UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER

# CS/IT Honours
# Final Paper 2020

## Title: Music Sharing in a Bandwidth Constrained Environment using a Community Network


## Author: Keegan White


## Project Abbreviation: OVCONNECT

## Supervisor(s): Dr. Hafeni Mthoko, Dr. Melissa Densmore

| **Category** | *Min* | *Max* | Chosen |
|---|---|---|---|
| Requirement Analysis and Design | *0* | *20* | 15 |
| Theoretical Analysis | *0* | *25* | 0 |
| Experiment Design and Execution | *0* | *20* | 0 |
| System Development and Implementation | *0* | *20* | 20 |
| Results, Findings and Conclusions | *10* | *20* | 10 |
| Aim Formulation and Background Work | *10* | *15* | 15 |
| Quality of Paper Writing and Presentation | *10* | | 10 |
| Quality of Deliverables | *10* | | 10 |
| Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*) | *0* | *10* | 0 |
| **Total marks** | | **80** | **80** |

# Music Sharing in a Bandwidth Constrained Environment using a Community Network

Keegan White
Computer Science
University of Cape Town
Cape Town South Africa
keeganthomaswhite@gmail.com

## ABSTRACT

Data constraints shape the manner in which digital media is shared and consumed. Musicians who do not have access to a reliable and constant internet connection are adversely affected by global music sharing trends as they do not have the means to upload their music to large streaming platforms. There is a large concentration of musicians facing these issues in Ocean View (OV), a low-income area in Cape Town South Africa. The aim of this project is to develop a music sharing platform that will be deployed on the local community-owned network and consequently be free to use for residences of OV. This music sharing platform will be synchronized with a globally available consumer centred version of the platform hosted on an Amazon Web Services server in order to allow the musicians to reach a larger audience. This will offer the musicians a way to promote themselves and gain a larger audience. Musicians based in OV were consulted at the beginning of this project to define a feature list, however, due to COVID-19 in-person interviews were not possible. This meant all interviews had to be virtual. As a consequence of this the sample size for the interview process was small due to the data constraints the musicians face. The interviews that were conducted reinforced the need for the music sharing website and there was a great deal of enthusiasm and excitement surrounding the project. The system developed is docker based with a WordPress front end, a MariaDB database and a python-based application programming interface that enables the systems to interact with customized frontend features that WordPress does not offer. The outcome of this project suggests that conducting virtual human-centred design projects in resource constrained environments is not a feasible approach.

## CCS CONCEPTS

• Community Networks • Bandwidth Constraints • Database Synchronization

## KEYWORDS

Ocean View, iNethi Community Network, Music Sharing

## 1 Introduction

This project focuses on deploying a music sharing website on the iNethi community owned network located in the low-income area of Ocean View (OV) in Cape Town South Africa. The iNethi network is a joint venture between academics at the University of Cape Town (UCT) and OVCOMM Dynamic, which is run by OV residents, referred to as ‘the directors’. As a result of the management dynamic, the decisions made regarding the infrastructure needed for the website, the features of the website and deliverables created had to be acceptable to both parties.

### 1.1 Motivation and Aims

This project was proposed by members of the OV community as they require a means to share their music with fellow OV residents and those residing outside of OV. Traditional music sharing applications and websites are not viable for OV-based musicians and their fans that reside in OV as these applications rely on a constant and reliable connection to the internet. These individuals do not have access to this type of internet connection as they face data constraints due to their economic standing and the comparatively high cost of data in South Africa [3].

The aim of this project is to create the aforementioned music sharing website that will allow local artists to share their music with the people around them and people residing outside of OV in a simple, cost effective and data-efficient way. The purpose of this website is not only to encourage local content creation and celebrate the local talent but stimulate the local economy by providing the musicians with a way of making money and marketing themselves online. This is accomplished by incorporating social media style profiles where artists can advertise themselves, accept donations and share contact details. Additionally, their music is uploaded to an e-commerce shop that allows users to purchase coupons to access restricted content or download music free of charge.

### 1.2 Deliverables and Methodology

The deliverables of this project are two instances of the music sharing website. One instance to be hosted on the servers in OV and the other to be hosted by Amazon Web Services (AWS). The AWS instance is accessible globally and requires a user to use their own data. The instance hosted on the iNethi servers in OV is zero-rated for anyone connected to the community network. These two instances need to be synchronized, which included not only synchronizing the databases but also the scripts used for front-end and backend processes.

Initially the design and implementation phase of the project was planned to be focused around co-design and rely heavily on user involvement. However, due to the COVID-19 virus and the restrictions put in place by the government, that limited human interaction, it became difficult to recruit members of the OV community. This was caused by the difficulty communicating with them due to their data constraints. However, there was still user involvement in the initial phase of the project, but not as much as had been planned. This resulted in a waterfall methodology to be adopted for the software development lifecycle (SDLC). Reason being it was not possible to collaborate and create iterations of the software based on user input in the development phase. The project timeframe was short meaning that an iterative approach to the SDLC was not viable.

The rest of the report is divided into 4 sections. The Related Work section focuses on the research already done on the core issues and topics that this project faces, mainly the means of internet access in low-income areas, music sharing and bandwidth constraints in low-income areas, as well as the outcomes of a co-design project based on music sharing in OV done by previous UCT students. There is also an exploration of global music sharing trends. The Design and Implementation section outlines the requirement analysis phase of the project and where this information came from, how the system was developed in terms of tech-stack and other details about features and how they are organized and implemented. The Discussion section explores the issues faced during development due to COVID-19 and the dynamic between UCT and OVCOMM Dynamic, as well as how to measure the impact of the project. Finally, the Conclusions section specifies future work that could benefit the community, the impact the website could potentially have depending on the points made in the previous section and highlights the difficulty of virtually co-designing during a pandemic in a resource constrained environment.

# 2 Related Work

Music sharing in low-income areas in South Africa and the data constraints faced by residents of these areas has been well documented. Additionally, there has been vast amounts of research into the significance of music sharing in low-income areas, not only in terms of the opportunities it creates for the artists, but the role it plays in creating communal identities and breaking down the racial barriers faced in modern-day South Africa. Global music sharing trends are also a well-documented area of study, with numerous studies surrounding music sharing platforms such as Spotify.

Additionally, there has been a previous attempt at a co-design study done by UCT students with OV musicians that investigated their ideal music sharing application. They wished to sell their music, advertise and share their music both within and outside OV, track interaction with their music and show their style with a profile.

## 2.1 Mobile Data Usage in Low Income Areas

Mobile data usage in low income areas in South Africa is limited due to the high cost of data relative to the income in these areas and variable coverage, resulting in low quality connections [3, 4, 5]. Mobile internet connections are the main way in which the internet is accessed in these areas as WiFi networks are not readily available to the public [5].

More than fifty percent of South African households have a monthly income of less than R1600 [3]. When compared to the cost of data bundles it becomes apparent why people are faced with bandwidth constraints.

**Table 1. Data Plans from Mobile Operators in South Africa [3]**

| Operator | Option 1 | Option 2 | Option 3 | Option 4 |
|---|---|---|---|---|
| MTN | 5MB 1-day voucher, R4 | 20MB 1-day voucher, R12 | 50MB 3-day voucher, R25 | 300MB 5-day voucher, R85 |
| Vodacom | 20MB 1-day voucher, R5 | 100MB 1-day voucher, R10 | 250MB 1-day voucher, R20 | 250MB 1-month voucher, R60 |
| Cell C | 20MB 1 - day voucher, R3 | 100MB 1-day voucher, R13 | 100MB 1-month voucher, R25 | 300MB 1-month voucher, R60 |
| Telkom | 25MB 1-month voucher, R8 | 50MB 1-month voucher, R15 | 100MB 1-month voucher, R29 | 250MB 1-month voucher, R39 |

Bandwidth related issues result in the residents of low-income areas having limited to no access downloading large files [6]. This plays a major role in the limited use of streaming websites and applications such as YouTube, [2, 3, 6, 7] leading to a different manner in which media is consumed. Residents of low-income areas prefer to download their media, given that it is only a few megabytes, directly to their device so that they can listen to or watch it as frequently as they want and not use data for the same media consumption [7].

However, while downloading files has become the norm there are further issues that come into play due to the limited storage space on their devices [2]. This is due to the fact that most of the devices used are predominantly low-spec smart phones [1, 2, 5, 6, 8]. This means that the digital persistence of an artist's music is important as their fans may have to delete their music to free up storage space, but then may revisit the website where they downloaded it at another time to re-download the song.

## 2.2 Music Sharing in Low Income Areas

One of the biggest issues South African artists face is their inability to share their music [9]. A web presence is believed to be a way to level the playing field with established artists and a way to reach new audiences [9]. Many artists believe it is more important to expose people to their music than receiving money for it [9]. This is due to the fact that they not only want to share their message but expanding their audience will allow them to have more live shows and get their music into stores, which many artists in South Africa struggle to do [9].

Music sharing in low-income areas has historically been dominated by Bluetooth and WhatsApp file sharing [1, 2]. There has only been one attempt to facilitate music sharing in these communities, which resulted in the development of a website called KasiMP3. It was designed for use on feature phones with data constrained users in mind [2]. It allows musicians to create a profile, upload songs, videos, pictures and has many features of a social media website [2]. However, musicians have started having difficulty accessing the website, which is believed to be due to latency issues [2]. In analysing the interviews done related to KasiMP3, it became apparent that the social media features of the website were not necessary, and the artists did not necessarily use the picture and video features. These issues were driven by the fact that both the artists and users faced bandwidth constraints. Their main requirement was a website that created an uncomplicated and data-efficient way to download music.

The use of Facebook is also a popular means of sharing music by artists in South Africa, but this requires the use of a third-party website for the downloading of the song, like the aforementioned KasiMP3 or an alternative called Datafilehost, which will be expanded upon below. The artist will create a post on their Facebook profile and share a link to a website where people will be able to go and download their song. Due to the fact that the majority of these artists set their Facebook profiles to public it becomes possible to search for their songs via Google and see these posts [2]. This is an important factor in the searchability of their music and offers the artist a digital presence which makes them discoverable to invisible audiences, an audience that the artist does not have a direct connection to [2]. However, the use of Facebook to share music in low-income areas becomes problematic due to the aforementioned data constraints residents of these areas face.

Datafilehost is an anonymous file sharing website that offers an online drop box which musicians can upload a file to without logging in, thus reducing the data needed to use the website [2]. It is not searchable, does not have a tagging process and does not offer descriptions for the file uploaded [2] which emphasises the importance of the use of public Facebook profiles to make the music uploaded to this website accessible. Even with all these issues the benefit of low data usage has made Datafilehost extremely popular. Datafilehost also offers a download counter which is important to the artists that use it; however, this has been found to be inaccurate [2]. Additionally, the file will be deleted from the website after 90 days of inactivity. When accessed from a device with antivirus software there are warnings of malware being present on the site [2].

## 2.3 The Significance of Music in Low Income Areas

Beyond the research into the technology related side of music sharing in low-income areas in South Africa, there has been numerous studies that have highlighted its social importance to these communities. There has been a focus on the Cape Town Hip-Hop scene, which is fundamentally different to the commercial Hip-Hop seen worldwide. It aims to challenge gangsterism and the inequality present in the country [2, 10, 11]. This is done by emphasising the importance of morals and ethics while highlighting the racial barriers that are still present in the country.

Many themes present in this music are inspired by Steve Biko and the Black Consciousness movement [2]. With musicians aiming to uplift their communities and disseminate positive and motivational messages.

Self-reflection, topical debates on Acquired immunodeficiency syndrome (AIDS) and globalisation are extensively present in the Cape Town Hip-Hop scene [11]. Artists aim to educate and change the way society views these issues using their music as their means of communication. Not only do artists tackle these socio-economic issues but they aim to encourage their communities to stay away from crime [10].

## 2.4 Global Music Sharing Trends

A general trend in the world at the moment is the increase in use of music streaming platforms such as Spotify, Apple Music and Deezer [14, 15]. The music world has moved away from purchasing individual songs and albums online, to a streaming model. This model is based on artists earning a certain amount of money per stream of their music [15].

Streaming of music is seen as a substitute for permanent downloads of music [15]. Globally consumers are moving away from storing music on their phone and rather stream it as they can consume whatever they want whenever they want without worrying about storage capacity.

These streaming services are offered on either a limited free to use basis or a premium version which the users pay a monthly subscription for [14, 15]. These services are offered on both fixed devices and mobile phones [14, 15], requiring an internet connection as data is used every time a song is streamed.

## 2.5 Means of Internet Access in Low-Income Areas

### 2.5.1 Internet Access via Phones.

Internet access is almost exclusively carried out using mobile phones in low income areas [1, 5, 6, 7, 8].

### 2.5.2 Internet Access via Community Networks

Some low-income areas in South Africa also have access to community WiFi networks which offer access to zero-rated services and general internet access for rates that are lower than mobile data rates.

The iNethi network in OV is an example of this. It was created to not only offer members of the OV community access to cheaper internet, but also encourage content sharing between the people in the community in an area where access to data is limited [13]. Residents of OV have access to a variety of free services and can browse the internet for twenty rand per gigabyte of data.

## 2.6 Insights Drawn from Related Work

There is a need for an application that allows data constrained users in low-income areas to share and consume music, with the special attention placed on mobile phone centred design. The motivation for such an application is not only economic but social in the sense that music in these areas can act as a medium for change and education.

With data constraints in mind this application cannot be based on a streaming model, like the conventional music sharing platforms globally, and should instead be download based. However, one factor that can be drawn from the modern music streaming trends is the social media-based features that allow you to see what other users are listening to as this could lead to users discovering new artists. Additionally, the economic status of the primary users, individuals living in low income areas, should be taken into account. While generating income is an important factor for the artists putting all the content behind a paywall will gatekeep a large portion of the primary user group. Artists also consider making a name for themselves more important than selling their music as live performances are more lucrative and exposure will lead to more of these. This could mean that, depending on the artist, the sale of music is of secondary importance when compared to the ability to share music more effectively.

With specific focus on the fact that this music sharing application will be zero rated for the primary users on the iNethi network, some features from KasiMP3, such as profiles and other social media features could be included. It is also evident that a download counter is a core requirement of the website.

# 3 System Design and Implementation

A waterfall model was chosen to structure the software development life cycle (SDLC). The waterfall model is a linear model in which a sequence of stages uses the output of the previous stage as input [12]. The SDLC is divided into the following stages: analysis, design, development, testing, implementation and maintenance [12]. The maintenance phase is not included in this project. The reason being that there was not enough time to deploy the system and have it used for long enough for any maintenance to be done.

This model was chosen as it was difficult not only to recruit participants due to their data constraints and COVID-19, but also challenging to stay in contact with them. This meant that an agile model, which was the planned method, was not feasible as there was no guarantee of being able to get consistent feedback from participants during the development process, which is a core aspect of the agile model [12].

Despite the difficulty engaging with prospective users the aim was still to involve them as much as possible. This meant that the analysis phase and testing phase was designed to incorporate them in the form of interviews and user-based testing.

All interviews conducted and the steps taken to document these interviews were approved by the UCT Ethics Committee with approval identity number 16920045. The interviewees were sent a consent form before interviews and informed consent was given by all participants. As compensation for their time all participants received twenty rand of airtime and a one gigabyte voucher for the iNethi network. All interviews were conducted virtually, via WhatsApp calls, due to the regulations the government had put in place that restricted in person interaction.

## 3.1 Requirements Elicitation and Analysis

The requirement analysis phase of the project revolved around interviewing prospective users, an OVCOMM Dynamic director and the members of the UCT staff that are involved in the iNethi network. These were the main stakeholders identified in this project.

The pre-interview phase, the planning phase, involved generating interview questions that were based on information gathered from the research in section 2. These questions were reviewed and validated during the planning phase by the project supervisor and the OVCOMM Dynamic director. This ensured the questions were appropriate and addressed all the important factors surrounding music sharing and creation in OV. Information gathered from interviews was integrated into future interviews in order to judge its validity and significance.

### 3.1.1 OV Based Interviews

The OVCOMM Dynamic director that was interviewed has a keen interest in music and was used as a point of entry into the music scene in OV. He was tasked with recruiting participants for the interview process. The participants that were interviewed consisted of two full time musicians and a musician that made music as a hobby. The musician that was a hobbyist was an avid listener to local music and offered insights into what a listener would want from the website.

### 3.1.2 UCT Based Interviews

The interviews with the UCT staff focussed on technology related topics and were used to assess what was feasible in terms of implementation. These occurred both pre and post interviews with prospective users in order to make sure the deliverables and milestones that were set throughout the project were acceptable and feasible.

### 3.1.3 Tech Stack

Interviews with the UCT stakeholders helped shape the tech stack used for this project and helped to identify the core complexities.

All software, except features that had to be run on the servers, had to be contained within docker containers. The use of docker containers was important as this is how the pre-existing system is structured. The reason being that docker provides a repeatable development, build, test, and production environment. This meant that all software could be worked on locally, in a sandbox environment and then seamlessly deployed on the iNethi and AWS servers.

WordPress had to be used for designing the website as WordPress does not require any coding experience to edit pages and configure plugins. This means it would be easy for a resident in OV with no programming knowledge to edit and adapt the website front-end and its features as they see fit. The goal of the iNethi project is to pass full ownership of all software on the network to OV residents, so the use of WordPress was non-negotiable.

The issue of synchronization between both websites was one that was identified as a critical point of importance. Music uploaded within Ocean View needed to be visible on the global instance of the website while downloads and coupon generation also had to be synchronized.

#### 3.1.4 User Access to Technology

All three artists had smart phones and said that their fanbase and the majority of OV residents did as well. Two artists had laptops that were mainly used as storage devices. One of these artists also pointed out that he faced storage issues, even though he had a laptop, and wanted to make use of cloud storage but did not have the data needed to upload his files. The artist without a laptop also faced storage constraints.

While all three artists had access to a device that could connect to the internet, one of the artists emphasised the fact that he, and many of his peers, had extreme data constraints. Due to this he would mainly use his data for WhatsApp. He did not use email due to data restrictions and wanted to know if he could receive notifications about his music's performance on WhatsApp.. The other two artists made use of email, Facebook and other social media applications but did not have access to uncapped data plans.

#### 3.1.5 Technical Ability of Users

The OVCOMM director suggested the website features a help page pre-populated with detailed explanations of the core functionality of the website and how to make use of these services. He indicated that this was necessary as the average user will not make use of technology beyond WhatsApp messaging. This meant that features of the website could not be assumed to be intuitive. This was corroborated by the other participants.

#### 3.1.6 Artist's Current Music Sharing Methods

All three musicians agreed that the most common way of sharing music was via WhatsApp groups and Facebook. One musician was of the opinion that Facebook was a dying platform, so he and his bandmates had stopped uploading their music there as it was seen as a waste of data. This particular artist had his music on SoundCloud and Spotify, but he had never listened to his music there and did not have access to these applications due to data constraints. His brother had set it up for him.

#### 3.1.7 Ideal Features of the Music Sharing Website

All the artists, when prompted about features they would like to see on the website, mentioned the need for a genre tagging system, the need for a profile page so that they share their details, an analytics page so they could track user engagement and a way for users to preview their music.

#### 3.1.8 Music Based Income and Income Generation Possibilities

The full-time musicians emphasised that their main source of income was from live performances. While they indicated that they would like to sell their music, they did not think it was viable for all music on the platform to be behind a paywall, specifically when referring to their audience within OV. They believed that the website would not be used if people had to pay for music as they said the majority of the community struggles to meet their everyday financial. This contrasted the director's opinion as he felt the most important feature of the website would be income generation.

There was more interest in using the website to grow their name and expand their audience. However, they were interested in a donation feature, which was viable as they stated that all the musicians they knew and the majority of people in OV had a bank account.

Another idea that came up was a system that allowed them to sell access codes to their music that is placed behind a pay wall. This would replace their need to sell Compact Discs (CDs) at live performances, which they said had become less lucrative over the years as digital consumption of music became more prevalent.

#### 3.1.9 Challenges Faced During the interview Process

Due to COVID-19 all contact with prospective participants and interviews had to be conducted virtually. Recruiting participants and maintaining contact virtually was difficult due to the data constraints they faced. Many messages to musicians did not deliver for days as they did not have data. This meant planning an interview was difficult as the artist were not certain as to when they would have data. As a result of this the possibility of continuous user participation throughout the project was not viable.

#### 3.1.10 Final Features List

Following the interviews with all the stakeholders, the following feature list was created:

- An upload page with genre tagging
- A profile system for musicians to share their details and banking information for donations

- A coupon (access code) generating system
- An e-commerce style shop for the music to be displayed in
- A music preview system
- An analytics system
- A help page with details on the website's features

## 3.2 Design

With the information gathered from the background research and interviews a feature list was created. This list was used to guide the creation of design documentation, see Appendix A. During the design phase it was decided that the two websites would differ slightly. The AWS instance would not have the option to register as a musician or upload music. This was done in order to make sure that only OV musicians made use of the upload system.

### 3.2.1 Architecture Diagrams

There are two instances of the music sharing website. One hosted by AWS and another to be deployed locally on the iNethi servers. The architecture for both is almost identical with the only difference being that the AWS server runs on Hypertext Transfer Protocol Secure (HTTPS) requests while the local iNethi server runs Hypertext Transfer Protocol (HTTP) requests, which will soon be changed to HTTPS (see A.1.1 and A.1.2). Additionally, the local architecture features a python-based application programming interface (API), the system management API, that is used to process user coupon requests.

All the docker containers are connected to a docker bridge network, which allows the connected dockers to communicate with each other. A HAProxy docker container is run as a reverse proxy server on both servers. This takes in HTTPS requests in the case of the AWS server, verifies and decrypts them and passes them on to the WordPress docker container as a Hypertext Transfer Protocol (HTTP) message. In the case of the local network The HAProxy container routes HTTP messages to the WordPress docker container or the system management API.

On both servers the WordPress container carries out the request and returns a HTTP response to the HAProxy server, which is then encrypted in the case of the AWS server and sent as a HTTPS response back to the user. In the case of the local server the process is the same except the response is sent in HTTP format. Optionally the WordPress docker queries the connected MySQL database, hosted in a MariaDB container, if data is needed from it. All docker containers are mounted to local folders on the server. This means even if the container crashes or is restarted none of the data will be lost.

In the case that a request is made to the system management API on the local server it will verify that the source of the request is from the Uniform Resource Locator (URL) of the local website and then carry out the request and send a response in HTTP format to the user.

### 3.2.2 Use Case Diagrams

The music consumer and musician have access to different features of the website. The musician can make use of all the features of the website while a customer can only browse and download music (see A.2.1). Whereas the musicians can upload music, create a coupon generation pin and subsequently request coupons (see A.2.2). The access to features is all based on the user's permissions which are generated at registration depending on which registration form they fill out. The form to register as a musician is not available on the AWS server.

### 3.2.3 Sequence Diagrams

The coupon generation process (see A.3.1) makes use of the system management API to process user input and coordinate coupon creation between the local and global server. The API uses calls to the WordPress WooCommerce API to create the coupon using information gathered from the local database, where the product ID I stored for both the local and global product instance.

The music upload process makes use of backend processes run on the server (see A.3.2) to coordinate product creation between the local and global server. These processes include updating the file name, moving the file into a permanent location, syncing files between the servers, reading from the database to get the user's submitted information and then using the WordPress WooCommerce API to create the product locally and globally.

## 3.3 Development

### 3.3.1 System Configuration

The project development phase started with choosing the correct docker images and setting up docker containers. This involved creating a 'docker-compose' file that is invoked from the terminal and starts and links the containers to the bridge network. The base image of WordPress was chosen while the 'mariadb/server:10.4' docker image was chosen as it allows the local host to connect to the docker container from outside of the docker bridge network. This allows the localhost machine to query data from the database. This is an important part of the product creation process.

Within the compose file the environment variables are set, such as database users, passwords and the folders where the containers' data will be mounted. Additionally, a custom PHP configuration file was written for WordPress in order for files larger than two megabytes to be uploaded as this the WordPress default.

There is a pre-existing HAProxy server container running on both the servers the websites will be deployed on. This meant there is no reference to the HAProxy in the compose file, but the configuration file for the HAProxy had to be edited to route traffic to the correct docker container.

### 3.3.2 Plugins

The following plugins were added to the website via the WordPress plugin library:

- Code Snippets: stores PHP code that runs on the website so that the code does not need to be added directly to the

current theme's 'functions.php' file. This means the theme can be changed and custom PHP code will not be lost.

- Contact Form 7: a form generation plugin that was used to create the music upload form and coupon processing forms.
- Contact Form 7 CFDB7: forms created with Contact Form 7 do no save the information that is submitted. This plugin enables the data to be written to the database.
- Ultimate Member: a plugin that creates social media style features for the website. It manages the profiles; the access different profiles have to tabs and creates a searchable directory of musicians.
- WooCommerce: this is an e-commerce toolkit that allows for product and coupon creation, however, only the website owner can access these features.

### 3.3.3 iNethi Music Sharing Package

The scripts used to interact with the MariaDB docker container and WooCommerce Store were added to the iNethi music sharing python package (see A.4.1).

#### *3.3.3.1 Database Manipulation*

All processes that required reading or writing from the database used the MariaDB python library. A JSON configuration file is used to gain access to the database. Subsequent queries are run using MySQL. The core functionality of this code is used to read in information related to uploads, update the downloads table as well as provide helper methods for the system management API.

Three tables were added to the default WordPress database. These tables store coupon data, download data and mappings of the AWS WooCommerce product IDs to the local WooCommerce product IDs.

#### *3.3.3.2 WooCommerce API*

All processes that required entries into the store were done using the WooCommerce python library. This required a secret and public key to be generated from the WooCommerce plugin dashboard. These were used, along with the URLs of the stores to add products, create coupons and calculate the number of downloads a song has. The same keys are used to access both stores as the database and WordPress files were duplicated upon the initial deployment.

This python script uses details from the database as well as details that are embedded during file uploads to facilitate product creation, coupon generation and to calculate the number of downloads each song had.

User product creation is started when a user uploads a file via the 'Music Submission' page. They have to be signed in to do this. This makes it possible to embed their username in their submission, which is added to the product description and attribute field. This is necessary in order to be able to track song ownership. Additionally, when the song is uploaded there is a check to see if the user has already uploaded a song with the same name. If they have, the name is updated so that every song a musician uploads has a unique name. This makes it possible for the user to select the song they wish to create a coupon for at a later stage. The product ID in the backend will be different for each song, even if the name is the same, however the user will not see this ID so unique names are necessary. However, different artists can have songs with the same name.

The product creation process sets the download count of the song to zero in the downloads table and enters the date and time it was uploaded. Additionally, the genres the user has specified, and their artist name is converted to unique IDs for the search and genre tagging system.

Due to the fact that users can only upload songs while in OV, the product creation file only exists on the local server. However, it also adds an entry to the AWS product library. If both of these actions are successful a JSON object is returned with the details of both songs, this is verified by the script otherwise an error is logged and saved to the server.

The invocation of product creation is done on the server when a file is saved in the mounted folders as this indicates the song was successfully uploaded. If it were triggered by the user submitting the form on the front-end there is no way to guarantee the song has successfully reached the sever. Additionally, this process was not migrated to the system management API due to the fact that the current local network runs on HTTP. This makes it easy for someone to intercept the data sent to the API and change it with ease due to the lack of encryption. With this being said all messages sent to the AWS server use HTTP however by the time they are sent they are already encrypted using aforementioned secret key.

### 3.3.4 The System Management API

The system management API (see A.4.2) uses the Flask python library to create a web application that is run in a docker container. The library allows the creation of URL endpoints that are used to invoke functions. By default, all Flask applications restricts Cross-Origin Resource Sharing (CORS) so that the API cannot be used by a domain outside the domain from which the resources were served. To circumvent this the URL of the locally hosted website was whitelisted within the Flask application.

When the user submits a pin creation form the API verifies that they are a registered user and have not already created a pin. If they pass these checks the user is sent a response verifying that their pin has been assigned. This pin is hashed using multiple rounds of the SHA-256 algorithm and then stored in the 'users' table in the database. This is a one-way hash so the user's pin can be verified if the original pin is submitted but it cannot be recreated from the stored hash. The pin is needed as there is no other way for the API to verify the user. The WordPress Plugin

used for user management hashes passwords before adding them to the database. While these hashes could be used to verify the user there is no documentation on the algorithms used. Although the user needs to be logged in to send the coupon creation form, the additional layer of security is added to protect the user's content in the case that a malicious attempt to create a coupon is sent to the API.

When a user submits a coupon-creation form they submit their pin and the song name that they wish to generate a coupon for. Their username is embedded in this request in order to verify if they uploaded the song. If their details pass the verification stage, then the WooCommerce API is used to create a coupon for the specified song. This requires the product ID to be retrieved from both servers as they may be different. Once these are obtained, the system generates a random 10-digit alphanumeric code that is checked against pre-existing coupon codes and returns this to the user. If the user did not enter correct details, they will be sent an error message.

The system management API is also used for tracking downloads. The WooCommerce API does not allow access to downloads on a per product basis. Downloads have to be checked using the order logs for both stores. When a song's page is loaded an API call occurs and the order logs are checked. Each time the download counter is updated an entry is made in the 'downloads' table specifying the date and time that this update occurred. The date and time are checked every time the API call is made and if the download count has not been updated within a thirty-minute period then the order logs are checked.

Although there were multiple approaches taken to secure the API the fact that the data is transferred by HTTP makes the data easily susceptible to interception. The HTTPS features that are present on the AWS servers are being migrated to the local network so this issue will be short lived. Although security was the core reason to keep product creation server side, there is no other feasible way to facilitate coupon creation. Adding triggers to the database to export a file with the request data to a locally mounted folder was an option. However, this implementation took far longer to process the requests as it required file reading and writing as well as a cronjob to run in order for the coupon creation process to start.

#### 3.3.5 JavaScript Scripts

JavaScript scripts were written and embedded in the web frontend to carry out API calls. Three scripts were written to send post requests to the system management API. Another was written to send a get request to a third-party API to verify musicians bank details. The jQuery library was used for this.

When a musician submits their profile details to signup via the 'Musician Registration' tab an API call is made if they have filled in their bank details. The 'FreeCDV' API is used to verify that the user has entered valid details. A response message is then displayed to them to indicate whether their details are valid or if there is a mistake in their submission. If there is a mistake a detailed error message is returned to them and they are told how they can change their details.

Additionally, two separate JavaScript scripts were written to deal with handling the coupon system. One script was embedded on the 'Create a Permission Pin' page and another on the 'Coupon' page.

On the 'Create a Permission Pin' page a user enters a 5-digit pin and a post request is sent to the system management API. The afore mentioned processes, from section 3.5.4, are run by the API and the response is displayed to the user.

Similarly, on the 'Coupon' page a post request is sent to the system management API. The aforementioned processes, from section 3.5.4, are run by the API and the response is displayed to the user.

The final JavaScript script was written to update the download counter displayed on a songs page. The song name and the username of the uploader is sent to the system management API when the page is loaded, and the aforementioned processes are carried out by the API. The response is then returned, and the counter is updated.

#### 3.3.6 Cronjob File Management

Every minute a cronjob runs on the WordPress upload folder that is mounted to the server. This job loops through each file in this folder and carries out four processes. Firstly, the files have their name appended with the date and time so that files uploaded with the same name will not be overwritten. Once the files are renamed the product creation script described in section 3.3.3 is run and entries. The files in this folder are then moved to a different directory as the default upload folder is cleared every twenty-four hours. Finally, the files are synced with the AWS instance using rsync and the secure shell file transfer protocol (SFTP)

#### 3.3.7 Plugin Customization

PHP code was added, via the Code Snippets plugin, to remove the required fields from the WooCommerce checkout page, such as the address fields and payment details.

#### 3.3.8 Difficulties Faced Due to the Tech Stack

Docker containers are designed to only share data and connect to other containers connected to the same docker network. This meant that the WordPress and MariaDB containers running on the AWS server and the iNethi server were completely independent. Additionally, the websites have a different feature lists which made the synchronization processes difficult as they could not be direct copies of each other.

Another issue at hand was running scripts on the server when an action was carried out on one of the websites. Since the WordPress image is built to run as a standalone system only interacting with other containers sharing its bridge network it is also difficult to run a script within the container and communicate with the host machine. This problem was solved by creating the system management API that runs in a separate 'python-3.8' docker container.

## 3.4 Testing

All the test detailed below were carried out in the development environment.

There were two types of testing conducted; manual testing for user interface (UI) features and automated tests for the API features.

All automated tests were run using Postman. Postman is a tool that allows API calls to be tested by entering the relevant data and creating a post request, in this case, that is sent to the relevant API URL. Tests are then run on the response data using JavaScript and the output is saved as JSON to a local directory.

While this was used to test the backend code, manual tests were used to check whether the UI based responses were displayed correctly based on input. These manual tests were conducted on both a computer and a mobile phone due to the fact that the majority of the target user group will be using phones to access the website

Tests for the following project specific processes were created and run:

- The song upload to product process
- The coupon pin creation process
- The coupon generation process
- File and database syncing between the two website instances
- The bank details check process
- The song downloads indicator

#### 3.4.1 Song Upload and Product Creation

Song uploads are a multi-part process that rely on server side and front-end processes. Due to the fact that the music submission page requires a user to be logged in, nonce verification and cookie verification it was a difficult process to get Postman to successfully send an acceptable post request to this page. This testing process failed and was subsequently tested manually. Built in functions throughout the 'create product' process display error messages to the docker logs and save errors to log files in the mounted folders on the server. This meant it was possible to upload songs and verify if they were successfully processed. Additionally, the store could then be manually checked for the song and a download could be attempted from both stores. This process was successful.

#### 3.4.2 Coupon and Coupon Pin Generation

The user pin creation and coupon generation process were tested with linked tests that were run using the 'Runner' Postman feature that runs a set of tests in a specified order. The following tests were run in this order: registering a pin with an existing user passes, the user from the last test tries to add a pin and gets a failure response, the previous user then creates a coupon for a song they uploaded which passes, that user then tries to create a coupon for a song they did not upload and it fails and finally the same user tries to create a coupon for a song with the incorrect pin and it fails. Additionally, a blank post request was sent the API which failed correctly.

All of the above tests, except the last one as it is not possible to send a request without a username, were carried out manually to test the UI based responses and they all passed.

#### 3.4.3 Database and File Sync between Instances

The database syncing occurs when the WooCommerce API calls are made. If the API call is successful, then the database has been successfully updated. This means that any failed attempts to synchronize the database would be added to the error logs. During the testing period there was no errors added to the log.

The file synchronization process is carried out using rsync which keeps error logs if a synchronization fails. However, because the synchronization is carried out in the files' permanent location this means that if the synchronization fails on one occasion the file will be synchronized on the next call. This means that the only way that this process will ever fail is if the connection to the AWS server is permanently severed or if the cronjob stops running.

#### 3.4.4 Bank Details Check

This was tested with Postman and manually. The Postman tests involved entering incorrect bank details, bank details missing fields and then correct bank details. These were all sent to the API using get requests and the responses for all tests displayed the correct output.

The manual tests were used to see whether the UI displayed the correct messages based on the details entered. All these tests passed.

#### 3.4.5 Song Download Counter

The songs download counter was tested using Postman and tested manually. The manual checks verified that the correct number was being displayed on the UI. The following Postman tests were run: querying the downloads for a newly uploaded song, querying the downloads for a song multiple times within a thirty-minute period and querying a song that had been downloaded a known amount of time. All these tests passed.

## 3.5 Implementation

#### 3.5.1 Test Environment

All features were developed using a local environment that synced with the AWS sandbox server that iNethi projects are tested on. The local environment as well as the AWS server mirrored the production environment. HTTPS was used on the AWS server while HTTP was used on the local machine to make sure the API, along with all the other processes could function even with the different communication methods.

#### 3.5.1 Deploying to Production Severs

The deployment onto the local iNethi server was unsuccessful and resulted in the inability to conduct the planned user testing. While

the AWS environment deployment was successful, the deployment on the local servers failed due to the libraires used for the backend processes, specifically the MariaDB python library. Although there is a stable release for the operating system the server runs it was not possible to get it to install and configure properly. This meant that there was no way to process song uploads. The front end, databases and API were all successfully launched on the local server, however, due to the core feature of the system not functioning the website has been taken down. However, successfully launching the website on the local server would only take a small revision in the code base. The system management API could be extended to incorporate all the database functions and WooCommerce API functions with very few changes necessary. The architecture of both systems would not need to change in this process and no changes would need to be made to the AWS code base as it was deployed successfully.

## 4 Discussion

While the intended co-design aspect of this project was not possible to carry out as planned, there was still valuable input gained from artists in the virtual interviews. Additionally, the aforementioned difficulties communicating with artists did limit the sample size of OV participants. However, the information gathered from participants aligned with the insights drawn from background research indicating that the feature list and direction of the project was appropriate and representative of what artists in OV want.

That being said, it is important to note that the ideal deliverable was not created as there is no way to purchase songs directly off the website, the analytics page was replaced with a download counter on each product page and there is no way to preview music. This was not only due to time constraints, but also the way in which OVCOMM Dynamic is run. There was no viable way to create a central bank account to process card payments through, as there was no one that would be able to monitor. This means that the only financial compensation artists can receive for their music is through the coupon system. However, the artists did not put financial compensation at the top of their priority list, so this set back does not severely affect the impact this system can make. Additionally, features such as WhatsApp notifications were not feasible due to the cost involved in deploying a system like this. The WhatsApp API is not free to use and as such a private company would have to be employed to develop and maintain a system like this.

While the deliverable was not the embodiment of all the artists requirements, if it is maintained and it becomes popular it has the ability to fulfil their core needs and provide them with a platform, they can use to share their music without the hinderances of bandwidth constraints.

In light of the unsuccessful deployment to the local server there are a number of lessons that can be drawn from this process. Despite the security concerns of hosting an API over HTTP, it is a safer option to develop dockerized APIs when working on the iNethi server rather than rely on the server to run backend processes. Especially with the knowledge that the system will be changed to HTTPS in the future. Additionally, there are socially based lessons that can be taken from this process. The project's supervisors warned of promising any potential users anything in terms of deliverables in case there was an issue at any point in the project that hindered deployment. This advice was adhered to throughout the process and while the implementation of the final deployment is achievable with the a few adjustments to the code base, the interviewees were not left disappointed by false promises. This is a valuable lesson that can be applied to all research projects in which the participants are heavily invested in the outcomes.

## 5 Conclusions and Future work

Conducting a user-centred project during the early stages of COVID-19 was extremely challenging. This coupled with the fact that virtual interviews were difficult to organise due to the potential users' bandwidths constraints meant that there was a heavy reliance on input from a small sample group. Although the participants feedback aligned with previous work it became clear that research that involves bandwidth constrained participants is heavily reliant on the ability to break virtual boundaries and interact with them in person. There was difficulty establishing lines of communication due to these factors so even incentivising participation, in the form of airtime and data, was not successful.

Over and above this it is important to contextualise research and understand the social dynamic at hand. This was evident by the fact that musicians would prefer to have their music freely accessible due to the financial restraints their fans face rather than placing it behind a paywall. A sentiment that was not shared by other stakeholders in the project that thought that economic gain would be at the top of the priority list for OV musicians.

With this being said it is also important to note that this system has the potential to introduce the musicians' work to an audience outside of OV in a way that was previously not possible. This in itself is reason to develop the system further and deploy it successfully.

The majority of the musicians requirements have already been met by the current system, however, future work would include dockerizing the product creation process so that the system will run on the iNethi local server, adding a separate analytics page for each artist and allowing the artist to edit and update their songs in the shop. All of these features would require small additions to the existing API as the core functionality is already present. Additionally, a way to automate the handling of the error logs would make the system more efficient.

## ACKNOWLEDGMENTS

I would like to thank my supervisors Dr. Hafeni Mthoko and Dr. Melissa Densmore for their support and guidance throughout my research. I would also like to thank Dr. David Johnson for his guidance and help developing my system.

# REFERENCES


[1] A. Schoon. 2014. Digital hustling: ICT practices of hip-hop artists in Grahamstown, Technoetic Arts a Journal of Speculative Research, 12 (2014), 207-217. DOI: 10.1386/tear.12.2-3.207_1

[2] A. Schoon. 2016. Distributing hip-hop in a South African town: from the digital backyard studio to the translocal ghetto internet. In Proceedings of the First African Conference on Human Computer Interaction, 104-113. DOI: https://doi.org/10.1145/2998581.2998592

[3] A. Phokeer, M. Densmore, D. Johnson, and N. Feamster. 2016. A First Look at Mobile Internet Use in Township Communities in South Africa, ACM DEV '16: Proceedings of the 7th Annual Symposium on Computing for Development., 15 (2016), 1-10. DOI: https://doi.org/10.1145/3001913.3001926

[4] A. Phokeer, M. Densmore and D. Johnson. 2016. Characterisation of Mobile Data Usage in Township Communities, Proceedings of Southern Africa Telecommunication Networks and Applications Conference, DOI: 10.13140/RG.2.2.36137.80486

[5] S. Hadzic, A. Phokeer and D. Johnson. 2016. Townshipnet: a localized hybrid TVWS-WiFi and cloud services network, International Symposium on Technology and Society., 1-6.

[6] I. de Lanerolle. 2012. The New Wave: Who connects to the Internet in South Africa, HOW they connect and what they do when they connect., DOI: 10.13140/2.1.1391.6485.

[7] A. Schoon. 2014. Digital hustling: ICT practices of hip-hop artists in Grahamstown, Technoetic Arts a Journal of Speculative Research, 12 (2014), 207-217. DOI: 10.1386/tear.12.2-3.207_1

[8] T. Kreutzer. 2009. Assessing cell phone usage in a South African township school, Int. J. Educ. Dev. Using Inf. Commun. Technol., vol. 5, no. 5 (2009), 43–57.

[9] G. Pritchard and J. Vines. 2013. Digital Apartheid: An Ethnographic Account of Racialised HCI in Cape Town Hip-Hop, Conference: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2537-2546. DOI: 10.1145/2470654.2481350

[10] H. Becker and N. Dastile. 2008. Global and African: exploring hip- hop artists in Philippi Township, Cape Town, Anthropology Southern Africa, 31:1-2, 20-29. DOI: 10.1080/23323256.2008.11499960

[11] D. Marco. 2011. Rhyming with "knowledge of self": the South African hip-hop scene's discourses on race and knowledge, Muziki, 8:2, 96-106, DOI: 10.1080/18125980.2011.631303

[12] S. Balaji and M Sundararajan Murugaiyan. 2012. Waterfall Vs V-Model Vs Agile: a Comparitive Study on SDLC, International Journal of Information Technology and Business Management, vol. 2, no. 1 (2012) 26-29.

[13] M. Lorini, M. Densmore, J. David S. Hadzic, H. Mthoko, G. Manuel, M. Waries and A. van Zyl. 2019. Localize-It: Co-designing a Community-Owned Platform, Locally Relevant ICT Research, 1-15. DOI: 10.1007/978-3-030-11235-6_16

[14] L. Aguiar and J. Waldfogel, 2017. As streaming reaches flood stage, does it stimulate or depress music sales?, International Journal of Industrial Organization, Volume 57, 278-307, DOI: https://doi.org/10.1016/j.ijindorg.2017.06.004

[15] L. Aguiar. 2017. Let the Music Play? Free Streaming and its Effects on Digital Music Consumption. Information Economics and Policy, volume 41, 1-14. DOI: https://doi.org/10.1016/j.infoecopol.2017.06.002

[16] S. Bhattacharjee, R. Gopal, G. Lawrence Sanders. 2003. Digital music and online sharing: software piracy 2.0?, Communications of the ACM, Vol. 46, No. 7, 107-111, DOI: 10.1145/792704.792707

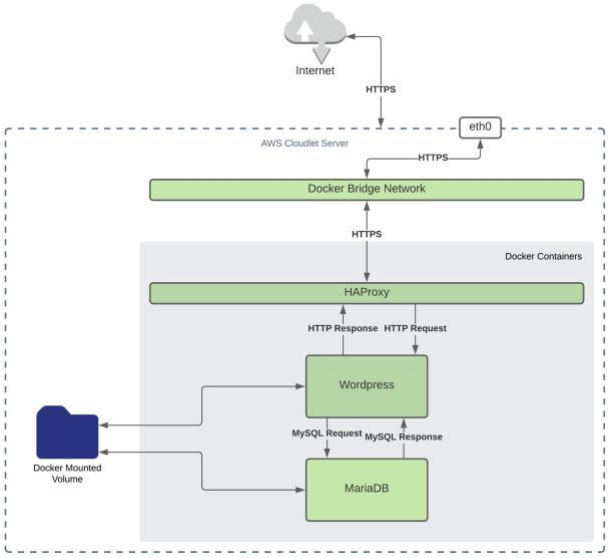
[17] V. Setty, G. Kreitz, R. Vitenberg, M. van Steen, G. Urdaneta, and S. Gimåker. 2013. The hidden pub/sub of spotify: (industry article). In Proceedings of the 7th ACM international conference on Distributed event-based systems (DEBS '13). 231–240. DOI: https://doi.org/10.1145/2488222.2488273

# Appendices

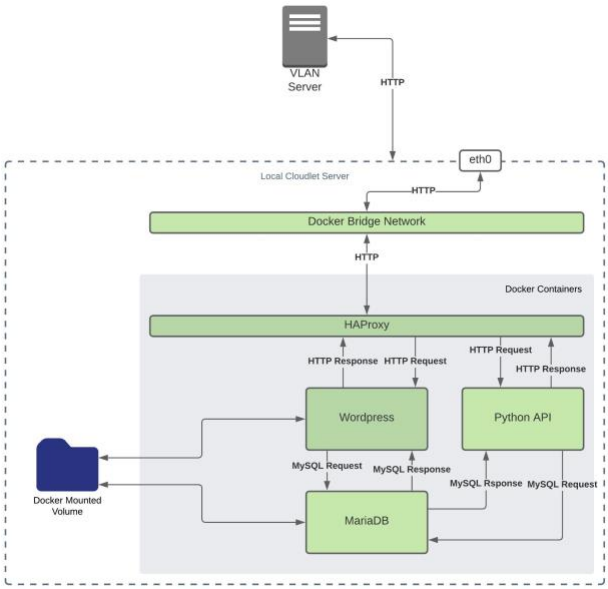
## A Software Design Documentation

## A.1 Software Architecture Diagrams

### A.1.1 AWS Cloudlet Server

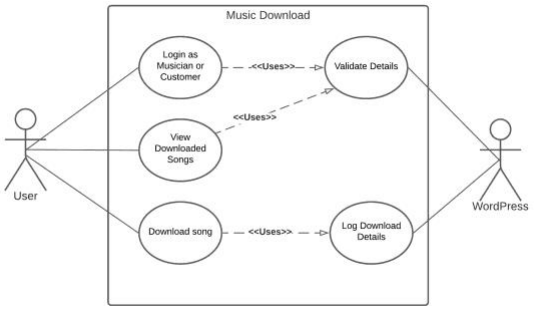



### A.1.2 Local Cloudlet Server Architecture

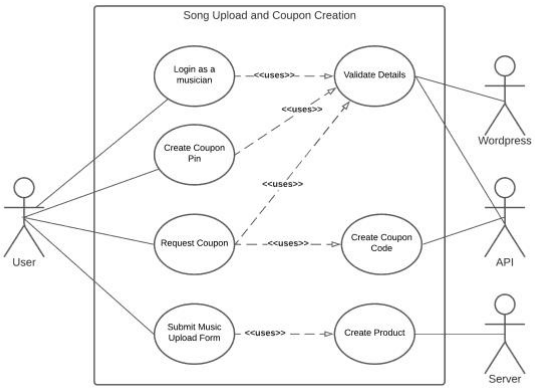



## A.2 Use Case Diagrams

### A.2.1 Music Download

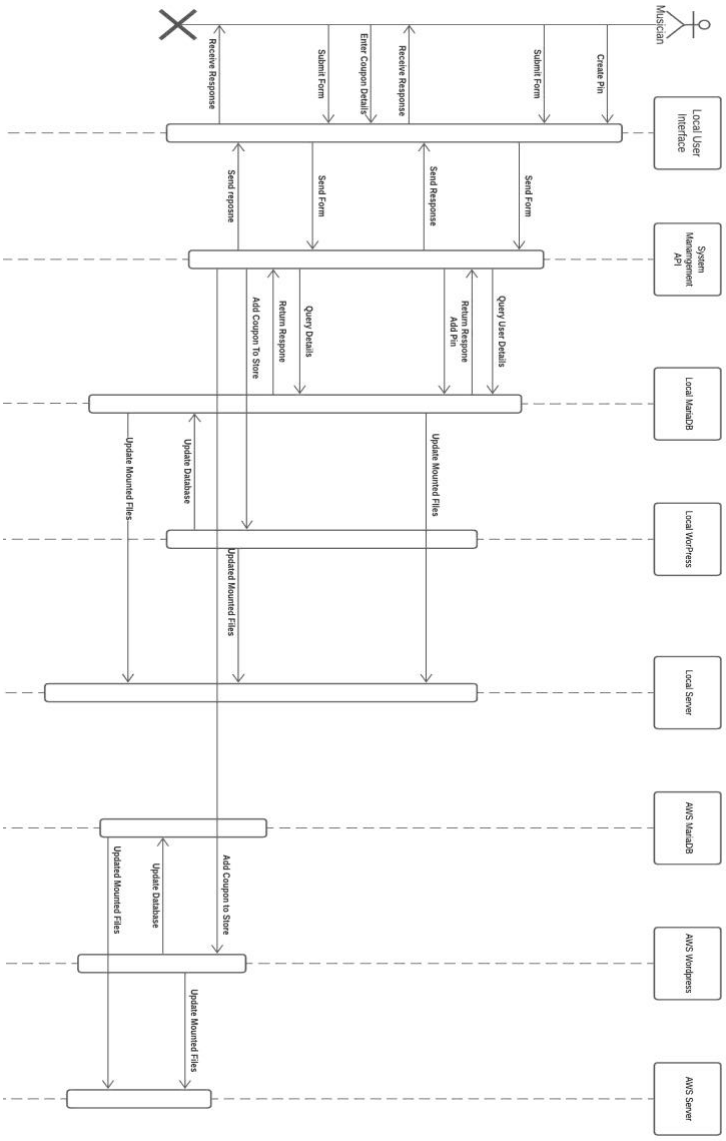



### A.2.2 Song Upload and Coupon Generation

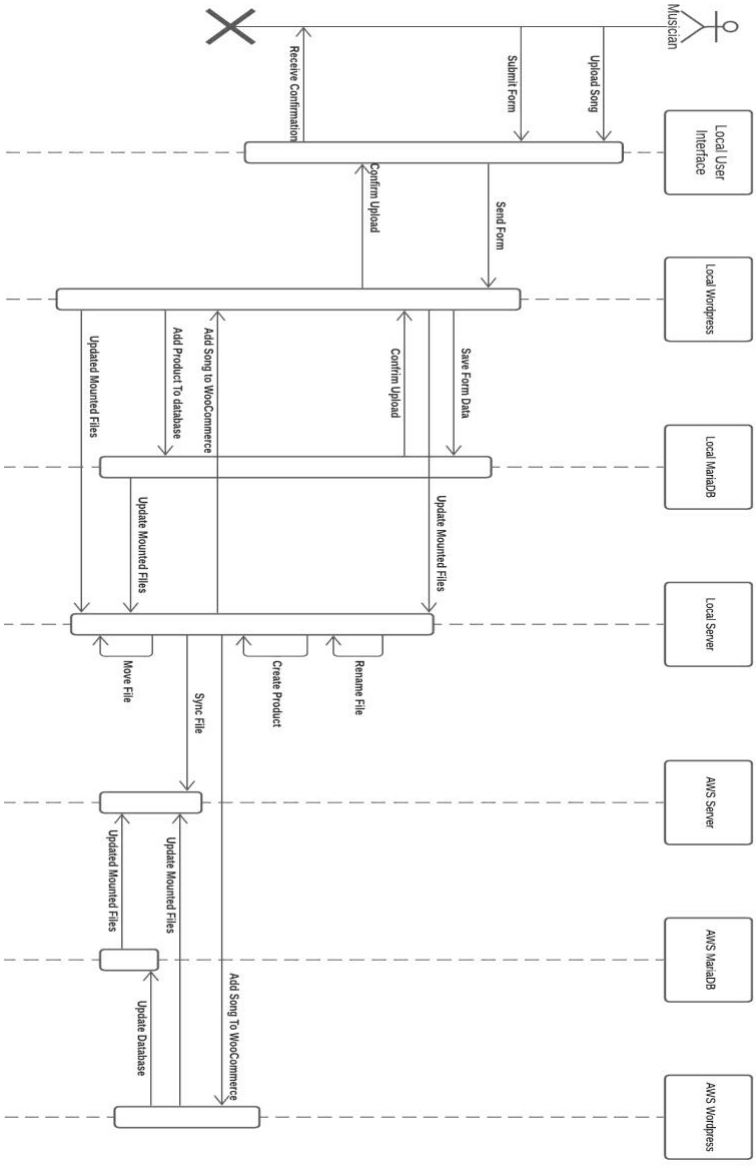


## A.3 Sequence Diagram

### A.3.1 Coupon Generation

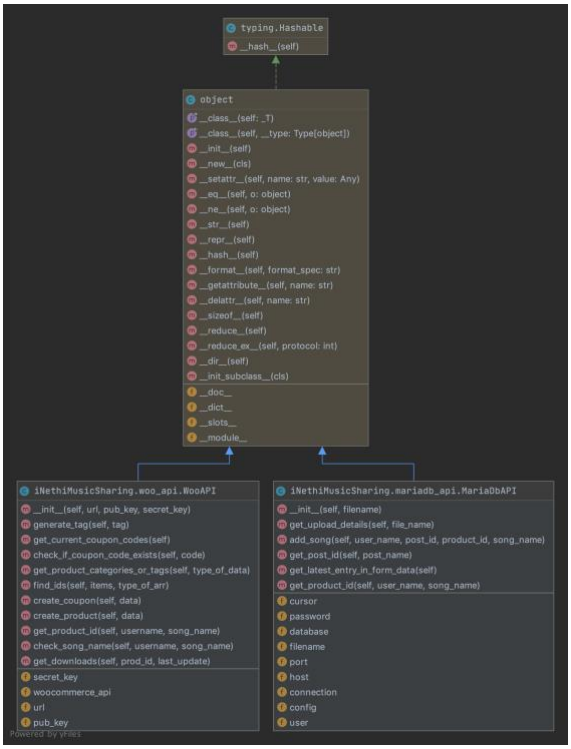



### A.3.2 Music Submission



## A.4 Class Diagrams

### A.4.1 iNethi Music Sharing Package




### A.4.2 System Management API