Dungeons & Dragons® 5e Campaign Assistant

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 are 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.

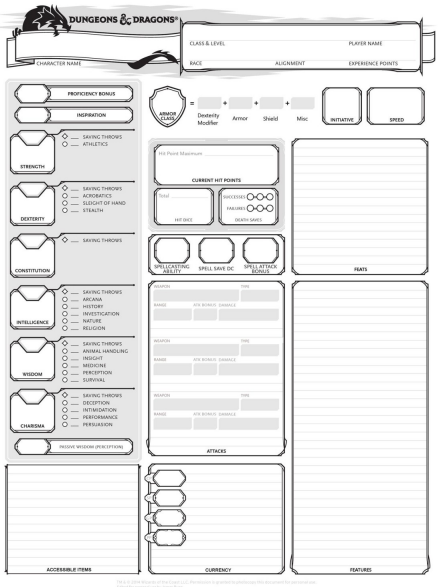

Figure 1. Blank Character Sheet

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. The project needs to be able to create a variety of items and item types.
- The project 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 program 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. Working closely with those familiar with D&D® will be useful in designing an intuitive UI. 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.

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 4.
- 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 them.
- User must be able to create customizable gear.

2.1.2. Non-Functional.

- Program should reliably load all open files from previous save state.
- 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.

- A "campaign manager," or a system by which an entire party's sheets can be managed by the DM for record-keeping purposes.
- 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.

2.2. Use Cases

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

TABLE 1. USE CASE TABLE

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.

1) User opens a new instance of the program.
2) User left-clicks on “Create New Character” option from the startup window.
3) A new tab is created containing a blank character sheet, and the main window is opened.

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

Use Case Number: 2

Use Case Name: Load existing character

Description: User has opened the program and needs to edit an existing character sheet. They will click the “Load Existing Character” option upon opening the program. 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 “Load Existing Character” option from the startup window.
3) A new tab is created containing the character sheet, and the main window is opened.

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 the save icon in the main window. 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 the save icon in the main window.

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 the save icon in the main window.
3) User is prompted to choose a save directory and file name.

Termination Outcome: The file has been saved to disk.

2.3. Interface Mockups

Figure 2. Mockup 1

Figure 3. Mockup 2

3. Project Timeline

Week	Date	Checkpoint	Plan to Meet Checkpoint
3	2/4/19	Project proposal drafted	Project will be named. We will draft an abstract, an introduction, and a scope.
6	2/21/19	Project proposal update	Proposal will have atleast 3 use cases, functional and non functional requirements, and interface mockups.
7	2/26/19	Project proposal presentation	We will have prepared project interface mockups and graphics necessary to pitch our project proposal
10	3/10/19	Working demo must compile	Project mockup must be finalized for program to be designed around.
11	3/21/19	Project Update and Demo Day	Project Timeline, Structure, and UML outline must be prepared, and program must be presented.
13	4/1/19	Inventory functionality completion	Inventory should be able to create weapons, armor, and equipment.
14	4/7/19	Save and Load character functions completion	Project should be able to load and save character information. This should be a simple read/write from/to file.
15	4/16/19	Penultimate Writeup and Demo Day	Project proposal should be pretty much done.
16	4/25/19	Project completion	Project must be completed by this date. That means that program must be be able to, at minimum, create and load character sheets successfully. Program will be able to edit spells and attributes.

TABLE 2. TIMELINE

As a team, our time management skills were well balanced. We set clear goals for ourselves and have met them within the time frame we estimated. There was some panic over time when a teammate didn't realize what 'penultimate' meant, but, other than that hiccup, time management went smoothly.

4. Project Structure

The structure of the project centers around a list of Character objects (one of which is active at any given time), to be passed between the various Windows of the program. Each individual Character object contains all information which would be maintained within a standard D&D 5e character sheet - most importantly a list of ability scores, skill values, inventory items, and spells. Logic for the calculation of many of these items is contained within the Character class, and the main purpose of each Window is to display these attributes in an easily digestable form.
For example, ability bonuses and skill values will not be directly accessible by the end user, but rather are internally maintained and calculated by the logic within the Character class (using the pertinent logic from the Player's Handbook which would generally be calculated by hand.)
Several fields (e.g. Feats, Abilities, etc.) will simply be text fields with arbitrary contents, to be freely edited by the end user. This is to allow for the greatest degree of flexibility in using the character manager – facets of D&D such as these are often not cut-and-dry enough to easily facilitate a better method of organization.

4.1. UML Outline

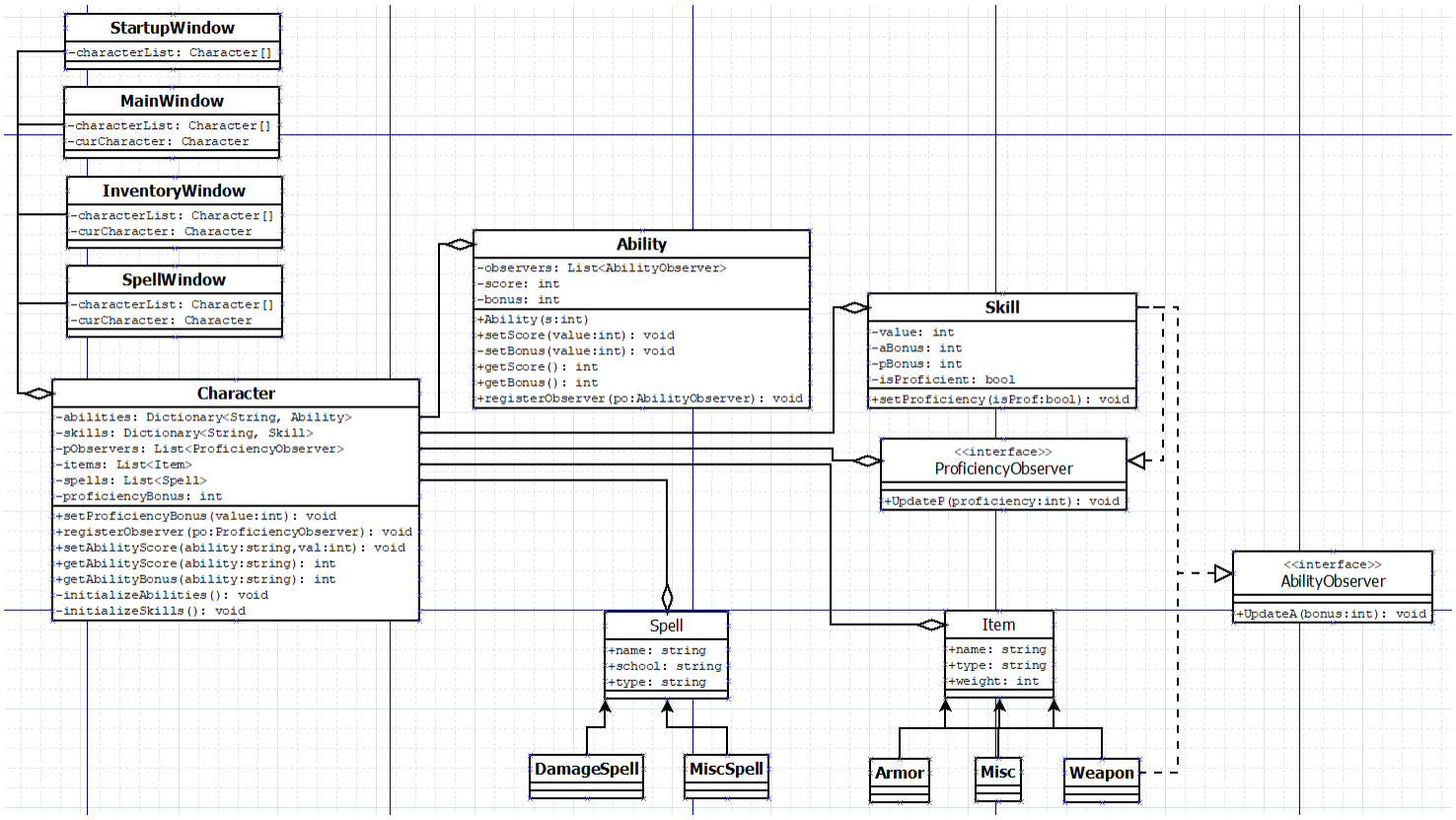

Figure 4. Project UML

4.2. Design Patterns Used

Observer – Both the ProficiencyObserver and AbilityObserver classes are implementations of the Observer pattern, and are used to keep ability bonuses and skill values in sync.

5. Results

As a team, we have accomplished almost all of what we set out to do. We implemented all of our desired use cases. A more significant achievement was meeting all of our functional requirements. For all intents and purposes, the project allows the user to input and keep track of all relevant information present on a standard D&D 5e character sheet. It also allows the user to create, save, and load different spells and equipment. The only non-functional requirement we did not meet was the ability to manage multiple character sheets simultaneously. We entertained the idea for a large portion of the development process, but it was dropped after feedback from testers deemed the feature unnecessary. As a bonus, the team added player avatars, armor icons, and weapon icons.

5.1. Future Work

If work on this project were to continue past project completion, further updates might include the implementation of our current stretch goals. Something that occurred to us fairly late in development was the idea of using a preexisting API to load possible spells to select from, rather than having the user make their own.