\_shoe** function updates the shoe details in the inventory, which is then saved to **database.json**.

**Deleting a Shoe:**

* User inputs the SKU in the GUI.
* The **delete\_shoe\_gui** function calls the **delete\_shoe** function.
* The **delete\_shoe** function removes the shoe from the inventory and saves the updated inventory to **database.json**.

**Generating Reports:**

* User selects a report type in the GUI.
* The **generate\_reports** function calls the appropriate report generation function.
* The report is displayed in a custom message box.

**Test Data**

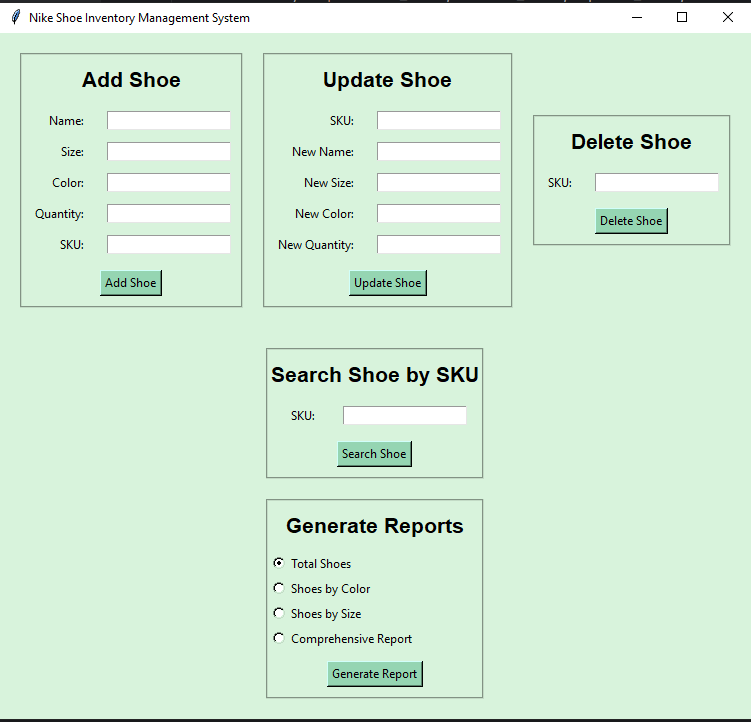
The **database.json** file contains sample data for testing purposes:

[  
 {  
 "name": "Onitsuka Tiger",  
 "size": "42",  
 "color": "Black",  
 "quantity": 79,  
 "sku": "XYZ-6789"  
 },  
 {  
 "name": "Jorden 11",  
 "size": 45,  
 "color": "Red",  
 "quantity": 66,  
 "sku": "GUI-6754"  
 },  
 {  
 "name": "Nike Dunk Low",  
 "size": 35,  
 "color": "Black",  
 "quantity": 78,  
 "sku": "XYZ-0987"  
 }  
]

**Screenshots**

To enhance the understanding of the system, here are some screenshots of the application:

**Main GUI Window:**



This shows the main window with sections for adding, updating, deleting, searching shoes, and generating reports.

**Add Shoe Section:**

A screenshot of a computer

Description automatically generated

This section allows users to input details for adding a new shoe.

**Search Shoe Section:**

A green box with black text

Description automatically generated

This section enables users to search for a shoe by SKU and view its details.

**Update Shoe Section:**

A screenshot of a update shoe

Description automatically generated

This section allows users to update the details of an existing shoe.

**Delete Shoe Section:**

A screenshot of a computer

Description automatically generated

This section allows users to delete a shoe from the inventory.

**Generate Reports Section:**

A screenshot of a screenshot of a report

Description automatically generated

This section provides options to generate various inventory reports.