Midterm Project Documentation:

Character Manager

March 14, 2025

Student - Connor Clawson Professor – Edward Weber

# Premise

The goal of this project is to create and build up a character sheet manager, using the rules of the pen and paper role playing game Pathfinder 1st edition, hereafter referred to as PF1e. By the end of the semester, the web application should feature fully functional character sheets for player characters and non-player characters alike.

# Database

Using MySQL, the database is to be tailored to ensure all values in the many fields of a character sheet are valid, saved, and made available to users.

## Tables

As the project evolves, more tables will be added to the database to achieve the desired functionality. The following tables have been set up with values and relationships:

* CHARACTERS
  + This is the main table that users will interact with.
  + V.1 contains the following:
    - Foreign key constraints to the CLASSES and RACES tables.
    - Base scores for the six core stats: Strength, Dexterity, Constitution, Intelligence, Wisdom, and Charisma.
  + V.2 contains the following:
    - fields for all known skills in PF1e, including preset indicators for which ability modifier is used, if a skill can be used untrained, and a placeholder column for denoting if a skill is a class skill for that character. User definable fields are as follows:
      * Skill ranks
      * Racial bonuses
      * Feat bonuses
      * Miscellaneous bonuses.
  + V.3 Contains the following:
    - A section for adding feats to a character with the following fields:
      * Feat name
      * Feat description
* CLASSES
  + This holds a fixed list of the 11 core classes in the Pathfinder Core Rulebook (CRB.)
  + V.1 contains the following:
    - Foreign key constraints to the CLASSES\_SKILLS and DICE tables.
* RACES
  + This contains the 7 core races playable by players in the CRB.
  + Includes fields for ability score modifiers, used at character creation
* DICE
  + This table lists the size or face count of the dice used in the game. This table is expected to be utilized by other tables in the future.
* SKILLS
  + Contains the complete list of skills useable by characters.
  + V.1 contains the following:
    - Foreign key constraint to the CLASSES\_SKILLS table
    - The names and Boolean flags for trained/untrained skills and armor check penalties.
* CLASSES\_SKILLS
  + This table handles the many-to-many relation between classes and the skills they have access to.
  + Currently, only the class skills for the Barbarian class are stored. The remaining classes will have their skills linked in the near future.
* ABILITIES – not shown in graph above
  + This table holds the names and abbreviations of the 6 ability scores.
  + This table is not yet utilized as of V.1

## Future Tables

Tables for feats, skill stats, and inventory will be forthcoming as the project continues to develop. These will require their own systems and implementation due to how they are structured.

# Visual Presentation

The theming and layout of the paces will be reminiscent of the tables and charts in the core rule book, as well as the print version of the character sheet. For now, a basic monochromatic layout will be used to aid in the design of the wireframe for the system. Character sheet designs are not ready currently.

By the end of the project, every page will feature the imagery, decorations, and stylings typical of a fantasy setting. A mood board is to be assembled to clarify the desired look.

A screenshot of a computer

AI-generated content may be incorrect.

Mockup of the character list.

# Pages

Aside from the main roster and character sheet, there will be pages for adding new sheets to the list. And the character sheet itself will be comprised of multiple pages navigated by a tab bar. The location of this tab bar is yet to be decided.

## Add Character & Edit Character

These two pages use a shared character sheet form that is included in the PHP. The two pages use a <form> block that includes the character sheet file and their own button for telling the server to add the character data entered or update an existing record with new data.

## The Character Sheet

The character sheet file itself will contain all the fields a player or dungeon master would expect of a full character sheet. Depending on how much room the character sheet takes up in a window, it will be necessary to make the page responsive using the features of both HTML5 and CSS3.

There are at least two ways this could be done:

1. Using <div> containers and “flex” in CSS, the different character sheet elements would be arranged to fit a single column on a non-wide viewport or be double column in a wider viewport.
2. Hide different core elements like skills, feats, spells, and inventory behind tabs the user can click on to access. This would be accomplished through the inclusion of JavaScript. <https://www.w3schools.com/howto/howto_js_tabs.asp>

The entire sheet also features text and numeric elements that update with whatever the user inputs into the sheet, such as ability modifiers for skills and armor check penalties depending on what kind of armor the character has equipped. All JavaScript functionalities will happen on the client side, its sole purpose is to calculate results and populate user inaccessible fields, it cannot and will not write any data for the database.

### Short Bio

This holds the fields for the character’s name, race, class, class level, gender, alignment, and which campaign they belong to. The character level is the only field that is strictly numeric. With a limited range of 1 to 20. Currently, it is not possible to create a multi-class character. The implementation of a multi-class system should be investigated.

### Ability Scores

This block contains the user defined scores and auto-calculated modifiers of the six ability scores that influence the characters capabilities in a game. When creating a character, the scores are either rolled at random with three 6-sided dice or with a points system indicated in the core rulebook. These scores are unsigned (always positive and no lower than zero) and directly affect the ability modifier value. The ability modifier is used in skills, attack, and defense calculations. These modifiers are calculated as (*ability score* – 10) ÷ 2.

### Skills

Skills are used both inside and outside of combat. They allow a character to overcome obstacles and challenges put before them by the dungeon master. Each skill has a total bonus that is added to the roll of a 20-sided die. This bonus is calculated simply as *Skill points* (a.k.a. ranks) + *ability modifier* + 3 (if it is a class skill) + *any racial bonuses* + *any feat bonuses* + *any miscellaneous bonuses* – *their armors check penalty* (if any.)

For the visual appearance, the skill list and its many fields will be represented as a table. The headers will be tilted 45 degrees clockwise, pivoting from the top of the column. This is to minimize the overall width of the table.

The skill name column is generated from the skills table in the database, this allows for any custom skills to be added to the list by a player or dungeon master. The Untrained column will be marked with a special character from the UTF-8 table such as a filled square. Next is the Skill Bonus column which shows aa calculated total from the remaining columns using the above mathematical formula.

The Class Skill column uses the same special character as the untrained column to denote that skill as a class skill for those characters chosen class. The Armor Check Penalty column (abbreviated ACP) provides a negative penalty to that skill based on the ACP of any armor worn. The Ranks, Racial, Feats, and Misc columns are numeric form fields entered by the user and MUST contain an integer value.

### Feats

Feats are used to provide various situational bonuses to characters as they level up. The basic version of this system is with two text fields: the name of the feat, and the description of that feat. Once a feat is added, the user should be allowed to add another feat. A list of feats tied to the selected character should be made viewable and modifiable.

The database will need a “Feats” table with columns for ID, Character ID, Feat name, and Feat Description. The Character ID will have a Foreign Key constraint. When deleting a character, this table should be updated by first removing the records tied to the character before removing the characters record in the characters table.