Dungeons & 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 & 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 & Dragons® and Dungeon Masters can use to better experience their game.

1. Introduction

The popular table-top game, Dungeons & 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 players' inventories and character sheets 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 & 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 & Dragons®. We found ourselves spending more time managing our characters than we did playing the game. Our program aims to make Dungeons & Dragons® less paper dependent.

1.2. Challenges

- D&D® 5e, while less complex than previous versions (see: D&D® 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

The goal of the D&D® 5e Campaign Assistant is to provide a user-friendly interface through which players of D&D® 5e can manage and keep track of their player characters. By project completion the user(s) should be able to, at the very minimum, seamlessly manage a character sheet which includes a spell list, inventory, and attributes.

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.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Create a new character	User	Med	1
2	Edit existing character	User	Med	1
3	Save the current character	User	Low	2

TABLE 1. USE CASE TABLE

2.1.1. Functional.

- User must be able to input and keep track of all relevant information present on standard D&D® 5e character sheets.
- Program must properly manage player characters' inventories e.g. item weight, weapon statistics, and item value.
- Program must be able to write character sheet info to disk and subsequently load from such files.
- User must be able to create items following certain templates – customizable gear is a staple of D&D®.

2.1.2. Non-Functional.

- Program should reliably save user data in the event of an unexpected failure.
- Program should be able to concurrently manage at least 10 character sheets (for assistance in managing an entire party of player characters).

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

Use Case Number: 1
Use Case Name: Create a new character
Description: User has opened the program and needs to create a new character sheet. They will click "File" in the toolbar followed by "New" from the drop-down list. This will create a blank character sheet.

1) User opens a new instance of the program.
2) User left-clicks on "File" button in toolbar.
3) User left-clicks on "New" option from the drop-down list.
4) A new tab is created containing a blank character sheet.

Termination Outcome: The user now has a blank character sheet instance opened in a new tab.

Use Case Number: 2
Use Case Name: Edit existing character
Description: User has opened the program and needs to edit an existing character sheet. They will click "File" in the toolbar followed by "Open" from the drop-down list. This should open an existing character sheet from disk in a new tab.

1) User opens a new instance of the program.
2) User left-clicks on "File" button in toolbar.
3) User left-clicks on "Open" option from the drop-down list.
4) A new tab is created containing the character sheet.

Termination Outcome: The user has now opened an existing character sheet instance in a new tab and is ready to begin editing.

Use Case Number: 3
Use Case Name: Save the current character
Description: User has a character sheet open in a tab. They will click "File" in the toolbar followed by "Save" from the drop-down list. This will save the current character sheet to disk, prompting if the file has not yet been saved.

1) User has a previously saved character sheet open.
2) User left-clicks on "File" button in toolbar.
3) User left-clicks on "Save" option from the drop-down list.

Termination Outcome: The file on disk has been overwritten with new character sheet data.

Alternative: Current character sheet has not been saved.

1) User has an unsaved character sheet open.
2) User left-clicks on “File” button in toolbar.
3) User left-clicks on “Save” option from the drop-down list.
4) User is prompted to choose a save directory and file name.

Termination Outcome: The file has been saved to disk.

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.