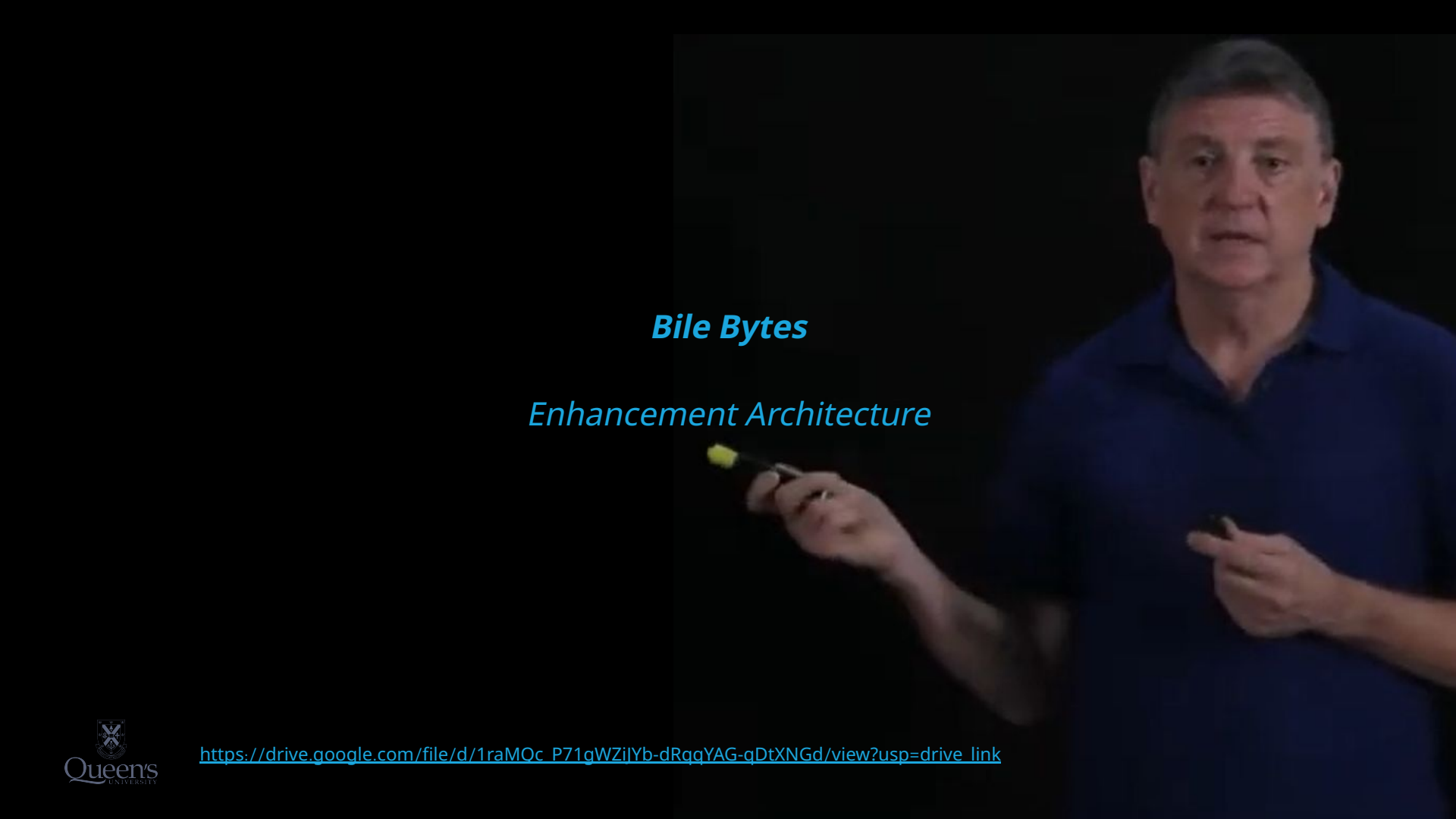


https://drive.google.com/file/d/1raMOc_P71gWZiIYb-dRqqYAG-qDtXNGd/view?usp=drive_link

Bile Bytes: Contributions

- John Li (Group Leader)
  - Abstract
  - Enhancement description
  - Effects of enhancement on system NFRs
- Ewan Byrne
  - Enhancement description
  - Use cases and sequence diagrams
- Brandon Kim (Presenter)
  - Alternative architectural styles for the enhancement
  - Potential risks due to the enhancement
- Norah Jurdjevic
  - SAAM architecture analysis
- Gavin Yan
  - Effects of the enhancement on existing architecture
- Owen Sawler (Presenter)
  - Plans for testing the impact of interactions between enhancement and other features
  - Conclusion

Enhancement Description

- Game browsing menu built into ScummVM
- Includes searching and filtering features
- Improves user experience through a more convenient way to find games

Effects of the Enhancement on Key NFRs

Maintainability
- Minimal effect on game engines
- New browsing page introduces new maintainability considerations

Evolvability
- Modular design of the enhancement ensures that it can evolve independently of the game engine or other core components

Testability
- The browsing page can be isolated for testing without affecting the game engine or other subsystems

Performance
- Minimal effect on the performance of the core game engine subsystem and its dependencies
- The enhancement is designed to retrieve and display game information efficiently with minimal latency for users when searching or filtering
- Network dependency introduces a potential bottleneck

Effects on Architecture

Application Layer (base/):

- Adds logic for transitioning between Game Library and Browse menu
- Manages browsing initialization, user interactions, and metadata retrieval

User Interface Layer (gui/):

- Adds a new browse menu for game metadata display, search, and filters
- Adds download/purchase links and extends existing GUI components
- Handles dynamic input and data presentation

Networking Utilities (common/):

- Fetches and manages game metadata efficiently
- Enables seamless data retrieval for GUI

Alternative Architectural Styles

Peer-to-Peer Style

- Each peer has a portion of the catalogue that they manage and maintain.
- Whatever user shares this catalogue are able to upload links and information about that game that is only viewable between peers.
- Extremely cheap to maintain.

Repository Style

- A shared repository is accessible to those who download ScummVM.
- Users will be able to upload links and information about a game that is viewable to everyone who has ScummVM.

Downsides of Alternative Styles

Peer-to-Peer Style

- Potential Legal issues.
- Data loss and fragmentation will occur.
- Security.

Repository Style

- High query uploads and queries can slow down the repository.
  - Requires more complexity to mitigate
- Security.
- Legal issues.
- Requires heavy moderation.

SAAM Architecture Analysis: Stakeholders

ScummVM Users

- **Performance:** How quickly are results returned when searching or filtering with the game browser?

ScummVM Developers

- **Modifiability:** List of games should be easy to update for addition of new games and removal or editing of existing games.
- **Maintainability:** External links should be routinely checked to ensure they lead to appropriate destination.
- **Security:** Authorization for trusted developers should be included to safeguard against potentially malicious links.

SAAM Architecture Analysis: Approaches

Client-Server Approach

- **Performance:** Dependent on a network.
- **Maintainability:** Centralized information supports consistent monitoring of external links.
- **Modifiability:** Supports access/editing by multiple developers.
- **Security:** Supports authorization for link monitoring, presents additional network security concerns.

Peer-to-Peer Style

- **Performance:** Dependent on a network of peers.
- **Maintainability:** Decentralized nature presents issues for consistent monitoring.
- **Modifiability:** Supports access/editing by multiple developers.
- **Security:** Supports authorization for link monitoring, presents additional network security concerns.

Repository Style

- **Performance:** Too many demands can harm performance.
- **Maintainability:** Centralized information supports consistent monitoring of external links.
- **Modifiability:** Supports access/editing by multiple developers.
- **Security:** Supports authorization for link monitoring.

Plans for Testing: Key Testing Approaches

White-Box Testing

- Granular unit testing of critical components
- Focus on metadata parsing, search algorithms, filter mechanisms
- Performance profiling using Valgrind and gprof

System-Level Testing

- End-to-end user journey validation
- Complex scenario simulations
- Seamless architectural integration testing

Black-Box Testing

- Edge case and vulnerability analysis
- Security-focused network interaction tests
- Boundary condition validation

Plans for Testing: Objectives and Tools

Primary Testing Objectives

- Ensure performance integrity
- Validate component interactions
- Identify and mitigate potential risks
- Maintain ScummVM's architectural standards

Testing Tools:

- Google Test (gtest): Unit testing framework for comprehensive code validation
- Valgrind: Memory profiling and error detection tool
- gprof: Performance analysis tool for identifying computational bottlenecks

Risks Due to Enhancement

Reliability

- Both websites need to be active.

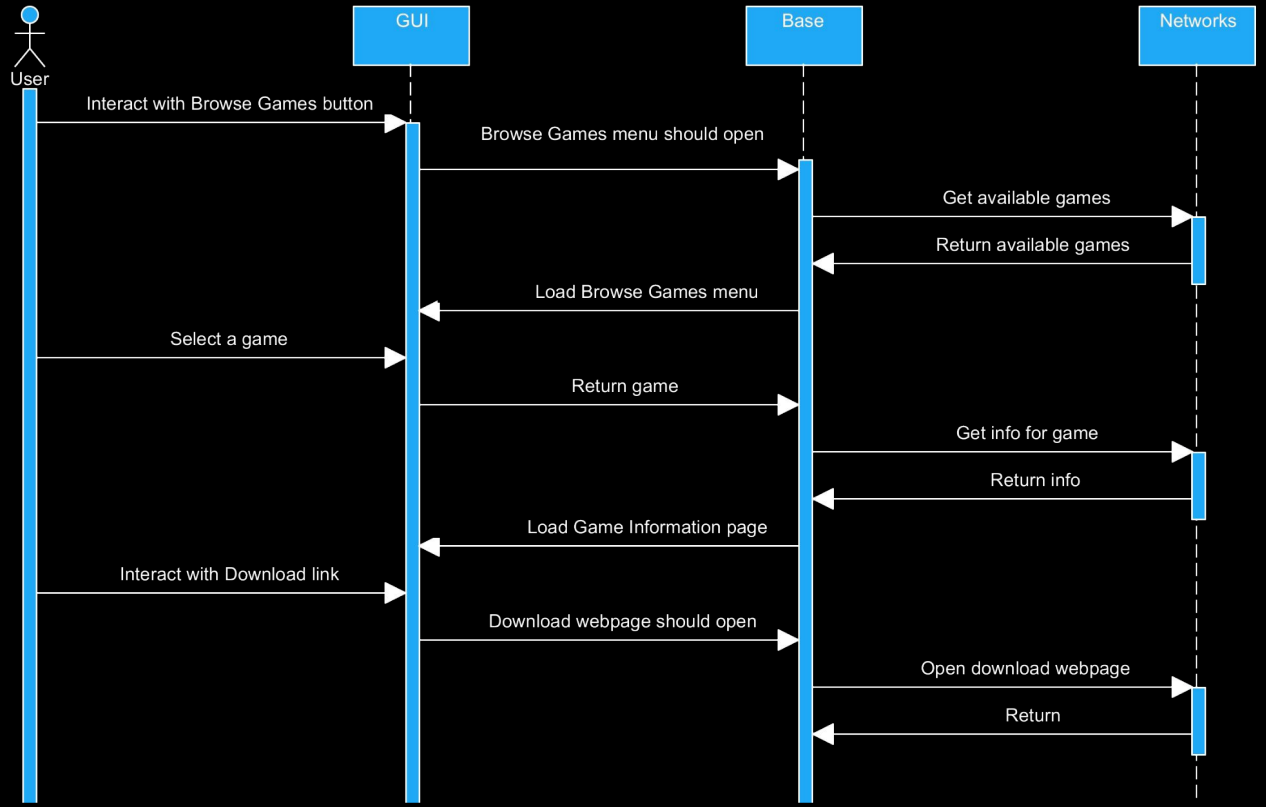
Requires funding

- More difficult due to niche market
- Would need to rely on funding from stakeholders.
- Money would be sent to developers of game, instead of ScummVM.

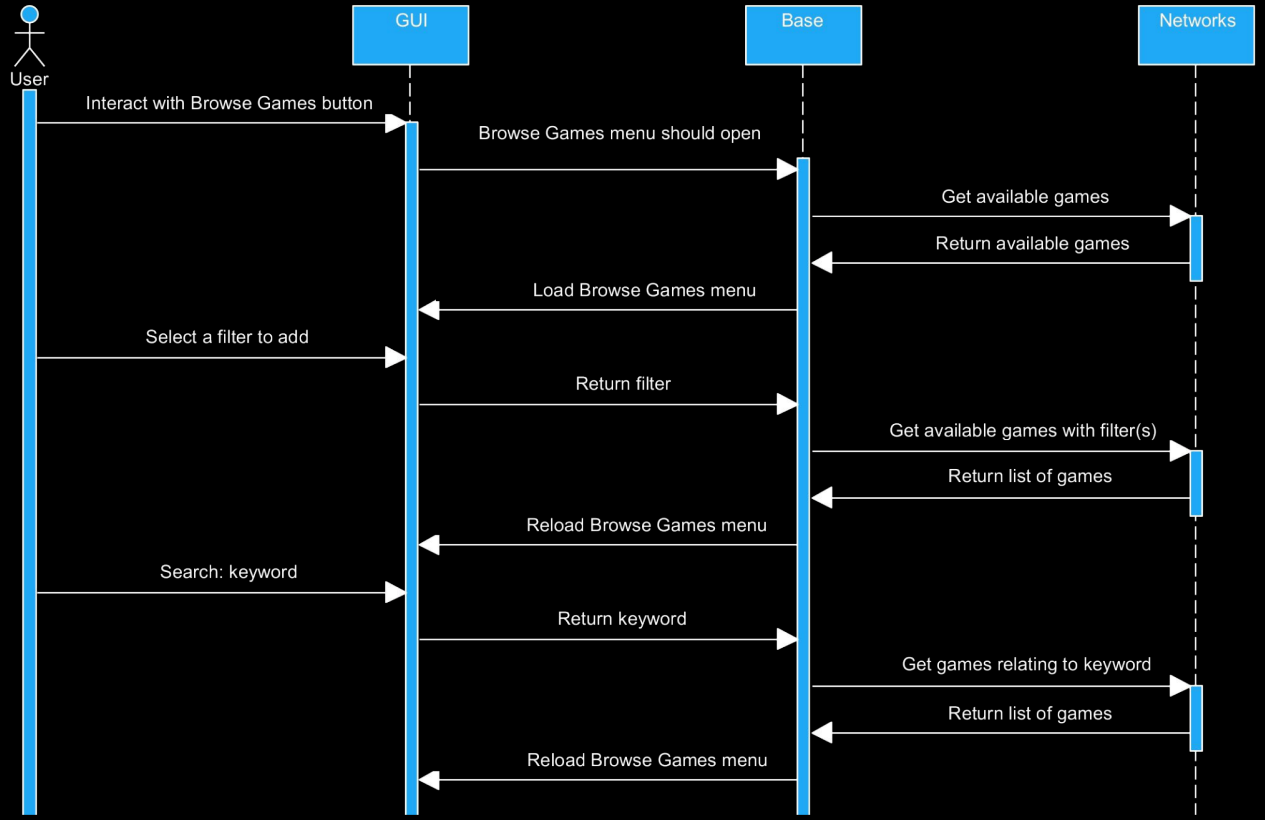
Security would need to be increased.

- Would need to make sure links that are provided are safe.
- Requiring more funding.

Use Case 1: Downloading a game



Use Case 2: Searching and Filtering



Lessons Learned

- Iterative process to improve architecture was effective at finding the best architectural style to use
- Performing research independently allowed maximum generation of ideas, while minimizing confusion
- Working on sections independently while maintaining communication with the group was a good way of ensuring that the architecture remained cohesive

Conclusion

Proposed Enhancement:

- Game browsing feature to improve user experience.
- Key functionalities: downloading, searching, and filtering games.

Architectural Design:

- Focus on User Interface and Application layers.
- Introduces networking utilities and GUI components.
- Maintains system modularity and core architecture integrity.

Evaluation & Implementation:

- Used SAAM evaluation to assess multiple approaches.
- Selected client-server style for balanced performance, maintainability, and security.

Testing Strategy:

- Comprehensive testing: white-box, black-box, and end-to-end testing.
- Mitigates risks from network dependencies and system interactions.