*"Player Count, Game Type, Online, Review, Genre, Price, Console, Age Rating, Release Year"*

Valid Console Search Parameters:

* Player Count - Game Type
* Connectivity - Review
* Genre - Price
* Platform - Age Rating
* Release Year

(Click for additional info)

Acceptable Values

Player Count = 1-4 Game Type = Single-Player, Co-Op

Connectivity = Online, Local Review = X (out of 100)

Genre = any Price = XX.XX

Platform = any Age Rating = E, T, M

Release Year = YYYY

Valid Mobile Search Parameters:

* Review - Price
* In-App Purchases - Age Rating
* Size - Genre
* Update Year

(Click for additional info)

Acceptable Values

Review = X.X (out of 5) Price = XX.XX

In-App Purchases = Y/N Age Rating = X+

Size = X.X (MB) Genre = any

Update Year = YYYY (last update)

**About the Project**

WaterVapor Gaming is a project that we are still working on and perfecting. It is supposed to mimic a game store and be able to display any type of game.

The basis of this project was built around the Hash Map and Priority Queue data structures. The Hash Map was used to store elements that were read from the files and used for the primary data handling elements such as parsing data and inserting elements into the priority Queue. The Priority Queue was created to handle how the search engine would look through different games and present them. Based on a user’s given parameters (in order of priority) the search engine would extract data points to be displayed.

The app display was rendered through the SFML library. Each thing displayed is its own created object, since SFML doesn’t support text wrapping. We implemented our own way of manipulating the text objects to display on the window using vectors. Each game icon is also its own independent object that has its own functionality when pressed.

The biggest challenges we faced include properly parsing the data from the files, loading the text objects sequentially, and communicating the user input from the front end to the backend. Most of our time was spent trying to debug and find solutions for the text object display and how the program would interact with the data. Towards the end, we mostly debugged visual errors as well as incorrect input.

Some things we would have done differently would be to research different graphics libraries to use to put less strain and how text is handled on the frontend and backend, or even research if using HTML would have been better in the long run. We also would have started the project with the newest C++ library. We found more effective ways to handle data with C++23.

Jayden McKenna

jgmckenna02@gmail.com | 505.514.6465 | linkedin.com/in/jaydenmckenna | github.com/jaydenm1751

* From Albuquerque, New Mexico
* 3rd year Computer Engineering Major at the University of Florida
* Worked primarily on user interface and the creation of the Custom Priority Queue Data Structure.
* Other similar projects created: Minesweeper, Uno, Connect 4, AVL Tree Creation.