Alexis Echano, Su Kara, Amelia Leon, & Priti Rangnekar
CS 151 Final Project Report
June 10th, 2023

College Reading Log

I. Background and Problem

College students have little time to enjoy reading for leisure and rarely have the bandwidth to enrich themselves with written texts other than textbooks. Students are also overburdened with academic text and need help to keep themselves accountable to reignite reading for fun again. There are also no comprehensive or logic-based reading logs out there that are geared toward college students. We aim to solve this issue by providing a worksheet that enables the logging of books and a way to keep track of book series, set personal reading goals, and explore new genres and languages.

This is an important endeavor and a project that is near to our hearts, as there are no solutions out there for students to mindfully include reading for leisure within their busy schedules while also providing a way for growth like increasing the difficulty of book reading across academic quarters or reading books from diverse authors. Furthermore, current platforms, such as Goodreads or Amazon, that provide information about books like page count, authors, and genre present it in a relatively scattered manner across their websites. Thus, for students wishing to incorporate such information into their reading schedules, the planning process requires significant manual information gathering and processing, such as in the form of Google spreadsheets or a Notes app. Through logic programming, we seek to encode this information in a concise and visualizable format presented in a manner directly in sync with a student's personal schedule and book choice.

Our project also provides a stepping stone to more possibilities. This includes other media formats, such as logging for TV shows, movies, and music, as well as other types of users, such as collegiate book club readers creating reading plans based on their custom club goals, or public libraries with summer reading challenges for elementary school readers, who would also enjoy experimenting with reading plans in an interactive format.

II. Solution

Our solution is a worksheet composed of HTML, Javascript, and logic rules that not only log books to read or have been read but also clearly outline guidelines and rules to fulfill like reading book series in order and not having duplicates. The website is divided into three key sections: goal setting, rule checks, and visualization of progress. These sections allow a user to properly see their reading goals and progress all in one simple page!

This solution is highly relevant to the logic programming world as it builds upon the basic idea of program sheets in a different context and more complicated rules and datasets. It

actually might have been more difficult to implement using traditional code because of the different cases of book arrangements like ensuring, simultaneously, that a user reads a book each quarter while being able to graph this data and fulfill other requirements like increasing difficulty levels.

III. Design, Logic Programming, and Data

The design of the system can be broken down into the user interface, the logic rules, and the data used which takes up over 1000 lines of code. First, the user interface is created totally from HTML with helper functions in Javascript to populate book data and load charts. The user interface has commented sections for the quarterly breakdown of books with dropdowns of all the possible books in our database/lambda. Every book is noted with a unique ID that we use within our logic rules. Then, the rules are highlighted in different colors to signify the status of the rules (red for unfulfilled requirements, green for completed requirements and suggestions, and yellow for unfulfilled suggestions). We then use Javascript to calculate averages and counts for the charts below which are bar charts and pie charts to visually show the user how many books they may read over the quarter, how the difficulty of reading changes, etc.

Furthermore, we created views to indicate that a book was selected in a given HTML selector and quarter. We also have views for the user's reading hours goal for each quarter, which we give them the autonomy to decide and enter themselves. Then, we had views for each of our rules as well as sub-aspects within each of those rules to indicate whether it was satisfied, based on the selection of a book(s) as well as data about the book(s)' information, and the data about the quarters in which books were selected as appropriate. We used elements of semantic programming to update the style (color) of a rule in HTML based on whether the corresponding view was fulfilled. Lastly, we had views for calculating statistics such as the sum and mean, and displaying the values as appropriate.

The data we used is hard coded into our worksheet. Within the dropdown sections, we have a list of books with their corresponding genre, series or language, page count, and difficulty level. For suggestions on which books to include and on estimates about difficulty level, we consulted reading lists such as those on Goodreads, and crafted OpenAI prompts to predict a book's difficulty level, although our final data was our own. We also have data needed for our rules, such as for indicating which books are sequels of one another. In addition, there is data for quarter-related information indicating which HTML elements correspond to which quarter (since there are 8 selector options per quarter), as well as which quarters come after the other. We included over 20 books as part of our current project but hope to expand to more books in the future!

IV. Challenges

Some challenges we faced in creating this final project include determining the project scope and working with an original design. Since the Program Sheet assignment was relatively

basic and utilized simple checkboxes, we had to quickly learn how to scale that idea to something bigger and enable data visualization, while using a mass amount of book data to make the log useful and efficient.

Since this type of project has not been done for our target audience and using logic programming, it was difficult to translate our idea into both working code like creating the user interface in pure HTML, and to architect how we may utilize data and logic rules. Thus, we built upon our knowledge from the Program Sheets assignment, while also researching relevant HTML and Sierra syntax and documentation from online sites and CS 151 published material to meet our needs.

Other challenges we faced were regarding the data. At first, we had issues narrowing down the scope of the project and figuring out the best way to encode them into lambda - for example, whether to have one line of data for each book containing all information about it, or whether to have separate lines of data for each piece of information about a book. To address these considerations, we thought about what would be the most scalable and concise approach and the extent to which having separate lines of data would be necessary.

For the rules, we had to be thoughtful in creating rules that represented a wide variety of logic programming capabilities (arithmetic aggregates, book selections for one quarter, book selections across quarters, etc) while also being of realistic interest to readers striving toward an educational but feasible reading plan. For example, for the rule about encouraging readers to maintain or increase their average difficulty from one quarter to the next, we searched information such as average pages read per hour and crafted our own mathematical formula to convert difficulty and page count into an estimate for the number of hours needed.

Another challenge was regarding the efficiency of our program. Certain rules, such as checking if a user has read multiple genres, were initially very slow, with the HTML sheet taking 1-2 seconds to update after a user selected a book. By using query optimization principles such as wiser subgoal ordering (e.g. having a subgoal about if a book is actually being read by a student before a subgoal about its genre), we were able to address this challenge and have our checkboxes and rule calculations happen within milliseconds.

Lastly, given our learnings over the quarter about appropriate logic programming style and syntax best practices, we sought to aim towards not only having code that was functional but also code that meet these standards; we made several passes over our project to strengthen our style.