January 1, 2026

C964: Computer Science Capstone

Task 2 parts A, B, C, and D

Part A: Letter of Transmittal

January 7, 2026

Gabe Newell

Valve Corporation

Bellevue, WA

Dear Mr. Newell,

I am writing this letter to propose an enhancement to Valve Corporation's online game retail platform, Steam. The platform already has a sizable catalog, and as that number grows, users will continue to face challenges in discovering new games. This will result in users investing more time and effort into researching whether a game aligns with their interests. If these issues remain unaddressed, decision fatigue and hesitation will continue to negatively affect user experience, platform engagement, and potential sales.

The solution I am proposing is a game recommendation system that will allow users to enter a game and receive a ranked, curated list of similar games. The system will generate the results based on game metadata, including descriptions, genres, and tags. The solution will serve as a discovery tool, providing users with an efficient, intuitive way to explore the catalog without extensive manual searching.

The proposed solution will benefit Valve Corporation by improving user experience, engagement, purchase confidence, and post-purchase satisfaction. By improving how users discover games, the recommendation system will increase exposure across the entire catalog rather than only well-known titles. This increased exposure will boost visibility for smaller, more niche games, allowing independent studios to develop more titles and continue the revenue cycle. Users will no longer have to rely on third-party recommendation and review sites, as Steam will function as a one-stop platform for game discovery. Furthermore, the scalable design will ensure that game discovery remains consistent for users as Steam continues to expand its catalog.

The estimated development timeline is four to five weeks, including data preparation, development, interface integration, and testing. Until official integration, the data will be sourced from a Kaggle dataset that mirrors the Steam catalog. Since the system will use existing metadata and infrastructure maintained by Valve Corporation, expenses will primarily be for labor. Because the system will operate exclusively on game metadata and does not use personal user data, there are no significant ethical concerns.

Overall, the project will provide a cost-effective enhancement with long-term scalability to the Steam platform. With my experience in machine learning and developing data-driven systems, I am confident that this solution will provide meaningful value to the Steam platform and its users.

Sincerely,

Cole Linke

Cole Linke, Developer

Part B: Project Proposal Plan

Project Summary

As the Steam catalog expands, users will continue to face difficulties finding games that match their interests, making manual searching or third-party recommendations increasingly inefficient. Users will often experience decision fatigue and hesitate when purchasing new games. The proposed project will provide Valve Corporation with a discovery solution in the form of a recommendation system. The application will analyze game metadata to generate ranked recommendations based on similarity. The analysis will be generated from existing features such as title, game description, genres, and tags. Because this metadata is consistent across all games, the application will remain functional even as new games are added to the dataset.

The primary deliverable will be a content-based recommendation application. The application will allow users to enter a game title and receive a ranked list of similar titles. The results will also include a graphic showing which features are most prevalent among the recommendations. Secondary deliverables will include an intuitive user interface, a cleaned dataset, and a user guide detailing installation and usage instructions. The solution will benefit Valve Corporation by providing a maintainable recommendation system that improves discovery and reduces user frustration.

Data Summary

The data will be sourced from Kaggle and serve as a mirror of the Steam catalog until the application is integrated with the live database. The final dataset will be a Comma-Separated Values (CSV) file containing metadata for all Steam games as of January 2026. This file format was chosen because it can be efficiently processed by data analysis tools such as pandas.

During the design phase, key fields will be identified to accurately represent a game’s features. In the development phase, data will be cleaned, and selected fields will be used to generate similarity scores. Optional filtering will be present to allow users to fine-tune their results after similarity calculations are performed. Because the dataset contains only catalog metadata, ongoing maintenance will primarily involve updating it as new games are added to Steam.

The dataset will meet the project's needs because fields such as title, description, genre, and tags provide sufficient detail to characterize games for similarity-based recommendations. Irrelevant columns that do not contribute to similarity calculations will be dropped from the dataset. Missing or incomplete values will be removed during processing and handled again during calculations to ensure consistency. There will be no ethical or legal concerns about using this dataset because it contains no identifying user information, and all the data is publicly available on Steam.

Implementation

The project will follow the industry-standard Cross-Industry Standard Process for Data Mining (CRISP-DM). This methodology will provide a structured framework for developing a data-driven solution. The first phase is business understanding, which will focus on defining the need for improving content discovery on the Steam platform. Next, the data understanding phase will involve collecting the dataset

and identifying requirements for similarity-based recommendations. During this phase, an outline will be created identifying which features are relevant for similarity calculations and filtering, and which can be excluded. Data preparation will entail executing the outline by cleaning and restructuring the data. Missing or incomplete values will be handled during preprocessing and calculations, and irrelevant features will be removed. Modeling will be performed by comparing games based on their metadata, yielding tangible similarity scores. In the evaluation phase, the system will be tested to ensure that the results contain sufficient metadata compared to the target game to be considered relevant. After validation, the system will be deployed with a user-facing interface that allows users to generate recommendations. The system will be maintained by updating the dataset with the most current catalog of Steam games.

Timeline

Milestone or deliverable	**Project Dependencies**	**Resources**	**Start and End Date**	**Duration**
Finalizing Project Plan	N/A	Stakeholders, Project Manager, Developers	02/01/26 – 02/03/26	3 Days
Data Collection & Review	Project Planning Finalized	Kaggle Dataset	02/04/26 – 02/05/26	2 Days
Data Preparation & Cleaning	Data Collected and Reviewed	Pandas, Data Cleaning Script, Kaggle Dataset	02/06/26 – 02/08/26	3 Days
Model Development	Data Cleaned and Prepared	scikit-learn, TF-IDF Vectorization, Cosine Similarity	02/09/26 – 02/13/26	5 Days
UI Development & Integration	Model Developed	Voila, ipywidgets, WordCloud, matplotlib, Jupyter Notebook	02/14/26 – 02/18/26	5 Days
Testing & Model Validation	Application Integrated	NumPy, scikit-learn, Validation Metrics	02/19/26 – 02/22/26	4 Days
Documentation & User Guide	Testing Completed	Microsoft Word	02/23/26 – 02/28/26	6 Days
Deployment	Documentation and User Guide Completed	Voila, Internal Deployment Environment	03/01/26 – 03/02/26	2 Days

Evaluation Plan

During data collection and preparation, verification will consist of reviewing the dataset for completeness and the presence of required components. This will include handling records with missing values and removing irrelevant features. During model development, verification will involve testing recommendation outputs for logical consistency and ensuring that similarity calculations run without error. Interface components will be verified by ensuring that user inputs, filters, and visualizations respond and render as expected. Upon completion of development, validation will assess the recommender system's overall effectiveness. This will be done using quantitative metrics such as similarity scores and feature overlap between recommended and target games. These results will confirm that the system produces relevant and consistent recommendations.

Costs

Item	**Description**	**Cost**
Hardware		
Developer Workstation	Existing Computer and peripherals	$0
Software		
Windows	Operating System	$100
VS Code	Integrated Development Environment	$0
Python	Programming Language	$0
Pandas scikit-learn NumPy matplotlib WordCloud	Data Processing Library Machine Learning Library Numerical Computing Library Visualization Library Text Visualization Library	$0
Voila	Application Deployment Tools	$0
Jupyter Notebook	Development Environment	$0
Steam Game Dataset	Kaggle Dataset	$0
Labor		
Machine Learning Engineer	30 Days x 8 Hours x $60	$14,400
	Total Cost:	$14,500

Part C: Application

Application Files

- game_recommender.ipynb > Source code for the recommender application
- requirements.txt > Requirements for the recommender application
- steam_games_cleaned.csv > Cleaned dataset of Steam games
- data_cleaning.py > Code to Clean and convert JSON file to CSV

Part D: Post-implementation Report

Solution Summary

Valve Corporation sought to improve the user experience of browsing the Steam marketplace by implementing a game recommendation system. The previous version of the platform lacked an efficient way for users to discover games, instead relying on third-party recommendations or manual searching. Due to the ever-growing size of Steam's catalog, this resulted in wasted time, wasted purchases, and decision fatigue from the overwhelming number of options. The implemented solution addressed this problem by developing a content-based recommendation system that generated game suggestions based on similarities in game data. The application utilized Term Frequency-Inverse Document Frequency (TF-IDF) vectorization and cosine similarity to compare textual features, including game descriptions, genres, and tags, derived from the Steam dataset. After the new system was implemented, users could input a specific game title and receive a ranked list of recommended games.

Data Summary

The data for this project originally consisted of a JavaScript Object Notation (JSON) file sourced from a Kaggle dataset, which contained metadata for over one hundred thousand games available on the Steam platform. During data preparation, the JSON file was converted to a Comma-Separated Values (CSV) file. Irrelevant columns were removed, and records with missing or incomplete fields were excluded. Any remaining missing values were handled by substituting empty strings or zeros to prevent processing errors, and common stop words were excluded during vectorization. Relevant text fields were combined into a single textual feature, which was used as input for the recommendation algorithm. Key fields used from the dataset included the game name, a short description, genres, tags, and Metacritic score.

During the design phase, key fields were identified to accurately represent game features. In the development phase, cleaned and transformed data were used to generate similarity scores. An optional Metacritic score filter was included to fine-tune the results after similarity calculations were performed. The data preparation process was designed to be maintainable, allowing the application to be rerun with updated data.

Machine Learning

The machine learning approach implemented was an unsupervised content-based recommendation system using TF-IDF vectorization and cosine similarity. The libraries used were pandas for flattening data into data frames and scikit-learn for vectorization and similarity computations. The system analyzed game metadata and compared the games based on textual similarity. For each user-submitted input, the system produced a ranked list of games along with similarity scores.

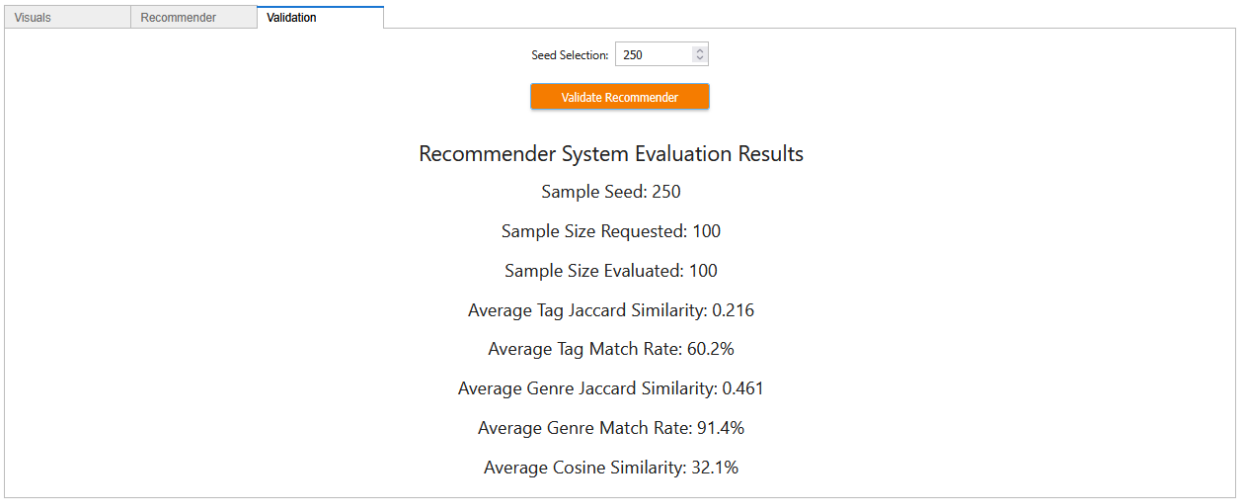
The system was developed by first creating a combined textual feature for each game, including its name, short description, genres, and tags. These combined features were converted into numerical representations using TF-IDF vectorization. TF-IDF was used to capture the relative importance of words across the dataset, while stop words were excluded to reduce the influence of common words. After that,

cosine similarity was applied to the vectors to measure the similarity between games. The similarity scores were then used to rank games, producing an ordered list of recommendations.

This method was selected because the available data consisted entirely of game metadata rather than labeled training data or user interactions. A content-based approach enabled games to be compared based on their features.

Validation

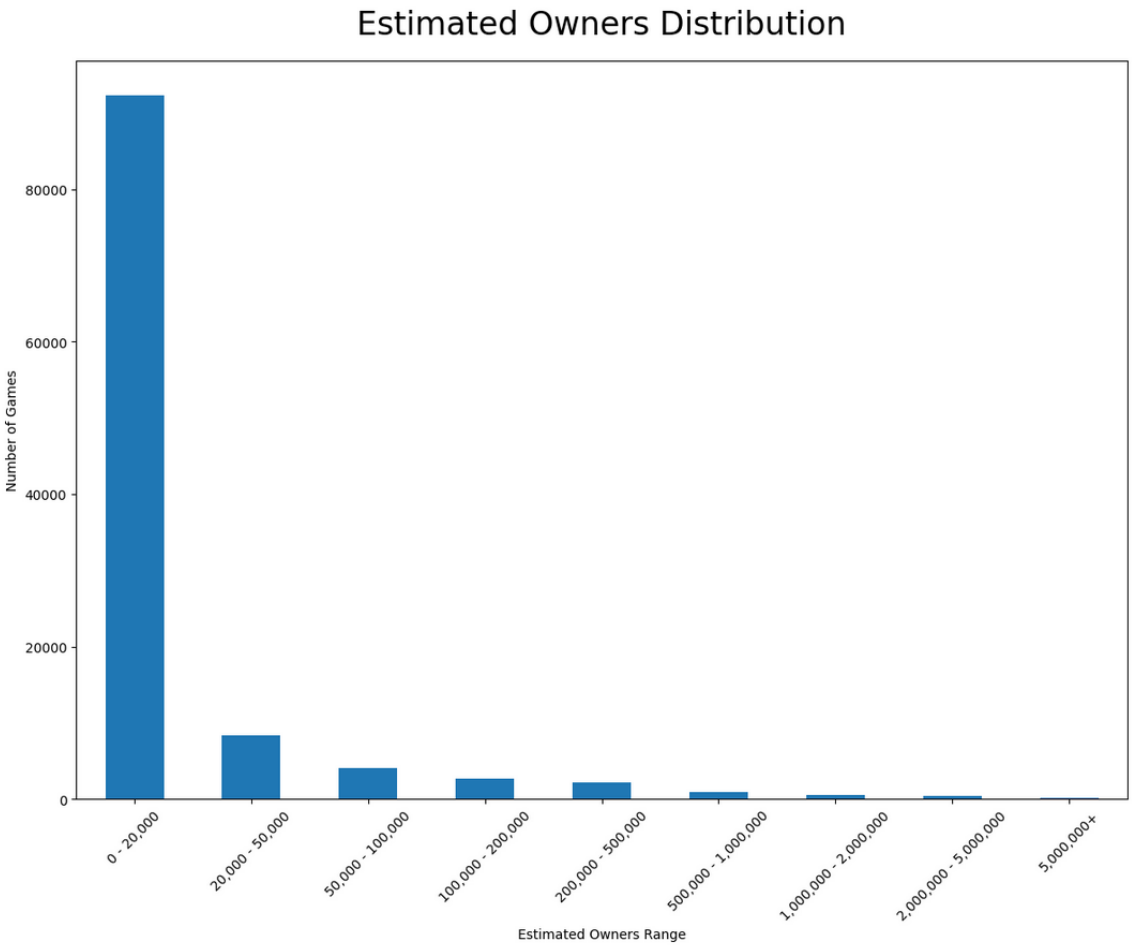
The recommendation system was classified as an unsupervised learning approach because it generated results based on feature similarity without labeled training data. Traditional unsupervised metrics were not applicable because the system did not perform clustering. Instead, model performance was evaluated using post-hoc internal consistency metrics. Validation was performed by running a simulation for 100 randomly selected games, and the top five recommendations for each game were evaluated for semantic relevance based on shared metadata features. An example of the validation output is shown with the accompanying visual.



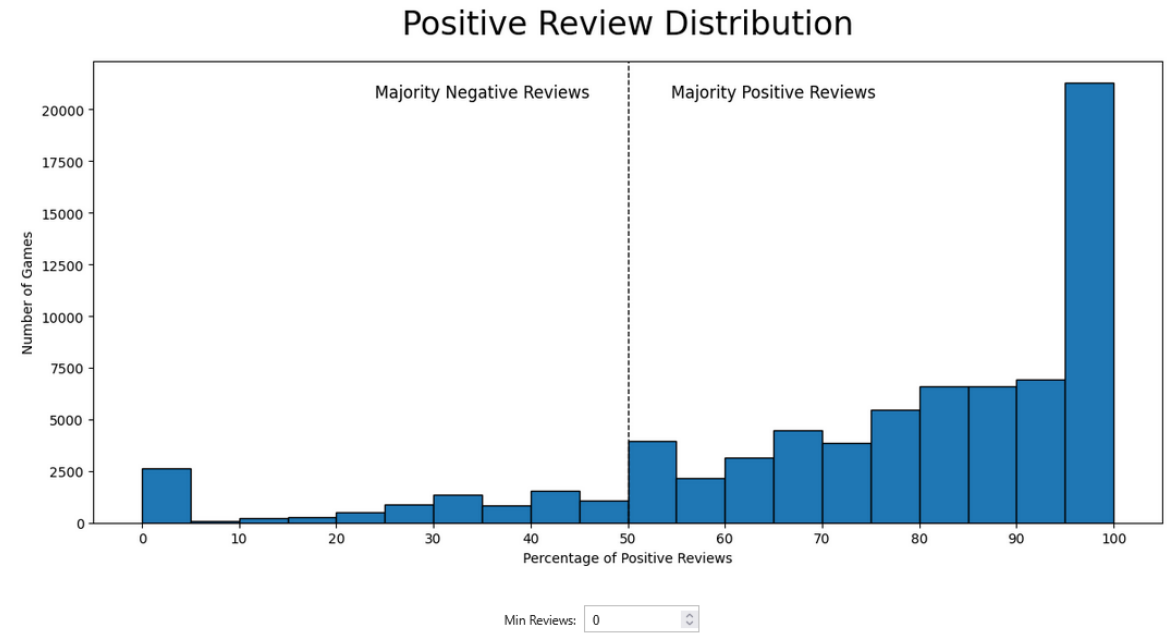
The Jaccard similarity measured the proportion of shared tags or genres between a target game and its recommendations, with values closer to 1 indicating stronger semantic overlap. The average match rate measured the number of recommended games that shared at least one tag or genre with the target game. The evaluation results showed frequent overlap in high-level features, such as genres, while overlap in fine-grained features, such as tags, occurred less often. This pattern suggests that recommendations are relevant and diverse. Cosine similarity scores further supported the results by showing that recommended games shared a meaningful degree of combined features with the target game.

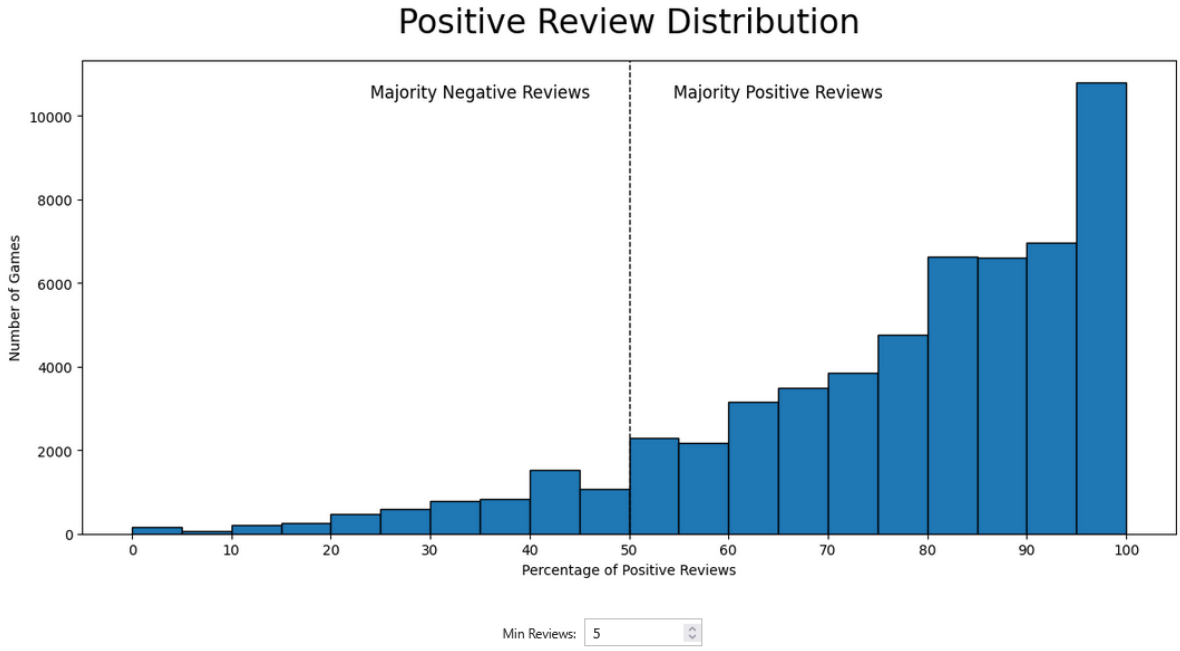
Visualizations

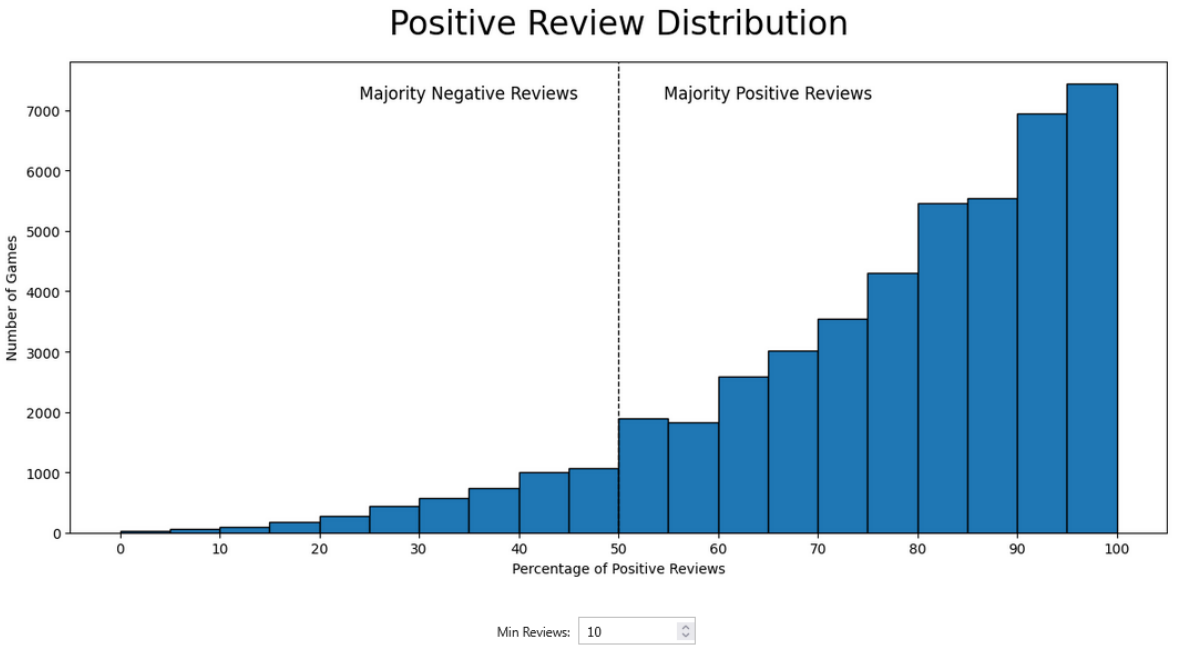
The application included four total visualizations, the first three of which were located in the Visualization tab, and the fourth was in the Recommender tab.

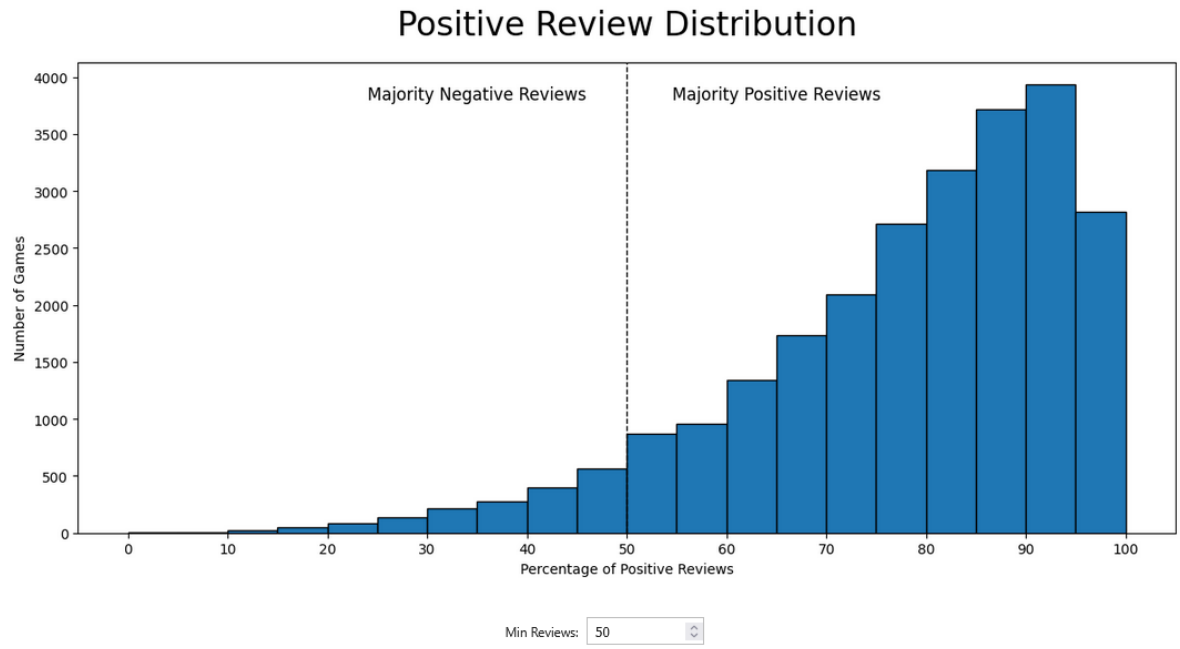


The first visualization was a histogram that displayed the distribution of estimated ownership across all games in the dataset. The data illustrated a drastic decrease from bucket one to two and a subtle decrease from bucket two onwards. This indicates that the vast majority of games in the dataset are relatively unknown to most players.

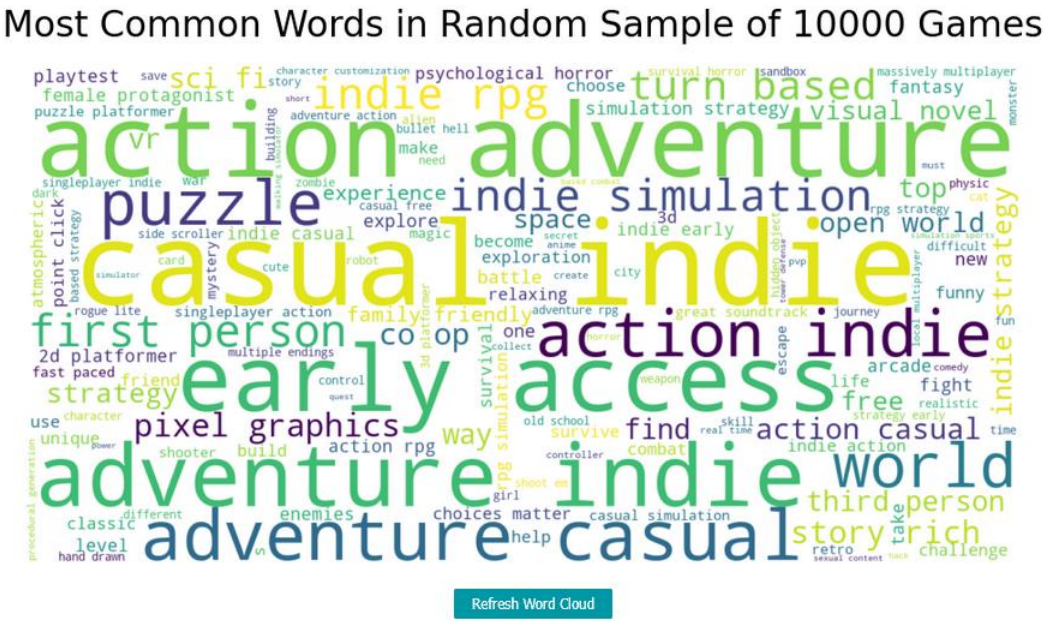

Positive Review Distribution
Majority Negative Reviews
Majority Positive Reviews
Number of Games
Percentage of Positive Reviews
Min Reviews: 0


Positive Review Distribution
Majority Negative Reviews
Majority Positive Reviews
Number of Games
Percentage of Positive Reviews
Min Reviews: 5

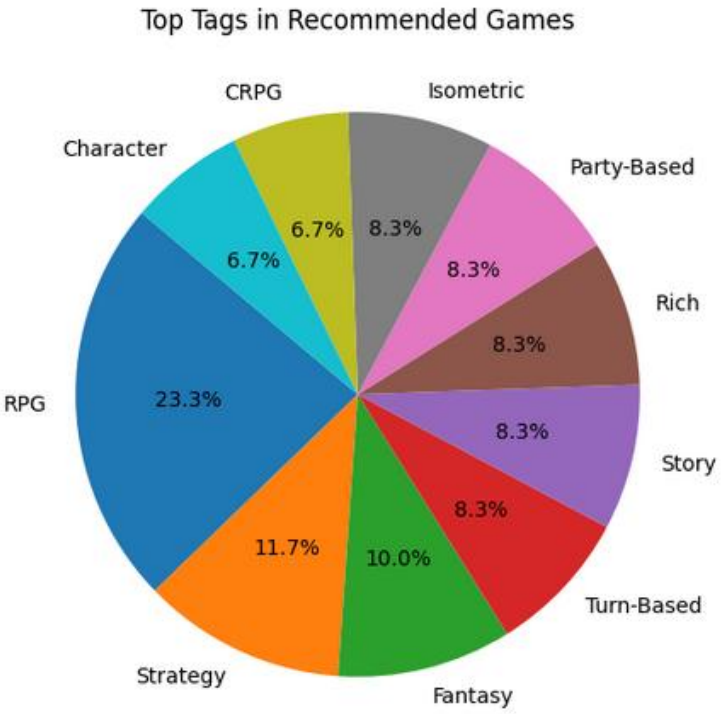




The second visualization was another histogram that depicted the distribution of positive reviews. This visualization included a minimum review input field to filter results. Games that had no positive or negative reviews were excluded from the initial rendering. The charts showed that as a higher minimum review was applied, the results stabilized and formed a steep curve with a noticeable drop in the 95-100% bucket. The results also showed that the majority of games received more positive reviews than negative ones.



The third visualization was a word cloud generated from the combined textual features used by the recommendation system. Due to the dataset's size, a random sample of 10,000 games was used, with a maximum word limit of 150. The visualization also included a refresh button to regenerate the word cloud with a different random sample. The results provided insight into the dominant themes across game descriptions, genres, and tags in the sample.



The final visualization was a pie chart created after the target game generated its recommendations. The pie chart displayed the relative frequency of the ten most common tags across the top five recommended games. This provided users with a tangible summary of the tags they were likely to be interested in.

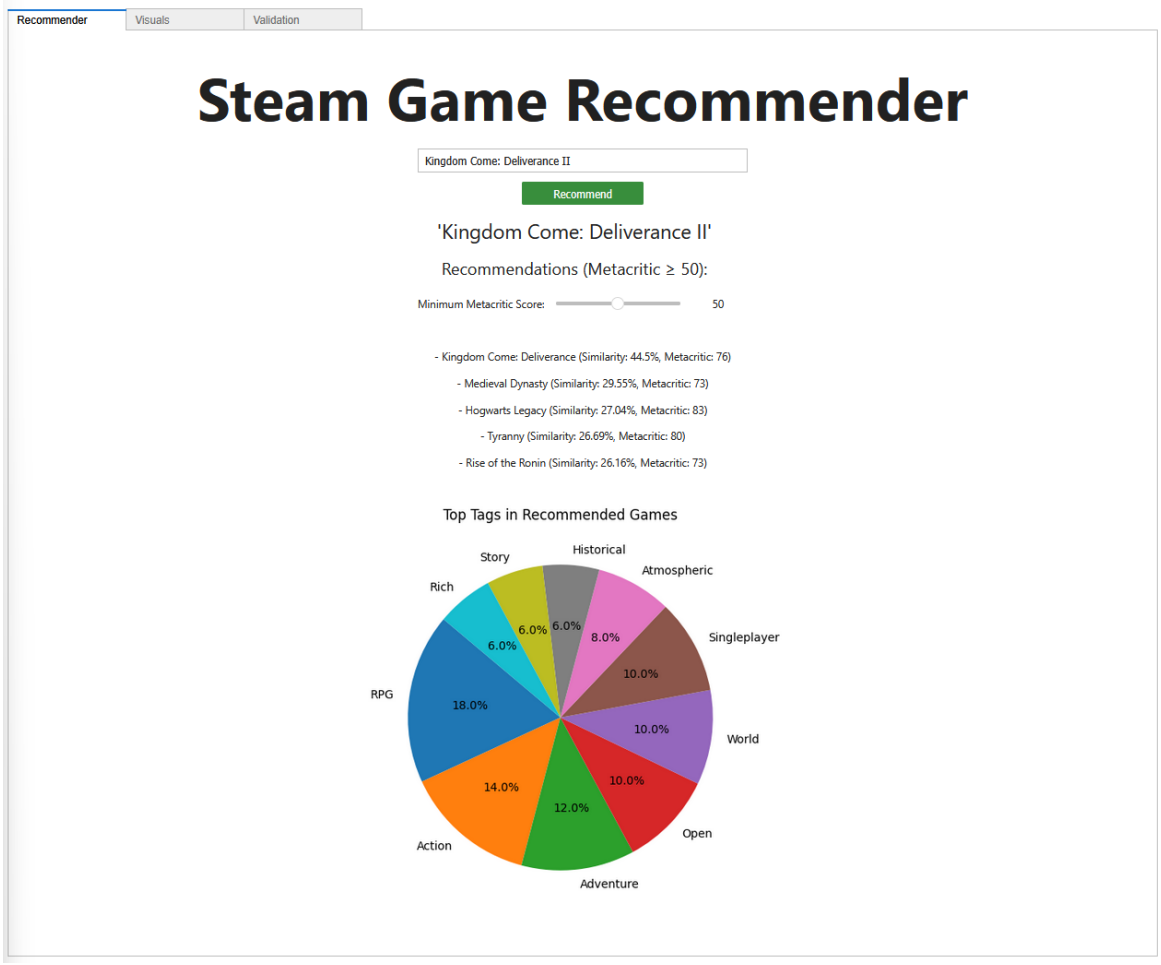
User Guide

User Guide (Local Setup)

1. Install Python 3.13
2. Download and unzip C964-steam-recommendation-capstone.zip
3. Open the Command Prompt and navigate to the project folder location
    a. cd C964-steam-recommendation-capstone
4. Make sure your current folder contains game_recommender.ipynb, requirements.txt, and steam_games_cleaned.csv
    a. dir
5. Install required dependencies
    a. python -m pip install -r requirements.txt
6. Run the application
    a. python -m voila game_recommender.ipynb
7. Your browser should automatically open, but if not, navigate to http://localhost:8866/
8. Notice that when the page loads, there are three different tabs at the top of the page

Recommender Tab:

1. To run the application, type a game title in the input text bar
2. The program will display a list of game titles after 4 characters, and you can choose your target game from the list if you see it.
3. Example input: “Kingdom Come: Deliverance II”
4. Click the “Recommend” button
5. The program will display five of the top recommendations for the target game
    a. The recommendations are in order of most to least similar
    b. Each recommendation is accompanied by a similarity score relative to the target game and a Metacritic review score
6. The “Minimum Metacritic Score” slider (default: 50) can be used to filter out games
    a. Games with 0 will be less known and usually from an independent studio
    b. Games with a score will usually be relatively well-known and from an established studio
7. The pie chart shows which tags are most common among the displayed recommendations
8. Changes to the “Minimum Metacritic Score” slider will update recommendations and tags in the pie chart



Visuals Tab:

1. Estimated Owners Distribution
    a. Shows the number of games where each range is the estimated number of owners
2. Positive Review Distribution
    a. Shows the number of games where each range is the percentage of positive reviews relative to total reviews.
    b. Ranges below 50 have a majority of negative reviews, and ranges above 50 have a majority of positive reviews
    c. Use the “Min Review” input box to filter games and change the histogram
    d. Use the arrows to increment by one or type a number and hit enter on your keyboard
    e. Example input: 0, 5, 10, 100
3. Word Cloud
    a. Shows the most common words in a random sample of 10,000 games
    b. Included game descriptions, genres, and tags
    c. Click the “Refresh Word Cloud” button to generate a new random sample of 10,000 games

Validation Tab:

1. Click “Validate Recommender” to run validation
2. Runs the recommender application on 100 random games and displays the average statistics
3. Jaccard Similarity: measures the overlap of shared features between a target game’s set and the recommendations’ combined set
    a. Example: 0 = no overlap, 1 = identical sets
4. Match Rate: measures the rate at which a recommendation shares at least one feature with a target game
    a. Example: 80% means that 4/5 recommendations share at least one feature with the target game
5. Average Cosine Similarity: measures the average cosine similarity score for all instances of the 100 game sample
6. The “Seed Selection” input box can be used to choose a new seed of 100 random games to validate