EPPS 6354 - Single-Player Video Game Library
Dr. Karl Ho
Final Report

Kasra Akbari, 2026-05-11

1. Introduction

In a traditional library, readers select a book based on genre, author, release date, and recommendations. A librarian can guide readers to books that match their interests by using organized information about the library's collection. For this project, a single-player video game library functions similarly to a traditional library, with the database serving as the library. Single-player games include numerous pieces of information, such as genres, developers, player progression, recommendations, platforms, publishers, completion status, and gameplay advice. A game may include different platforms and different progressions for different players.

2. Project Purpose

The goal is to allow users to explore game information, view recommendations, track player progression, and access gameplay advice. This project is set in four steps. First is schema design, where the author defines tables, keys, and bridge tables. Second is the database build, in which the tables and data are created in PostgreSQL and then converted to SQLite. Third is the analysis logic, where R is utilized to rank unplayed games by similarity to a player’s completed games. Fourth is the app display, where Shiny is used to demonstrate outputs to users.

3. Data and Scope

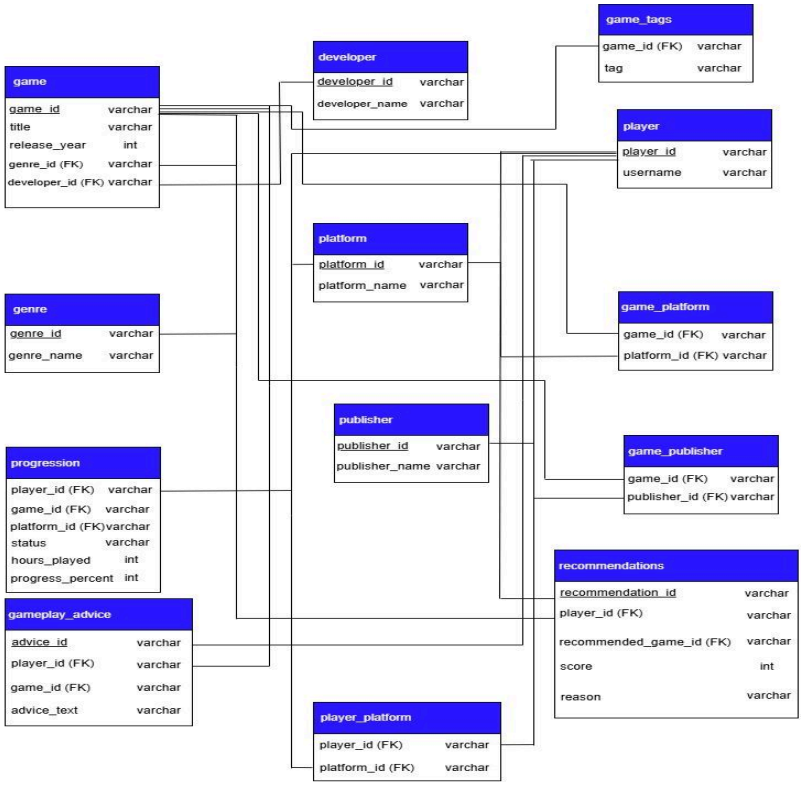
The games are based on real single-player video games, but player records are simulated for the project demonstration because simulated players show progress tracking, recommendations, and gameplay advice without requiring real personal data.

4. Database Design

4.1 Relational Schema

The database was first built and tested in PostgreSQL, with 13 tables implemented in the schema. These are genre, developer, publisher, platform, player, game, game_platform, game_publisher, progression, game_tags, player_platform, recommendations, and gameplay_advice. Each table contains attributes that are connected to other tables through primary and foreign keys, where the primary keys are underlined, and the foreign keys are labeled as (FK).

Figure 1. Schema Diagram



4.1.1 Genre Table

The *genre* table contains data on single-player video game genres, with two attributes: *genre_id* (the primary key) and *genre_name*. This table is necessary because it provides the broad genres for each game.

Table 1. Genre

genre_id	genre_name
G1	Action
G2	RPG
G3	Survival/Psychological Horror

4.1.2 Developer Table

The *developer* table stores information about the studios that developed the games in the database. Each developer is assigned a *developer_id*, which serves as the primary key, along with a *developer_name*. The full table contains 16 developers, but the table below shows a sample of six rows to demonstrate the table structure.

Table 2. Developer

developer_id	developer_name
D1	id Software
D2	QLOC
D3	Capcom
D4	Sandfall Interactive
D5	Sucker Punch Productions
D6	Motive Studio

4.1.3 Publisher Table

The *publisher* table contains data about the companies that published the games in the database. Similar to the *developer* table, each publisher is assigned a *publisher_id*, serving as the primary key, along with a *publisher_name*. This full table contains 10 publishers, but it shows a sample of six rows to demonstrate the structure in Table 3.

Table 3. Publisher

publisher_id	publisher_name
P1	Bethesda Softworks
P2	Bandai Namco
P3	Capcom
P4	Kepler Interactive
P5	Sony Interactive Entertainment
P6	Electronic Arts

4.1.4 Platform Table

The *platform* table stores gaming platforms in the database and contains two attributes: *platform_id* (the primary key) and *platform_name*, which identifies the platform. The platforms incorporated for the project are shown in Table 4.

Table 4. Platform

platform_id	platform_name
PL1	PC
PL2	PS5
PL3	Xbox Series X
PL4	Nintendo Switch 2

4.1.5 Player Table

The *player* table stores players in the database and comprises two attributes: *player_id*, which serves as the primary key, and *username*, which identifies users. Table 5 shows the structure of the *player* table.

Table 5. Player

player_id	username
U1	GamerLegend25702
U2	SohaAkbari052027
U3	Cryptococomydawg
U4	TheDarKnight
U5	SilentBlade
U6	PrinceOfPersia
U7	ItzSpidey

4.1.6 Game Table

The *game* table is the central table in the project, containing the main information for each game in the database. It comprises five attributes: *game_id (primary key); title (identifies the game); release_year (records the year of the game's release); genre_id (connects the game to the genre table); and developer_id (connects the game to the developer table)*. Table 6 shows the structure of the *game* table.

Table 6. Game

game_id	title	release_year	genre_id	developer_id
GM1	Doom: The Dark Ages	2025	G1	D1
GM2	Dark Souls: Remastered	2018	G2	D2

GM3	Resident Evil: Requiem	2026	G3	D3
GM4	Clair Obscur: Expedition 33	2025	G2	D4
GM5	Ghost of Tsushima	2020	G2	D5

4.1.7 Game_Platform Table

The *game_platform* table is a bridge table that handles the many-to-many relationship between games and platforms. Connecting one game to one platform per row, the table uses two foreign keys: *game_id* and *platform_id*. This structure prevents platform information from being repeated in the main *game* table and allows the database to show which platform each game is available on. Because the table contains 73 rows, Table 7 shows a sample of the *game_platform* data.

Table 7. Game_Platform

game_id	platform_id
GM1	PL1
GM1	PL2
GM1	PL3
GM1	PL4
GM5	PL1
GM5	PL2

4.1.8 Game_Publisher Table

Similar to the previous table, the *game_publisher* table is a bridge table that handles the many-to-many relationship between the games and publishers. This structure is necessary because one publisher can be associated with multiple games, and a game may also be connected

to more than one publisher. Each row utilizes two foreign keys, *game_id* and *publisher_id, to* connect the two tables. Table 8 shows a sample of the *game_publisher* table.

Table 8. Game_Publisher

game_id	publisher_id
GM1	P1
GM2	P2
GM3	P3
GM4	P4
GM5	P5
GM6	P6

4.1.9 Progression Table

The *progression* table tracks each player's activity in the game library, linking players to specific games and platforms via three foreign keys: *player_id*, *game_id*, and *platform_id.* Additionally, the table includes *status, hours_played, and progress_percent*, demonstrating whether a player has completed a game, is currently playing it, or has placed it in their backlog. This table helps the Shiny app function by allowing users to view their individual game history and progression. Since the table contains numerous rows, Table 9 shows a sample of the *progression* table.

Table 9. Progression

player_id	game_id	platform_id	status	hours_played	progress_percent
U1	GM1	PL2	Playing	40	60
U1	GM2	PL1	Completed	60	100
U2	GM1	PL3	Backlog	0	0
U2	GM3	PL4	Playing	12	25

U3	GM3	PL1	Completed	40	100

4.1.10 Game_Tags Table

The *game_tags* table provides more specific descriptions of each game beyond the genre categories. While the database includes three main genres – *Action, RPG,* and *Survival/Psychological Horror* – they do not fully describe the specific features of each game. The *game_tags* table addresses this limitation by assigning descriptive keywords to games, such as *fps, dark_fantasy, story_rich*, or *turn_based_rpg.* An example would be *Doom: The Dark Ages*, which is categorized under the *Action* genre, but the tags provide detailed descriptions by identifying it as *an action, first-person shooter,* and *dark fantasy game*.

This table is essential because the R recommendation system uses tags to compare games more precisely. The app uses shared tags to identify games with similar themes, gameplay styles, and difficulty levels. The *game_tags* table contains two attributes: *game_id*, serving as the foreign key linking each tag to a game, and *tag*, which contains the descriptive keyword. Table 10 shows a sample of the *game_tags* table.

Table 10. Game_tags

game_id	**tag**
GM2	soulslike
GM2	dark_fantasy
GM2	open_world
GM2	difficult
GM4	action_rpg
GM4	fantasy
GM4	story_rich
GM4	turn_based_rpg

4.1.11 Player_Platform Table

Like other bridge tables, the *player_platform* table handles the many-to-many relationship between players and platforms, where each row connects one player to one platform via two foreign keys: *player_id* and *platform_id*. This makes the table essential because one player can own multiple platforms, and one platform can be owned by multiple players. Table 11 shows a small sample of the *player_platform* table.

Table 11. Player_Platform

player_id	platform_id
U1	PL1
U1	PL2
U2	PL4
U1	PL4
U2	PL1
U2	PL3

4.1.12 Gameplay_Advice & Recommendations Tables

The *gameplay_advice* and *recommendations* tables are added to support the application's generated outputs. Unlike the main tables, they are not manually inserted with data because recommendations and gameplay advice are generated dynamically based on the player's history, platform ownership, and game preferences, allowing the app to produce personalized results rather than storing the same advice for every user.

5. Database Implementation

The database was first implemented and designed in PostgreSQL, so that the relational structure could be established and tested before connecting it to the user-interactive app. After the schema design, the tables were created with primary and foreign keys to establish and maintain relationships among games, players, platforms, genres, developers, publishers, tags, and progression data. The main entity tables were developed first, followed by the bridge tables that handle many-to-many relationships, such as *game_platform*, *game_publisher*, and *player_platform*.

After the tables were developed, the data were inserted into each table, including single-player video game titles, developers, publishers, platforms, release years, genres, and tags. The player data were simulated for demonstration purposes so that the application can present player histories, platform ownership, progress tracking, and recommendation results without using real player data. Once the data were implemented, the author ran SQL queries to test whether the tables were correctly connected and functioning.

After the database was created in PostgreSQL, it was converted to SQLite for app integration and web connectivity. SQLite was used because it was simpler to package with the Shiny application and allowed the app to query the database directly. It also helped establish a connection once published online, since PostgreSQL only functioned locally on the author's device, where the database was created. The Shiny app uses the SQLite database to display the game library, player platform ownership, player progression, generate recommendation results, and establish a leaderboard output.

6. SQL Queries and Testing

With the tables established and the data inserted, SQL queries were used to verify the database's relationships, which were important because the Shiny application relies on joined data from multiple tables. This paper will provide two SQL query samples: Game Library and Player History. These queries were selected because they demonstrate two major outputs of the application: the full game library and each player's individual game history.

6.1 Query 1: Game Library Query

```
select
    g.game_id,
    g.title,
    g.release_year,
    ge.genre_name,
    d.developer_name,
    pub.publisher_name
from game g
join genre ge
    on g.genre_id = ge.genre_id
join developer d
    on g.developer_id = d.developer_id
join game_publisher gp
    on g.game_id = gp.game_id
join publisher pub
    on gp.publisher_id = pub.publisher_id;
```

This query tests whether the *game* table is correctly connecting to the *genre, developer, game_publisher,* and *publisher* tables. The results produce a complete game library by displaying each game's ID, title, release year, genre, developer, and publisher, supporting the Game Library tab in the app.

6.2 Query 2: Player History Query

```
select
    p.username,
    g.title,
    pl.platform_name,
    pr.status,
    pr.hours_played,
    pr.progress_percent
from progression pr
join player p
    on pr.player_id = p.player_id
join game g
    on pr.game_id = g.game_id
join platform pl
    on pr.platform_id = pl.platform_id
order by p.username, g.title;
```

This query tests whether the *progression* table correctly connects players, games, and platforms, and the output shows each player's game title, platform, status, hours played, and progress percentage. This query supports the Interactive App tab, where users can select a player and view their history.

7. Recommendation Logic and Shiny App

7.1 About the App

The Shiny app contains four main features. The first feature is the "About the App" tab, which gives the user an introduction to the app, what it offers, and what they can do with it.

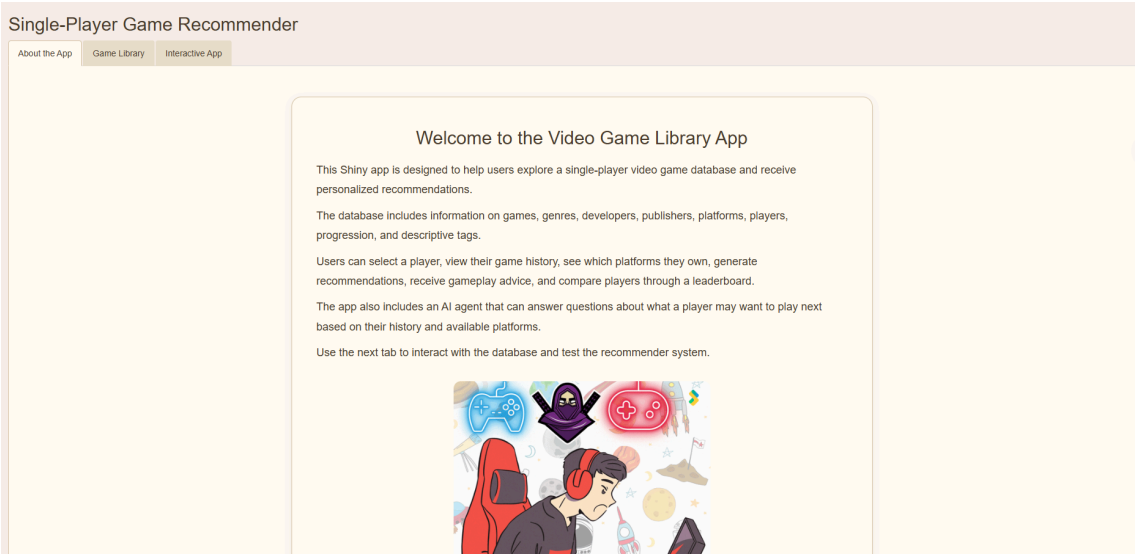


Figure 2. About the App Tab

7.2 Game Library Tab

The second feature is the "Game Library" tab, which displays the games in the database, including game covers, game_id, title, release_year, genre_name, developer_name, and *publisher_name*.

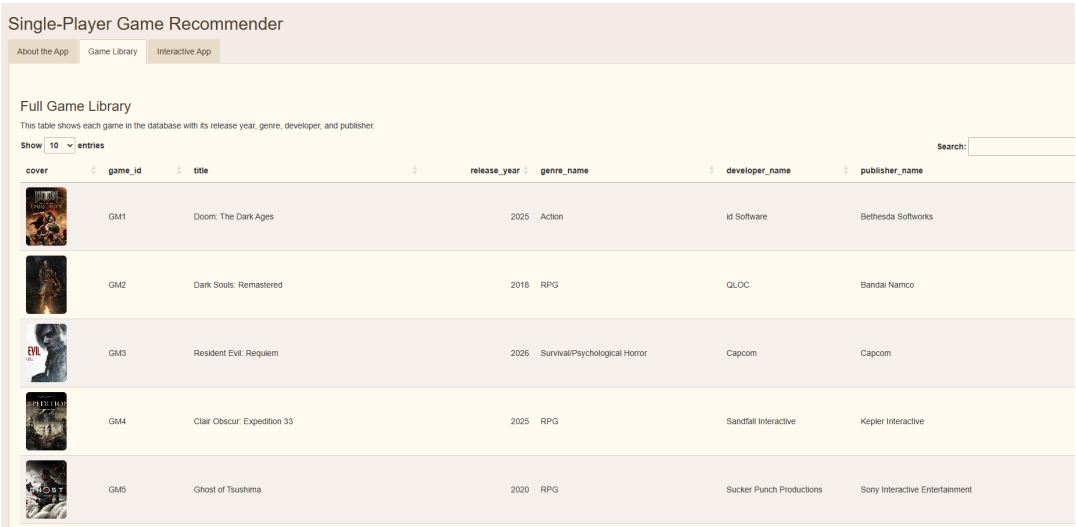


Figure 3. Game Library Tab

7.3 Interactive App Tab

The third feature is the “Interactive App” tab, where the user can select their username from the database, and the tab then displays the platforms they own. Once the username is selected, the app will also display the player's history, including the game cover, title, platform, player status, hours played, progress percentage, and release year. Below the “Select player” button is the “Number of recommendations” UI, where the player can generate as many recommendations as they wish.

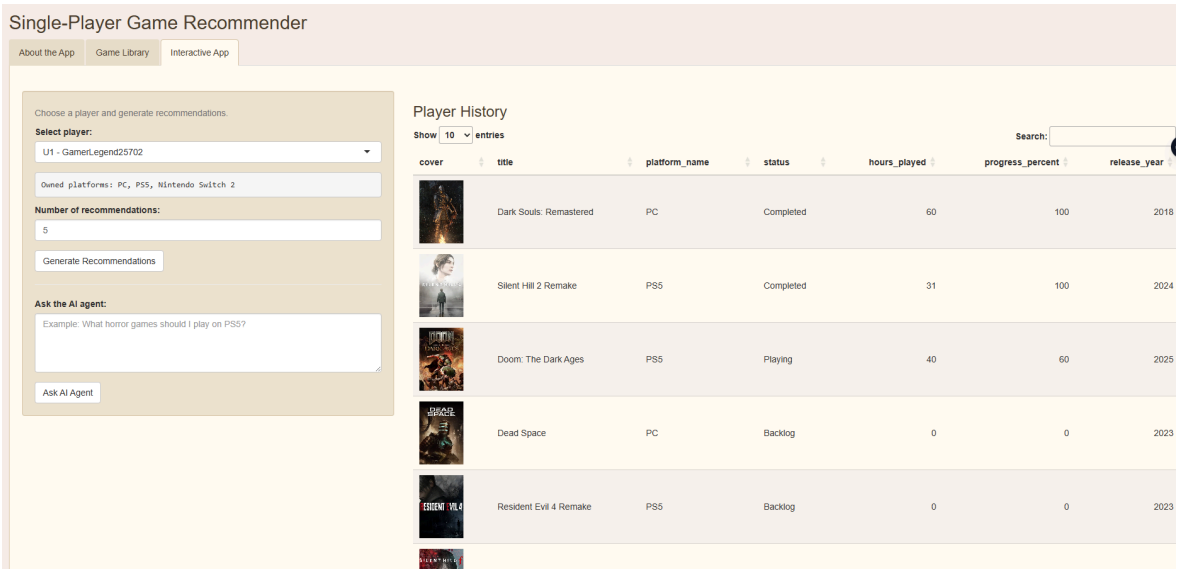


Figure 4. Interactive App Tab

The Shiny app is available at: https://019dd61f-326b-aec6-a86f-4499e1e19461.share.connect.posit.cloud/. The app allows users to view games, select players, review player history, generate recommendations, receive gameplay advice, and interact with the AI agent.

7.4 Recommendation Logic

Once the user selects the number of recommendations, R calculates a similarity score based on the player's history by comparing the items to games the player has already played or completed. The formula is *score = 2 * same_genre + same_developer + shared_tags_n*. Once calculated, the recommendations and gameplay advice will appear alongside the AI agent's text output. The recommendations explain to the user what the game is about and why it aligns with their interests. Below the recommendations section, the gameplay advice explains how to play the game and improve one's gameplay. Every recommendation and gameplay advice will differ for each user based on their player history, so the explanations won't be exactly the same.

7.5 Leaderboard

The Shiny application includes a leaderboard at the bottom of the Interactive App tab. This feature pulls information from the progression table and calculates the number of completed games, total hours played, and average progress percentage for each player. The output is then ranked for users to see which players have the highest completion activity.

8. AI Agent and Posit Connect Cloud

The *OpenAI* agent is included as an additional feature that extends the app beyond the database-driven recommendation system. While the main recommendations are generated with the database fields and R scoring logic, the AI agent allows users to ask broader or more specific video game questions in natural language. For example, a player who owns a Nintendo Switch 2 can ask which games fit their player history. The AI agent then generates a response based on the

user's question, the player's history, and the available platforms. Some suggestions may come from outside the database, showing the AI agent's ability to provide broader recommendations when the user asks an open-ended question.

The Shiny app is published through *Posit Connect Cloud* because it allows the app to safely store the environment variables (API key and AI model) required for the AI to function. The app was also tested through Shiny deployment, but the AI component did not work there, so the author used Posit Connect Cloud for the final deployment.

9. Recommendations and Future Directions

Several future directions could strengthen the project. First, the Shiny app could use actual player accounts rather than simulated ones, enabling the database to track real player data. Second, the database could be expanded to include more games, along with their developers, publishers, platforms, release dates, and tags. Lastly, a rating and feedback system could be implemented to allow users to rate games and provide feedback on the recommendations they receive.

10. Conclusion

Overall, this project designed, implemented, and tested a relational database for a single-player video game library. It organizes game metadata, player records, platform ownership, progress data, developers, publishers, genres, and tags through connected relational tables. The database structure was tested and built on PostgreSQL, then converted to SQLite to connect the database to the Shiny application. The app allows users to explore the game library, view player history, generate recommendations, receive gameplay advice, and engage with the

AI agent. Although the project currently uses simulated player records, it provides a functional basis for a larger video game recommendation and advice system.

11. AI Appendix/ Tools Used

Game titles, developers, publishers, and release dates were based on publicly available online game information. Furthermore, player records were simulated for demonstration purposes.

Additionally, ChatGPT and Google Gemini were used as assistant tools for coding support, brainstorming, troubleshooting, organizing explanations, and wording for presentations. Google Search and AI Overview helped identify basic game metadata, such as developers, publishers, and release dates.

12. References

Baidu Baike. n.d. “Spider-Man: Miles Morales.” Baidu Baike. https://baike.baidu.com/en/item/Spider-Man:%20Miles%20Morales/1447017.

Batman: Arkham Wiki. n.d. “Batman: Arkham Asylum.” Fandom. https://batmanarkham.fandom.com/wiki/Batman:_Arkham_Asylum.

Cosmocover. 2025. “Clair Obscur: Expedition 33 Launches April 24, 2025.” Cosmocover, January 23. https://www.cosmocover.com/newsroom/clair-obscur-expedition-33-launches-april-24-2025/.

Cyberpunk Wiki. n.d. “Cyberpunk 2077.” Fandom. https://cyberpunk.fandom.com/wiki/Cyberpunk_2077.

D’Angelo, William. 2022. “The Last of Us Part I Remake Details Released.” *VGChartz*, June 9. https://www.vgchartz.com/article/453946/the-last-of-us-part-i-remake-details-released/.

Dead Space Wiki. n.d. “Dead Space (2023).” Fandom. https://deadspace.fandom.com/wiki/Dead_Space_(2023).

Dornbush, Jonathon. 2023. “The Last of Us Part II Remastered Coming to PS5 on January 19, 2024.” *PlayStation.Blog*, November 17. https://blog.playstation.com/2023/11/17/the-last-of-us-part-ii-remastered-coming-to-ps5-on-january-19-2024/.

Game Informer. n.d. “Resident Evil Requiem.” Game Informer. https://gameinformer.com/product/resident-evil-requiem.

GameSensor. 2022. “Marvel’s Spider-Man Remastered—Sales Amounted to Almost $55 Million in the First Month of Release.” GameSensor, December 25. https://gamesensor.info/news/marvels_spiderman_remastered.

Games Press. 2025. “Digital Version of The Rogue Prince of Persia Arrives on Nintendo Switch and Switch 2 on 16th December.” Games Press, December 11. https://www.gamespress.com/DIGITAL-VERSION-OF-THE-ROGUE-PRINCE-OF-PERSIA-ARRIVES-ON-NINTENDO-SWIT.

God of War Wiki. n.d. “God of War (2018).” Fandom. https://godofwar.fandom.com/wiki/God_of_War_(2018).

Harradence, Michael. 2025. “Ghost of Yotei Developer Sucker Punch Productions Looks Set to Remain a Single-Project Studio.” *PlayStation Universe*, October 14. https://www.psu.com/news/ghost-of-yotei-developer-sucker-punch-productions-looks-set-to-remain-a-single-project-studio/.

Haug, Kim. 2019. “Resident Evil 2: Review (2019 Remake).” *Ulvespill*, February 7. https://ulvespill.net/resident-evil-2-review-2019-remake/.

Hill, Kip. 2020. “‘Ghost of Tsushima’ a Playstation Masterclass in Open World Design.” *The Spokesman-Review*, December 7. https://www.spokesman.com/stories/2020/dec/07/ghost-tsushima-playstation-masterclass-open-world-/.

LeBlanc, Wesley. 2024. “Prince of Persia: The Lost Crown Review: A Royal Resurgence.” *Game Informer*, January 11. https://gameinformer.com/review/prince-of-persia-the-lost-crown/a-royal-resurgence.

Makuch, Eddie. 2012. “Batman: Arkham City Ships 6 Million.” *GameSpot*, February 8. https://www.gamespot.com/articles/batman-arkham-city-ships-6-million/1100-6349980/.

Mikaelian, Alex. 2025. “Everything We’ve Learned About Silent Hill F So Far.” *XDA Developers*, April 3. https://www.xda-developers.com/everything-about-silent-hill-f-so-far/.

Robinson, Andy. 2022. “Resident Evil 4 Remake Is Finally Official.” *Video Games Chronicle*, June 2. https://www.videogameschronicle.com/news/resident-evil-4-remake-is-finally-official/.

Romano, Sal. 2024. “Marvel’s Spider-Man 2 Coming to PC on January 30, 2025.” *Gematsu*, October 18. https://www.gematsu.com/2024/10/marvels-spider-man-2-coming-to-pc-on-january-30-2025.

Skovlund, Eira. 2018. “Dark Souls: Remastered: Review.” *Ulvespill*, May 30. https://ulvespill.net/dark-souls-remastered-review/.

Steam. n.d. “Batman: Arkham Knight.” Steam. https://store.steampowered.com/app/208650/Batman_Arkham_Knight/.

Steam. n.d. “DOOM: The Dark Ages.” Steam. https://store.steampowered.com/app/3017860/DOOM_The_Dark_Ages/.

Steam. n.d. “SILENT HILL 2.” Steam. https://store.steampowered.com/app/2124490/SILENT_HILL_2/.

Ultimate Pop Culture Wiki. n.d. “The Witcher 3: Wild Hunt.” Fandom. https://ultimatepopculture.fandom.com/wiki/The_Witcher_3:_Wild_Hunt.

VideoGameGeek. n.d. “Prince of Persia: The Forgotten Sands.” VideoGameGeek. https://videogamegeek.com/videogame/71860/prince-of-persia-the-forgotten-sands.