# Kodi Conceptual Architecture

**CISC 322**

**Assignment 1 Report**

**Sunday, October 22, 2023**

Group 8 : KidKodi
Aidan Gardner - (20agg2@queensu.ca)
Gavin Chin - (21gc9@queensu.ca)
Daniel Garami - (21deg1@queensu.ca)
Barkev Keyork Sarkis - (19bks5@queensu.ca)
Mahir Khandokar - (21mhk5@queensu.ca)
Adrian Putz-Preyra - (16app2@queensu.ca)

**Table of Contents:**

## Abstract

This report offers a comprehensive analysis of the conceptual architecture of Kodi; an open-source multimedia application that caters to a diverse set of users via its broad compatibility among many different devices, operating systems, and platforms. As a centralized platform designed for playing and managing various types of media, Kodi is a highly modular and extensible system. Outlined in the report is the layered structure of the platform, featuring Client, Presentation, Business, and Data layers, with each layer having its own set of roles, functionalities, and sub-components within the system. Furthermore, the components within each layer are identified and explored, as the report specifies the functionality and interactions the layers have within the system. To gain perspective into these interactions of the system's architectural components, two major use cases within Kodi are analyzed, with these use cases being media playback and media downloading. In addition to this, the report delves into the details of several external interfaces and also provides a glossary of key terms and naming conventions described in the architecture to better understand the system. Through this analysis, the report highlights Kodi's robust and adaptable conceptual architecture, attributing its widespread popularity as a multimedia platform to these key characteristics. The Figure below provides an overview of Kodi's dependencies, interactions, and relationships.

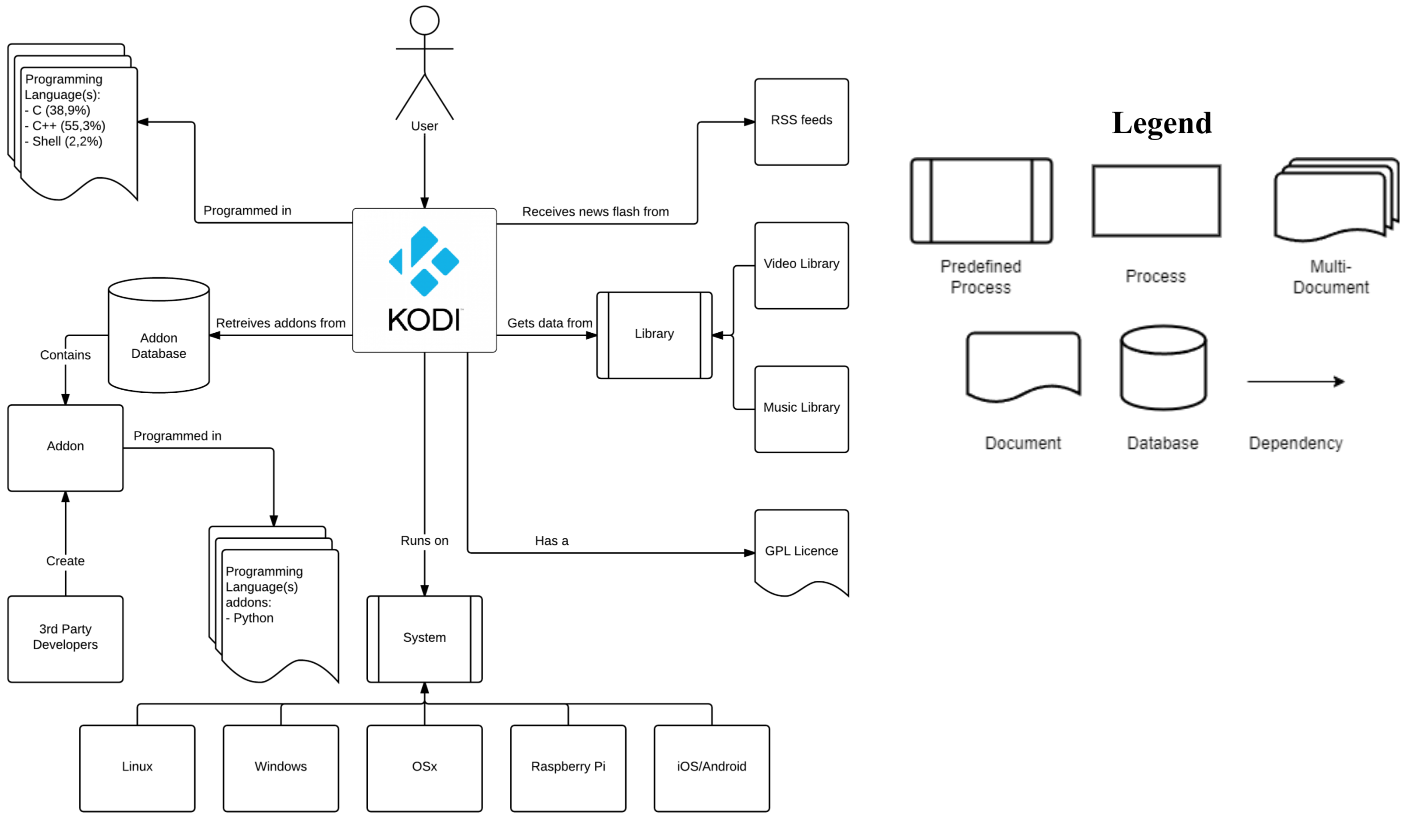

**Figure 1:** Overview of Kodi's Dependencies, Interactions, and Relationships

## 1. Introduction and Overview

**Introduction:**

Kodi, formerly known as XBMC Player, is a versatile and open-source multimedia center developed by the XBMC Foundation, with contributions from a global community of developers mainly collaborating on GitHub. This report delves into the architecture of Kodi Media Player, shedding light on its multifaceted layers, features, and architectural aspects. It aims to provide insight into the robustness and extensibility of Kodi's architecture, exploring how it has evolved over time.

Kodi offers an engaging and user-friendly design, allowing users to access and enjoy a wide range of media, including movies, music, and pictures. Its interface serves as a comprehensive media center, offering a clear and interactive overview of multimedia content. Moreover, Kodi supports an array of add-ons, enabling users to stream or download content from external sources. Kodi's journey began with its development for the Xbox game console, originally named "XBOX Media Center (XBMC)." Due to its increasing popularity, the XBMC Foundation decided to expand the software to various platforms. Today, Kodi is available on Windows, OSX, iOS, Android, and Linux. In 2014, the software underwent a name change, evolving from "XBOX Media Center" to "Kodi" Entertainment Center. It is noteworthy that Kodi has always been an open-source project, freely available for all to use.

The conceptual architecture of Kodi was derived by analyzing the documentation on their open-source GitHub repository, as well as the Report “A Report On the Journey of Team Kodi” and the Kodi Wiki. This report offers a comprehensive analysis of the architectural structure underpinning Kodi Media Player, providing insights into the various components that make up Kodi's software architecture and its functionalities. By scrutinizing this architecture, we aim to gain a deeper understanding of how a multimedia center's software architecture can be designed and operated. Additionally, we will explore potential improvements and adaptability of the architecture to enhance Kodi's capabilities. In our exploration, we also consider how Kodi's architecture accommodates future changes and expansion. To illustrate the practical operation of these components, we present two use cases, demonstrating how the components collaborate to accomplish specific tasks. The first use case demonstrates how Kodi's components work together to facilitate media playback, which is the core functionality of the software. The second use case showcases Kodi's capabilities in media downloading, allowing users to seamlessly retrieve and store their desired multimedia content.

## 2. Conceptual Architecture

### 2.1 Overview

The architectural style of Kodi can be classified as layered architecture. The system is built up of multiple component layers that can interact with each other, however, these layers are not limited to only interacting with adjacent layers. Rather each layer is largely independent from each other, allowing a significant amount of interactions between different components, therefore the architectural style of Kodi does not strictly adhere to all of the elements of a normal layered style architecture. As a result, Kodi is able to employ a multi-threaded architecture which allows for the concurrent execution of tasks. This is essential in a versatile multimedia player such as Kodi that relies on a smooth and responsive experience. Regarding the interactions between layers, they feature many methods of passing control, such as message passing, event-driven and service-oriented communication, and the use of APIs to support user customizability. The different component layers of Kodi are the following:

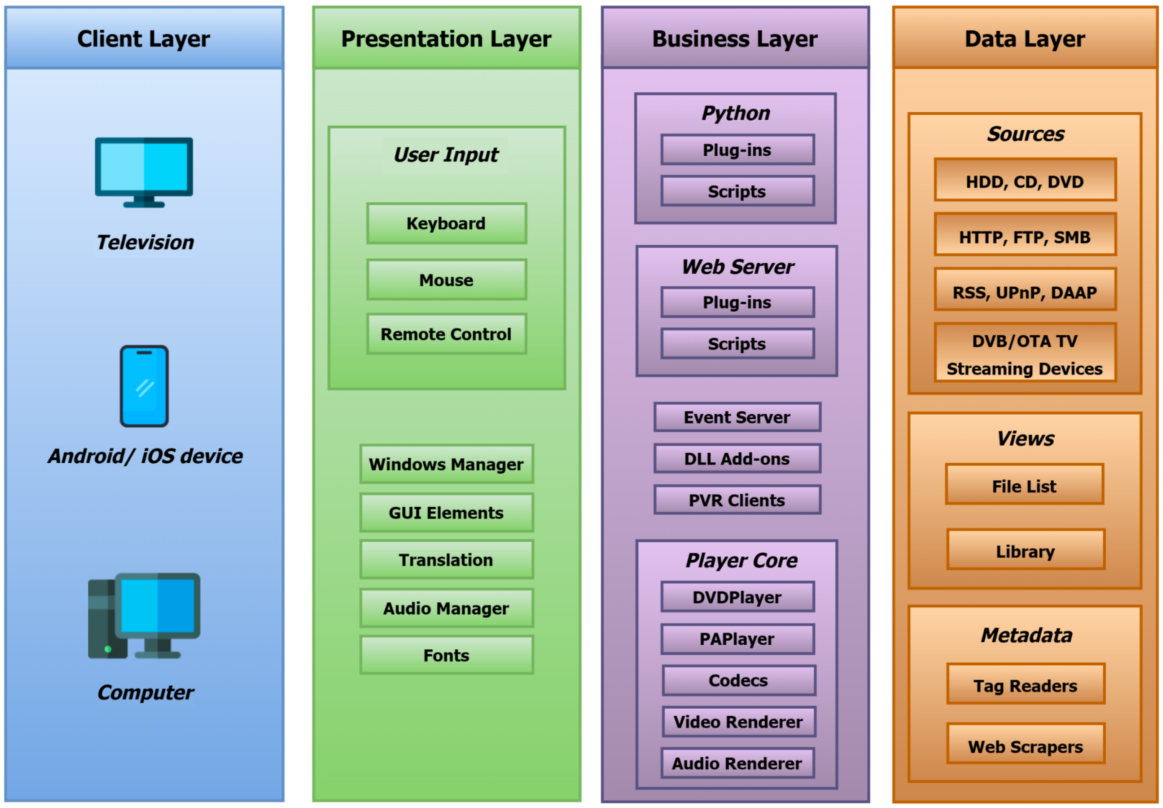

**Figure 2:** Architectural overview of Kodi

- **Client Layer:** This layer is related to the specific system that is being used to run the application, whether it is TV, Android, IOS, Windows, Linux, etc.

- **Presentation Layer:** This layer includes the files and packages in charge of the user information display part. This includes things such as fonts, language packs, user input, view management, etc.
- **Business Layer:** Contains files and packages related to server access, external libraries, and add-ons. Additionally, this layer provides a client that will establish a connection with a streaming server, and is also in charge of reading and displaying audio and video files.
- **Data Layer:** This layer contains everything in reference to files, functions, and content management packages. This includes saving, placing the files on the disks used, streaming them from an external server, transforming data, and more.

**2.2 Client Layer**

The client layer of Kodi plays an important role in ensuring that the application is accessible and performs well across multiple systems and devices. In other words, it is responsible for allowing for the execution and functionality of the Kodi application on a wide range of platforms such as iOS, Android, Linux, Windows, macOS, among others.

This layer acts as the interface between the Kodi application and the hardware and software components of the user's device. It ensures that the application can access the resources and capabilities of the host system through its provided adaptation and compatibility mechanisms. This includes managing features that are specific to certain devices, handling hardware interactions, and taking care of system-level requirements to help achieve the best possible performance and functionality of the application.

Additionally, the client layer is highly responsible for the overall consistency of user experience throughout the many platforms the application is available for. This serves the purpose of guaranteeing that a user can seamlessly interact and access multimedia content across any of their preferred devices. It leverages particular capabilities and functionalities of a variety of different operating systems which significantly strengthen the overall usability and accessibility of the Kodi application.

**2.3 Presentation Layer**

The presentation layer of Kodi is largely responsible for the overall user experience of the multimedia application. This is mainly due to the layer encompassing the role of the user interface which typically is directly correlated with user experience. The layer includes a variety of components and modules that handle the visual display, user interactions, and user customization options within the application. The following are the key elements of the presentation layer.

- **User Input Module:** Responsible for handling peripheral devices that detect user input. This allows the users to interact with the Kodi interface using a variety of input devices including remote controls, keyboards, touchscreens, etc.
- **Windows Manager Module:** Kodi relies on the window management libraries provided by the operating system of the host to manage tasks such as window arrangement, navigation, and display management. This ensures a seamless experience regardless of the user's device or platform.
- **GUI Elements:** Kodi's presentation layer provides and utilizes a variety of visual components such as windows, buttons, dialog boxes, and lists, among many other graphical user interface features. This provides the user with intuitive navigation and content interaction with the Kodi platform.
- **Translation Module:** This module offers support for multiple languages, enabling the user to select their preferred language within the Kodi interface. The module ensures that the application can be accessed and understood by a wide range of users from different linguistic backgrounds.
- **Audio Manager Module:** This module manages the audio settings within the Kodi platform. It enables users to configure various audio parameters, such as selecting audio output devices, output modes, volume control, and audio synchronization across devices.
- **Fonts Module:** The fonts module allows users to customize the fonts that are present throughout the Kodi interface. It includes customization options, such as adjusting font styles, font sizes, and other font-related features to improve the readability and visual appearance of the text elements throughout the Kodi interface.

## 2.4 Business Layer

The business layer of Kodi includes essential files, packages, and functionalities that are important for the management and delivery of the wide range of content that the platform offers to the user. This layer is responsible for handling server access, external libraries, management of add-ons, handling connections to streaming servers, and processing video and audio files using codecs (DIVX and AC3). The overall goal of the layer is to act as the middleman between the backend data and the presentation layer to ensure that the multimedia content is correctly processed and displayed to the user. We can break down this layer into three main modules listed below.

- **Python Module:** This module manages custom add-ons that allow users to enhance their experience on the platform by integrating additional add-ons and functionality. Users with knowledge of Python are able to create and integrate their own add-ons that they can use within the Kodi platform.

- **Web Server Module:** This module allows users the option to remotely manage their multimedia content through the means of web browsers or other applications. It enables accessibility of the content for multiple users that are within a common local network. As a result, users are strongly encouraged to implement secure practices such as password protection to prevent any security threats.
- **Player Core Module:** The player core module manages and controls the playback of Kodi's multimedia content within the platform. It serves as an interface between the media files and the underlying hardware or software components that are required for high-quality playback. The main functions of the module include decoding multimedia files, ensuring the synchronization between audio and video components, managing playback controls (pausing, stopping, and seeking), and overall delivering a high-quality playback experience for the user. This module is largely responsible for the overall performance and reliability of the multimedia playback within the Kodi application.

**2.5 Data Layer**

The data layer of Kodi is responsible for managing a variety of multimedia-related files, functions, and content management packages. It specifically manages the storing, organizing, and handling of multimedia content within the platform. Some of the key functionalities within this layer include the following:

- **File Management:** This layer enables the storage and organization of multimedia files. It handles tasks related to saving, categorizing, and managing multimedia files on the local storage disks used by the application.
- **Streaming Management:** This layer allows for the seamless streaming of multimedia content from external servers. This provides the user with access to a wide variety of online multimedia services and resources. This is achieved through protocols such as FTP, RSS, and HTTP to ensure the efficient retrieval of media files from remote sources.
- **Data Transformation:** This layer serves the purpose of transforming multimedia files into a variety of formats that allow for transmission and storage. It enables the conversion of media files into compatible formats so that they can be efficiently transferred across different devices and platforms.
- **Content Management Packages:** A variety of content management packages are integrated within the data layer to allow for the organization and management of multimedia content within the platform. These packages facilitate efficient management of multimedia files, metadata, and related information that enable the content to be categorized and organized within the Kodi platform.

**2.6 Control & Data Flow**

Now that the layers of Kodi have been identified, we can look at how data flows between the layers. A user begins with having Kodi operating on some sort of OS, handled by the Client Layer. With Kodi now running, the Presentation Layer handles the visual aspect seen by the user, and what the user is interacting with. From here, a user then decides what they would like to do, ultimately submitting some sort of input request to be handled. This is handled by the Business Layer, in which all the core functionality is held and processed. This layer handles the request by fetching resources from the Data Layer and performing an operation on the data, whether it be downloading, playback, or more. After the request has been handled, the user will see the response of their input on their screen. As summarized above, Kodi handles its flow in a very modular way, where each step is handled by a different layer.

**2.7 Non Functional Requirements**

As a multimedia center that supports the use of a variety of systems, the non-functional requirements of Kodi refer to qualities and attributes that focus on performance, usability, scalability, compatibility, and more. Ensuring the Kodi is quick to respond to user input and provide a smooth and lag-free experience are highly desired components that Kodi strives to achieve. Kodi also aims to be as user-friendly as possible as seen through its customizable UI settings. All going hand in hand are the NFRs of compatibility, scalability, and ease of maintenance which can be seen through Kodi's integration of multiple platforms and modular architecture to support it.

**3. System Evolution**

As an open-source software, Kodi is continuously evolving. With the developers behind Kodi working mostly voluntarily, the evolution of the system starts with a select group of developers who act as stakeholders, able to grant or deny potential contributions. One of the major roles they have is to map out system evolution and decide which platforms to support and which to not. This evolution is key in balancing the time and work of its developers. To make sure that improvement of the software is as easy as possible, the layered structure mentioned earlier also provides for the easy refactoring of code to add-ons. Keeping this in mind, there are some tradeoffs to the evolution of Kodi, as the developers must measure the importance of keeping support on a wider variety of devices resulting in a larger audience versus the ability to maintain the code and the addition of new features.

**4. External Interfaces**

As an entertainment hub, Kodi has a variety of external interfaces that allow it to interact with external services, applications, and devices. One of the external interfaces of Kodi is the web interface, which allows users to control and interact with their Kodi installation through a web browser. This can be used for remote control, library management, visual feedback, and more. To use the web interface, users need to enable it in the Kodi settings and enter a username and password to prevent unauthorized access to their media. Users can also control Kodi remotely or from an external application using Kodi JSON-RPC API. Additionally, the use of network interfaces can be seen, such as Universal Plug and Play (UPnP), Digital Living Network Alliance (DLNA), and Server Message Block (SMB) and Common Internet File System (CIFS) protocols. These network interfaces allow for the sharing of media content across compliant devices on the local network. The use of external media sources can also be seen, as Kodi can integrate with many different streaming services to access online content and metadata. Reliable transmission of data to and from the system is essential, and the use of external interfaces helps to improve the functionality and usability of Kodi. In regards to the data being transmitted, each layer produces and receives different information. For example, the Presentation layer focuses on user interaction information in terms of user input commands and preferences, which it would send to the Player Core Module of the Business Layer. As mentioned earlier, network communication information between networked devices and external content providers is also a major key in providing Kodi with its easy accessibility and sharing properties. To summarize, these external interfaces make Kodi adaptable to different use cases and preferences, and allows users to tailor their setup to work seamlessly with multiple devices and media content.

## 5. Diagrams & Use Cases

Kodi has many use cases for a wide range of features. However, the two main features of Kodi are:

1. Media Playback
2. Media Downloading

These specific use cases will be demonstrated using Sequence Diagrams to model the flow of execution. See Figure 3 to gain an understanding of the symbology being used in these diagrams.

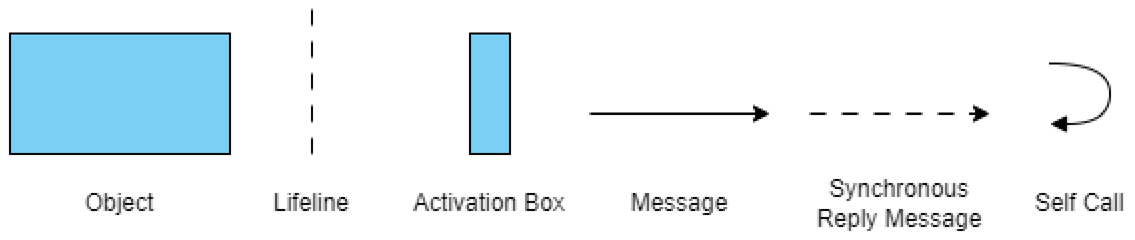

**Figure 3.** Sequence Diagram Legend

**5.1 Media Playback**

The main function of Kodi is media playback. To start media payback, the user must initiate it by selecting some form of media on the Kodi UI. The User Input Module allows the user to interact with the Kodi interface, initiating this process. A majority of Kodi's media management is done in the Player Core Module, which acts as an interface between the media files and the hardware/software required for media playback. The module retrieves media data from the data layer, which includes metadata elements and source elements and decodes the newly retrieved data to convert it into a viewable format for the user. It then retrieves the user's audio settings from the Audio Manager Module and initializes audio playback and video playback to the Client Layer. At this point, the desired media is displayed to the user, with the process being summarized in Figure 4.

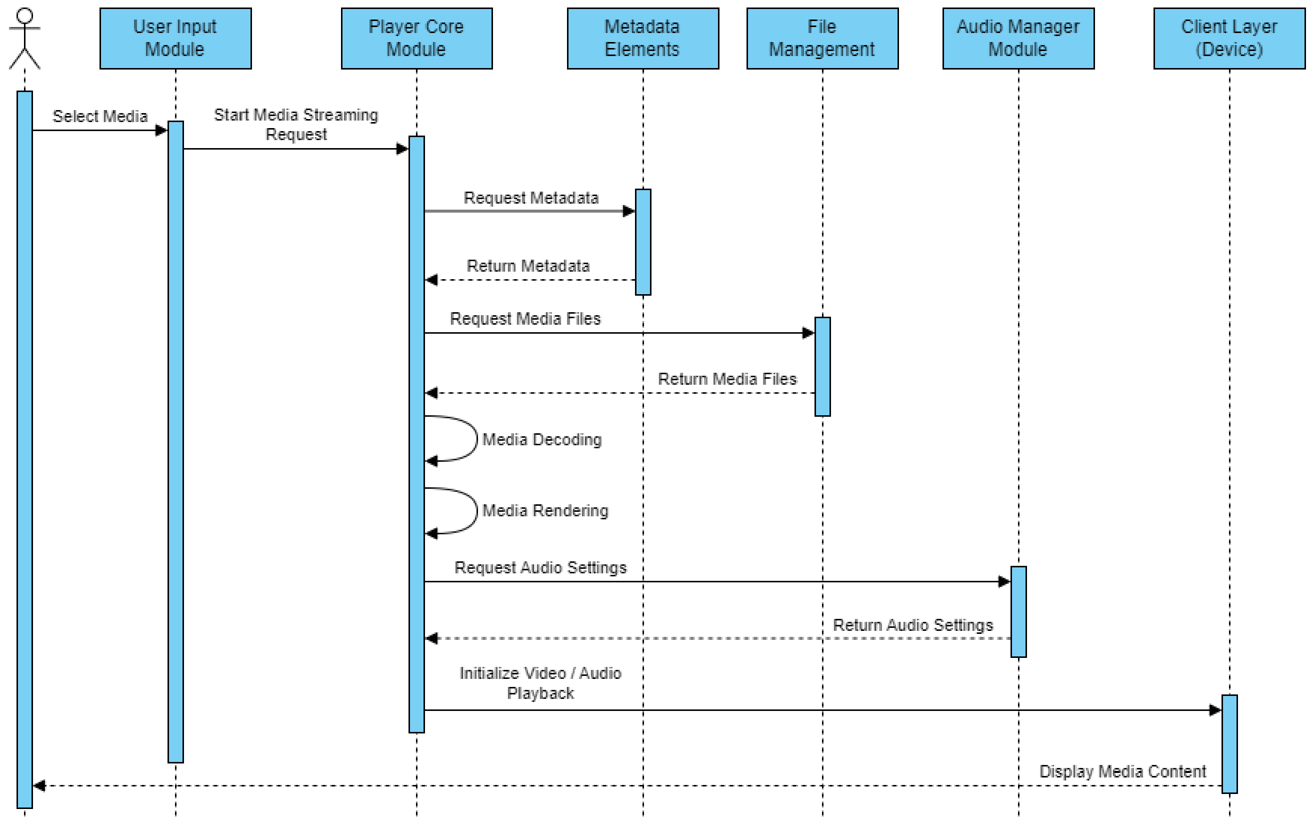

**Figure 4.** Sequence Diagram of Use Case 1: Media Playback

### 5.2 Media Downloading

The second use case of Kodi is the downloading of media to storage systems. To start this process, the user selects some sort of media to download. The User Input Module allows the user to interact with the Python Module by sending a media download request. Responsible for managing custom add-ons and functionality to Kodi, the Python Module can use these to initiate the download of the desired media. The add-on being used in this case would retrieve media download contents including metadata, source elements, and more. Finally, it transforms the multimedia files into an appropriate format that allows for transmission and storage. This process can be seen in Figure 5.

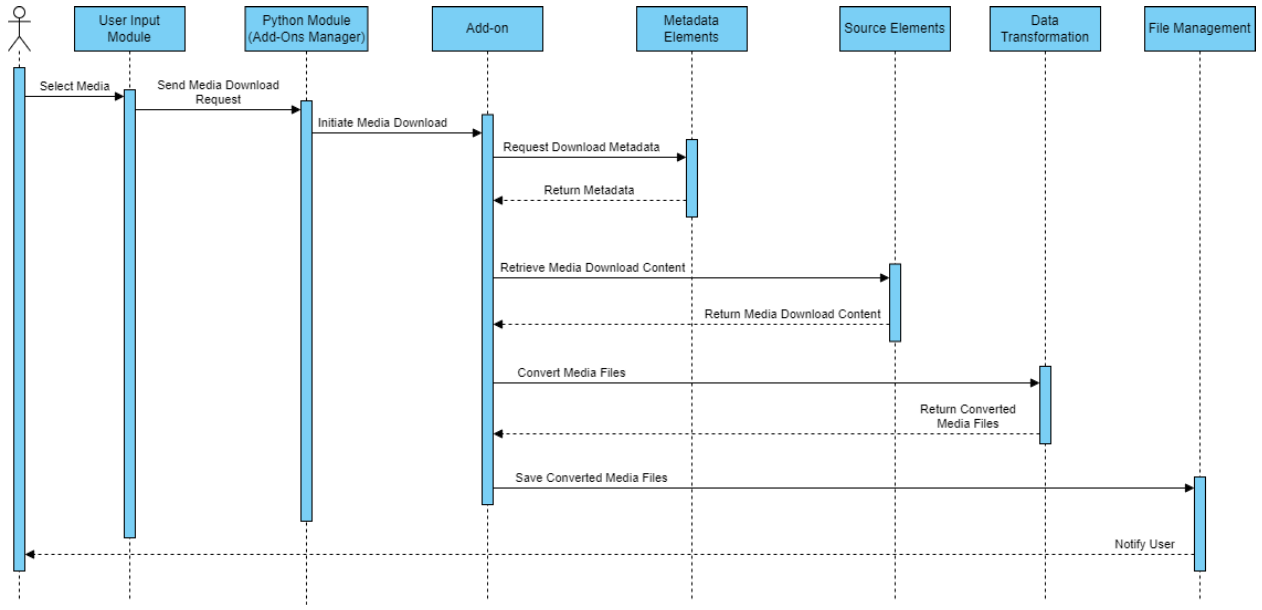
User Input Module
Python Module (Add-Ons Manager)
Add-on
Metadata Elements
Source Elements
Data Transformation
File Management
Select Media
Send Media Download Request
Initiate Media Download
Request Download Metadata
Return Metadata
Retrieve Media Download Content
Return Media Download Content
Convert Media Files
Return Converted Media Files
Save Converted Media Files
Notify User

**Figure 5.** Sequence Diagram of Use Case 2: Media Downloading

## 6. Data Dictionary / Naming Conventions

### 6.1 Data Dictionary

- **Graphical User Interface (GUI):** A visual interface which allows users to interact with software using menus, controls and graphics
- **Add-ons (Feature):** Small extensions or programs that add functionality
- **Media Library (Component):** A database that allows the user to catalog and store media content
- **Metadata (Data):** Information about media content such as title, actors, genres, and more

- **Scrapers (Component):** Components responsible for collecting information from various sources
- **JSON-RPC API:** A remote procedure call specification used for data interchange primarily between a client and a network
- **Web Interface (Feature):** A web-based interface that allows for the control of Kodi from a browser
- **Codecs (Component):** Used in applications to play, create, and share media files

### 6.2 Naming Conventions

- **API:** Application Programming Interfaces
- **UPnP (Universal Plug and Play) (Protocol)**: A network protocol that enables communication with other UPnP-compliant devices
- **DLNA (Digital Living Network Alliance) (Protocol):** Allows for the sharing of media with DLNA-compatible devices
- **SMB (Server Message Block) / CIFS (Common Internet File System) (Protocol):** File managers and accessors used in storage systems
- **FTP (File Transfer Protocol):** Standard network protocol used for the transfer of computer files between a client and server on a computer network
- **RSS (Really Simple Syndication):** Web feed that allows users and applications to access updates to websites
- **HTTP (Hypertext Transfer Protocol):** Application protocol for distributed, collaborative, hypermedia, information systems.

## 7. Conclusions / Lessons Learned

### 7.1 Lessons Learned:

In the course of analyzing Kodi's conceptual architecture, we have gained valuable insights into the multifaceted layers and components that make up this versatile multimedia center. This report has provided a comprehensive understanding of how Kodi's architecture is structured and operates, and we have identified several key conclusions and lessons learned from this exploration.

1. **Versatility of a Layered Structure:** Kodi's architecture allows it to function seamlessly across a wide range of platforms, including iOS, Android, Linux, Windows, macOS, and

more. This adaptability ensures that users can access their multimedia content consistently, regardless of the device they are using.

2. **Security Considerations:** While not explicitly mentioned, it's important to highlight the importance of security in Kodi's architecture. The external interfaces of Kodi allow it to interact with devices across a local network and share media. The Web Server Module, for instance, allows remote management of multimedia content, but it should be used with secure practices, including password protection in order to prevent any potential security threats.
3. **The Importance of Add-Ons:** Add-ons provide huge functionality improvements to Kodi and are crucial when it comes to efficiently refactoring code. By using these add-ons, functionality can be added or removed without having to recompile the whole system or worry about specific dependencies.

Furthermore, as a group, we have collectively learned multiple lessons on teamwork and potential strategies and improvements when it comes to collaborating on a group project. As with the majority of team projects, communication is a must. When it came to creating the report, in order to have a cohesive and efficient working environment, it may have been a good idea to have started earlier or set guidelines or deadlines for getting certain aspects done along with weekly check-ins to assess the completion process.

**7.2 Conclusions:**

In wrapping up our exploration of Kodi's architecture, we've uncovered some essential insights. Kodi's structure combines a user-friendly design with adaptability across various devices. With its extensive support of various operating systems, Kodi strives to be as independent as possible. As a result, the use of modular architecture is key, and we discovered that Kodi's strength lies in its layered architecture, combining elements of traditional layered design with dynamic interaction between diverse components. This helps to make Kodi easier to maintain since it is a constantly evolving open-source project. Ultimately, the architecture of Kodi is well-designed, and will no doubt continue to grow and improve given its adaptability and support for future evolution.

## 8. References:

[1] "What is HTTP?," What is HTTP, https://www.w3schools.com/whatis/whatis_http.asp (accessed Oct. 22, 2023).

[2] Davide, "How do RSS feeds work?," RSS.com, https://rss.com/blog/how-do-rss-feeds-work/ (accessed Oct. 22, 2023).

[3] S. M. Kerner and J. Burke, "What is FTP? file transfer protocol explained," Networking, https://www.techtarget.com/searchnetworking/definition/File-Transfer-Protocol-FTP (accessed Oct. 22, 2023).

[4] R. Sheldon and J. Scarpati, "What is the server message block (SMB) protocol? how does it work?," Networking, https://www.techtarget.com/searchnetworking/definition/Server-Message-Block-Protocol (accessed Oct. 22, 2023).

[5] "What is the DLNA feature and how does it work?," Sony, https://www.sony.co.uk/electronics/support/articles/00106319(accessed Oct. 22, 2023).

[6] "What is UPnP? yes, it's still dangerous in 2023: Upguard," RSS, https://www.upguard.com/blog/what-is-upnp (accessed Oct. 22, 2023).

[7] "What is an API? (application programming interface)," MuleSoft, https://www.mulesoft.com/resources/api/what-is-an-api (accessed Oct. 22, 2023).

[8] "Kodi (software)," Kodi (software) - Wikipedia (accessed Oct. 20, 2023).

[9] "Open Source Home Theater Software," Kodi, https://kodi.tv/ (accessed Oct. 20, 2023).

[10] K. Dreef, M. van der Reek, K. Schaper, and M. Steinfort, "Architecting software to keep the lazy ones on the couch," Kodi, https://delftswa.github.io/chapters/kodi/ (accessed Oct. 20, 2023).

[11] L. Corrales, M. Fonseca, G. Gonzalez, J. Loria, and J. Sedo, "Architecture," Architecture - Official Kodi Wiki, https://kodi.wiki/view/Architecture (accessed Oct. 20, 2023).