CISC 322/326: Architectural Enhancement of ScummVM

Ryan Jacobson
21rkj6@queensu.ca
20356136

Evan Kreutzwiser
21ejk12@queensu.ca
20352619

Reid Stobo
reid.stobo@queensu.ca
10186536

Jacob McMullen
jacob.w.mcmullen@queensu.ca
20341079

Kashan Rauf
22kr11@queensu.ca
20369391

Mike Stefan
21mgs11@queensu.ca
20342569

Contents

***Abstract*—**

I. Introduction

In the previous two reports, we explored ScummVM's conceptual and concrete architectures. In both reports, all of the 5 primary subsystems of the architecture are the same; these are the Backend, the Engines, the Launcher GUI, the Common Library, and the Games. The biggest difference between the conceptual and concrete architectures are the dependencies, with the concrete architecture having all components dependent on each other with the exception of the Game, which only shares a two way dependency with the Engine. The concrete architectures subsystems and dependencies are shown below by the diagram in Figure 1.

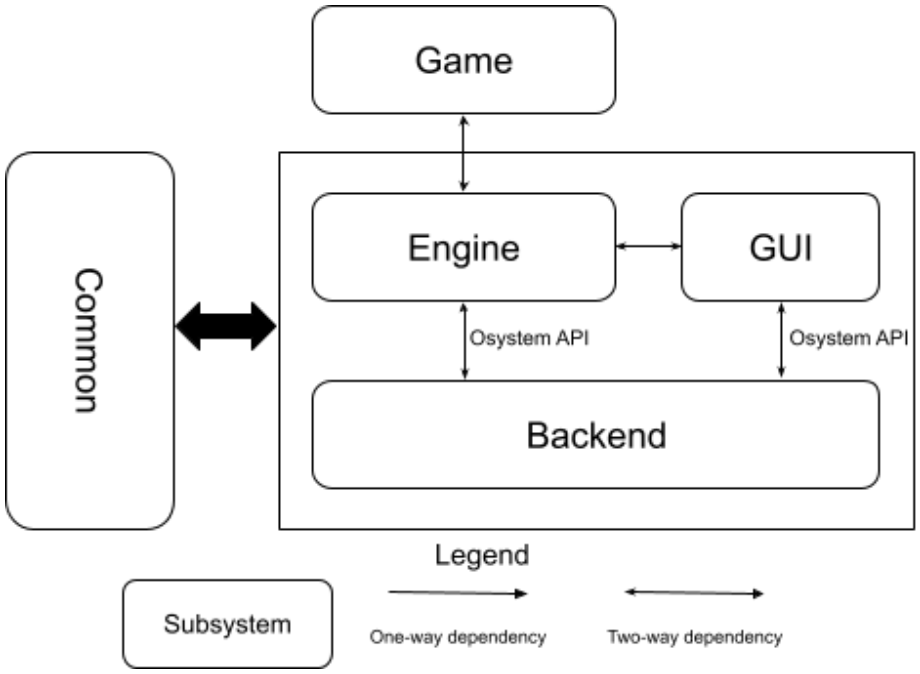


Figure 1: The dependency diagram of ScummVM's concrete architecture from our previous report.

In this report, we propose an architectural enhancement for ScummVM. When a user wishes to play one of the many games that are supported by ScummVM, not only do they need to have ScummVM installed, but they must also have the game files for each game they want to play. ScummVM does not directly provide any of these games, although they do direct users to places where they can purchase them. Our proposal for an architectural enhancement is a game storefront feature that would allow users to buy compatible games directly within ScummVM. More specifically, the games would be purchased from the ZOOM Platform, an affiliate partner of ScummVM that sells DRM-free game (DRM meaning digital rights management). It should be noted that ZOOM Platform currently does not sell all the games that ScummVM supports and that other video game suppliers would also be required to sell all compatible games. However, for the purposes of this report, we will focus on an implementation of the storefront feature with the ZOOM Platform.

- "Storefront" feature that allows users to access the affiliated ZOOM Platform game store to buy games compatible with ScummVM
- Store is accessed from the launcher gui (launcher <-> store)

- Purchased games can be installed from the store with little hassle
- Store uses data from engine to filter games by engine
- Store uses data from backend to only show games compatible with the platform

II. Possible Implementations of Storefront Feature

The storefront feature can be implemented in more than one way with respect to the concrete architecture of ScummVM that already exists. This section will outline two potential implementations from an architectural level.

i. Dedicated top-level component

The figure below demonstrates the first way of implementing the storefront feature; the core functionality will be part of a new component called "Store", which is responsible for opening the store's UI where games can be searched and filtered based on the platform and an engine of the user's choice. Logging in with their ZOOM Platform account would allow the user to securely purchase and install the games they've purchased. The storefront would be opened from the Launcher's gui subcomponent, and games installed through the store would be accessible from the launcher. The Store component would also access the Engine and Backend components to allow the user to search by games that are available on their platform and an engine of their choice.

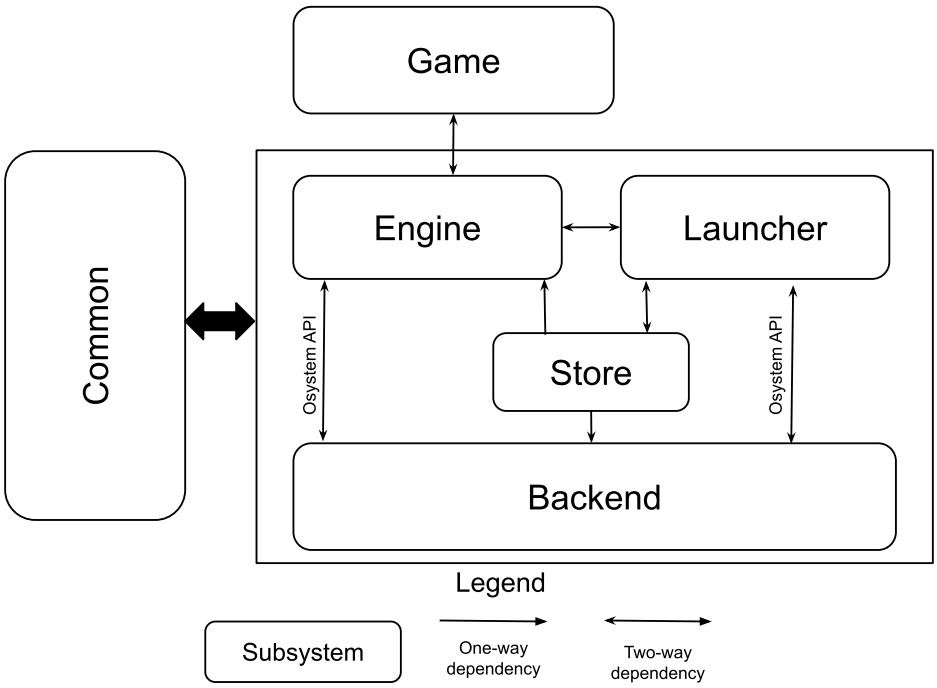


Figure 2: The dependency diagram of ScummVM with an added top-level "Store" component

A benefit of this implementation is that it makes the most logical sense from a top-level. The component is most closely related to the GUI subcomponent as they both involve menus and UIs, especially considering the fact that users add games to ScummVM using the gui despite being part of the Launcher. These subcomponents have widely different functionality since the Store would integrate with a third-party and is entirely dedicated to finding and buying new games to play on ScummVM, which is why distinguishing them is beneficial from a developer's point-of-view. A downside of this implementation

is that this component's placement in the architecture may not align with the layered architectural style of the current system. Since architectural styles are used as a guideline to organize the components of a system, this implementation may violate existing design philosophies used by ScummVM developers.

ii. Integrated into the GUI

The second possible implementation is to integrate the storefront into the Launcher component as a subcomponent of the GUI. The functionality of the ZOOM store would be hardcoded into the GUI component, making use of existing graphical assets as well as new ones developed for the storefront interface. This would prevent the necessity of creating any new dependency relationships, as the functionality could be achieved through the use of existing calls between the Launcher and the Backend using the OSystem API and between the Backend's components. For instance, the GUI's calls to the operating system to dynamically adjust its behaviour (such as during scaling and rendering) would be used by the storefront to determine what games to present to the user that are compatible with their system. Functionality for displaying a user's list of purchased games would be built directly into the component, ensuring the data is always available and eliminating the risk of not displaying due to missed calls.

The greatest benefit of implementing the ZOOM store as part of the GUI component is the preservation of ScummVM's existing layered architecture. Implementation would require less development time and be available for users to access more quickly. As the ZOOM storefront would be part of the core ScummVM codebase and not its own isolated component, better support for the service would be kept in mind during updates to ScummVM which would help prevent the risk of GUI updates breaking the storefront and leading to deprecation. On the other hand, this additional attention to ZOOM could slow ScummVM's development in other areas while they ensure that changes they make are not inadvertently impacting the storefront's functionality—an issue they would be much less likely to face with the storefront as its own component. As well, although integrating the store directly into the GUI would preserve the layered architecture, it would be adding even more functionality to an already busy Launcher component. Ultimately, this approach would save development time up-front and better preserve the existing architecture of ScummVM, but could come at the cost of a bloated backend layer and longer development times in the future.

III. Stakeholders

The operation of ScummVM is made possible by its dedicated group of volunteer developers. As an open-sourced volunteer project, the interests of the volunteer developers are synonymous with the interests of the project as a whole. The Developers are undeniably major stakeholders for any major architectural change made to ScummVM. The development of ScummVM has been made with modular interworking components such that no one implementation dominates the way the system is developed and operates. The backend obfuscates the OS by packaging its functionality within the OSystem API; The Engine component allows for any implementation of an engine by providing an overarching engine class which connects to the rest of the system. The ScummVM developers care most about the *modifiability* and *scalability* of their software. No change to the system should affect the amount of games or operating systems they support. No change should affect or limit the ability for developers to change the way the system grows and evolves. The integration of ZOOM's storefront should cause as little impact to the overall modular structure of ScummVM and not limit future developments or enhancements.

This enhancement is an expansion of the existing relationship between the ZOOM marketplace and ScummVM project. Their increased involvement surrounding the implementation and support for this enhancement makes them a major stakeholder. The ScummVM system rarely, if ever, collects or processes sensitive user data and can be run completely offline without issue. In contrast as an online games marketplace, ZOOM must interact with sensitive user data such as user accounts, contact details, and payment information in order to operate its storefront. This enhancement is also a new facet of their store, and ZOOM will be concerned about their ability to support it long term. Thus, the operators of ZOOM care most about the *security* and *supportability* of this integration into ScummVM. The ScummVM storefront should easily integrate into their existing purchase-processing infrastructure without introducing any major differences that would complicate support for the enhancement. The integration with ZOOM's account, payment and game delivery systems should not create any security risks which expose ZOOM's or any user's data.

ScummVM allows users on many different operating systems, and of varying skill levels, access ScummVM's massive library of games. So there is no doubt that ScummVM's userbase are the last major stakeholder. Any user could expect ScummVM to do its job effectively, without any limitations on when it could be used or on what platform it can be accessed from. Therefore, the userbase of ScummVM cares most about *Performance*, *Availability*, and *Portability*. The implementation of the ZOOM storefront shouldn't introduce significant delays to the process of launching the software or adding new games. A user who doesn't want to interact with the addition shouldn't face any new significant delays to their experience. If connection to ZOOM or the storefront is ever inaccessible, a user's experience shouldn't be affected, and they should still have access to their purchased and downloaded games. The introduction of the storefront should be made with OS in mind, since there may be some hardware or network limitations which prevent certain OS's from accessing it. The enhancement must be integrated in a way which doesn't impact the number of OS's that ScummVM supports.

IV. Evaluation of Proposed Implementations

Implementing the integrated ZOOM storefront as its own separated component has the largest advantage of having the least impact on the rest of the structure of ScummVM. While the component does rely on the functionality of the engine and backend components, as well as being directly integrated with the GUI component, the Storefront can operate in a standalone capacity. This aligns with the developer's interests in modifiability, and scalability, as well as ZOOM's interest in supportability. The developers can work on other components of the system without any concerns of having to "work around" the ZOOM storefront's integration. Additionally, since the storefront would only be relied on by the GUI component, any improvements, changes, or issues surrounding the storefront would have very little impact on the rest of the system. ZOOM would be free to improve or change the way they interact with ScummVM with the knowledge that their impact will be localized to only a single component.

Implementing the integrated ZOOM storefront into the GUI component comes with some benefits to developers. It would likely be much easier to develop then if it were a separate component since it could rely on the existing framework of the GUI. This means a faster development time and a shorter road to production, as least initially. However, its introduction also comes at the expense of the design philosophy of ScummVM. Fully integrating the storefront into the GUI means that the correct operation of the GUI will rely, at least in some part, to the correct operation of the storefront. If the storefront faces

an availability issue, either from store updates or maintenance, there would be concern that it could also affect the availability of the GUI or ScummVM as a whole. No user wants to be prevented from using ScummVM because ZOOM is preforming an update. This implementation also provides new challenges to the modifiability and scalability of ScummVM. Any future updates or changes the ScummVM developers make to the GUI component will have to be made with the storefronts direct integration in mind. Any change made to the GUI must now be discussed with the ZOOM marketplace team to ensure that it wont impact their integration, which introduces delays and scope limitations.

The remaining concerns from ZOOM and the ScummVM userbase about the security, performance, and portability are implementation agnostic, since they don't differ between our two proposed implementations. More information about these concerns will be discussed later in this report in the "Effects of The Enhancement" [V], and "risks" [V.i] sections. Thus, the decision between which implementation to pursue will come down to an evaluation of the affects the enhancement will have on the modifiability, scalability, supportability, and availability. Given the modular philosophy of ScummVM, the ease with which the developers can modify the system, the flexibility afforded to ZOOM, and the low risk of greater unavailability the best option is to implement the storefront as a top-level component. It fits in with the existing structure of ScummVM without committing the team to a grand rework of the GUI. If ZOOM and ScummVM ever decided to end their relationship, this implementation also has the added benefit of being the easiest to remove from system.

V. Effects of The Enhancement

This enhancement will have a significant impact on the evolvability of both ScummVM and the Zoom Platform. By implementing this feature, ScummVM is now limiting the evolutions of Zoom. When proposing future improvements, the Zoom team will have to consider how these changes will affect this integration, and whether their proposed changes will cause a breaking change, this would effectively lock Zoom into whichever stable API system ScummVM uses to tie into their marketplace, account, payment and game delivery systems. Conversely, This change will improve the evolvability of ScummVM. Since ScummVM is focused on using freeware or legally obtained games [1], this integration will significantly expand the library of games which the system could support. This will likely require the develop new engines but with how easy this system will make acquiring the games, that should be no problem for the developer teams to evolve into that space.

Similar to the effects on evolvability, this change will create struggles with the maintainability of the system. With the integration of two systems, any issues or bugs in the implementation, or in the proposed storefront system will require both the Zoom and ScummVM development team to work together to maintain the system. This interworking between the teams could go smoothly and have minimal impact on the maintenance process. That is not very likely, by requiring these two teams, any man-hours spent on maintaining this system will have to align with the schedules and goals of both the team of volunteer ScummVM developers and the professional Zoom developers [2]. This will likely reduce the overall maintainability of the system, but if both teams work together they should be able to maintain a working relationship which keeps this proposal running. This is one of the major risks of our enhancement.

This proposal will have little to no effect on the performance of ScummVM, once the game is purchased and downloaded to a user library, this feature is not used again and the game itself and its corresponding

engine will be able to perform as intended and as they have been before our enhancement. Likewise, this feature will not majorly affect the testability of all features in the existing system. Our proposal will almost entirely be localized to the impacted directories and it's specific use case. There will obviously need to be test cases implemented that test all use cases of our system and its interactions with the gui, engine and backend systems. These tests might require changes to their testing infrastructure as they will now require the tests to be able to mock purchase games from the Zoom Platform, which means the testing server will require internet access. Beyond the addition of new test paths and use cases, our enhancement should have no other impacts on the testability of the system.

i. Risks

One major risk of our proposed system is security. Implementing the storefront feature will inherently introduce a number of security concerns which will have to be overcome. By integrating the storefront, this will require the ScummVM system to have some amount of internet interface, something the current platform does not support. This inherently introduces a major security attack point, something which a malicious agent could take advantage of and cause any amount of damage unless proper precautions and security principles are followed. Beyond opening the software up to internet-based downloads, this will also require ScummVM to integrate with the purchasing and account system on the Zoom Platform, this system will also have to be designed with security in mind, as they must protect all sensitive user data.

As mentioned above, this feature will also pose a risk to the maintainability of ScummVM. Any issues found in our proposal have the possibility of requiring developer effort from either the ScummVM or Zoom teams. This dependency on external teams to perform some maintenance will inherently reduce the maintainability of the system, and will require the goals of the Zoom team to align with supporting the ScummVM integration. This will also limit some improvements that can happen within the Zoom Platform, as they will have to be cautious of new breaking changes in the integration.

This system poses a risk to the portability of ScummVM. The marketplace system will not be available to all platforms which ScummVM currently runs on. Some systems which ScummVM supports do not include the required hardware to connect to the internet, and require the games to be preloaded onto the SD card with the ScummVM image, like the Nintendo DS [3]. This inherently means that our feature will not be portable to all platforms, and the architecture must withstand the exclusion of this feature on a platform basis. This should not be too big of a problem, as the ScummVM backend system is already designed to account for some hardware differences, but our system still must be designed to impact the portability as little as possible, and have its absence not affect the function of the rest of the system.[4]

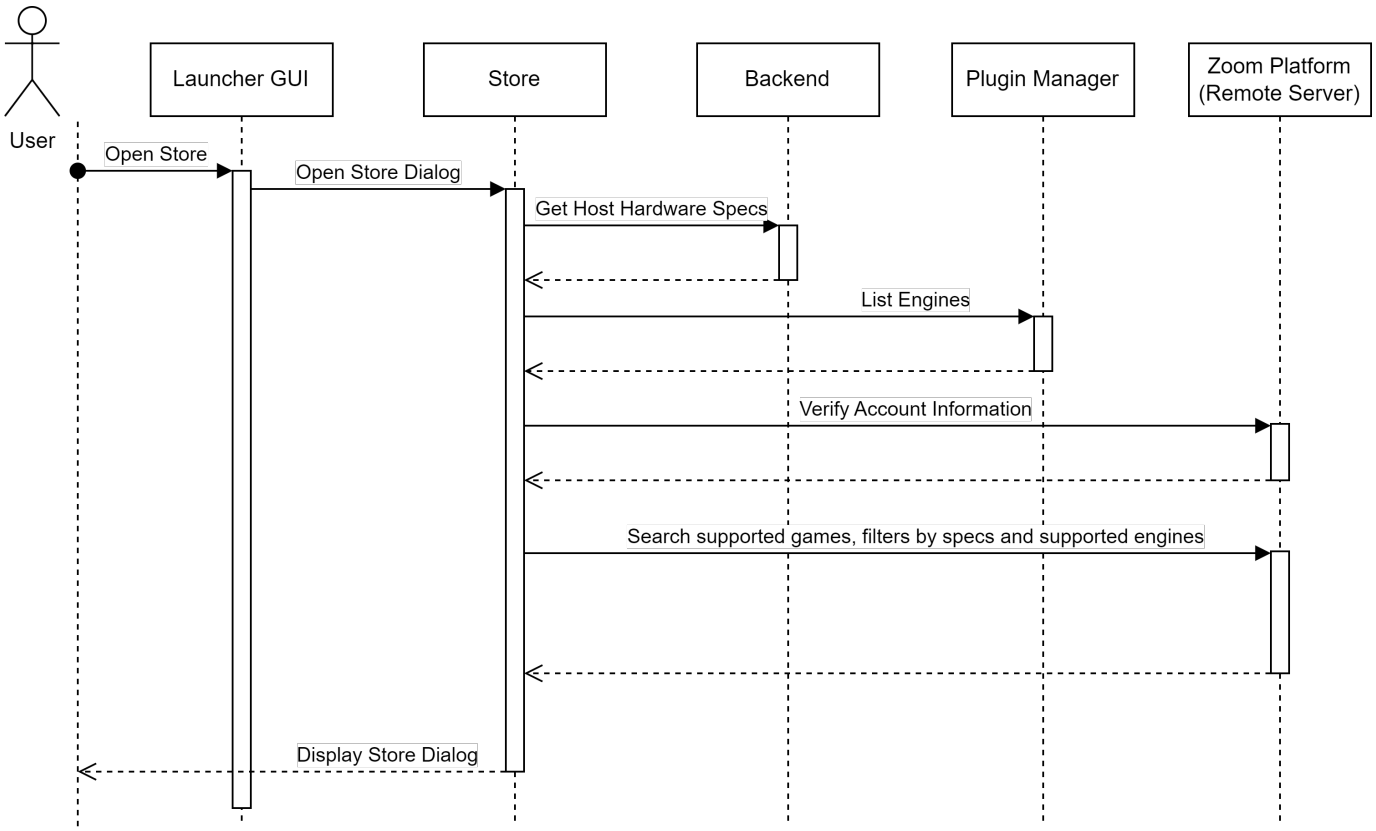
VI. Testing

The testing for our component will come in two parts, the existing system and the enhancement. As mentioned above ScummVM must be functional when using our system, and remain functional on platforms where our system is unavailable. That's where the two stages of our proposal come in. We will have one set of tests for ScummVM where it is running on a system without the required support for the enhancement, and one set of tests where the enhancement is fully operational. Both of these test suits will be comprised of a series of unit tests. The unit tests will test the functionality of each component and the data flow between components. From these unit tests, we will also create a suite of integration tests, and end-to-end tests which will test the complete process for each of our use cases. These tests will call and chain together unit tests for every step of the use cases, and ensure that they function. Once released,

there should be a final regression test suit which ensures that the system is still functional and that the integration between ScummVM and the Zoom Platform is performing as intended.

VII. Use Cases

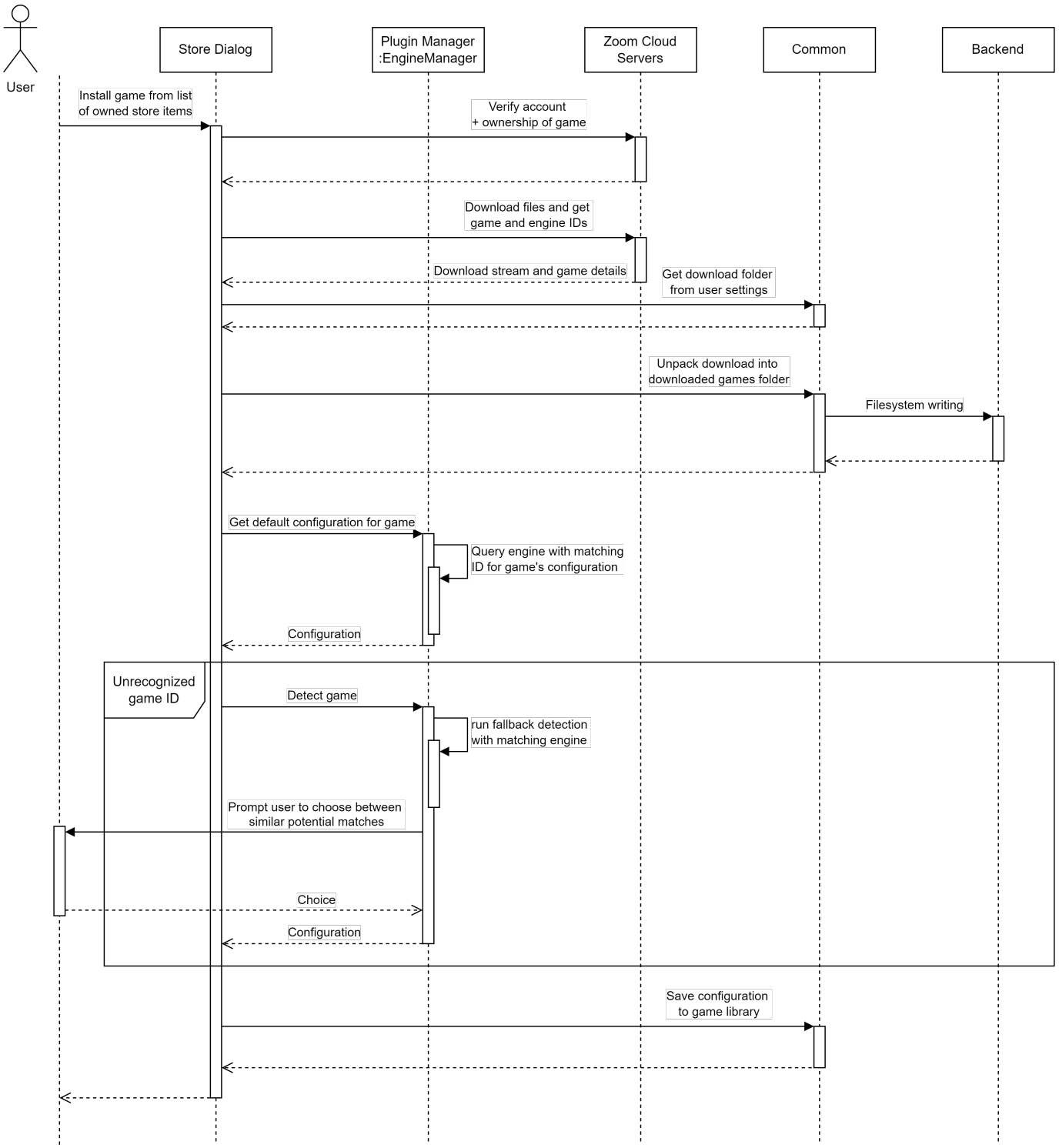
I Finding New Games



A key component of the user experience with a built-in store is to only display games the user can actually play on their device by default. With the addition of a new interface to the backend and a list of installed engines from the plugin manager, the store could filter visible games by the included engines and the power of the user's hardware. Although the majority of the games supported by ScummVM are quite old and computing power continues to improve, ScummVM is designed to be very portable at its core and supports several low-end devices where processing power may limit the range of playable games.

For an optimal user experience, these compatibility filters should be enabled by default, but possible to disable should the user wish to see what else ZOOM has to offer. Combined with additional filters such as the game's genre, publisher, release date, or even game engine, the store can make the process of discovering new games more accessible.

I Installing a Game From the Store



Retrieving games from a central storefront both aids the end user in managing their library of games, as well as simplifying the installation process. Users will be able to easily install purchased games on multiple computers, and reinstall or repair games they may have deleted earlier.

The ZOOM store can track additional information alongside ScummVM-supported games such as the engine and game ID, which significantly eases the process of configuring known games: replacing the hashing and complex checks performed by every engine in the system at once with a quick table lookup to

determine the appropriate ScummVM game configuration. Some games have a large number of different versions and regional variants released that require different configurations and the user may not always know which particular version they downloaded. This method of adding games requires even less user input and knowledge about the game than ScummVM's already considerably user-friendly process - omitting the need even to select which variant of the game was downloaded in most cases - reducing the process to the press of a single button.

However, the existing fallback detection might still be required in the event a user tries to install a game that had support officially added sometime after they last updated their installation of ScummVM. Even this case is optimized by the store-provided metadata since only the individual engine matching the game needs to have its detection process run. This would attempt to let the user play their games even with ScummVM versions that did not explicitly support the game, as long as the engine is present.

References

[1] "About." [Online]. Available: https://wiki.scummvm.org/index.php?title=About

[2] "About us." [Online]. Available: https://www.zoom-platform.com/about

[3] "Nintendo DS." [Online]. Available: https://docs.scummvm.org/en/v2.8.0/other_platforms/nintendo_ds.html

[4] "HOWTO-Backends." [Online]. Available: https://wiki.scummvm.org/index.php?title=HOWTO-Backends