# Kodi - Proposal for Enhancements

Date: December 5, 2023

**CISC 322 - Software Architectures**

Professor Karim Jahed

**Prepared by Group 24:**

**Kobi**

Edward Ng (Team Lead) - 20en3@queensu.ca

Arjun Devnani (Presenter) - arjun.devnani@queensu.ca

Raif Karkal (Presenter) - 20rrk2@queensu.ca

Abdul Moez Akbar - 20ama12@queensu.ca

Danyaal Sahi - 20dhs4@queensu.ca

Drake Li - 19dl53@queensu.ca

**Abstract:**

Kodi, originally known as Xbox Media Center, is an open-source software media player that has been available since 2002. This media player allows users to view every type of media such as movies, TV shows, music, etc. The source of media can come locally, through network media storage or from the internet and was designed to be used with televisions and remote control. However, Kodi has evolved to be available on other devices like mobile phones and laptops. XBMC Foundation developed this app using Python, C, C++, Assembly and Objective-C and can function on Android, Windows, Linux, and iOS. A Watch Party can enhance the viewing experience for users on Kodi. The Watch Party can allow synchronous video playback for users in the same room, allowing family and friends to enjoy any form of media on Kodi without being in the same room. This report discusses the addition of the feature to Kodi and how that impacts stakeholders, interactions of components and the conceptual architecture presented.

## Watch Party Feature

Our proposed enhancement is watch party integration within Kodi. Watch parties are a popular feature incorporating a social aspect to video streaming. Watch parties allow users to select a video and host a synchronized viewing session. Participants are also able to interact while viewing the media in the form of comments, reactions or a chat box.

There are many benefits for both consumers, and Kodi as a media player. First, it incorporates a social aspect to Kodi. Currently, there are no features encouraging or enabling social interaction within Kodi. Watch party integration would foster a community and a sense of connectedness within friends and family, allowing for many new use cases for the software. This is further strengthened by the fact that virtual watch parties do not rely on geographical location. This also increases Kodi's user engagement by increasing Kodi's use cases, as well as encouraging users to spend more time on the software. Moreover, Kodi is already cross-platform with availability on android mac and windows. A Kodi watch party also encompasses cross-platform capabilities further increasing its reach.

Furthermore, many streaming services have already included watch party integration such as Amazon Prime, Disney+ and Apple TV. However, amongst Kodi's direct competitors involving local media organizations Plex is the only software to include any sort of watch party functionality. Kodi implementing watch parties allows them to be ahead of the curve instead of lacking behind. The current state of Kodi has no native watch party support.

In addition, there are no widely utilized third party add-ons adding multi-streaming functionality to Kodi. The most relevant add on is TeamWatch. TeamWatch is a third party add-on that allows users to watch television together. The add-on allows users to send a url and watch videos concurrently. The add-on also allows users to message each other during the video. This add-on can be used as the building blocks for a more comprehensive watch party system described above, allowing synchronized viewing and viewer interaction.

## Interactions

At the core of the Watch Party feature is the user interaction design. The success of this feature hinges on an intuitive and seamless user interface that allows for easy hosting, joining, and management of watch parties. Essential elements include:

**Streamlined Interface:** A user-friendly interface that facilitates effortless navigation, enabling users to initiate or join watch parties with minimal effort.

**Interactive Communication Tools:** Incorporating chat and voice communication features within the watch party interface to foster real-time interactions, thereby enhancing the communal viewing experience.

**Personalized Recommendations:** Leveraging user history and preferences to suggest relevant watch party events or content. This feature could use algorithms to analyze a user's past viewing habits and suggest watch parties hosting shows or movies that align with their interests.
**User Authentication:** Prior to initiating or joining a watch party, users must undergo a process of authentication. This process acts as the gatekeeper, confirming that only legitimate users gain access to the feature.

**Session Management:** Managing user sessions to ensure that actions such as 'Play' or 'Pause' are executed by authenticated users.

## Architecture

### Architectural Design Considerations

The architectural design of the Watch Party feature is critical, especially with the integration of P2P technology. Key considerations include:

**P2P Streaming Synchronization:** Developing a robust system to ensure synchronized streaming across various devices in the P2P network is crucial. This system must manage latency and buffering issues to provide a seamless viewing experience.

**Scalability and Load Balancing:** The P2P architecture inherently offers scalability advantages. Efficiently distributing the load across the network enhances the system's capacity to support multiple watch parties simultaneously.

**Security and Privacy:** In a P2P network, where data is shared among users, implementing stringent security measures is paramount. This includes safeguarding user data and ensuring the integrity and confidentiality of the shared content.

### Integration with Existing KODI Features

For the Watch Party feature to be effective, it must be seamlessly integrated with KODI's existing functionalities. This integration involves:

**Library Compatibility:** Ensuring access to KODI’s vast media library for watch party content selection.

**User Profile Synchronization:** Utilizing user profiles to provide personalized watch party recommendations and settings.

**Plugin and Add-On:** Guaranteeing compatibility with existing KODI plugins and add-ons to maintain the platform's versatility and user experience.

**Conceptualizing the P2P Architecture**
The incorporation of a P2P model is a defining aspect of this project. This decentralized approach, where each participant in a watch party acts as both a client and a server, promises enhanced streaming efficiency and reduced reliance on central servers. The P2P architecture not only optimizes bandwidth usage but also fosters a more robust and scalable solution for the Watch Party feature.

**Maintainability**
Maintainability refers to the ease with which a system can be modified to correct faults, improve performance, or adapt to a changing environment. The introduction of a P2P Watch Party feature could have mixed effects on KODI's maintainability:

**Complexity:** The P2P architecture adds complexity to the system, potentially making maintenance tasks more challenging.

**Modularity:** If implemented with modularity in mind, the feature can be maintained and updated without affecting other parts of the system, enhancing maintainability.

**Evolvability**
Evolvability is the ability of a system to accommodate future changes, either in technology or user requirements. The P2P Watch Party feature impacts evolvability as follows:

**Adaptability:** The decentralized nature of P2P systems could make KODI more adaptable to changes in network environments and user numbers.

**Future Enhancements:** The feature's modern architecture could facilitate the integration of new technologies and functionalities, thereby increasing the system's evolvability.

**Testability**
Testability is the degree to which a system allows for the evaluation of its functionalities. The P2P Watch Party feature influences testability in several ways:

**Testing Complexity:** The decentralized and dynamic nature of P2P networks can complicate testing procedures, as it requires extensive and varied test scenarios.

**Automated Testing:** The feature may necessitate advanced automated testing strategies to efficiently manage the increased testing complexity.

**Performance**
Performance is a critical attribute, particularly in media streaming platforms. The P2P Watch Party feature can significantly affect KODI's performance:

**Efficiency:** By distributing the load across peers, the P2P model can improve streaming efficiency, especially under high demand.

**Scalability:** P2P networks excel in scalability, potentially enhancing KODI's capacity to handle simultaneous watch parties without compromising performance.

**Latency and Synchronization:** Ensuring low latency and synchronization in a P2P network can be challenging, which might impact the performance during real-time interactions.

**A SAAM analysis is performed for the alternatives:**

**Advantages of P2P Architecture**
**Scalability:** P2P networks are inherently scalable, as each new user contributes additional resources to the network. This can lead to improved handling of simultaneous watch parties, distributing the load across a broader base. Additionally, as demand increases, the network's capacity to handle traffic grows organically, potentially reducing the need for significant infrastructure investment.
**Efficiency:** By sharing the distribution of media content, P2P networks can deliver high streaming efficiency. Resources are used more effectively since content is served from peers rather than a central server. Additionally, Bandwidth utilization is optimized as content is streamed directly between users, often leading to faster delivery and reduced lag.

**Disadvantages of P2P Architecture:**
**Complexity in Maintenance:** The introduction of P2P technology can complicate maintenance due to the added complexity of managing a distributed network. Additionally, Diagnosing issues may require more sophisticated tools and expertise, as problems can arise from the network's many interdependent parts.
**Security and Privacy Concerns:** In P2P networks, content and data are distributed among users, raising concerns about data integrity and user privacy. Additionally, implementing security measures that protect against unauthorized access and ensure secure communications between peers is essential.

**Advantages of Client-Server Architecture:**
**Controlled Streaming Synchronization:** Streaming synchronization is centrally managed, potentially reducing latency and buffering issues. The server provides instructions to clients, ensuring a synchronized viewing experience.
**Simplified Maintenance:** The centralized nature simplifies maintenance tasks. Updates and corrections can be applied centrally, potentially making the system more manageable.
**Security:** Security measures are more predictable and controllable on a central server. Encryption and data protection measures can be applied more consistently.

**Disadvantages of Client-Server Architecture:**
**Scalability:** Scalability relies on the capacity of the central server. Handling multiple watch parties simultaneously may require a distributed server architecture, introducing potential complexity.

**Dependency:** The system's performance depends on the central server. If the server experiences issues or becomes overloaded, it can impact the entire watch party experience.
**Bandwidth:** The central server bears the brunt of streaming demands, potentially leading to higher bandwidth utilization and increased infrastructure costs.

**Why Peer-to-Peer over Client-Server**
We chose P2P due to its inherent scalability and operational efficiency. P2P networks dynamically scale with user participation, distributing the load seamlessly and organically expanding capacity as demand grows. This scalability feature not only ensures improved performance during simultaneous activities like watch parties but also mitigates the need for substantial upfront infrastructure investment. The efficiency of P2P networks shines as resources are utilized optimally, with content being served directly from peers rather than relying on a central server. While challenges in maintenance and security considerations exist, the advantages of scalability and resource efficiency make P2P the better architecture, particularly for prioritizing flexible growth and streamlined content delivery.

**The effects of the enhancement on the high- and low-level conceptual architectures**
The effects of the enhancement on high-level conceptual architecture include the following: User Interface, Networking and Media Playback. For UI, the effect is the addition of interactive elements for initiating and joining watch parties. For Networking, the effect is networking modules have to be enhanced to facilitate data exchange among party members. For Media Playback, the effect is the modification of media player modules to support synchronized playback functionality.

The effects of the enhancement on low-level conceptual architecture include the following: User Management, Event Handling, Synchronization and Security. For User Management, the effect is the implementation of user management logic to control access to the watch party features. For Event Handling, the effect is that the architecture needs to be event-driven to capture user interactions for initiating, joining, and leaving watch parties. For Synchronization, low-level modules need to be adapted to manage synchronization data between members. For Security, the effect is the integration of encryption protocols to ensure the privacy and security of data exchange.

**Impacted directories and files are flagged**
The designated directories for flagging encompass various crucial aspects of the system's functionality. These include UI, Networking, Media Playback, User Management, Event Handling, Synchronization, and Security. Within the UI directory, specific files such as `skins/your_skin/`, `addons/skin.your_skin/`, and `gui/Skin.xml` are to be flagged. Networking files, are found in directories such as `network/` and `services/`, as well as `addons/plugin.video.watch_party/`. The Media Playback section requires attention to files like `system/playercorefactory.xml`/, `system/players/`, and `addons/plugin.video.watch_party`. User Management files include those within `userdata/Database/` and `addons/plugin.program.watch_party`. The Event Handling directory is flagged with files from `system/events/` and `addons/plugin.video.watch_party`. Synchronization files include

`system/sync/` and `addons/plugin.video.watch_party`. Security files include `system/encryption/` and `addons/plugin.video.watch_party`

## Use Cases

### Use Cases #1: Creating a Watch Party

Figure 1 focuses on the essential aspects of setting up a watch party and synchronizing playback. The primary user creates a watch party, and this action triggers a series of events involving the Kodi Interface, the Media Playback System, and the Database Management system. The secondary user joins the watch party using the provided ID, and the systems work together to synchronize the playback between all participants.

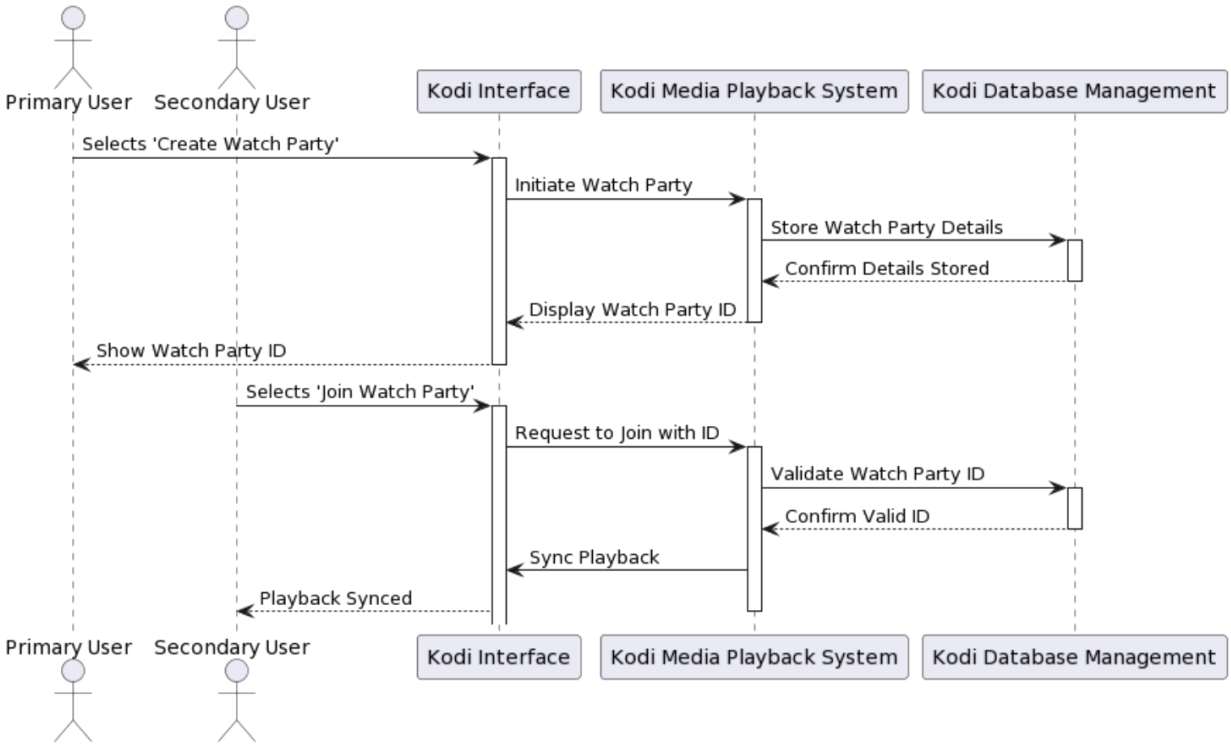


*Figure 1. Use case for creating a Watch Party.*

### Use Case #2: Synchronized Video Playback in a Watch Party

In Figure 2, the primary user initiates playback, which is communicated to the Kodi Watch Party Manager and then to the Media Playback System. The action is logged in the Database Management system. The secondary user's system receives the sync instruction and begins playback, confirming the action through the Kodi interface. Similar to the play action, the pause

command is sent by the primary user, processed by the Watch Party Manager, and executed by the Media Playback System. The pause action is also logged. The secondary user's system synchronizes with the pause command and confirms the pause state.

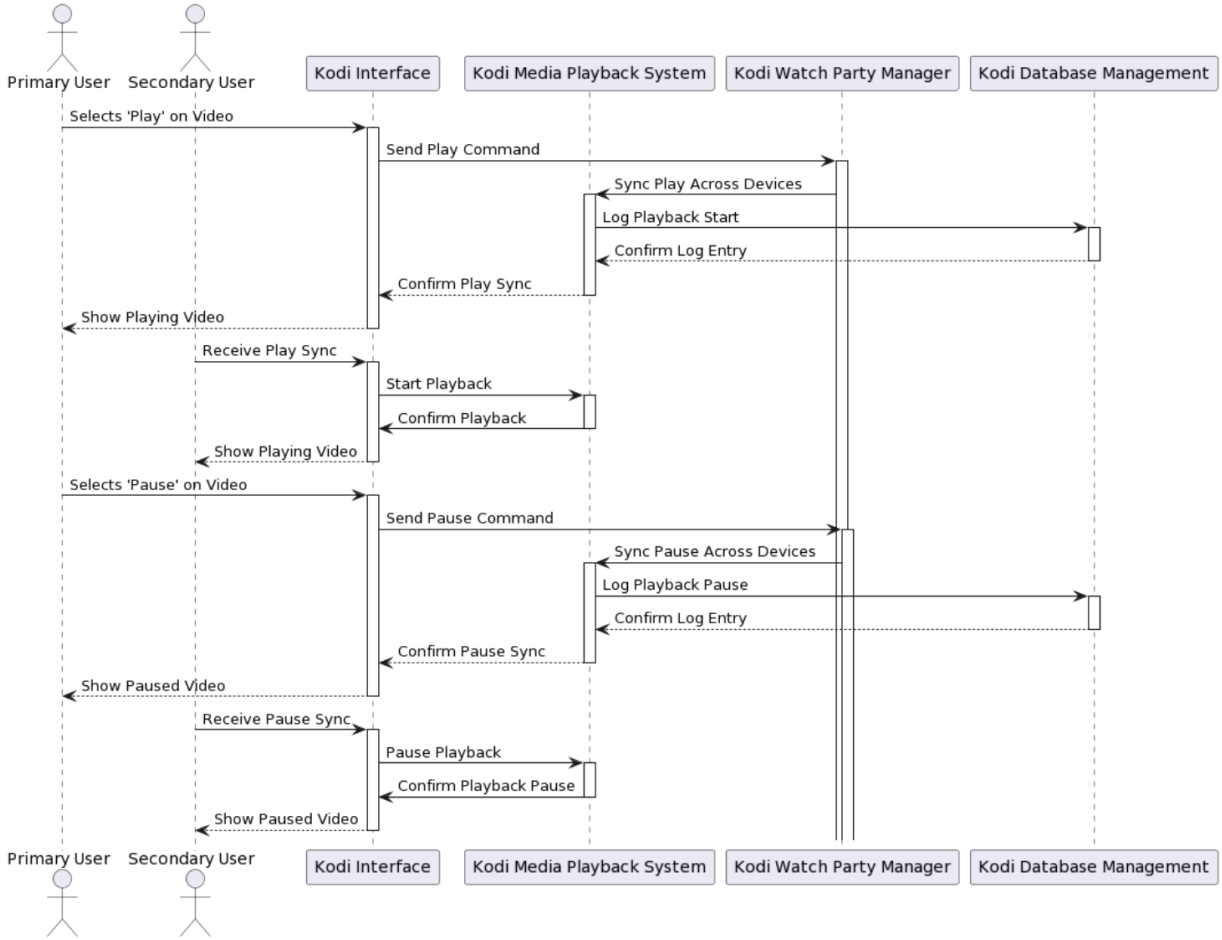


*Figure 2. Use case for synchronous video playback in Watch Party.*

## Major Stakeholders

There are three major stakeholders that are affected with the addition of a Watch Party feature: board members, developers, and users (Dreef et al., 2015). The board members will be affected by how well the feature does and is the communicator between users to developers on how well the feature performs. The developers are the ones who need to develop the feature and have a huge role in making sure the enhancement is rolled out to the public to use. The last

stakeholder are the users. They will be the ones who use the Watch Party feature and the developers are dependent on the feedback for this feature.

In the view of the users, the most important non-functional requirement (NFR) should be the synchronous feature of the Watch Party with less than 1 second delay. In order for the users to have a seamless Watch Party experience where all peers should be viewing the content at the same time, minimal latency should be involved. If there is too much delay between peers watching, the user's may not use the feature anymore because it is hardly synchronized as advertised. The board members would care about the reliability of the Watch Party, such that this feature is required to have a 99% uptime so that users can use this feature whenever they want. This is important for board members because they cannot market something that is constantly unavailable to the users. Finally, the developers have a couple NFRs they care about such as the feature should keep user data encrypted, the feature should run on all Kodi platforms, along with the performance and reliability mentioned above. However, the most important for them would be scalability. The Watch Party feature should be able to handle up to 20 users in a lobby and so the developers need to make sure their servers can handle the traffic. They may even look into other alternatives to client-server architecture and try peer-to-peer architecture so there is no single point of failure.

The two use cases we analyzed above through the sequence diagrams (Figure 1 and Figure 2) are users connecting to another user's Watch Party and video playback while using the Watch Party feature. The use case of connecting to another user's party affects the NFR of scalability because the server needs to be able to handle up to 20 concurrent users in a Watch Party. The stakeholders affected by this are the developers and the users. The developers would need to be able to handle the traffic of users along with ensuring user security and all platforms are compatible. The users are affected by how well they are able to connect with their peers to join the party. The second use case involving video playback affects the same stakeholders. The developers need to ensure the performance of the system is good enough, while users need a seamless viewing experience while being synced with all peers. This means this use case impacts the NFR regarding performance the most. Where all users should be synchronously watching a video with less than a second delay.

Among the two use cases, the most important would be the video playback while in the Watch Party. The main feature of the Watch Party is to be able to enjoy any form of media at the same time as your friends and family while being in different rooms. Therefore, it is essential that synchronous video playback among peers is the most important feature and is focused on when developing the Watch Party.

## Testing

In order to ensure the proposed watch party feature is properly integrated with Kodi without inhibiting current functionality, proper testing must be conducted. There are many tests

conducted when developing software which could be implemented in our proposed scenario as well. These tests, and how they apply to the Watch Part feature integration, are outlined below:

**Initial Testing Scenario - Definitions:**
At the beginning of any testing process, test scenarios are defined. In the proposed scenario, specific use cases and scenarios for the watch party feature would be defined. For example, test scenarios could include starting a watch party, inviting friends to a watch party, playback synchronization, and testing the chat feature. Within the test cases, it is also important to determine and define how the watch party feature will integrate with pre-existing features and UI elements such as playback controls, library management/access, and subtitles/audio options.

**Compatibility Testing:**
Compatibility testing would be used to ensure that the watch party feature is compatible with different media types, codecs, and any streaming services that Kodi supports. We would also have to verify cross-platform and cross-device compatibility that Kodi currently supports (macOS, Windows, Linux, Android, iOS etc.)

**Integration Testing:**
Integration testing would be used to ensure the proper integration of the proposed watch party feature with all of Kodi's current features and functionalities. Some test cases for integration testing would include checking if library updates are updated in real-time during a watch-party, and if playlists and queues are properly managed and updated.

**UI and UX Testing:**
The front-end of the new watch party feature would also have to be tested through UI and UX testing. We would have to evaluate the impact of the watch party feature on Kodi's UI. Testing would include ensuring the UI for the new feature is clear and intuitive, while keeping Kodi's current UI clean and easy to use. Parts of the UX testing would include testing how users navigate between a watch party and other existing Kodi functionalities.

**Performance Testing:**
Performance testing is also a crucial testing process which ties closely to UX. Performance testing would be performed to assess the performance of Kodi while the watch party feature is in use. The tests would check for issues such as latency, playback synchronization problems, and conducting simultaneous resource-intensive processes. This testing process would also involve performing load testing to see how well Kodi handles multiple users simultaneously participating in a watch party.

**Security and Privacy Testing:**
Security and privacy testing would be conducted to ensure secure user authentication and authorization for users participating in a watch party. We would also assess the privacy implications, particularly pertaining to user data sharing during watch parties ensuring it complies with privacy regulations.

**Feedback Collection:**
Since Kodi is a large open source code base with many developers and users, the proposed software would first be released to beta testers from the Kodi community. Feedback would be collected from these users through forums, surveys, and bug-tracking logs within the beta software. This would gather information on experience, issues, and suggestions. Social media and community forums/channels could also be monitored for discussions and feedback.

**Regression Testing:**
Regression testing would be used to ensure that the final feature integration does not negatively impact existing features that were working as intended. This step would determine any unforeseen bugs.

**Iterative Testing:**
Lastly, ongoing iterative testing would be used as updates to Kodi are released. The watch party feature will continue to be tested with each new version in an attempt to catch and address any unforeseen issues that may arise due to changes in the software.

Overall, standard testing methods and practices would be utilized for the proposed Kodi watch party feature enhancement.

## Potential Risks & Mitigations

There are many potential risks and challenges that may arise due the proposed watch party enhancement, as is true with any new feature or functionality in any piece of software. The foreseen risks and challenges are outlined below:

**Security Risks:**
Authentication and authorization issues are one of the possible security risks of the watch party feature. It must be ensured that the new watch party feature has a robust authentication system to prevent unauthorized access. It is imperative that proper authorization controls are implemented to restrict actions within a watch party to only authorized users. Another security risk is in regards to data privacy. When the watch party feature is being developed, the developers must be cautious about the privacy of user data and how it is shared during a watch party. Encryption and secure transmission protocols should be implemented to protect sensitive information.

**Network Security:**
It is important to consider the risk of network security during development to guard against the interception of data that is exchanged during watch parties. Secure communication protocols must be employed to prevent data tampering and “middle-men” attacks.

**Performance Risks:**
It is important to recognize that the proposed watch party feature may put a strain on servers due to playback synchronization and multiple simultaneous connection management. As

suggested earlier, load testing could be performed to ensure the servers can handle the necessary/expected load without any performance degradation to help mitigate this risk. Latency is another issue associated with performance risks. Developers and testers must ensure that real-time synchronization between watch party participators does not affect user experience by introducing latency.

**Maintainability Challenges:**
Implementing the watch party feature may involve complex code which would have to be integrated into the existing codebase. This may make it challenging in the future when making updates and maintaining the code. To mitigate this, it is important that developers follow proper organization and documentation guidelines Kodi has followed thus far.

**Compatibility Issues:**
SInce the watch party feature must work on all platforms currently supported by Kodi, it may be challenging to have the feature be cross-platform and cross-device compatible. Proper testing must be done in order to ensure seamless and consistent functionality on all operating systems and devices currently supported.

**User Experience Challenges:**
One risk associated with new feature introductions is cluttering the user interface and thereby hampering the user experience. Furthermore, introducing a new feature of this magnitude may require users to learn new interactions with Kodi. It is important to keep the learning curve relatively small to ensure user satisfaction. One way to mitigate this would be to provide clear documentation and in-app guides to assist in user adoption of the watch party feature.

**Community and User Acceptance:**
Another risk commonly associated with new major feature introductions is user acceptance. The user base may have varying opinions, so it is important to consider feedback and criticism for ongoing improvements. The benefits of the new feature must also be clearly communicated.

**Regulatory Compliance:**
It is important that the new watch party feature complies with any copyright and licensing regulation to mitigate any risk in this regard. Some aspects to consider for regulatory compliance could be playback synchronization methods and content sharing among users.

**Scalability Issues:**
Scalability potential is a risk to consider prior to implementation. If the feature becomes more popular than expected among users, it may create scalability issues. It is important to plan for scalability and consider various options in advance in case a growing user base is to be accommodated.

**Testing and Quality Assurance:**
In order to best attempt to mitigate any of the risks above, it is important to conduct proper testing. Incomplete test coverage is a risk that may result in unforeseen bugs and vulnerabilities

in the new feature's code. It is crucial to ensure comprehensive test coverage for various use cases, devices and platforms. Furthermore, another risk is that future updates to Kodi may affect the watch party feature. Regular regression testing is necessary in order to identify bugs that may arise from new updates.

There are many potential risks involved with releasing a new software feature such as the watch party feature for Kodi. The proper steps must be taken as described above in order to best attempt to mitigate these risks.

## Conclusion

In conclusion, we propose to introduce a watch party feature to Kodi that allows users to concurrently view media content and interact through a chat feature. We explored the various stakeholders involved and proposed two possible architectures to implement this new feature. The two considerations were a peer to peer architecture and a client server architecture. Through our SAAM, peer to peer architecture is recommended with all advantages and disadvantages considered. Lastly, we identified potential risks and suggested various testing processes in order to ensure a seamless integration of our proposed feature with Kodi and its current functionalities.

## References:


*About Kodi. Kodi. (2023). https://kodi.tv/about/*

*Architecture. Architecture - Official Kodi Wiki. (2023). https://kodi.wiki/view/Architecture*

*Video playback. Video playback - Official Kodi Wiki. (2023). https://kodi.wiki/view/Video_playback*

*Dreef, K., Reek, M. van der, Schaper, K., & Steinfort, M. (2015). Architecting software to keep the lazy ones on the couch. Kodi. https://delftswa.github.io/chapters/kodi/#stakeholders---the-brainiacs-behind-your-tv-screen*

Ng, E., Devnani, A., Karkal, R., Akbar, A., Sahi, D., Li, D.. (2023). *Kodi - Conceptual Architecture [Unpublished paper]. Computing Department, Queen's University.*