KODI Architectural Enhancement Report

CISC 322/326 - Group 4

Kabeer Adil - 20ma101@queensu.ca
Kate Edgar - 20kie@queensu.ca
Nicholas Tillo - 20njt4@queensu.ca
Raksha Rehal - 20rsr3@queensu.ca
Yu Xuan Liu - 20yxl3@queensu.ca
Xiyun (Victoria) Cao - 18xc17@queensu.ca

TABLE OF CONTENTS

ABSTRACT

Our previous reports covered KODI's conceptual and concrete architecture, its dependencies, structure and the various architectural styles at play, this report aims to accomplish a different goal. For this report, we will take into consideration everything we have covered and put forth an idea for the implementation of a feature that will enhance the user experience. Additionally, the impact of the new feature on KODI's subsystems will be discussed as well as the potential risks and issues it may concur on the existing architecture.

INTRODUCTION & OVERVIEW

With our age of information constantly expanding and reaching new heights, something that has been brought up in every field as of late has been the usage of AI. While some view it as a threat to livelihood (which it very well could be), others see endless potential for automation and ease of access. Other services similar to KODI employ recommendation ratings, i.e Netflix displaying a percentage based on how interested they think you'd be in a certain show. With our KODI feature, we intend to show the possibility of utilizing AI in order to enhance user experiences while still allowing for individual engineering to occur. As a result, we decided that any enhancement or new feature that was added to KODI's architectural base would be one that would benefit its users while maintaining enough distance so as to not be perceived as a threat or nuisance. After some deliberation, our proposed feature consisted of recommending content to consumers based on their unique preferences and usage habits. These recommendations could include films, tv shows, songs, or other add-ons/apps. There are multiple ways this could be implemented in order to benefit the user with the usage of AI being one such potential path. By analyzing patterns and observing consumer usage, it is our hope that this feature will supplement KODI's existing features and services whilst also showcasing that AI, when used to augment and not fabricate, can be an incredible way to enrich pre-existing notions and ideas.

OVERVIEW OF ENHANCEMENT

KODI does not currently have a serviceable recommendation system implemented within its architecture. Presently, it simply provides prompts for the user to pick up on newer episodes of shows or movies that they left off on. There have been attempts at making recommendation systems for KODI in the form of widgets and plugins. Some of the plugins that provide recommendations in their own ways include LazyTV (which provides recommendations based on movies saved in the user's library) and TraktTV (which provides recommendations based on trending and popular shows and movies). None of these services, however, are curated uniquely to the user's interests and preferences. In other words, no data on the user is collected and utilized for accurate recommendations. Our proposed enhancement would be a recommendation system that is precise to the user, showcasing movies, TV shows, and music that would be curated to their taste based on the data collected by their previous viewing/listening histories.

There are various benefits that an all-encapsulating recommendation system would bring to KODI…

- **Cohesivity and variety**: The plug-ins that are available are solely exclusive to movie (and sometimes TV show) recommendations. An inherent recommendation system would be able to highlight the strength of KODI - i.e. that it is a multimedia player, and hence would be able to provide recommendations for various types of media including movies and shows, but also music, live TV programs, and even pictures. This is a more specialized service than what most recommendation systems do; i.e. provide recommendations for videos only.
- **Contemporary advancements** : KODI using an AI-based implementation would show that it keeps up-to-date with the latest features that technology can offer. Many media players (such as Spotify, with their new AI-run DJ recommendation system, or even Netflix) are starting to utilize this new technology in furthering the user experience for its clients. Incorporating a similar feature will demonstrate the modernity of KODI as a system, while also demonstrating how versatile and modifiable the system is. Users will feel reassured in their choice of using KODI, and will not consider switching media players in hopes of a more modern system, as this feature would be able to demonstrate that KODI is on top of its game.
- **Marketability** : Having a recommendations feature will encourage further use of the system, as these recommendations will ensure that users continue to consume content supported by the KODI platform. This would especially be effective if - added alongside the recommendations - are easily accessible links to *access* these recommendations.

Sample scenarios of the implementation of the feature:

- Upon opening KODI, recommendations would be presented on its landing page. This would be a direct upgrade of the “recommendations” that KODI currently provides on its homepage (i.e. simply prompts to continue the last unfinished media that was requested by the user, such as an unfinished movie or a new episode for a TV show they were watching). These recommendations would be based on the watch/listening history of the user.
- After completion of a movie or TV show, or upon exiting the music or live TV player, the user could be prompted with a window that provides them with recommendations for media similar to the one they had just engaged with. E.g. Upon completing a horror movie, users would be prompted with a variety of horror movies with a similar plot/atmosphere as the one they just watched, and would be able to select to play/rate any of the options provided.

Implementation Approaches

1. Through the use of generative AI to collect data and create recommendations for users based on the media they consume. The model would collect data on the user's watch history (including factors such as genre, duration of engagement, and metadata of the content) to provide personalized recommendations for the user. This ML model would be more fine-tuned with the more data gathered on the user's interests.
2. Through the use of a data collection system that would survey users' opinions on other works of media relevant to one which they consumed. Specifically, there may be a pop-up at the end of a movie or a song that prompts them to rate how similar the media they just consumed is to the options provided. This will serve almost as a poll system to collect data from the user which will be utilized across KODI systems likewise. Architectural styles that would need to be utilized in this approach include a repository style to store the poll results.

Interactions with Other Features

The user's experience with KODI changes when adding this feature in a variety of ways. For one, they are likely to have a better time engaging with the features that KODI has to offer. Immediately on start-up, their homepage will be filled with specially curated media recommendations for them, whether that be shows, movies, music, or channels. The accuracy of the recommendation model would compel them to use the system more. Many interfaces would be impacted in the sense that new windows would have to be created if utilizing the polling implementation approach.

SAAM ANALYSIS

This section of our report will detail our SAAM architecture analysis in regards to our two approaches to our consumer recommendation feature. The SAAM analysis will compare KODI's major stakeholders, pertinent NFRs, and how each implementation of the approach will affect them overall. We will compile this information in order to decide which implementation will have the best affect on all the important parties. The major stakeholders in this setup comprise of KODI's users and its many developers. When considering users, some of the NFRs being discussed will include security, usability, reliability, and performance. Meanwhile for the developers, the relevant NFRs may include the likes of scalability, maintainability, interoperability and the breadth of data integrity. The following charts sum up the impact of all these NFRs, how they will affect the relevant stakeholders and how our different approaches will change how they operate within the context of the architecture.

Stakeholder: Users

NFRs	**Implementation 1**	**Implementation 2**
Security	Using AI means gathering and iterating through important and potentially private information, so it is of utmost importance to ensure that it is anonymized and encrypted in order to ensure user privacy.	Because this implementation depends on the user answering a survey every time (something they may not do), it may be less accurate than the AI model, however, it does not have to iterate through much user information to do its job.
Usability	The users would be offered an opt-in mechanism for the AI recommendation system, as well as customization options allowing them to further specify recommendations based on genre, content, length, etc.	In order to gather user opinions easily, a focus would be a user-friendly interface with clear instructions regarding how to contribute to the data gathering. It would also be important that the integration of the poll system is not intrusive and does not affect the user's usual KODI use.
Reliability	The algorithm will need extensive testing and validation in order to ensure that the recommendations borne from it reflect on the user's actual preferences. Ensuring the processes related to this implementation take place in the background and do not affect KODI's regular services will also be a key factor.	In this situation the implementation would need to be robust enough to handle varying user loads without needing downtime in order to be effective. Additionally, should the system run into errors, mechanisms need to be in place to ensure that data collection from the user is not interrupted.

Stakeholder: Developers

NFRs	Implementation 1	Implementation 2
Scalability	As the user base grows, the amount of user data that the AI would have to sort through would only grow. As a result, designing modular systems with more resources and nodes would need to be created in order to scale up processing needs.	Similarly, the data collection system in this implementation should be able to handle large volumes of user submissions. Partitioning the data and balancing the load of information is key in making sure that the software can keep up with growing demand as KODI's userbase increases in size.
Maintainability	Implementing automated testing and continuous version control, as well as following design practices that ensure that code is written modularly and in an easy-to-understand manner will be key to ensuring a strongly maintained codebase.	In this case, both implementations operate relatively similarly. Both must ensure that code is well documented, easy to understand, and contains modular components. This will allow for easier bug fixing, rolling out updates and enhancing the feature overall.
Interoperability	Ensuring that the AI recommendation system can seamlessly interact with various data sources, media formats, and potentially external APIs or services is necessary to guarantee interoperability for this implementation. This involves creating interfaces that allow easy integration while maintaining accuracy in recommendations.	In this case of this NFR, both implementations inhabit similar ideals. It would be pertinent for the data collection system to interact with diverse platforms, devices, and data sources where users consume media. Would need to work on establishing protocols or interfaces that allow seamless integration with different media platforms, ensuring the system can adapt to changes in media consumption.

After analyzing both implementations and how they affect the stakeholders through various non-functional requirements, our team decided on adopting the first approach, the AI recommendation implementation. This is based on how the NFRs in question are reflected across the user base. In terms of developers and how they would have to handle aspects such as scalability, maintainability and interoperability, the similarities between implementations are quite high. Therefore, deciding which one works best using just the developer input would yield an inaccurate result. However, when factoring in the stakeholders on the user's side, a clearer picture is painted. By analyzing the security, reliability, and usability of NFRs, the AI recommendation implementation showed more positive additions to KODI's existing infrastructure. If implemented correctly, it would present an optional feature that enhances the user's experience while keeping their information safe and working almost entirely in the background; thereby not interfering with a user's daily use of KODI's platforms. While it would share some of the technical needs of our second implementation, it would fall short of presenting a totally hands-off, automated experience in the same way that our first implementation would.

IMPACTED SUBSYSTEMS

User Interfaces,

The main section that gets impacted by the newly implemented function. The high level architecture that gets affected is a new sub-component potentially called a "Polling Manager"/"AI Generator" or a child of the current subcomponents "Interfaces" and "User Input". Each way comes with its own benefits and drawbacks,

For the extension of the current subcomponent: Its benefits are that it creates a less cluttered, higher level of subcomponents, and allows for the inheritance of functions from the parent classes. The drawbacks are that it is unclear which subcomponent the new subcomponent should be a child of, as it requires both the inheritance of "Interfaces" and "User Input". Additionally, this model is not very scalable, as it is largely self contained and the expansion of this system will create too many dependencies to be viable.

For the creation of a new subcomponent: The benefits are that it allows for its own separate functionality, and other things are able to connect and create dependencies with this component. It also has a focus on scalability, in the future if it grows to larger functionalities it will have to make fewer changes in comparison to the other option. The drawback is that it is unable to inherit from any of the previous subcomponents, and thus must create new dependencies, cluttering the upper level of implementation.

- On the lower level,
    - Functions like "create_poll()" and "push_entry_to_database()" would have to be implemented and strength the dependency from the UI component to the Data & File Management component.
    - Functions like "display_poll()" are required with many connections and dependencies to other subcomponents within the UI.
    - Functions implemented for the AI creation and data analysis must rely on functions defined in the Utilites compoent, further increasing that dependency.

Impacted Directories: All, mainly Interfaces, Input & Dialogs

Add-ons,

Add-ons may be added by the community to enhance the accuracy of said component, or to extend its capacities to other media formats outside of KODI. An example Add-on that can be created is an extension of the recommendations to other subcomponents, it could use recommendation functions to create recommendations for file sharing, using the information of what files are usually downloaded together. This addon creation requires the functions used by the recommendation system, and the connected subcomponents to be public, shifting the required implementation tactic.
Impacted Directories: Kodi Developmental Kit

Data & File Management,

The data and file manager is largely impacted by the introduction of this feature. An entirely new subcomponent will have to be formed in order to accommodate for the required data collection and analysis. This component will manage most of the data that is required for these two implementations. No matter how the system get implemented, it will require a database to store either,

- Through Generative AI implementation, it will require a large amount of testing data to train the AI to give accurate results that the user will like.
- Through the poll and database implementation, the polling system requires a section of the database to puts its entires into, and the recommendation system requires a database to pull its information from so therefore the a whole new database component must be created.

Impacted Directories: File System & Multi-Media Database

Utils,

As utils is largely a collection of independent functions used all around the system, many smaller functions will likely get added to the subcomponent. These functions are mainly going to be to assist the data analysis algorithm implemented of choice.

- Things like “calculate_vector_dot()” or “perform_linear_regression()” must be added to the utilities so that the generative AI will be able to accurately manipulate data.
- The method of polls and data bases requires little utilities, the most likely option is some sorting algorithm, maybe a merge or binary sort can be created. Some other auxiliary functions like “get_most_polled()” will also have to be implemented.

In addition, like any new functionality that gets added, it must be tested, this will be located in the Utils subcomponent. A test suite and testing systems must be added to ensure accuracy and reliability of the feature.
Impacted Directories: Settings & Test Code

Cores,

The cores component is largely unaffected, as recommendations are based on data and user interfaces, having no intersection with how things are rendered. Curiously the only possible change that can be made to cores is specifically to the retro player in which the recommendation algorithm can be applied to in order to recommend new games.
Impacted Directories: Retro Player

PVR,

As the PVR is largely separated from the streaming service the KODI provides, this subsystem is generally unaffected. A possible architectural change could be a way to help assist in the data collection that is required by both possible implementations. A change that could be made is that the recorded media can be used for information about what people tend to view together. As if someone records multiple movies using the PVR, those relationships between the recorded movies can be recorded, either to help train the generative AI, or to add to the data polled.
Impacted Directories: Recordings

Network,

The Network component is completely unaffected, its focus on relationships between users is separate from the user based recommendation system proposed.
Impacted Directories: None

USE CASES

1. User receives personalized movie recommendations upon opening KODI

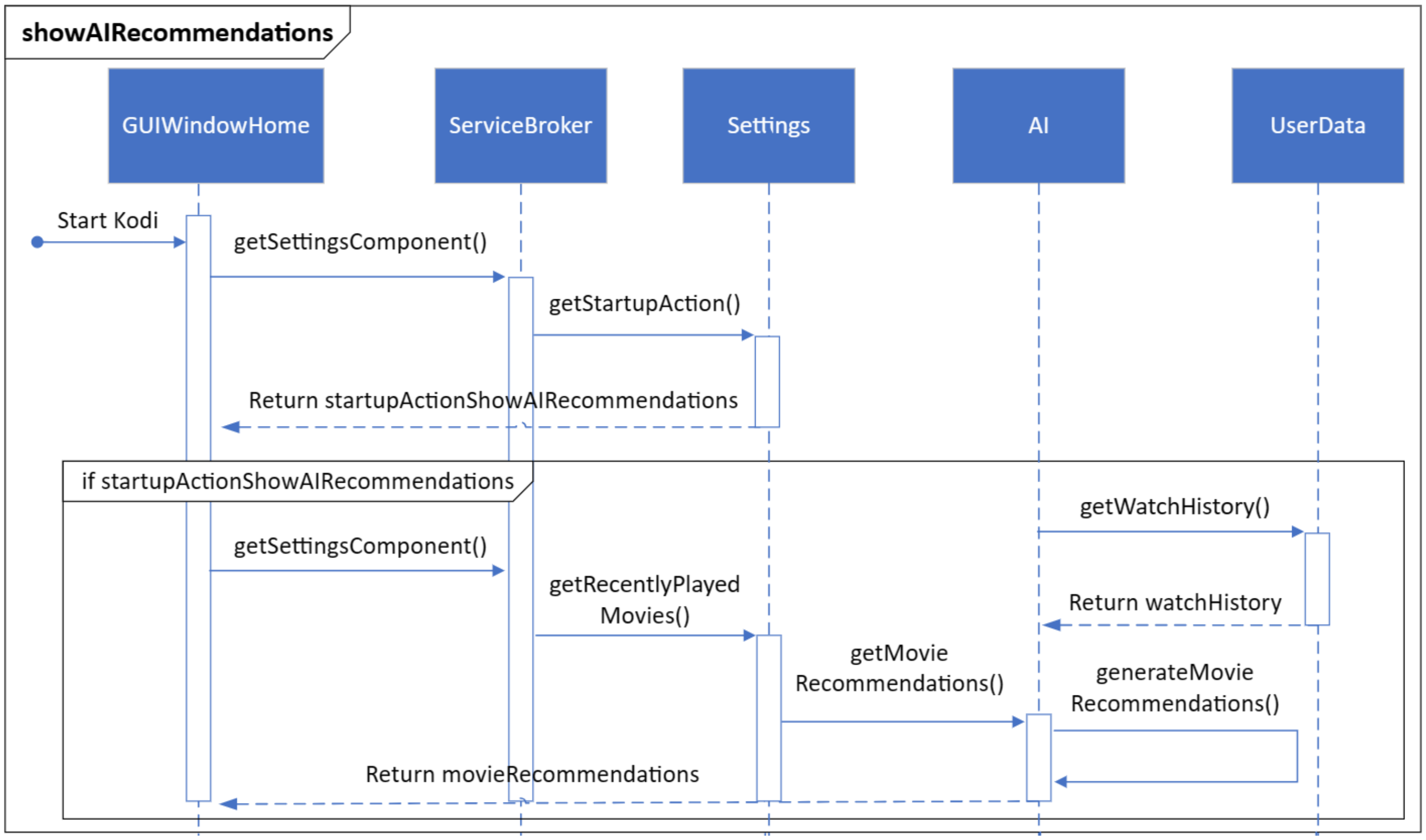
showAIRecommendations
GUIWindowHome
ServiceBroker
Settings
AI
UserData
Start Kodi
getSettingsComponent()
getStartupAction()
Return startupActionShowAIRecommendations
if startupActionShowAIRecommendations
getSettingsComponent()
getRecentlyPlayed Movies()
getMovie Recommendations()
getWatchHistory()
Return watchHistory
generateMovie Recommendations()
Return movieRecommendations

Figure X: Sequence diagram illustrating how AI movie recommendations could work

2. After completing a TV show or movie, the user will be provided with dynamic recommendations

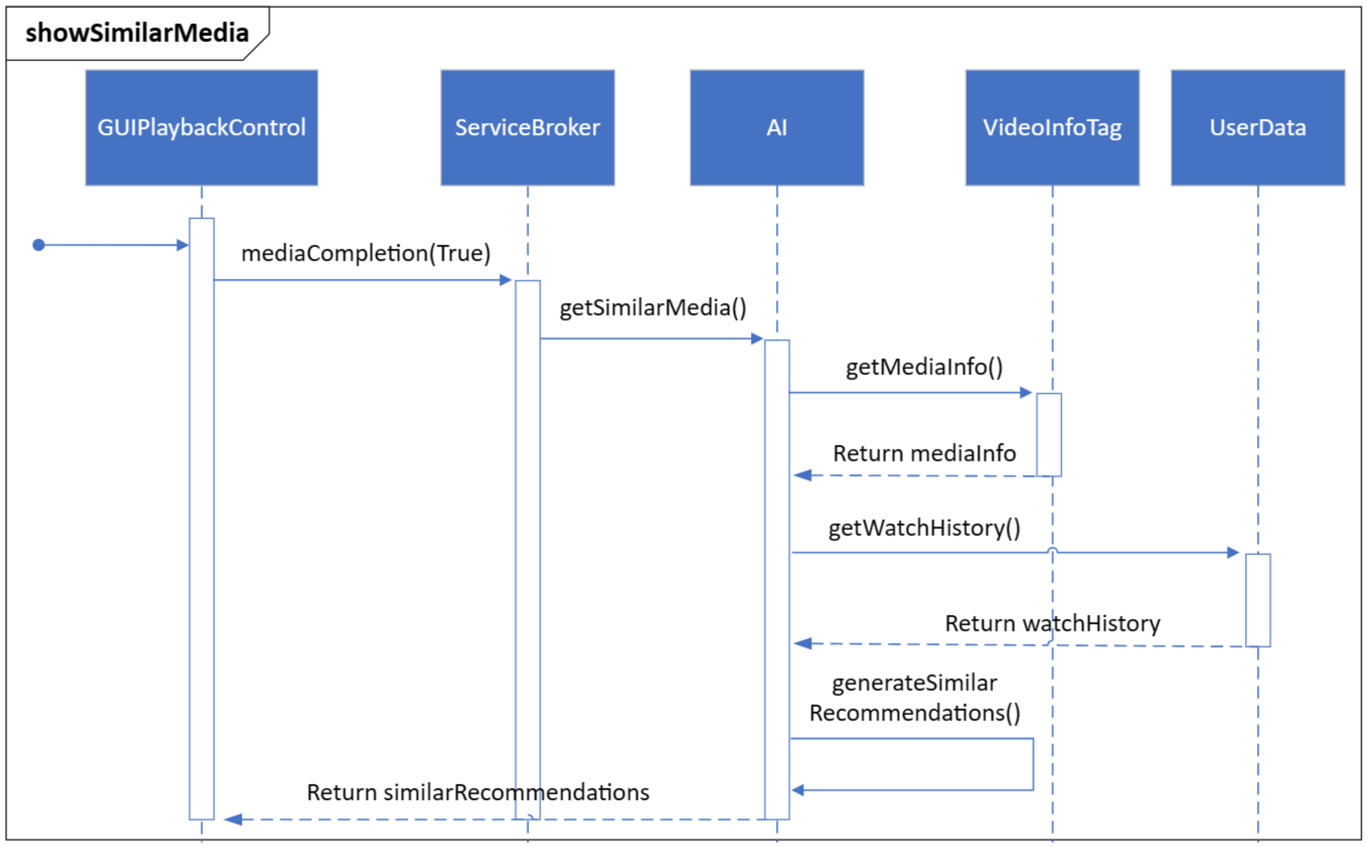

Figure X: Sequence diagram illustrating how our AI enhancement can recommend similar media

- The AI Recommendation System combines the recently played media's information (eg. genre) from the VideoInfoTag module and the user's watch history from the UserData database to generate similar media recommendations.

TESTING

The testing plan for our proposed AI recommendation system enhancement involves a multifaceted approach to ensure a seamless integration and positive impact on user experience. The primary testing stage will be focused on the installation and configuration procedures, ensuring a smooth deployment for users. Emphasis will be placed on ensuring a smooth integration with existing functionalities and compatibility across diverse input devices and operating systems.

For early usability and accuracy testing, we will provide the AI with a limited subset of movies. The system will categorize these movies, allowing for manual evaluation of accuracy. To enhance the scope of this testing, we will request user input using methods such as a thumbs-up or thumbs-down system, to scrutinize the recommendations provided to them. In addition, usability testing will evaluate the GUI for intuitiveness and non-invasiveness.

Security testing will evaluate the impact on user privacy, with emphasis on the anonymization and encryption of sensitive information. For reliability, scalability and maintainability testing, we will gradually roll out the enhancement to users. This approach allows us to assess the AI's ability to handle varying user loads without experiencing downtime. Additionally, it provides us with an opportunity to gauge the efficiency of our maintenance process without significant repercussions.

POTENTIAL RISKS

There are three aspects of concern when discussing the potential risks and limitations of the AI recommendation system, which are data privacy and security, algorithm bias and accuracy, and resource scalability. Data privacy and security are usually a major concern for any service that needs data collection to perform. In the AI implementation proposal data being collected is strictly restricted for protecting the user's privacies. There is still a security risk as long as the users' data has been collected. If users feel their data is not being handled securely, it could lead to a loss of trust in the user base. Ensuring that user data is securely stored and processed is crucial in the development process. As the data being collected is restricted. There is a risk that the AI algorithm might be biased, leading to skewed or inappropriate recommendations. Ensuring the accuracy and fairness of recommendations is essential to maintain user satisfaction which is a challenge with the limited data. For resource scalability, AI systems that particularly involve data analysis and generative processes can be resource-intensive. Ensuring that the system can scale effectively to handle large user bases without compromising performance or increasing operational costs is a significant challenge. There is the potential risk that the AI system might consume more resources than expected and slow down KODI's general performance in other aspects.

The concern for data security also exists for the other implementation that requires active data collection from the users. Active data collection can be more disruptive to user activities, potentially affecting the usability and security of the system, and may cause data redundancy since the data collected from users might not be highly reliable.

LESSONS LEARNED

Through our examination of both implementations approaches we've come to see an example of how when we're comparing different implementations or contemplating between two different approaches. It's important to examine the way in which each Stakeholder and their respective NFRs correspond and react to the method being implemented. In this instance, despite how the NFRs are reflected across the developers in a way which presents both as very feasible and plausible approaches to the enhancement. Upon examining the User side of things, we come to realize that the methodology involving active data collection is far more disruptive of user activities and makes accuracy far more difficult due to the lack of reliability and security. As such, we can determine that the A.I. approach/passive data collection system is better than the active data collective system.

Furthermore, upon our research on what could possibly improve KODI’s functionalities and usage as a streaming app/platform. We’ve come to discover the importance of incorporating functions commonly found as a part of the app’s contemporaries. As that would improve marketability/credibility since it is now comparable to current industry standards.

CONCLUSION

In conclusion, by implementing a serviceable recommendation system utilizing artificial intelligence we could enhance KODI’s user experience and update its system to one on par with modern streaming platforms services. To be specific, we propose the use of generative AI which collects data on user media consumption in order to provide recommendations tailored for said user. This will be done through passive data collection, with the data collected being based on the user’s watch history. Where factors such as the genre of the content it’s content will end up determining the recommendations received by the user. We found that this implementation of a data collection system is effective in fulfilling the specific non-functional requirements (NFRs) of the two identified major stakeholders, the Users and the Developers. The Users require Security, Usability and Reliability. While the developers need to fulfill scalability, maintainability and interoperability requirements. Although this will require an immense amount of work and will result in the modification, if not complete reconstruction, of KODI’s numerous subsystems. We feel that leveraging generative AI for data collection and recommendation generation would ultimately enhance KODI’s system/software architecture. As it would allow KODI to be capable of tailoring content that meets the specific preferences and needs of its users. Significantly enhancing its marketability as a streaming app, since it now boasts similar features that modern streaming platforms utilize, as well as the general cohesivity of KODI’s system as a whole.