MMORPG Database Concept and Implementation

Term Project for ACS575 – Dr. Yoo

Purdue University, Fort Wayne

Alekzander Green

Project Overview

I am an avid video game player and so I have always been interested in video games and the code behind them. This semester, I am also taking a Game Design course, which has made me even more interested. Among all game genres, MMORPGs are my favorite, but they are also the most difficult to develop. However, for this project, I would like to attempt to develop a basic database system that is able to store data for an MMORPG. This would store data like items, player data, and maybe more.

Problem Statement

MMORPGs need to store a lot of data. A large part of the genre is collecting and finding new items to use in the game. This project will attempt to build a database that is capable of storing and retrieving different kinds of items in an MMORPG and allowing the game to store a user's inventory in a reliable manner. This will require the planning and implementation of a database as well as a backend system to support API calls to the database.

Objectives

Market Research – I will look around at a few RPG inventory systems to see how they classify items and organize inventories. The games selected will be from an array of different RPGs from different subgenres. This will ensure that we have a good idea of how we might implement an inventory system.

Schema Design – I will design conceptual, logical, and physical schemas to plan and demonstrate how the database should function and how it will be implemented.

Database Implementation – I will implement a database capable of storing and retrieving data concerning items, inventories, and players. This database system will mostly focus on how items and inventories will be stored in a wide array of tables, but limited player information will also need to be stored to connect players to their inventories. This will likely be done using PostgreSQL as it is widely used and supports OR technology.

Backend Implementation – A backend API will need to be implemented to allow for streamlined and controlled database querying. Due to my prior experience with Django and the Django REST Framework, I will be implementing a REST API backend using Django. This API will allow me to retrieve information for any item in the database, add new items to the database, add items to player inventories, remove items from player inventories, retrieve player states, and store player states. Further actions may be needed, but these requirements will be necessary without a doubt.

Demo Application – After implementing the database and backend API, a demo application will be necessary to showcase the database working. Depending on time constraints, the demo application could be the Django REST Framework default view, Postman queries, a simple web app built with Svelte, a simple desktop app built with C# or Kotlin, or a simple game interface built in Unity. In any case, the demo should be able to show items in the inventory (even if it is just the inventory query).

Testing and Refining – Testing will be done continuously as prototyping is being carried out. After the prototypes are all put together, final testing and refining will take place. Exact tests will need to be determined later in the process.

Final Report and Presentation – Once the prototype is properly refined, a final report and presentation will be developed to highlight the technology used and the project results.

Scope

Given that the system will be able to store custom data, there will not be much data available for the database, but there should be data in each of the tables. This system will focus on an MMORPG's item and inventory system. Other systems within an MMORPG may be implemented but are considered stretch goals. The item system will consider different types of items and will implement extensive tables to accommodate them.

This project will:

- Implement a database system.
- Include tables for items of different types.
- Include tables containing player inventories and/or state.

This project will not:

- Consider other aspects of an MMORPG.
- Include a game that uses the database.
- Contain extensive data within tables (all tables will be populated).

Timeline

Item	Date
~~Proposal~~	~~February 9, 2024~~
~~Market Research~~	~~February, 2024~~
~~Planning and Diagramming~~	~~February, 2024~~
~~Mid-way Report~~	~~March 12, 2024~~
Database Implementation	March, 2024
API/Demo Implementation	April, 2024
Testing and Debugging	April, 2024
Project Presentation	May 1, 2024
Final Report	May 3, 2024

Market Research

World of Warcraft (WoW) [1]

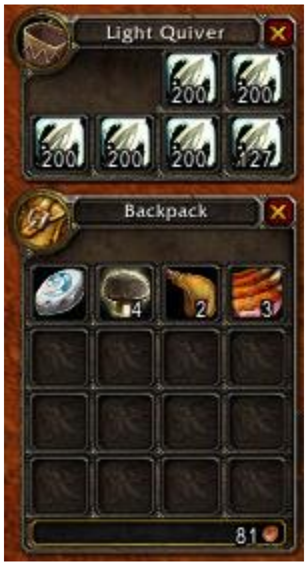

FIGURE 1 - EXAMPLE WOW INVENTORY

World of Warcraft (WoW) is an industry leading MMORPG where players take on the role of heroes in the mythical world of Azeroth. WoW allows players to collect all sorts of items each with their own purpose. We can broadly group items into five categories:

- Equipment (Weapons, armor, bags, and ammunition),
- Usable Items (Consumables and reusable items with effects),
- Quest Items (Items used for quests and no other purpose),
- Junk Items (Items with no specific use but that can be sold), and
- Ingredients (Items used in crafting)

However, despite the different types of items available, all items go into the same bags, and it is up to the player to organize their inventory. Player inventories in WoW consist of a Backpack that all players have equipped (and cannot remove) and up to four additional bags that must be equipped. Each of these bags has different sizes which determines how many items the player can place within the bag. For instance, the Backpack has 16 inventory slots. Additionally, some bags have restrictions on what kind of items can go into them e.g., only arrows can go into a quiver.

Multiple items will be grouped together into “stacks” of items. It should be noted that different items have different maximum stack sizes, with items like armor having a maximum stack size of 1 and items like potions having a maximum stack size of 20. Once a stack reaches its maximum stack size, excess items will begin a new stack in another inventory slot.

Additionally, players have access to a large bank where players can store items that they do not need immediate access to on their adventures. These banks are expandable by adding bags to them in the same way a player equips bags.

WoW Research Summary

- There are five basic categories of items,
- Items are collected into stacks of like items with a limit depending on the item,
- Stacks are contained within equipped bags and take up one inventory slot each,
- Players have between 1 and 5 bags equipped with different sizes,
- Some bags have restrictions on the types of items that can go into them,
- Player inventories are not otherwise divided, and
- Players have access to a bank where they can store more items.

Dungeons and Dragons Online (DDO) [2]

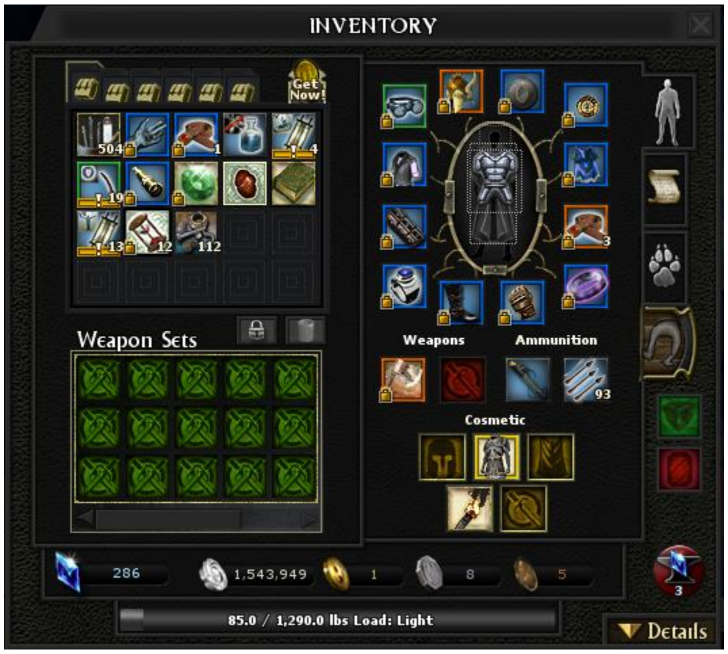

FIGURE 2 - EXAMPLE DDO INVENTORY

Dungeons and Dragons Online (DDO) is a relatively dated MMORPG set in the Dungeons and Dragons Eberron campaign setting. Players in DDO take up the mantle of questing adventures and inevitably collect all sorts of items in their journeys.

Items in DDO are categorized very similarly to those in WoW, with two additional categories:

- Storage items (used to store items within), and
- Special items (items that themselves have entire systems devoted to their use)

Unlike WoW, DDO does not have an intricate bag system. Players all have access to three 20-slot bags and can purchase up to three additional bags for a total of six bags. Like WoW, items are placed into stacks within the bags and have different stack sizes depending on the type of item. Finally, DDO also has a bank system that allows players to store additional items.

Due to some complexities unique to DDO, some items get very complex in their use, which is why the categorization of items can become difficult.

DDO Research Summary

- Many complex items make categorization difficult,
- Items are collected into stacks of like items with a limit depending on the item,
- Stacks are contained within bags and take up one inventory slot each,
- Players have between 3 and 6 bags that are all the same size,
- The inventory has no restrictions or divides, and
- Players have access to a bank where they can store more items.

Guild Wars 2 (GW2) [3]

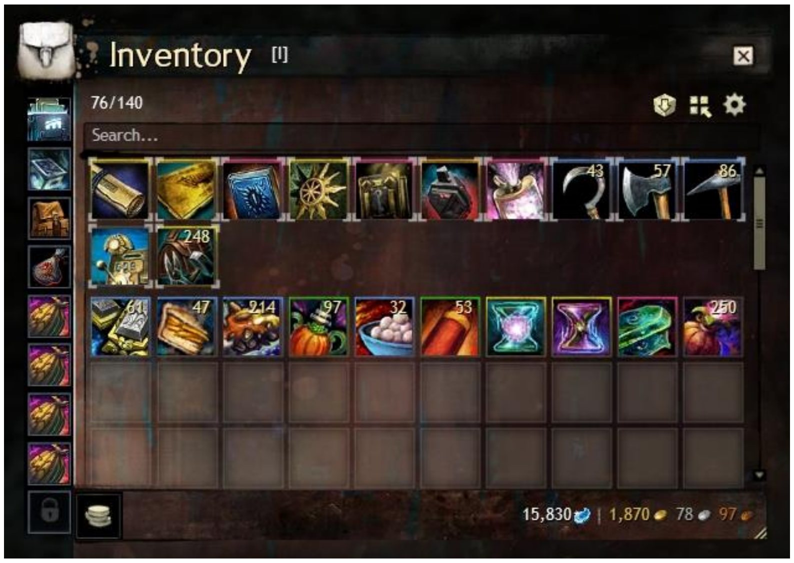

FIGURE 3 - EXAMPLE GW2 INVENTORY

Guild Wars 2 (GW2) is another fantasy MMORPG like both WoW and DDO. The Inventory system in GW2 is nearly identical to that of WoW. The one key difference is that items are either stackable or not stackable, and all stackable items stack up to 250 items per inventory space. Additionally, players can equip many more bags (up to 13) but all items exist within a single scrollable inventory view (unlike in WoW and DDO where you have to open specific bags to view items within them).

GW2 Research Summary

Similar in most respects to WoW, but with the following differences:

- Simplified item stacking system, and
- A central inventory for all inventory items where bags increase the inventory size.

Final Fantasy XIV (FFXIV) [4]

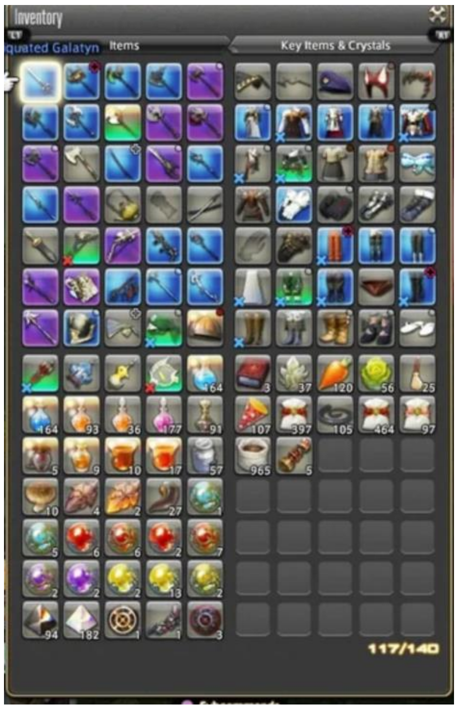

FIGURE 4 - EXAMPLE FFXIV INVENTORY

Final Fantasy XIV (FFXIV) is another fantasy RPG, and it allows us to begin seeing a trend. The inventory system is very similar to that of DDO, with stacking like GW2 (but up to 999 of an item).

FFXIV Research Summary

No additional information was gained besides reinforcement of inventory ideas from other games in different ways.

Overall Market Research Summary

With the addition of FFXIV, we can see that many MMORPG inventory systems follow a similar formula with the largest differences being how they handle bags (bags as equipment or bags as inventory tabs) and how they handle stacking (simple stacking with stackables/non-stackables or complex stacking with different stack sizes for different items).

I will be putting into practice what I believe is the simplest inventory system. The inventory system will be very similar to FFXIV's inventory system, but with item categories more in line with WoW. UsaThis means that the inventory system that will be implemented has the following features:

- Five item categories with various subcategories as necessary,
- Items that are classified as stackable or non-stackable,
- Stackable items have the same maximum stack size regardless of the item,
- Item stacks take up exactly one inventory slot,
- Players have a set amount of bags all with the same size, and
- Players will have access to a personal bank to store additional items.

Conceptual Schema

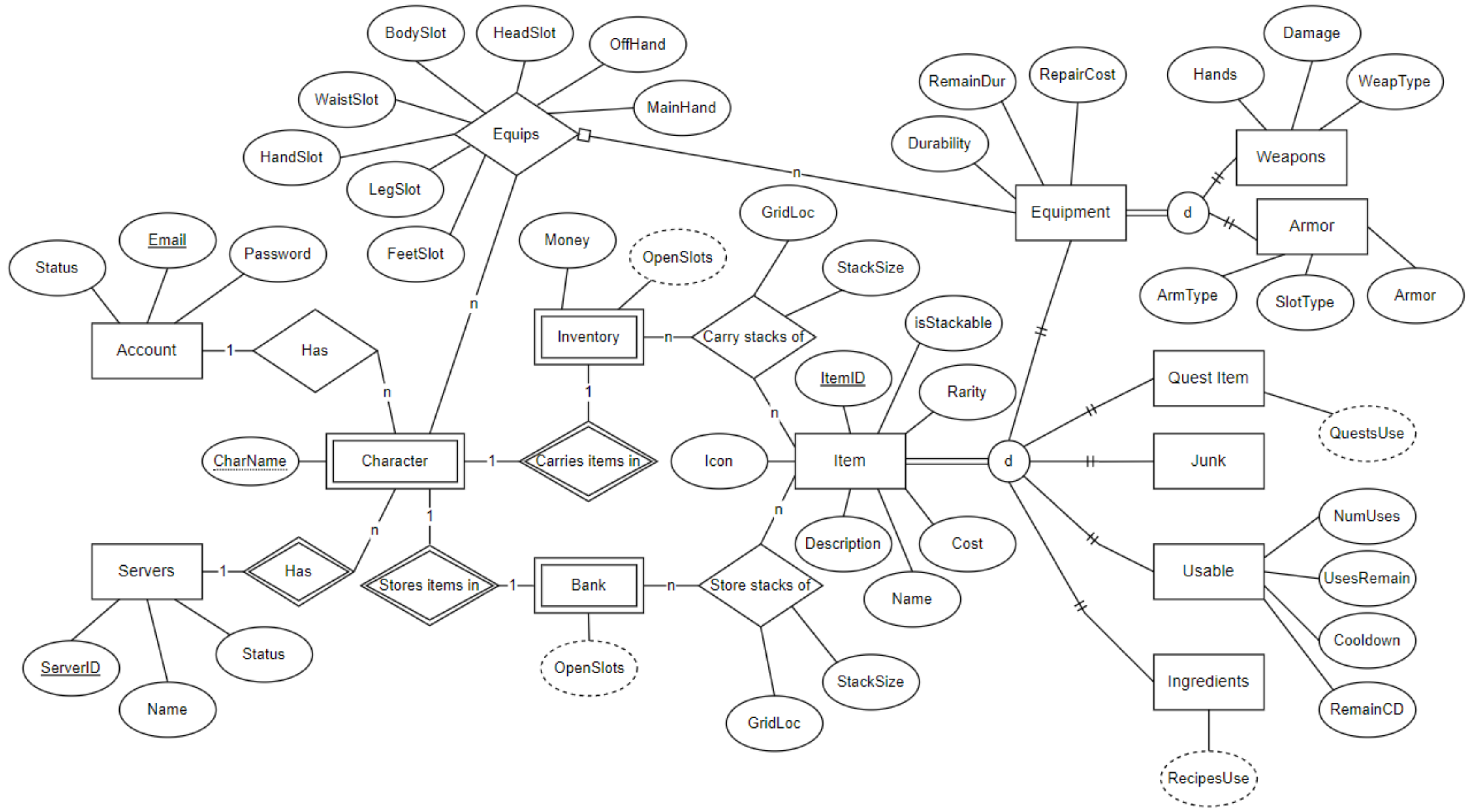

FIGURE 5 - CONCEPTUAL SCHEMA FOR THE INVENTORY SYSTEM

Logical Schema

The following is the logical schema built from the conceptual schema. This schema is built from the perspective of a relational database, but I am interested in using an Object-Relational schema instead. This schema is in a minimum of 3NF. Note that some of the item categories rely on external systems that will not be implemented in the prototype, leading them to feel empty.

Account (Email, Password, AccountStatus {ONLINE, OFFLINE, BANNED, INACTIVE})

Server (ServerID, Name, ServerStatus {UP, DOWN})

Character (*ServerID*, CharName, *AccountEmail*)

Inventory (*ServerID, CharName*, Money)

InventoryItem (*ServerID, CharName*, GridLoc, StackSize, *ItemID*)

Bank (*ServerID, CharName*)

BankItem (*ServerID, CharName*, GridLoc, StackSize, *ItemID*)

EquippedItems (*ServerID, CharName*, *MainItemID, OffItemID, HeadItemID, BodyItemID, WaistItemID, HandItemID, LegItemID, FeetItemID*)

Item (ItemID, Name, Rarity, Description, Cost, isStackable, Icon)

Equipment (*ItemID*, Durability, RemainDur, RepairCost)

Weapon (*ItemID*, Hands, WeaponType, DamageValue)

Armor (*ItemID*, SlotType, ArmorType, ArmorValue)

QuestItem (*ItemID*)

JunkItem (*ItemID*)

UsableItem (*ItemID*, NumUses, RemainUses, Cooldown, RemainCD)

Ingredients (*ItemID*)

Prototype System Functionality

The prototype system will allow developers to develop expansive item lists that are each categorized based on the type of item that exists. Developers can then create accounts, create servers, create characters for a specific account and server, update character inventories, and update character banks.

From the client perspective, the developer should be able to easily move items between character equipment, inventories, and banks. This serves as a backbone for a traditional MMORPG economy. Further details on exactly which CRUD operations should be able to be done on each table in the database can be found in the Data and CRUD Operations section further in this document.

Additionally, a simplified Account-Server-Character system that is common to most MMORPG systems. This will allow developers to create accounts and then select servers on which to create characters. Due to the focus of this prototype being on the item system, concepts like account authentication, server transfers, and character name changes will not be implemented (though they are common in MMORPGs). These tables instead serve to simulate a more dynamic database and MMORPG ecosystem.

If possible, the database will take advantage of OR technology. This will reduce the clutter of some tables and create opportunities for a more dynamic database. For instance, an “Inventory” type could be created to host items within an array such that moving items around in an inventory is more intuitive from the standpoint of the database. This would eliminate the need for additional “Inventory Item” tables that would otherwise become extremely large as the number of players in a game increase and those players collect more and more items.

System Architecture Overview

The project will implement a Client-Server architecture. The backend will consist of a database and web server to serve data from the database. The frontend will consist of the demo application. It should be noted that a traditional MMORPG system would have a more complex game server rather than a web server. The game server would be necessary to carry out game logic, ensure synchronization between clients, and ensure integrity between clients in addition to serving data from the database. However, this is a prototype system, so a development server will take the place of a game server in our example. This development server will allow us to develop storage and API logic without the need for a game server. Additionally, when a game server is developed, it can easily either make calls through the development server (treating the development server as an intermediate data server) or connect directly to the database using similar logic.

Database and DBMS

This project will use the PostgreSQL DBMS. PostgreSQL is an Object-Relational DBMS that uses SQL for querying. This database will store data in tables where the DBMS can manage the storage and retrieval of data.

Backend Web Server

This project will use the Django REST Framework to develop a REST API through which the DBMS will be accessed. Django REST Framework is an API framework that is used alongside the Django web framework for the express use of creating REST APIs. Django is a Python web framework which is simple to use and set up, but still offers more complex operations. The use of a REST API will allow us to create an interface that will make it simple to work with the database.

Frontend Demo Application

To demonstrate the database in action, a demo application will be developed. There are several different ways in which this could be done, which will be chosen depending on the resources available. The simplest application would be a basic CLI built using Python and the requests library; with more time a web application could be built using JavaScript and the SvelteKit framework or a desktop application could be developed using the .NET framework and written in C#; and with more time, a basic game interface could be developed using the Unity game engine wherein the code will be developed using C#. I have experience with each of these approaches in different contexts, but the overall outcome will depend on timing.

Data and CRUD Operations

The prototype system will need custom data to be created to develop each table in the database. Data will include basic information for the following:

- Player accounts,
- Game Servers
- Player characters,
- Character inventories & Inventory Items,
- Character banks & Bank Items, and
- Item data.

Each will require operations to be carried out on the data. These operations are detailed in the following sections.

Player Account Operations

Players and game systems should be able to do the following operations with the account information:

Create – Create a new account

Read – Get account status, get login info (should be done with an authentication service)

Update – Update account status

Delete – N/A (Accounts should not be easily deleted)

Game Server Operations

Players and game systems should be able to do the following operations with the server information:

Create – N/A (Servers should not be created frequently)

Read – Get server status

Update – Change server status

Delete – N/A (Servers should not be easily created)

Player Character Operations

Players and game systems should be able to do the following operations with the character information:

Create – Create new characters for a given account on a given server

Read – Get name, get equipped items list, get inventory, get bank

Update – N/A (Current character information does not require updating; however, expansion of this database would allow other character state information to be stored here long-term)

Delete – Delete characters (CASCADE delete when server or account is deleted)

Character Inventory Operations

Players and game systems should be able to do the following operations with the inventory information:

Create – Create a new inventory for a new character

Read – Get money

Update – Add money, remove money

Delete – N/A (CASCADE delete when character is deleted)

Inventory Item Operations

Players and game systems should be able to do the following operations with the inventory item information:

Create – Create a new inventory item for a given inventory slot (item added to inventory)

Read – Get items list for a given character

Update – Update inventory slots when items are moved

Delete – Remove items from an inventory slot (CASCADE delete when items are deleted)

Character Bank Operations

Players and game systems should be able to do the following operations with the bank information:

Create – Create a new bank for a new character

Read – N/A

Update – N/A

Delete – N/A (CASCADE delete when character is deleted)

Bank Item Operations

Players and game systems should be able to do the following operations with the bank item information:

Create – Create a new bank item for a given bank slot (item added to bank)

Read – Get items list for a given character

Update – Update bank slots when items are moved

Delete – Remove items from a bank slot (CASCADE delete when items are deleted)

Equipped Items Operations

Players and game systems should be able to do the following operations with the equipped items information:

Create – Create an equipped items table for a new character

Read – Get equipped items for each equipment slot

Update – Equip items to a slot, dequip items from a slot

Delete – N/A (CASCADE delete when character is deleted)

Item (Equipment, Quest, Junk, Usable, Ingredients) Operations

Developers and game systems should be able to do the following operations with the item information (note that these operations are similar for each item table, but there exists 7 item tables):

Create – Add new items to the database

Read – Retrieve item information given an item ID

Update – Tune item information and statistics, update dynamic stats (equipment & usable only)

Delete – Remove items from a database

References

[1] Fanbyte, "Inventory (WoW)," 17 June 2008. [Online]. Available: https://wow.allakhazam.com/wiki/Inventory_(WoW). [Accessed March 2024].

[2] DDOWIKI, "Inventory System," 16 October 2023. [Online]. Available: https://ddowiki.com/page/Inventory_system. [Accessed March 2024].

[3] Guild Wars 2 Official Wiki, "Inventory," 4 March 2024. [Online]. Available: https://wiki.guildwars2.com/wiki/Inventory. [Accessed March 2024].

[4] Square Enix, "Gear and Inventory," [Online]. Available: https://na.finalfantasyxiv.com/uiguide/equipment/#category-equipment. [Accessed March 2024].