# CISC 322

## Assignment 3 Report
## **ScummVM: Enhancement Architecture**


**Group: BileBytes**

| | |
|---|---|
| Brandon Kim | 21bmkk@queensu.ca |
| Ewan Byrne | 21emb10@queensu.ca |
| Owen Sawler | 21ors3@queensu.ca |
| John Li | 21jl263@queensu.ca |
| Gavin Yan | 21gly@queensu.ca |
| Norah Jurdjevic | 20nrtj@queensu.ca |

**Abstract**

We propose enhancing ScummVM with a game browsing feature to improve user experience by centralizing game discovery and access. This includes GUI updates to display ScummVM compatible games with metadata, reviews, and links to external sites for purchasing or downloading. Users will be able to search and filter games by attributes like genre and engine. The enhancement introduces minimal architectural changes, focusing on GUI and networking components while preserving modularity. While peer-to-peer and repository styles were considered, the new component will use client-server architecture to best address usability, maintainability, performance, and security. Key non-functional requirements include efficient data retrieval and seamless integration without impacting ScummVM's core systems. This feature aims to simplify game discovery for users and streamline updates for developers, enhancing ScummVM's usability and community value.

**Enhancement Description**

Our proposed enhancement is a game browsing page within the ScummVM application. From the main launcher screen, at the bottom of the Game Library, a browse button would be available. Clicking this button would bring the user to a new Browse menu, which would display all the games that can be played in ScummVM. Upon selecting a game, information relevant to the game would be displayed, along with reviews and a button to download or buy the game. The download/buy option would redirect the user to an external site where the game can be downloaded or purchased from the original creators of the game. This is to avoid issues with payment. Additionally, users would be able to search for game titles, or filter games by tag. Some possible tags include the game engine that the game uses, as well as the genre of game.

This enhancement would be beneficial to avid ScummVM users looking for more games to play. It offers a more convenient way of finding compatible games by gathering download links into one place, within the app. The search and filtering features would further this convenience by allowing users to directly search for games that they know they have an interest in, or doing a more broad search for specific features or characteristics that they would like to see in the game.

**Effects of the Enhancement on Key NFRs of the System**

The enhancement focuses on introducing a game browsing page into the ScummVM application, which will have certain effects on key non-functional requirements (NFRs), such as maintainability, evolvability, testability, and performance of the whole application.

The enhancement does not significantly affect the maintainability of the core game engine subsystem and its dependencies, as it is implemented as a separate module. However, the

browsing page itself introduces new maintainability considerations. One example involves maintaining accurate and up-to-date external links, which will require periodic updates from developers or automated validation systems. Additionally, adding or modifying game metadata, such as descriptions or tags, should be streamlined for contributors, particularly as ScummVM is open-source and has distributed contributors.

When it comes to evolvability, the modular design of the enhancement ensures that it can evolve independently of the game engine or other core components. Features such as advanced filtering, personalized recommendations, or integration with more robust game repositories can be added in the future without requiring major architectural changes.

The browsing page can be isolated for testing without affecting the game engine or other subsystems. Automated tests can validate key functionalities such as searching and filtering games, and ensuring external links work as expected. However, testing external links may require specialized integration tests to ensure external systems remain accessible.

The enhancement would have little to no effect on the performance of the core game engine subsystem and its dependencies. Regarding the browsing page, the enhancement is designed to retrieve and display game information efficiently, ensuring minimal latency for users when searching or filtering. The choice of a client-server architecture supports this performance goal by centralizing resources and enabling fast data retrieval. However, network dependency introduces a potential bottleneck if the server experiences downtime or high traffic. Implementing caching strategies for frequently accessed data could mitigate these risks and further enhance user experience.

**Effects of Enhancement on Existing Architecture**

The proposed enhancement to ScummVM would have significant implications across the User Interface Layers. At a high level, the Application Layer (`base/`) would need to incorporate logic for transitioning between the existing Game Library and the new Browse menu. This layer would be responsible for initializing the browsing functionality, handling user interactions such as clicking the “Browse” button, and managing the flow of metadata retrieval. Additionally, the Application Layer would oversee communication between the Browse menu and other components, ensuring seamless integration with the rest of the system.

The User Interface Layer (`gui/`) would see the most substantial changes. A new Browse menu interface would need to be developed to display game titles, descriptions, tags, and reviews, along with options for searching and filtering games. This interface would also include links for downloading or purchasing games, redirecting users to external sites. The design and implementation of these new GUI elements would require extensions to the existing menu and widget components within the GUI subsystem. Furthermore, input handling for searches and

filters would be added, with results dynamically retrieved and presented to users. This makes the User Interface Layer central to the functionality and usability of the new feature.

To support networking features, such as fetching data from the existing database of available games, utilities related to networking will be stored in the `common/` component. If a large number of files are required for this, a separate component could be broken off as `networks/`, similar to other utilities. In addition to networking utilities, we would also need utilities to handle parsing, caching, and managing the game metadata efficiently. With these utilities in place, the GUI subsystem will simply make calls to the utilities to gather the required data to be displayed.

Overall, this enhancement would primarily increase the complexity of the GUI component, and would require adding more utilities to support data fetching and data handling. With the Game Engine Layer being largely unaffected, the new browsing feature would integrate into ScummVM's architecture without compromising the modularity of its design. Careful attention to performance and maintainability would be critical, especially for ensuring that metadata retrieval and filtering operations do not degrade the user experience.

**Alternative Architectural Styles for the Enhancement**

One of the alternate architectural styles that can be used is the peer-to-peer style. Rather than relying on a central server storing the links and information about the game, a catalog would be created where each user would maintain a piece to update. This design would help make a cheap but effective way to distribute games between several people without the reliance on a server.

However, this approach would have a variety of issues, one of them being security. Without a way to validate the authenticity of the game, it can lead to potentially unsafe or malicious links being shared. Furthermore, legal trouble can arise if a game were to be distributed, even unintentionally, through the network. Finally, if a person were to uninstall ScummVM or go offline, their portion of the catalogue would become unavailable, potentially removing several games from the network. Things like data replication could help mitigate such occurrences from happening, but inevitably data fragmentation and loss would occur.

Another alternate architecture that could be used is the repository style. This style would function through a centralized repository that is a shared source of information that the community can use to upload and share games. Users can upload data about the game, including store links, descriptions, and updates from external sources. The repository ensures that all ScummVM clients can access consistent and up-to-date game information.

The issue that would arise due to this style is that if high columns of upload and queries were to happen could be bottlenecked and slow the repository down. This can be mitigated with more repositories, but it can add complexity to the overall system. Additionally, much like the peer-to-peer style, there would be the issue of security, where links uploaded on to the repository could be unreliable and could be dangerous to the user. Legal issues would also arise because of this style, as games shared on the repository can have unauthorized content. Furthermore, it would require more heavy moderation, resulting in being more expensive as moderators will be needed to remove or alter links that prove to be malicious or poor quality.

## SAAM Architecture Analysis

*Stakeholders*

There are two major stakeholders for our proposed enhancement, ScummVM users and its developers. The game browsing page serves primarily to enhance the user's experience of the application by providing information on and access to the games supported by ScummVM. Because of this, users of the ScummVM application are major stakeholders. When considering any new features or enhancements to an application, the people responsible for implementing these features are always major stakeholders in the project.

*Important Non-Functional Requirements (NFRs)*

For users of the application, the most important NFR is performance. This NFR focuses primarily on how quickly results are returned when searching or filtering with the game browser.

For developers of the application, the most important NFRs are modifiability, maintainability and security. The list of games displayed in the game browser should be easy to update, so games can be added, removed or have their details modified. This is especially important considering ScummVM is open source, and many different developers may need to update the list with their contributions. Each game's download/buy button redirects to an external site, so it is important to ensure these links lead to the appropriate pages. These external links should be routinely checked to ensure they are performing as expected. Additionally, the inclusion of external links can create a potential security risk. Including a form of authorization whereby only trusted developers can make these links available to users would provide a safeguard against this risk.

*Comparison of Approaches*

Our first approach focuses on a client-server style implementation for the enhancement. The use of a centralized server would make its performance reliant on a network, meaning if the network were to go down, the enhancement would no longer be accessible. Other than this consideration, this approach has no particular effect on the performance of the search and filter within the game browser. A client-server style implementation would be helpful for both the maintainability and modifiability concerns. Having a centralized server ensures that all external

links can be monitored and verified by the same system while also making it easier for developers to add new or modify existing components. As with any network-dependent approach, there are additional security concerns like the risk of a breach. However, this solution enables monitoring and authorization features for developers to ensure malicious links are not shared to users.

We considered both peer-to-peer style and repository style approaches as alternative options for our enhancement. The decentralized nature of a peer-to-peer style implementation results in easier adding or editing of information, supporting the modifiability of the enhancement. However, this also results in difficulties with maintenance, as a lack of centralized information makes it difficult to moderate effectively. This means that continuously verifying the safety and accuracy of external links will be more difficult. Additionally, this approach presents the same network security and dependence concerns as the client-server style, with additional reliability concerns as the approach depends on a number of individual 'peers' remaining active. For the repository style, the centralized information supports the maintenance required for external link monitoring, similar to the client-server approach. While this approach also allows for multiple developers to access and edit the centralized information, too many demands of the repository can have a noticeable effect on its performance. This style does not share the same network security or dependence concerns as the other approaches and allows for authorization to ensure the safety of external links.

*Choice of Approach*

After careful consideration of our approaches, we decided a client-server style implementation is the best fit. All approaches share similar security risks and features with respect to the authorization of links, and while the use of a network produces some additional risk, it ensures a better performance for the user. Because of the importance of a user's experience of the enhancement, and its potential to harm performance, we decided not to use a repository style approach. While both the peer-to-peer and client-server approaches are similar in security and modifiability, the peer-to-peer style does not support the consistent monitoring and maintenance of external links the way a centralized server would. Overall, a client-server style implementation better meets the needs of both the users and developers of the ScummVM application.

## **Plans for Testing**

To ensure the seamless integration of the game browsing enhancement into ScummVM, a robust and multifaceted testing strategy will be implemented that specifically addresses the unique interactions and potential challenges introduced by the new feature. The testing approach will focus on intricate connections between the Base component, GUI layer, and new Network utilities, with particular emphasis on validating the complex interactions involved in game metadata retrieval, searching, filtering, and external link handling.

White-box testing will be crucial in examining the internal mechanics of the game browsing enhancement, with a targeted approach to unit testing key functionalities. Using frameworks such as *Google Test (gtest)* developers will conduct granular tests on critical components, such as the metadata parsing system, search algorithm, and filter mechanisms. These tests will specifically validate the nuanced behaviors of the new feature, including handling partial game titles, managing multiple filter combinations, and ensuring robust performance when dealing with large game databases. Performance profiling tools like *Valgrind* and *gprof* will be crucial in monitoring the enhancements impact on system resources and identifying computational bottlenecks, ensuring that the new game browsing enhancement does not compromise ScummVM's existing performance standards.

System-level and end-to-end testing will provide a holistic validation of the Game Browsing enhancement, replicating complete user journeys from initial application launch through selecting the "Browse Game" button, game browsing, filtering, and potential download processes. These comprehensive tests will simulate real-world scenarios, such as searching for games across different platforms, applying complex filter combinations, and handling network-dependent interactions. Special attention will be paid to the transition between the existing Game Library and the new Browse Games menu, ensuring seamless integration within ScummVM's architectural framework.

Black-box testing will complement this by addressing edge cases, unexpected interactions, and potential vulnerabilities specific to the game browsing functionality. This will include rigorous testing of boundary conditions scenarios, such as searches yielding no results, handling incomplete or corrupted game metadata, and validating the robustness of external link redirections. Security-focused tests will be implemented to protect against potential risks introduced by the network interactions, including validation of URL parsing, protection against potential injection attacks, and secure handling of external links. By simulating various potential user interactions and system states, black-box testing can ensure complete robustness of the game browsing enhancement across multiple configurations and unpredictable conditions.

By implementing this comprehensive testing strategy, we aim to mitigate potential known and unknown risks associated with the game browsing enhancement, including performance degradation, unexpected component interactions, and potential security vulnerabilities. The testing approach goes beyond simple functionality validation and will not only ensure the functionality of the new feature but also confirm that it integrates seamlessly with ScummVM's existing architectural framework, preserving the application's reliability and core standards.

**Risks Due to Enhancement**

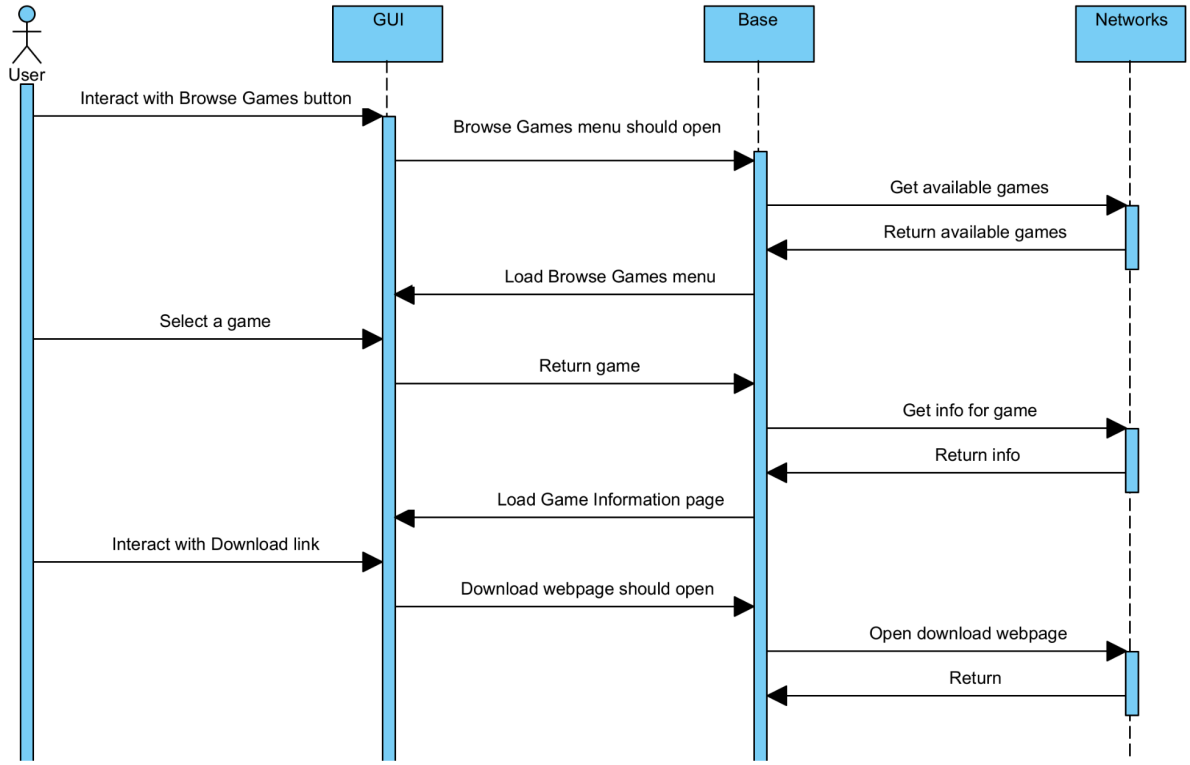
Due to our choice of enhancement, ScummVM would become reliant on a server to display information, reviews, and where to redirect the user. If at any point, ScummVM's server goes down, then the user will have to resort to how they were originally able to locate and obtain their games. Additionally, because of the nature of this enhancement, ScummVM would also be dependent on the redirected website to be active. This can also cause an issue of reliability, as both sides will need to be active in order for this enhancement to remain effective. If at any point ScummVM or the redirected web page goes down, the enhancement will be ineffective.

Additionally, ScummVM would need to acquire some form of funding as the servers will need to be maintained. Because the user is redirected to another web page, money that is spent on that website will not be sent to ScummVM, but rather to the developers and publisher of the game. This will result in the server being heavily reliant on receiving funding from the stakeholders. Furthermore, because ScummVM fills a niche market of running old adventure style games, receiving funding to maintain the servers could prove to be challenging.

Furthermore, because a server is being utilized, security would be a risk that would need to be considered. This is because if the server were to be breached or the incorrect link was used, the user could be redirected to a potentially malicious website. This would require the server to undergo consistent maintenance and have enough security to prevent such a thing to occur. However, this would result in it being more expensive as it would need to be updated to remain safe and reliable.

## Use Cases

**Use Case 1: Downloading a game through ScummVM Game Browser**



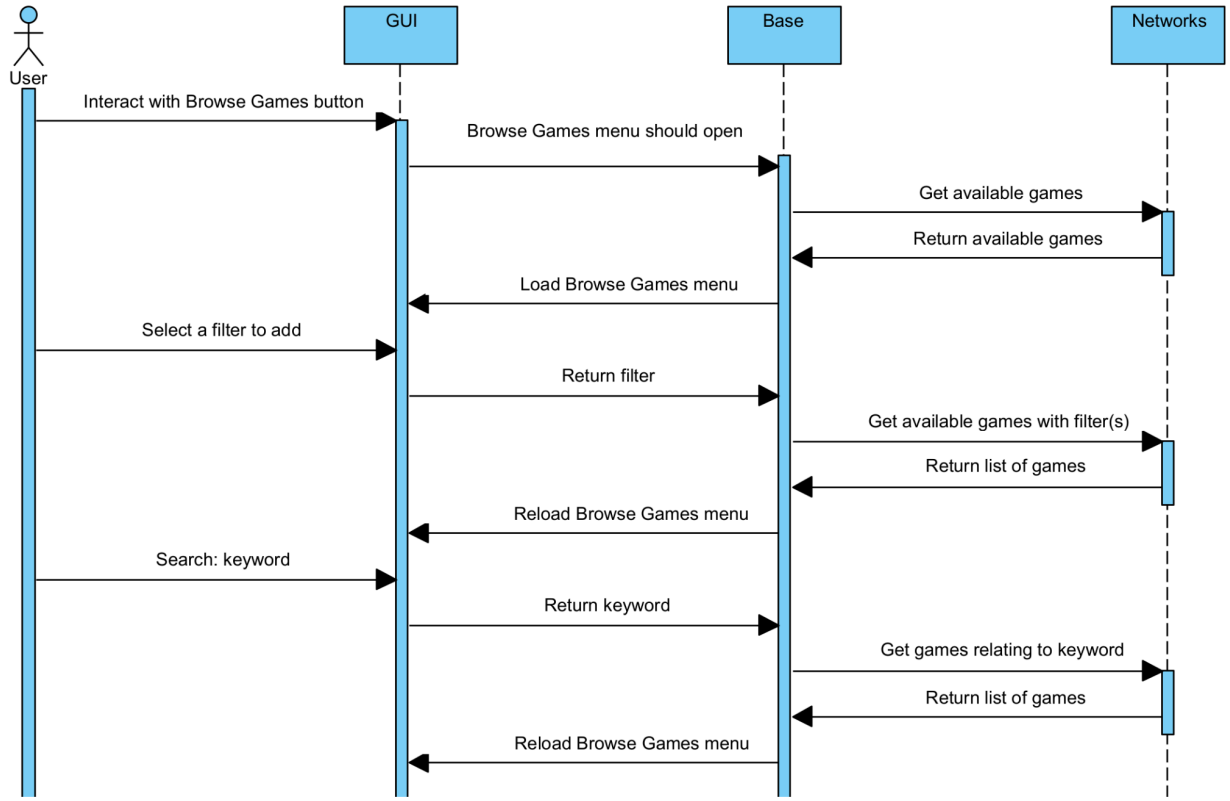

The first sequence diagram showcases the most important part of the enhancement, which is to download a game through the added menu. We will assume that the game launcher is active at the start of the diagram, since this is where the added menu will be accessed from. This means, according to our concrete architecture derivation, that the Base component is running the `scummvm_main()` method, and the GUI has loaded the game launcher. Additionally, to avoid complexity in the diagram, we will assume that the user doesn't use the filter or search features to find the game that they wish to download.

To open the new menu, the user interacts with the new "Browse Games" button that would be present in the ScummVM launcher. The GUI will alert the Base component that the Browse Games menu should be opened. The Base will then make a call to the new Networks component, since the database of available games needs to be accessed. Then, the base will get the GUI component to load the new "Browse Games" menu, which displays a list of games compatible with ScummVM.

Once they are on the Browse Games menu, the user will be able to select games to see more information. The GUI will return the selected game to the Base component, which will make another call to the Networks component. The Networks component will return the information for the specific game that was selected. The base will prompt the GUI to load an interface containing game-specific information, such as a description and user reviews. A

download link will also be present, along with a price if applicable, which directs the user to an external website where the game can be purchased or downloaded. When the user clicks on the download link, the base component will make a call to the Networks component, which will load the webpage related to the game in the user's default browser. From here, ScummVM is no longer in charge of what the user does to download the game.

**Use Case 2: Searching and filtering for games in the ScummVM Game Browser**




The second sequence diagram shows how users can use the extra features in the game browser to find games more conveniently and effectively.

The user will first interact with the new "Browse Games" button, which will cause the sequence of calls to load the Browse Games menu to occur as in the previous use case. The user will then be able to search for specific titles, or apply filters to their search. This can be done in any order, but filters will be covered first in this example. The user will be able to add a filter by interacting with a GUI element, such as a checkbox. Upon interacting with the corresponding GUI element, the Base component will make a call to the Network component to fetch all the games that fit into the filters that have been added.

The user will also be able to search for games by title. The Browse Games menu would contain a text box where the user can type text, then a search button which the user can interact with. When the user interacts with the search button, the Base component will be notified, and

will then send a request to the Network component to get the games with relevant titles, while still taking the filters into account.

## Conclusion

Through carefully considered architectural design and analysis, we proposed a game browsing feature as a new enhancement which stands as a strategic approach to expand ScummVM's user experience. Our investigation focused on introducing a game browsing feature which addresses key user experience challenges such as downloading, searching, and filtering a game from the game browser while maintaining the system's fundamental architectural integrity. The proposed enhancement primarily impacts the User Interface and Application layers, introducing networking utilities and GUI components without compromising the core software's modularity. Through a rigorous Software Architecture Analysis Method (SAAM) evaluation, we systematically examined multiple implementation approaches, ultimately selecting a client-server style that strategically balances performance, maintainability, and security considerations. By carefully analyzing stakeholder needs and architectural implications, we demonstrated the feature's potential to extend ScummVM's utility while preserving its existing architectural principles. Our comprehensive testing strategy—encompassing white-box, black-box, and end-to-end testing—provided a robust framework for mitigating potential risks associated with network dependencies and system interactions. This report provides a detailed exploration of the proposed enhancement, from its conceptual foundations to practical implementation considerations. We have illustrated how the new feature can potentially attract new users and enhance the ScummVM community's experience, while maintaining the application's core architectural extensibility.

## Lessons Learned

An iterative process, where the architecture for the enhancement is compared against the concrete architecture, then updated to fit better, proved very helpful in determining the optimal architectural style to use. This process can be easily done for multiple different architectural styles, since not many iterations are required to get a good idea of how it will integrate into the concrete architecture, but it provides much more insight into the optimal way to integrate the enhancement following a specific architectural style into the project.

We found while working on the project that doing research individually and then discussing our findings relating to the enhancement architecture, in-person, was an effective way of maximizing the generation of ideas while minimizing confusion later on while diving deeper into the architecture and writing the report. Working independently on different sections of the report while maintaining communication with other people working on related sections was a good way of ensuring that the architecture remained cohesive, despite working remotely and independently.

## Data Dictionary

Open Source: Software with freely available original source code which may be redistributed and modified.
URL: Address of a webpage

## Naming Conventions

SCUMM – Script Creation Utility for Maniac Mansion
GUI – Graphical user interface
SAAM – Software Architecture Analysis Method

## References


- Newman, J. 2020. What is SCUMM?, SCUMM reference guide :: Introduction. Retrieved from https://www.scummvm.org/old/docs/specs/introduction.php
- ScummVM API Documentation. Retrieved from https://doxygen.ScummVM.org/index.html.
- ScummVM Github Repository. Retrieved from https://github.com/ScummVM/ScummVM/blob/master/README.md.
- ScummVM Wiki. 2023. Developer Central. Retrieved from https://wiki.ScummVM.org/index.php?title=Developer_Central.
- ScummVM Wiki. 2023. HOWTO-Engines. Retrieved from https://wiki.scummvm.org/index.php?title=HOWTO-Engines.