**Smart Item Management System using Object Oriented Programming in Java**



Name : Ahmad Zaydan

Student ID : 2702358393

Lecturer : Jude Joseph Lamug Martinez, MCS

Course : Object Oriented Programming

Faculty : Computer Science

Binus International University 2023/2024

**Table of Contents**

**Table of Contents1**

**Abstract2**

**Chapter 1 – Project Specification**3

1.1 Introduction3

1.2 Idea and Inspiration 3

**Chapter 2 – Solution Design4**

2.1 Data4

2.2 Data Processing4

2.3 Storing Items4

2.4 Capabilities4

2.5 Class Diagram5

**Chapter 3 – Discussions on Implementation 6**

3.1 How It Works6

3.2 Algorithm6

3.3 Data Structure9

**Chapter 4 – Screenshots 11**

**Chapter 5 – Lesson Learned 12**

**Chapter 6 – Source Code 12**

**Chapter 7 – References 13**

**Chapter 8 – Poster 13**

**Abstract**

This report was made to document my work on the final project for the Object Oriented Programming Course, the project is about a program for a machine or system that is able to store inputted items and store them based on the kinds of items they are and in which category they belong to. I hope that by making this report, anyone who reads it are able to learn something and that the project itself may fulfil people’s needs.

**Chapter 1 – Project Specification**

* 1. **Introduction**

This project was made in order to fulfill my final project assignment for Object Oriented Programming course. For this assignment, I made a simple program with the Java programming language that can store inputted items and store them based on the kinds of items they are and in which category they belong, by implementing object-oriented programming properties such as List and Hashmap.

* 1. **Idea and Inspiration**

The idea for this project came to me when I was playing a game called Minecraft, at the time, I was making a machine in Minecraft that could sort, store, and retrieve items automatically using called Redstone mechanism. I was astonished by what I was able to make in the game using only the mechanics provided, and I thought to myself, if only such machinery existed in real life, it could solve so many people’s problems. Imagine someone just got home from somewhere bringing items such as groceries, kitchenware, bathroom essentials, and furniture, and they needed a way to store those items. With a machine like the one in Minecraft, it can automatically store them based on what kind of items they are. All they need to is to put the items in the machine. Now, to make a machine like that is not an easy feat and beyond my expertise. But, making the system for it is doable. So, I decided to make an item management system that can automatically categories the inputted items without the user telling the categories.

**Chapter 2 – Solution Design**

**2.1 Data**

For it to be “smart”, I utilize a “.txt” file containing a list of household items along with their category and sub-category. I chose “.txt” because it’s easy to modify and read with java. The category names each start with “###” symbols. The sub-categories start with a “-” symbol followed by the name and a “:” symbol. After the “:” symbol comes the name of the items. For example, the category name is Food, and inside food, there are sub-categories like groceries, fruits, etc. Followed by the sub-categories there is a list containing the item’s names like apple, oranges, etc.

In the “.txt” file it would be written like this:

### Food

- Fruits: Apple, Banana, Orange, Grape, Berry

- Vegetables: Carrot, Broccoli, Spinach, Potato, Tomato

- Dairy: Milk, Cheese, Yogurt, Butter, Egg

- Meat: Chicken, Beef, Pork, Fish, Bacon

- Grains: Bread, Rice, Pasta, Cereal, Oats

- Snacks: Chips, Cookies, Nuts, Chocolate, Crackers

- Beverages: Water, Juice, Coffee, Tea, Soda

- Condiments: Ketchup, Mustard, Mayonnaise, Soy Sauce, Vinegar

**2.2 Data Processing**

For the data processing, I created a “HouseholdItems” class that reads the list in the “.txt” file, by first searching for category names tagged with “###” symbols. Then it will search for the sub-category names by scanning for a “-” symbol. When that symbol is found, the program will scan for a “:” symbol in that same line and it will search whether the item inputted exists or not from the list of items that came after the “:” symbol separated by commas. If the item does exist, it will return each category and sub-category name.

**2.3 Storing Items**

After identifying in which category and sub-category the item belongs, the item will be registered with the ”Item” class which serves as an inheritance to the “Inventory” class. The “Inventory” class is the one managing the storage itself. It creates a new storage based on the category and sub-category of the item. What’s amazing is that the user is not required to state the category by themselves, the program already did that for them.

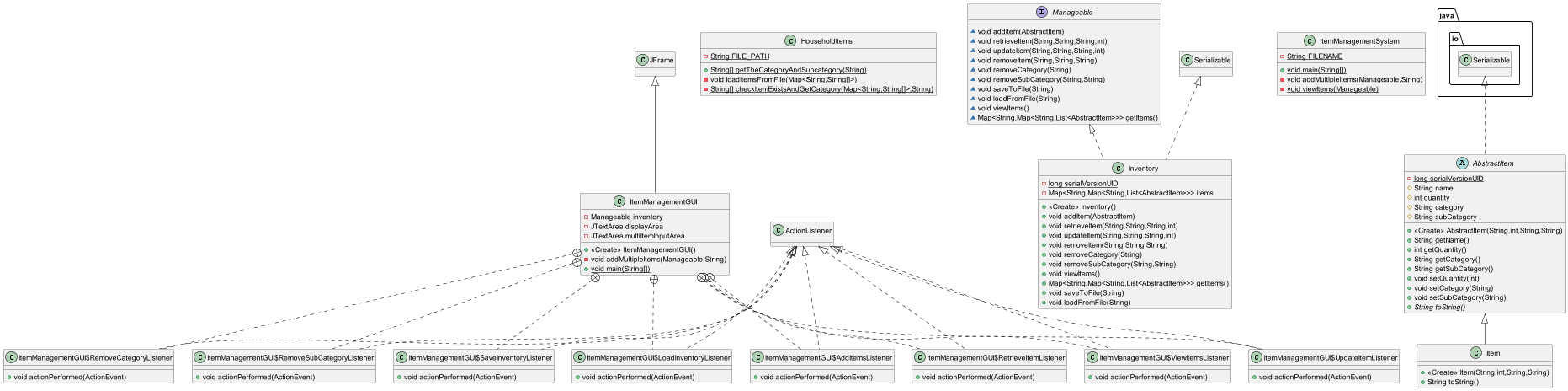
**2.4 Capabilities**

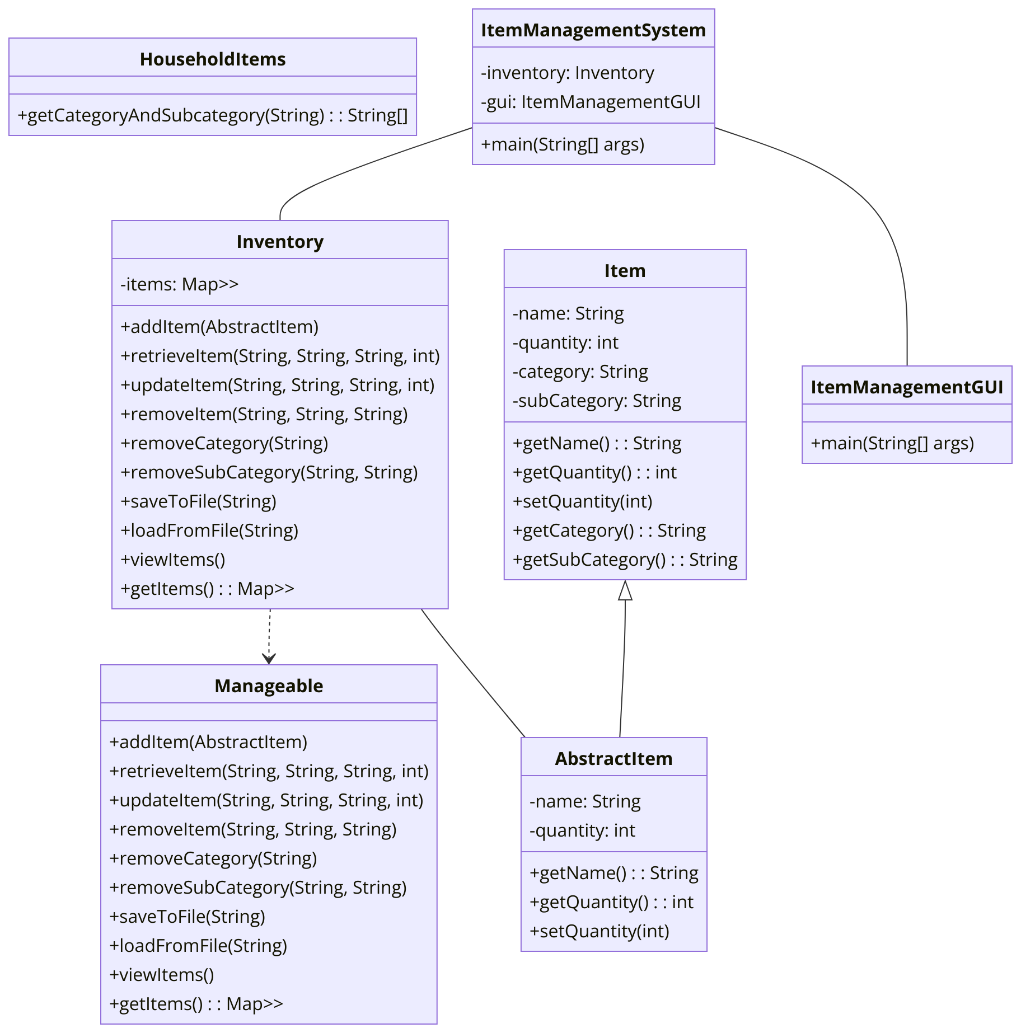
Other than adding items, the program also has other functionalities, such as:

1. Adding item(s),
2. Removing item,
3. Viewing items,
4. Update item quantities,
5. Removing storage categories
6. Removing storage sub-categories
7. Saving current inventory, and
8. Loading saved inventory.

All of these functionalities are run from the ItemManagementGUI, where I use the Swing and JFrame libraries for the GUI, it can also be run in the terminal with the “ItemManagementSystem” class.

**2.5 Class Diagram**

* ****All Classes
* The specific classes for the main mechanics

****

**Chapter 3 – Discussion of Implementation**

**3.1 How it works**

This project used a .txt file containing a list of household items and the categories and sub-categories each items belong. The file is read using a class that can read which are category names, sub-category names, and item names. It is then able to return categories and sub-categories of items that are inputted. As for taking inputs, another class it can take inputted items and determine their category and sub-category. Then the items are stored in an inventory file using a class that manages things such as adding, removing, saving, loading, and viewing the items. All of those functions are run using an item management class. And lastly, for user convenience,

**3.2 Algorithm**

The essential algorithms used in this Java Project are listed here

1. **Adding an Item**

Class: Inventory

Method: addItem()

Algorithm:

* Check if the category of the item exists in the items map. If not, create a new entry.
* Check if the subcategory of the item exists within the category map. If not, create a new entry.
* Iterate through the list of items in the subcategory to check if the item already exists.
* If the item exists, update its quantity.
* If the item does not exist, add it to the list.

1. **Retrieving an Item**

Class: Inventory

Method: retrieveItem()

Algorithm:

* Check if the category and subcategory of the item exist in the items map.
* Iterate through the list of items in the subcategory to find the item.
* If the item is found, decrease its quantity by the specified amount.
* If the new quantity is zero or less, remove the item from the list.

1. **Updating an Item**

Class: Inventory

Method: updateItem()

Algorithm:

* Check if the category and subcategory of the item exist in the items map.
* Iterate through the list of items in the subcategory to find the item.
* If the item is found, update its quantity to the new value.

1. **Removing an Item**

Class: Inventory

Method: removeItem()

Algorithm:

* Check if the category and subcategory of the item exist in the items map.
* Iterate through the list of items in the subcategory.
* If the item is found, remove it from the list.

1. **Removing a Category**

Class: Inventory Method: removeCategory()

Algorithm:

* Remove the entire category from the items map.

1. **Removing a Subcategory**

Class: Inventory

Method: removeSubCategory

Algorithm:

* Check if the category exists in the items map.
* Remove the subcategory from the category map.
* If the category map is empty after removing the subcategory, remove the entire category.

1. **Saving to File**

Class: Inventory

Method: saveToFile

Algorithm:

* Use ObjectOutputStream to write the Inventory object to a file.

1. **Loading from File**

Class: Inventory

Method: loadFromFile

Algorithm:

* Use ObjectInputStream to read the Inventory object from a file and replace the current items map with the loaded data.

1. **Loading Items from a File**

Class: HouseholdItems

Method: loadItemsFromFile

Algorithm:

* Read the file line by line.
* Parse the category and subcategory information.
* Populate the map with items categorized accordingly.

1. **Checking Item Existence and Getting Category**

Class: HouseholdItems

Method: checkItemExistsAndGetCategory

Algorithm:

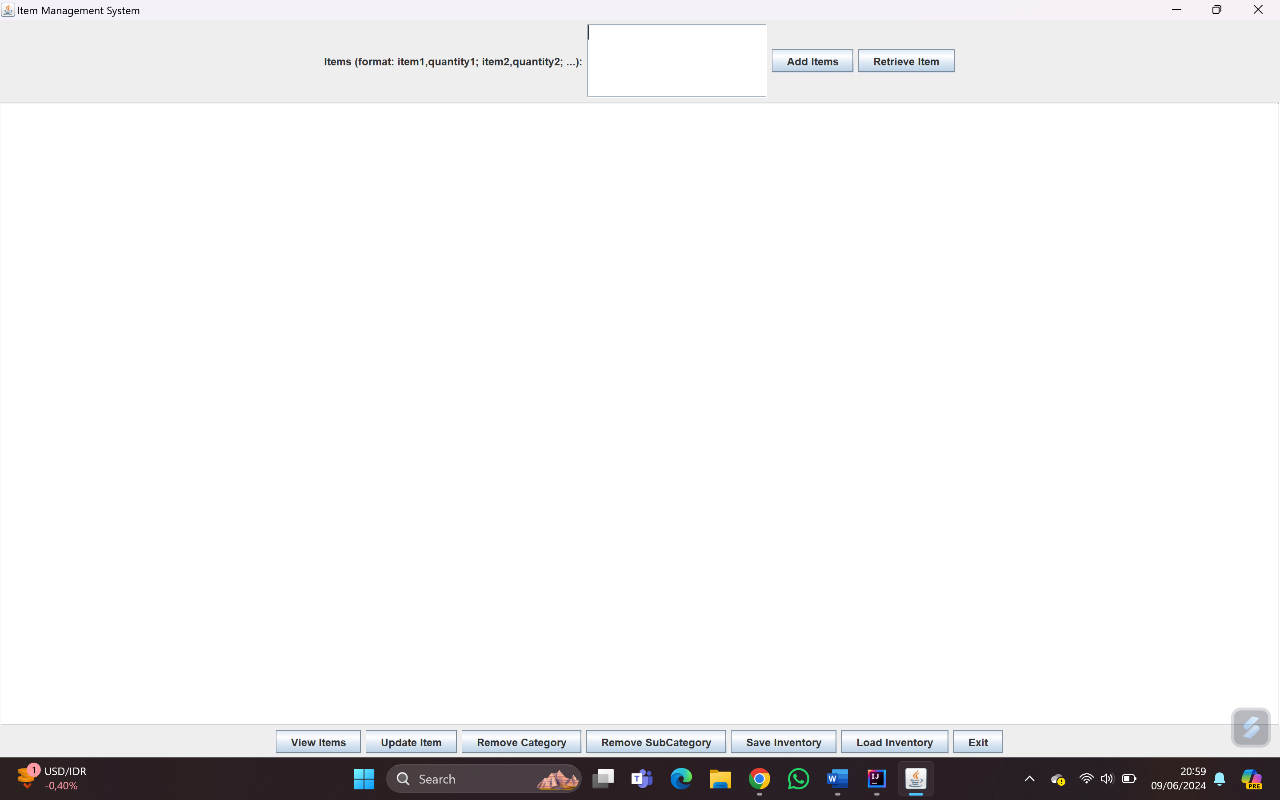
* Convert the item name to lowercase.
* Check if the item exists in the map.
* Return the category and subcategory if found, or default to "Miscellaneous" and "General".
  1. **Data Structure**
* Map: Used extensively to maintain the hierarchical structure of categories and subcategories, as well as to map items to their respective categories.
* List: Used to store items within each subcategory.
* Array: Used to split and parse strings.
* Set: Implicitly used in methods like removeIf for efficient removal operations.
* Scanner: Used for reading user input from the command line.
  1. **Solution Scheme**

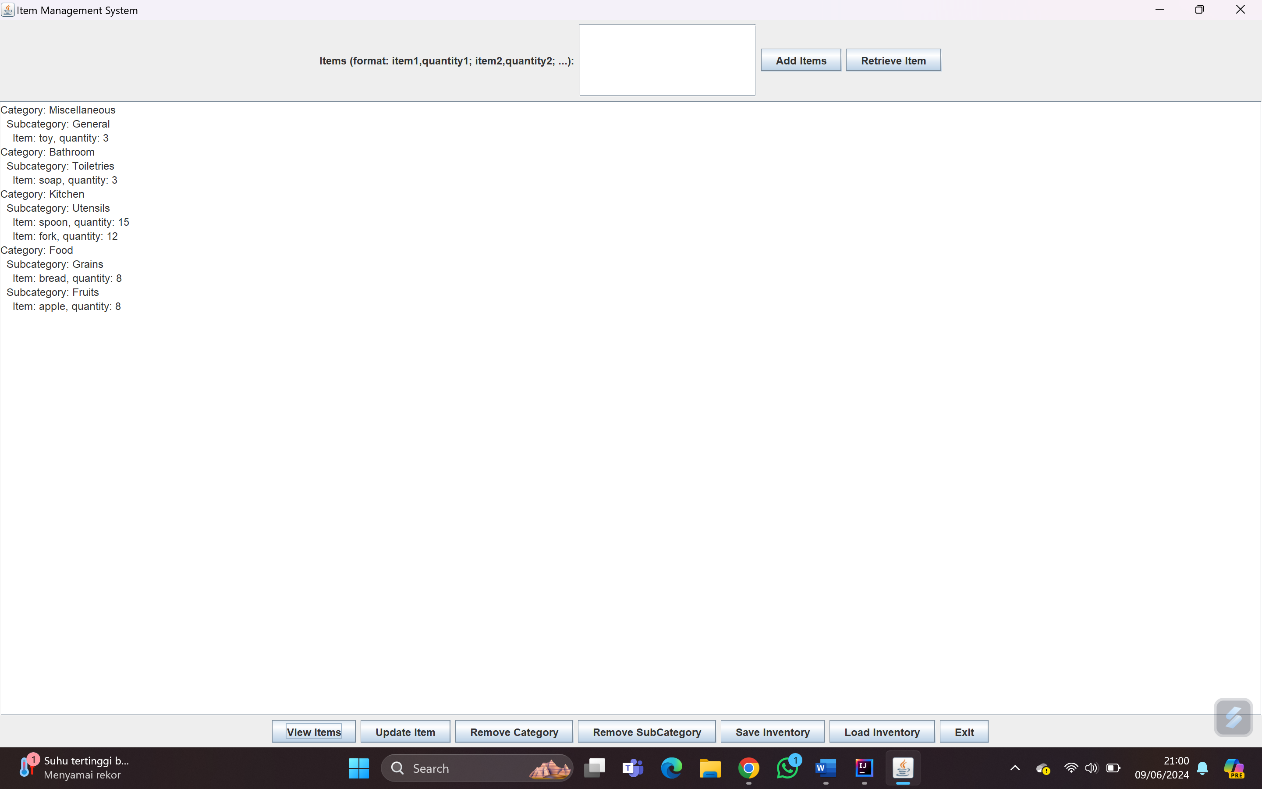
1. **Class Design**

* AbstractItem Class:
  + Represents a general item with attributes like name, quantity, category, and subcategory.
  + Provides getter and setter methods for these attributes.
  + Abstract class to allow extension for more specific item types.
* Item Class:
  + Extends AbstractItem.
  + Provides a concrete implementation for items with a specific string representation.
* Manageable Interface:
  + Defines methods for managing items, such as adding, retrieving, updating, removing items, saving to file, and loading from file.
  + Enforces a contract for any class that implements it to provide these functionalities.
* Inventory Class:
  + Implements Manageable interface.
  + Uses nested Map data structures to store categories, subcategories, and lists of items.
  + Implements methods to manage the inventory (add, retrieve, update, remove items, save/load inventory).
* HouseholdItems Class:
  + Utility class to categorize items by reading from a file.
  + Provides methods to get category and subcategory for a given item.
* ItemManagementSystem Class:
  + Command-line interface to interact with the inventory.
  + Uses Scanner to read user input for various operations.
  + Demonstrates adding multiple items, retrieving, viewing, updating, removing items/categories/subcategories, saving, and loading the inventory.
* ItemManagementGUI Class:
  + Graphical user interface to interact with the inventory.
  + Uses JFrame and other Swing components to create a window-based application.
  + Provides buttons and text areas for user interactions.

**Chapter 4 – Screenshots**

1. When user first run the program



1. After inputting the items

**Chapter 5 – Lesson Learned**

The things that I learned by making this project are varied. All of the thinks that I learned from the OOP class are implemented. The things are:

1. Coding with Java
2. Implementing object-oriented programming principles such as inheritance, polymorphism, interface, etc.
3. Implementing data structures such as List, Set, and Map.
4. Implementing built-in packages such as JFrame for GUI
5. Making UML diagram

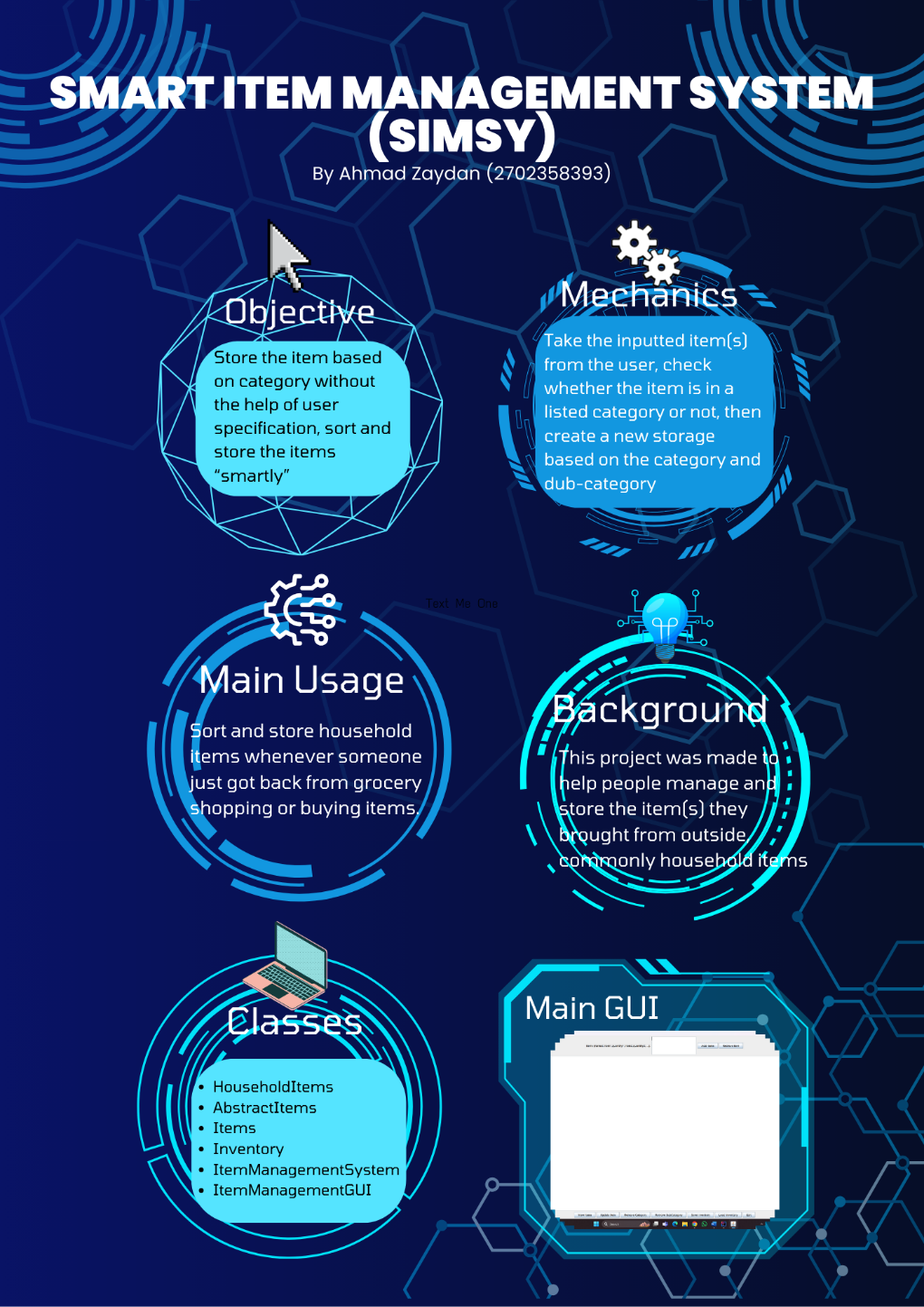
**Chapter 6 – Source Code and Video Demo**

* Github Link: <https://github.com/AhmadZaydan/Smart-Item-Management-System_OOP-Final-Project_Ahmad-Zaydan_2702358393>

**Chapter 7 – References**

* ChatGPT: <https://chatgpt.com/>
* Geeksforgeeks: <https://www.geeksforgeeks.org/>

**Chapter 8 – Poster**

****