Project: Tool Rental Inventory Management System

Description

This project sets up a tool rental inventory management system. It allows users to add, remove, update, and view inventory items. The system supports multiple user roles, specifically administrator and manager, each with different inventory management permissions. The system also includes a report generation feature, which allows users to create reports for low-stock items or tools that are about to expire. Data persistence is achieved through the use of a JSON file, which allows the inventory to be saved to and loaded from. The interactive menu allows users to manage inventory using a terminal interface, making it simple and intuitive to use.

Key Features:

1. Inventory Management: Add, remove, and update tools in the inventory.

2. User Roles: Admins have full access to all operations; managers have limited access.

3. Report Generation: Generate reports for low-stock items and items nearing expiry.

4. Data Persistence: Save and load inventory data using a JSON file.

5. Command-Line Interface: An interactive menu to guide users through available actions.

Structure Overview:

- Item Class: Represents individual items (tools) with attributes such as item_id, name, category, quantity, rental_price, and additional attributes.

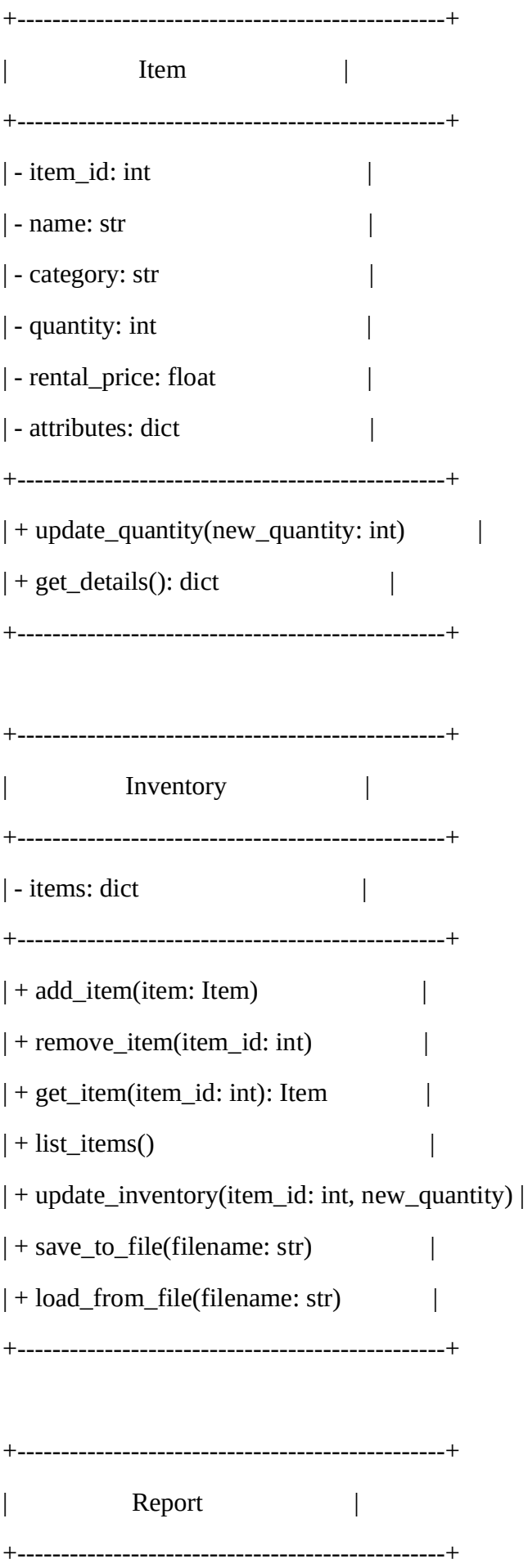
- Inventory Class: Manages a collection of Item objects. Provides methods to add, remove, update, and list items.

- Report Class: Generates reports based on the current inventory data (low stock or items nearing expiration).

- User Class: Handles user authentication and permissions (admin vs. manager). Ensures that users can only perform actions allowed by their role.

- Menu System: Provides a terminal-based interface for interacting with the inventory system.

Class Diagram:


Item
- item_id: int
- name: str
- category: str
- quantity: int
- rental_price: float
- attributes: dict
+ update_quantity(new_quantity: int)
+ get_details(): dict
Inventory
- items: dict
+ add_item(item: Item)
+ remove_item(item_id: int)
+ get_item(item_id: int): Item
+ list_items()
+ update_inventory(item_id: int, new_quantity)
+ save_to_file(filename: str)
+ load_from_file(filename: str)
Report

| - inventory: Inventory |

+--+

| + low_stock_report(threshold: int) |

| + expiry_report() |

| + generate_report(report_type: str, *args) |

+--+

+--+

| User |

+--+

| - username: str |

| - role: str |

| - permissions: dict |

+--+

| + get_permissions() |

| + perform_inventory_actions(action: str, ...) |

+--+

Explanation of the Diagram:

- Item: Represents the tools in the inventory, with methods to update quantities and retrieve details.

- Inventory: Manages all `Item` objects, including adding, removing, and updating tools. It also supports saving and loading data from a file.

- Report: Provides methods for generating reports on inventory status, such as low stock or expiring items.

- User: Ensures only authorized users (based on their role) can perform specific actions on the inventory.

Instructions for Use

1. Starting the System:

- Run the program in a terminal using the command: python <script_name>.py.

2. User Login:

- When prompted, enter your username and role (either "admin" or "manager").
- The system will validate your role and provide access to the functionalities available to that role.

3. Menu Options:
- The system presents a menu with the following options:
    1. Add an item to the inventory.
    2. Remove an item from the inventory.
    3. Update the quantity of an existing item.
    4. List all items in the inventory.
    5. Generate a report (low stock or items nearing expiry).
    6. Save the current inventory to a file.
    7. Load inventory from a file.
    8. Exit the system.

4. Adding an Item:
- Select option 1 and enter the details of the item (ID, name, category, quantity, rental price).
- The system will add the item to the inventory.

5. Removing or Updating Items:
- Select options 2 or 3 and enter the item ID to remove or update the item's quantity.

6. Generate Reports:
- Select option 1 and choose between low_stock or expiry reports. Provide any necessary inputs (e.g., stock threshold for low stock reports).

7. Saving and Loading:
- Use options 6 or 7 to save the current inventory to a file or load inventory data from a file.

Code Sanity and Functional Verification

1. Adding Items:

- Test adding multiple items to ensure they are correctly added to the inventory.
- Verify by listing all items in the inventory.

2. Removing Items:

- Test removing items and check that they are removed from the list.
- Try removing an item that does not exist to ensure the system handles this correctly.

3. Updating Inventory:

- Update the quantity of items and verify the new quantity is correctly reflected in the inventory.

4. Report Generation:

- Generate a low-stock report by adding items with different quantities and verifying that only the correct items are listed.
- Generate an expiry report by adding items with expiration dates and verifying the output.

5. Data Persistence:

- Save the inventory to a file and check the JSON file for accuracy.
- Reload the inventory from the file and ensure the items are correctly reloaded.

6. User Permissions:

- Log in as both admin and manager to verify that users have the correct permissions based on their role (e.g., only admins can remove items).

Findings, Challenges, and Areas for Improvement

Findings:

- The project provides a functional inventory system that manages tool rentals with user-specific access control. It allows for essential operations like adding, updating, and reporting on inventory.

Challenges:

- Handling permissions: Ensuring that different user roles have the correct access control was challenging but essential for security.

- Data persistence: Managing file operations (saving/loading inventory data) while ensuring data integrity required careful handling, especially when items have additional attributes.

Areas for Improvement:

1. Error Handling: More robust error handling could be implemented for invalid inputs, such as non-existent item IDs or improper data types.

2. Sales Tracking: Adding functionality for tracking tool rentals (e.g., who rented what, when) would enhance the usability for a real tool rental business.

3. User Authentication: Currently, the system only handles basic user roles. Implementing a proper login system with password encryption would improve security.

4. Database Integration: Instead of using a JSON file, switching to an SQLite or other database system would allow for more scalable and complex data handling.