- \*\*Graph Relationships\*\*: User-game interactions and game similarities.

- \*\*Genre Hierarchy Tree\*\*: Hierarchical organization of games by genre/popularity.

- \*\*3D Visualization\*\*: Interactive exploration of game connections.

- \*\*Hybrid Recommendations\*\*: Combines graph and tree models for accuracy.

1. Data:

- Raw Data: Steam datasets (`users\_reviews.json`, `users\_items.json`, `bundle\_data.json`).

- Processing: Clean playtime data, normalize scores, extract genres, and save structured data (e.g., `user\_game\_edges.csv`).

- Tools: `pandas` for data handling, log-transform for playtime normalization.

Columns:

User\_id

Item\_id

Item\_name

Playtime

Recommend

Review

Sentiment score

2. Graph:

- Structure: Bipartite graph with users, games, and game-game edges.

- Edge Weights: Playtime + recommendations (user-game), Jaccard/cosine similarity (game-game).

- Algorithms: Jaccard similarity for shared users, cosine similarity for playtime vectors.

3. Tree:

- Hierarchy: Root → Genres → Games.

- Popularity Sorting: BST-like ranking based on playtime, recommendations, and user count.

4. Recommendation:

- Hybrid Scoring: Combines graph-based (70%) and tree-based (30%) recommendations.

- Filtering: Removes already played games and deduplicates results.

5. UI & Visualization:

- Streamlit UI: Search bar, recommendation list with reasoning (e.g., "Similar to Left 4 Dead 2").

- 3D Visualization: `plotly`-powered interactive graph with zoom/rotate functionality.

---

3. File Structure

│ ├── raw/ # Input JSON files

│ └── processed/ # Cleaned CSVs/Parquet

│ ├── data\_processing.py # Load, clean, preprocess data

│ ├── graph\_model.py # Build user-game graph

│ ├── tree\_model.py # Create genre hierarchy

│ ├── recommendation\_engine.py # Hybrid recommendation logic

│ ├── visualization.py # 3D graph rendering

│ ├── main\_app.py # Streamlit UI entry point

4. Workflow

1. Data Loading & Validation:

- Load raw JSON files, validate schema, log errors.

2. Preprocessing:

- Clean missing values, normalize playtime, encode recommendations.

3. Model Construction:

- Graph: Build user-game and game-game edges.

- Tree: Insert games into genres, sort by popularity.

4. Recommendation Generation:

- Hybrid scoring merges graph (collaborative filtering) and tree (genre-based) results.

5. UI/Visualization:

- Streamlit app displays recommendations and 3D graph.

5. Key Algorithms

- Graph Edge Weighting:

`weight = log(playtime) + (recommend \* 5)`

- Similarity Metrics:

- Jaccard: `shared\_users / total\_users`

- Cosine: Playtime vector dot product.

- Hybrid Ranking:

`final\_score = (graph\_score \* 0.7) + (tree\_score \* 0.3)`.

6. Tools & Libraries

- Data: `pandas`, `numpy`.

- Graphs: `networkx`, `scipy.spatial.distance`.

- UI/Visualization: `streamlit`, `plotly`.

- Utilities: `logging`, `json`.

Model Construction and Recommendation Generation

1. Model Construction

Graph

- Nodes:

- User Nodes: Represented by `user\_id`.

- Game Nodes: Represented by `item\_id`.

User to game:

- Playtime

- Review

- Recommend

- Sentiment

Game to game

- Similarity (game to game)

b. Tree Model

- Hierarchy:

- Root: "All Games".

- Intermediate Nodes: Genres (e.g., "Action", "RPG").

- Leaf Nodes: Games (e.g., "Left 4 Dead 2").

- Popularity Score:

- Sorting Mechanism:

- Within each genre, games are stored in a priority queue sorted by `popularity\_score`.

- Example: For the "Action" genre, games are ranked as `[(game1, 95), (game2, 88), ...]`.

- Dynamic Updates:

- Recalculate popularity scores periodically (e.g., weekly) to reflect new user interactions.

2. Recommendation Generation

a. Graph-Based Recommendations

1. Find Similar Users:

- Identify users who played the same games as the target user.

Age

Gender

Tone

- Rank them by overlap in playtime and recommendations.

2. Suggest Games from Similar Users:

- Aggregate games played by similar users but not yet played by the target user.

- Score games using:

b. Tree-Based Recommendations

1. Extract Genres:

- Identify genres of the user’s input games (e.g., "Action", "Horror").

2. Recommend Popular Games:

- Fetch top-`N` games from each genre’s priority queue.

c. Hybrid Ranking

- Combine Score:

- Filtering:

- Remove games the user already played.

- Deduplicate overlapping recommendations from both models.

Example Workflow:

1. Input: User has played "Left 4 Dead 2".

2. Graph:

- Find users who played "Left 4 Dead 2" → they also played "Killing Floor".

- Calculate `graph\_score` for Killing Floor based on playtime and recommendations.

3. Tree:

- "Left 4 Dead 2" is in the "Action" and "Horror" genres.

- Fetch top games in "Action" (e.g., "DOOM Eternal") and "Horror" (e.g., "Resident Evil 7").

4. Hybrid:

- Merge and rank:"Killing Floor” (graph), "DOOM Eternal" (tree), "Resident Evil 7" (tree).

UI/UX Design and Implementation

The Streamlit-based UI serves as the frontend for the Interactive Game Recommender, designed to be intuitive, visually engaging, and responsive. Below is a detailed breakdown of its components, functionality, and user experience:

1. Input Interface

- Game Search Bar

- Auto-Complete:

- Dynamically suggests game names as the user types (e.g., typing "Left 4" → suggests "Left 4 Dead 2").

- Uses a precomputed list of game names from `game\_metadata.csv`.

- Multi-Select:

- Users can input multiple games they’ve played (e.g., "Left 4 Dead 2", "Killing Floor").

- Validation:

- Checks if the input game exists in the dataset.

- Displays an error message for invalid entries (e.g., "Game not found. Did you mean...?").

- Genre Filters (Optional)

- Dropdown menu to prioritize specific genres (e.g., "Action", "Horror").

- Adjusts tree-based recommendations by boosting scores in selected genres.

2. Recommendation Display

- Primary Recommendations

- Table View:

- Columns: `Game Title`, `Genre`, `Similarity Score`, `Reasoning`.

- Example row:

| Game Title | Genre | Score | Reasoning |

|---------------------|-------------|-------|-----------------------------|

| Killing Floor | Action | 92 | "Similar to Left 4 Dead 2" |

- Sorting Options:

- Users can sort by score, genre, or playtime.

- Secondary Recommendations

- "Popular in Your Genres":

- Lists top games in genres the user’s input games belong to (e.g., "Action", "Horror").

- "Trending Now":

- Displays recently popular games (updated weekly from `tree\_model` popularity scores).

- Reasoning Tags

- Visual badges explaining why a game is recommended:

- 🎮 "Liked by players of [Game X]" (graph-based).

- 🌟 "Top-rated in [Genre Y]" (tree-based).

- 💰 "Frequently bundled with [Game Z]" (from `bundle\_data.json`).

3. Interactive 3D Visualization

- Graph Visualization

- Nodes:

- Games: Colored by genre (e.g., Action = red, RPG = blue).

- Users: Smaller gray nodes (optional toggle).

- Edges:

- User-game edges: Faint lines with opacity proportional to playtime.

- Game-game edges: Bold lines scaled by similarity (e.g., Jaccard > 0.7).

- Interactivity:

- Hover: Displays game details (title, genre, popularity score).

- Click: Highlights connected nodes (e.g., clicking "Left 4 Dead 2" shows games played by its users).

- Zoom/Rotate: Users can explore the graph in 3D space.

- Embedded Visualization

- Rendered using `plotly.graph\_objects` and embedded directly into the Streamlit app.

- Performance optimization:

- Limit displayed nodes/edges for large datasets (e.g., top 200 games).

- Use `scipy.sparse` matrices to handle similarity calculations efficiently.

4. Layout and Navigation

- Dashboard Layout

- Left Sidebar:

- Search bar, genre filters, and settings (e.g., toggle user nodes).

- Main Panel:

- Top: Recommendation table.

- Bottom: 3D visualization.

- Responsive Design:

- Adjusts layout for mobile/desktop using Streamlit’s grid system.

- Tabs for Advanced Users

- "Graph Explorer": Focuses on the 3D visualization with advanced controls (e.g., similarity threshold sliders).

- "Genre Deep Dive": Shows genre-specific popularity rankings from the tree model.

5. Error Handling and Feedback

- Input Error:

- Graceful handling of invalid games/genres with suggestions (e.g., "Did you mean Killing Floor 2?").

- Empty Results:

- Displays fallback recommendations (e.g., "Top 10 Games of 2023").

- Loading States:

- Progress bars/spinners during data processing or graph rendering.