Make It Easy
for UTM CSCI 352

Chase Perritt and Danny Lillard

Abstract

"Make it Easy" is a modular asset management system for tabletop roleplaying games. The targets for "Make it Easy" are those who wish to "make it easy" when working with TTRPG assets: dungeon masters and players alike. The project has only just begun.

1. Introduction

"Make it Easy" is a modular asset management system for tabletop roleplaying games built to assist the user in TTRPG character building and upkeep via gathering information from online databases and organizing the information in an accessible and dynamic manner. We expect the target audience to benefit from "Make it Easy" due to its ability to better keep track of TTRPG elements in the heat of a game: initiative scores, health points, spell slots, etc.

1.1. Background

Tabletop RPGs, or role-playing games, are an increasingly common hobby in modern society that involve creating and acting out characters in fictional scenarios. One game is typically called a "session," and multiple sessions are called a "campaign." Two such examples of tabletop role-playing games are the widely popular Dungeons and Dragons (5th Edition) and Pathfinder (1st Edition). D&D [5e] and Pathfinder [1e] sessions tend to take place in high fantasy, magical settings; however, their rulesets are incredibly different. D&D [5e] is a simpler model and is thus easier to introduce to new players. Pathfinder [1], however, is better suited to players with some amount of tabletop experience, as there are an abundance of rules, some of which build off of an older model of D&D, that being D&D 3.5 edition. Some terms that would be useful to know are as follows:

Sheet: A page in a character's portfolio, typically used to track their stats, equipment, and/or spells.

1.2. Challenges

A particular challenge in mapping out both the rulesets of D&D 5e and Pathfinder 1e is that, though we have access to source reference documents and particular websites that compile all of the information from several source books and pdfs, there is simply too much information out there (in Pathfinder specifically). In Pathfinder alone we have to compile all of the feats, classes, races, class archetypes, traits, alternate racial abilities, spells, and common equipment. This is a gargantuan undertaking, and a simple solution would simply be to add less content than we hope: to cut out the equipment databases and alternate features in favor of the minimum required amount.

2. Scope

Completion of the project entails the existence of a D&D 5e and Pathfinder 1e character maintenance tool that rivals traditional pen and paper sheets in terms of quality of life assurance and ease of access to information. While this software may not render the pen and paper style obsolete, it should prove valuable as a beginner's introductory tool to tabletop roleplaying or as a reference for rules and monster stats for Dungeon Masters. As for stretch goals we would like to give the user to save their sheet at a .pdf or .txt, and have an automatic initiative roller.

2.1. Requirements

Most requirements for "Make It Easy" are derived directly from the standard pen and paper style of managing a character. Each requirement listed below deserves to be here because it makes the player's life easier, whether that requirement does something better than pen or paper, or adds new functionality.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1.1. Functional.

- User is able to create new characters - From a drop down menu similar to a Microsoft program, the user should be able to create a new character sheet
- User is able to load old sheets - using the same drop down menu player should be able to load a sheet file
- User is able to delete a sheet - using the same menu user can delete a sheet through the GUI
- User can change program theme - using the drop down menu user can change the theme of the program, such as light mode, dark mode, etc
- User needs to have a save file for each character - analogous to a sheet of paper this will keep track of all his or her last saved values for said character
- User needs to be able to change the values of a character - through text boxes, he or she will be able to alter the character's values
- The software must not allow incorrect input - users should not be able to set their health too high, raise or lower skill levels to strange numbers, etc.
- The software must have all the important information found in the user's handbook - each part of the character should be laid out and described for the player, if so desired
- Different tabs for different information - a character is usually complex, having backstory, ability stores/feats, class and level, spells, proficiencies and an inventory. This data should be broken down logically into different tabs
- User has the ability to take notes - note taking is an important part of TTRPGs there should be a seperate tab for these notes in the program

2.1.2. Non-Functional.

- Interchangeability - The program is easy for the developers to adapt to new TTRPG frameworks
- Speed of loading and saving - previous saves and the act of saving must load quickly for the user, in less than two seconds

2.2. Use Cases

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 item to cart
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 1). 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: Checkout
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.

Figure 1. First picture, this is a kitten, not a use case diagram

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

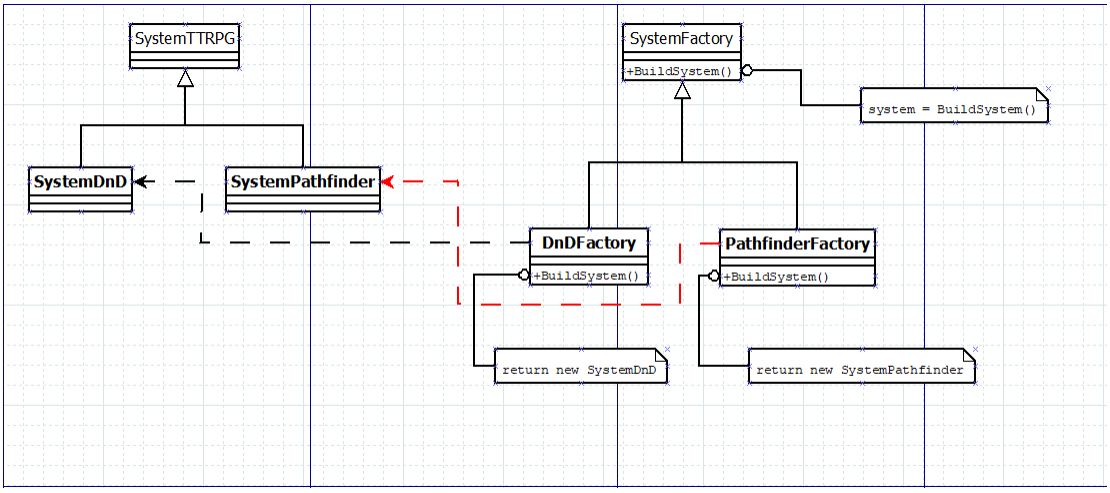
1) Requirements:
   We plan to have all of our requirements established by the submission and review of the first project proposal draft on the 10th of February, 2020.
2) Design:
   We plan for our major design decisions to be fleshed out by the end of February or early March.
3) Implementation:
   We hope to have finished implementing the primary, promised functionality of "Make it Easy" by the end of March or early April.

4) Verification:
   We hope to have fixed all bugs and leaks by the end of April for the timely submission of the project.
5) Maintenance:
   After the end of April, any additional functionality and systems could be added at any point in the future.

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



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.

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens