Dungeons and Dragons 5e Campaign Assistant for UTM CSCI 352 Spring 2019

Bryan Downing, Cole Weston

Abstract

The formidable complexity of the popular table-top game, Dungeons and Dragons, has created a demand for digital tools that help simplify the game's mechanics. One of the most fluid, and often hardest to manage, aspects of the game is the inventory and character sheet management system. Our program will provide an easy to use character manager that players of Dungeons and Dragons and Dungeon Masters can use to better experience their game.

1. Introduction

The popular table-top game, Dungeons and Dragons, is very complex and requires a lot of time to play. The game's inventory and skill systems are some of the most complex systems in the game, and they requires a great amount of paper resources to keep up with. Our proposed program will be able to efficiently manage an inventory system and will facilitate the management of each player's own character. Users of our program will benefit from its ability to automatically keep up with a player's inventory and character sheet without forcing them to look through the player guide or exhaust their paper supplies.

1.1. Background

A table-top adventure game, such as Dungeons and Dragons, is played using a large quantity of paper. Every player has a *character sheet* containing information about a player's character, an *inventory sheet* containing information about a character's inventory and equipment, and a *spell sheet* containing the list of spells and abilities each character has access to. A player must also use paper to make story notes and record enemy positions during *encounters* (fights). There are many different classes and races that players must choose from when making their character. Because of this, the skills and inventories of every player is uniquely constructed. The decision to create a new character management application was born from our shared annoyance with the time-consuming tediousness of sorting and sifting through stacks of paper every time we play Dungeons and Dragons. We found ourselves spending more time managing our characters than we did playing the game. Our program aims to make Dungeons and Dragons less paper dependent.

1.2. Challenges

- DnD 5e, while less complex than previous versions (see: DnD 3.5e), still has a massive degree of complexity in terms of interworking parts. Stats are affected by items, items affect stats, and there is simply a massive variety of permutations.
- Choosing a design pattern to work with is going to be crucial. It needs to be able to create a variety of items and item types.
- It must be able to implement search and sorting functions to find certain items quickly and with as few clicks as possible.
- Designing a save system for the game will require learning how to make a save system or, at the very least, require the program to read and write user information to/from a file using a consistent format. Either way, it will be challenging.
- Making an user friendly UI will be difficult due to our team's lack of experience with making UI's. Further research into graphic design might be for the best.

2. Scope (TODO)

This section is a bit tricksy. You are going to do your best to set up ground rules: How will you know when your project is done?

If you were doing this under contract for a company, this would be your checklist to make sure you get paid. We will be going into this in more detail over time, but you should start planning your major goals of the project as soon as possible.

For every sub(sub)section below, make sure to mark which items are basic goals (project won't be done without it) and which ones are stretch goals (it would be really cool to do...). We will be meeting one-on-one to help identify which goals go where.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add/Remove item to/from inventory	Player	Med	1
2	Add/Remove gold to/from inventory	Player	Med	1
3	Sell item from inventory	Shopper	Med	1
4	Create new inventory	Player	Med	1
5	Create new store and randomize inventory	DM	High	2
6	Search Store	Player	Med	1
7	Sort Store	Player	Med	1
8	Save Store	Player	High	1
9	Select Character Kit	Player	Med	2

TABLE 1. EARLY BRAINSTORM CASE TABLE

2.1. Requirements

The functional and non-functional requirements for this program are listed below. Alongside those, we have included a list of stretch goals that will potentially become basic (immediate) goals depending on how the semester progresses.

2.1.1. Functional.

- User must be able to input and keep track of character skills, feats, spells, and all other vital character information.
- User needs to have atleast one inventory – this inventory must be completely manageable with limits placed on it by class restrictions and potentially weight
- User needs to be able to save their inventory – must be able to be loaded from and saved to file for player convenience
- User must be able to access item descriptions, gold value, and weight – this will help users choose between items more easily
- User must be able to create items following certain templates – customizable gear is a staple of DnD.
- User must be able to sort items by type – weapons, potions, rations, etc.

2.1.2. Non-Functional ????????????.

- The program must have a usable, non-abrasive, minimal GUI.
- you'll typically have fewer non-functional than functional requirements

2.1.3. Stretch Goals.

- Separate permissions added for *dungeon master*. – It would be cool to see special DM only privileges for non-player character (NPC) inventory management.
- Inventory randomization for NPC stores, or generation of unique procedural items.
- A "campaign manager," or a system by which an entire party's sheets can be managed by the DM for record-keeping purposes.

2.2. Use Cases (STOPPING HERE PER ASSIGNMENT INSTRUCTION)

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table 1.

Use Case Number: 1
Use Case Name: Add/Remove item to/from inventory
Description: A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.

3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:

Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2

Use Case Name: Add/Remove gold to/from inventory

Description: A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.