**Vision document**

# Revision History

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 04.10.21 | 1.0 | Initial version | Group 14 |
|  |  |  |  |
|  |  |  |  |

[Revision History](#_rp4qga7whlx) **1**

[1 - Introduction](#_1d6d28jei1yq) **2**

[1.1 - Purpose](#_b2vbu97fde8w) 2

[1.2 - Solution Overview](#_bhaokrrz9lcy) 2

[1.3 - System requirements](#_8h81wo2fwtx3) 2

[1.3.1 Functional requirements](#_gha90guec180) 2

[1.3.2 Non-functional requirements](#_9l78fd29r5if) 2

[2 - Plan](#_1903qa48wri5) **3**

[2.1 - Iteration plan](#_376c491vmihb) 3

[2.2 - General architecture](#_jejmpmdqw57r) 3

[2.2.1 - Model](#_lo4bxuls3elf) 3

[2.4 - Risk assessment](#_n4bkao87jzks) 3

[3 - Tools & responsibilities](#_pu6mcwr37ugh) **4**

[3.1 - Storage](#_ynugouogyrva) 4

[3.2 - Communication](#_klvaetsw0v7u) 4

[3.3 - Development Environment](#_7sgssbp51abc) 4

# 1 - Introduction

## 1.1 - Purpose

The system primarily consists of two lists. One list contains the products to be selected, and another containing the already selected products.

## 1.2 - Solution Overview

Our solution is to scrape their system and collect the names of the different missing products, and their ID, add the product to a document to be prepared to be printed out. This is to digitalize the struggle of having to manually write down the missing product out on paper.

## 1.3 - System requirements

### 1.3.1 Functional requirements

The system will scrape a specific website for information regarding products that a customer has ordered. The system will note each product it has selected and each product it should have selected, but could not because the product is not in stock in the store.

The system will then return a list to the user, showing which products it has selected, and so forth.

### 1.3.2 Non-functional requirements

The non-functional requirements for our system are simple and consistent. The interface should be easy to use and navigate through. What we mean by this is having the different functions of the system clearly labeled and visible in the system so all the workers using the system can properly use it without massive issues.   
  
Regular people, as in people that are not a part of the staff team of the organization using that specific system should not be able to connect to the system and use it, due to them not being a part of the organization.   
  
Obviously, we want the system to be as good as possible when it comes to system up-time. The goal is 24/7 uptime, but this is also a very far-fetched goal, and likely impossible due to updates to the system, shutdowns, bugs, which causes the system to be turned of or restarted.   
  
We also want the system to be built in such a way that it can easily receive updates without massive issues figuring out what is in our code, database, etc. FInding out these problems if not solved for you, can take massive amounts of resources and time, and should be avoided if possible.

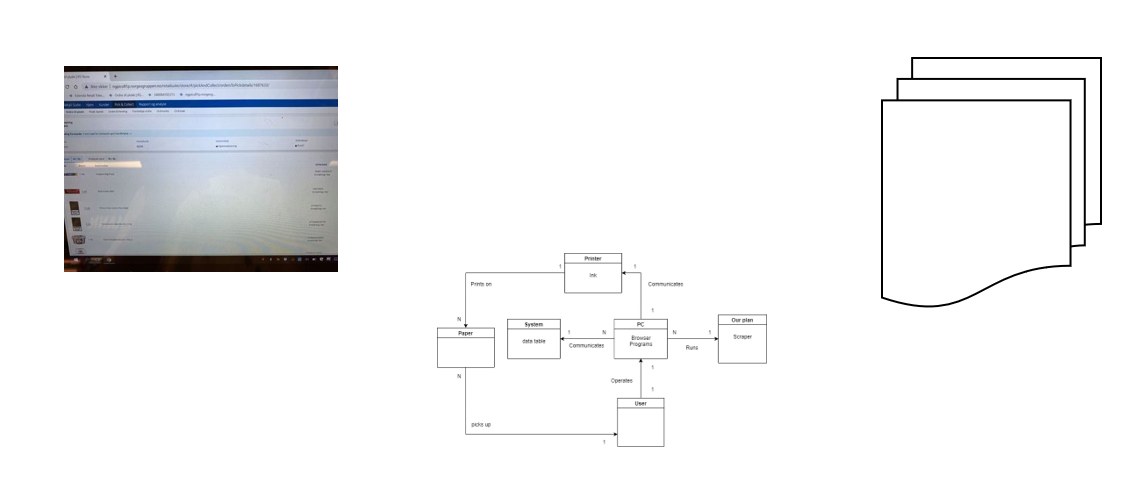
# 2 - Plan

## 2.1 - Iteration plan

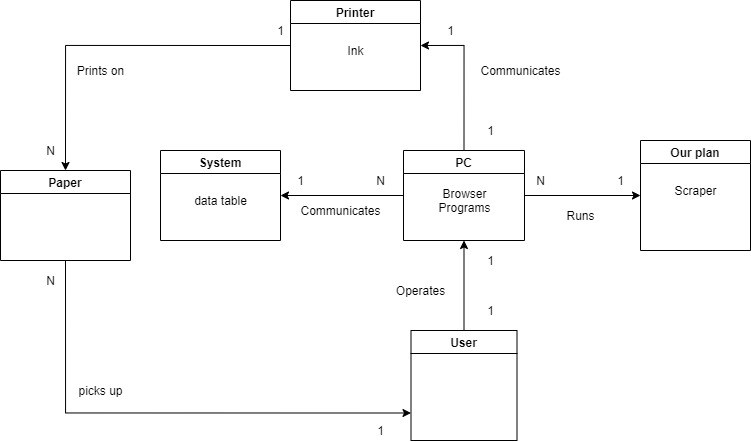
For our project, we have chosen to include a service to keep track of our iterations and what we have completed, and what is to be completed. This service is called Trello and hosts a service to create a digital Kanban board.  
We want to include this Kanban service in an agile development method like Scrum, where we can use the Kanban boards for sprint meetings, or more general meetings.

We have currently decided that we currently are working together on Thursday, and four of us on Tuesdays during seminars. We are still working on finding suitable time’s to work together on our project.

## 2.2 - General architecture



### 2.2.1 - Domain Model



## 2.4 - Risk assessment

Currently, there are no major risk factors that could potentially damage the systems. There are risks to every system, that it might have some downtime due to the servers being down, and/or having bugs that would cause the system to be unavailable. These risks are fairly low at this moment in time, due to not too many people using the system, so not too many people will notice the bugs till the bugs are resolved.  
  
The system would not hold sensitive or massively important content that would cause damage to the establishment if the content got leaked or stolen after a security breach of some kind.

# 3 - Tools & responsibilities

## 3.1 - Storage

For storing our documentation, and projects, we use a collaboration platform for sharing files and working in groups called GitHub. In GitHub, we can upload files and keep them synchronized with the rest of the group. This is to avoid having multiple versions of the same file stored many places at one time, containing different pieces of data that are not contained inside the rest of the files.

We also use a tool called Trello to keep track of the tasks that we need to do, are being done, or have been done, and we use it as a sort of kanban table.

## 3.2 - Communication

For communication within our project, we use two different programs for maintaining digital communication for our project when we are not working physically on campus. Two programs we use are Discord, and Facebook Messenger.

## 3.3 - Development Environment

For our project, we will most likely use a small array of development environments. The IDEs we have so far is PyCharm, which is an IDE for python code. We have also chosen to include Atom (general text editor), and Visual Studio Code (IDE) which can write in a vast amount of languages, even though we will mostly be using python to develop.

Backup plan - Music Streaming Service

**Solution overview:**

[Music] The solution will consist of a large database of music which is stored on a server, or something similar. A user will be able to access all the music on the database, and either stream music directly from it, or download music from the database to play it at a later time.

**Purpose:**

[Music] The system will provide a service which will allow a user to browse music through an application. It will feature simple ways of creating playlists, adding songs to the playlists, and browsing your friends’ playlists.

**Iteration plan:**

[Music] For our project, we have chosen to include a service to keep track of our iterations and what we have completed, and what is to be completed. This service is called Trello and hosts a service to create a digital Kanban board.  
We want to include this Kanban service in an agile development method like Scrum, where we can use the Kanban boards for sprint meetings, or more general meetings.

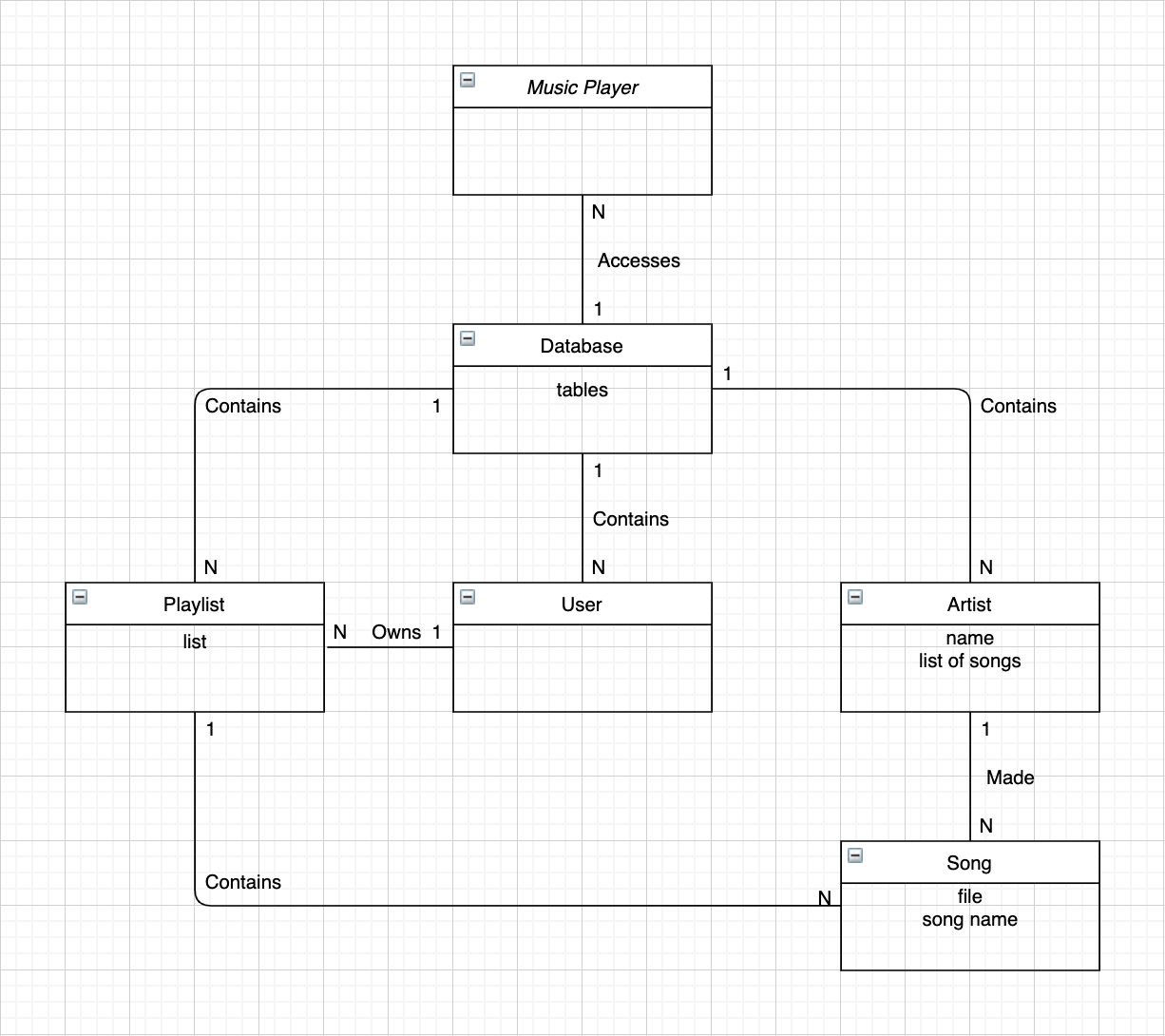
**Risk assessment:**

[Music] There will always be risks to a data system or an application, but we generally consider all of these risks as quite small. With the system depending on a server, there will be a potential risk of the server being down, and as a consequence, the system could be unavailable. There will have to be a lot of users on the system, for the servers to go down, and we consider that highly unlikely and therefore a small risk. When the users log in, they will have to use some information, but we will not require a lot of sensitive information.

There will always be a risk of a security breach, but it will be a small risk because it would not be likely to occur, and we will not hold any sensitive information which could possibly get leaked. Human errors with incorrect data processing or programming is a small risk because of the consequences being not so bad. Human errors will of course be likely, but we will be ready to solve it if it occurs.

Our music player will not hold any sensitive information, and therefore the risks are generally quite small.

**Domain model:**



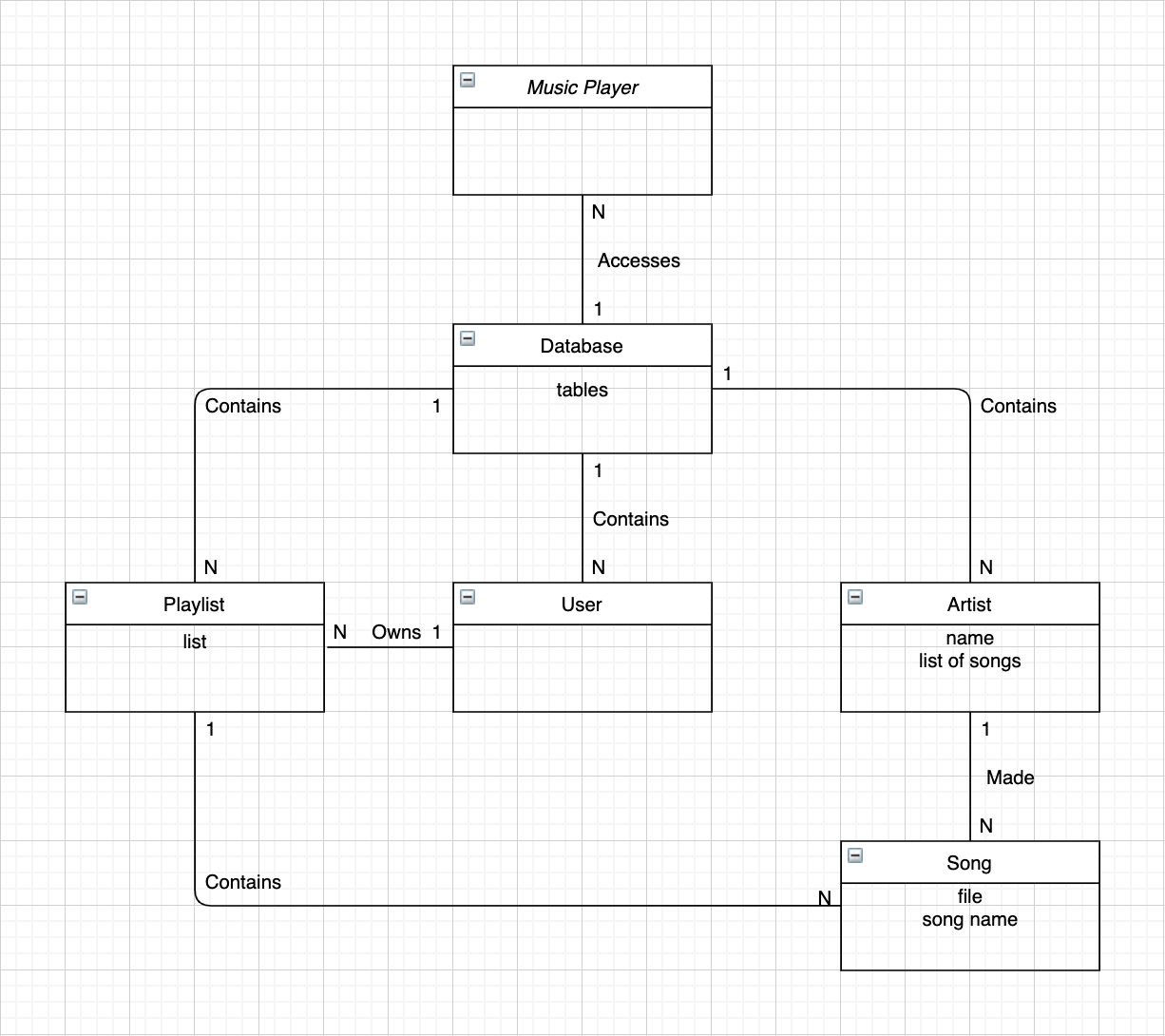
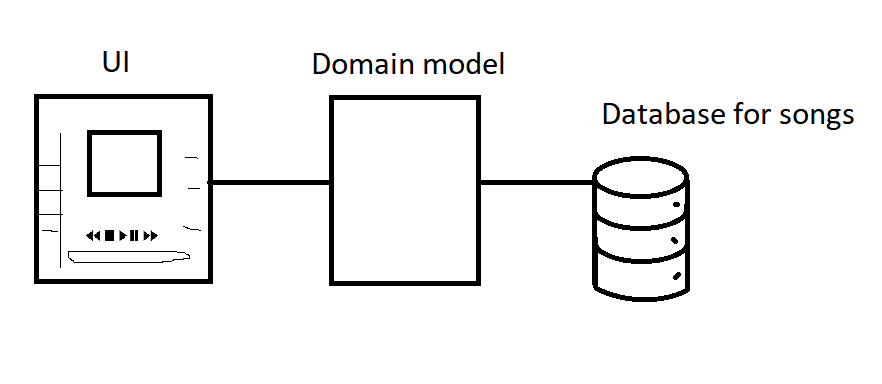
**Functional requirements**

[Music] We mainly use three requirements for our system. The user should be able to play songs, able to play and pause songs. The second requirement is where the user can create a playlist where they can later add songs to it and have for example different playlists for different genres. The last requirement is simple, to be able to search for songs.

**Non functional requirements**

[Music] The non-functional requirements for our system are simple and consistent. The interface should be easy to use and navigate through. What we mean by this is having the different functions of the system clearly labeled and visible in the system so all the workers using the system can properly use it without massive issues.   
  
Regular people, as in people that are not a part of the staff team of the organization using that specific system should not be able to connect to the system and use it, due to them not being a part of the organization.   
  
Obviously, we want the system to be as good as possible when it comes to system up-time. The goal is 24/7 uptime, but this is also a very far-fetched goal, and likely impossible due to updates to the system, shutdowns, bugs, which causes the system to be turned of or restarted.   
  
We also want the system to be built in such a way that it can easily receive updates without massive issues figuring out what is in our code, database, etc. FInding out these problems if not solved for you, can take massive amounts of resources and time, and should be avoided if possible.

**Architecture sketch**



Both the UI and the domain model goes through the database.

**Project tools**:

[Music] For our project, we will most likely use a small array of development environments. The IDEs we have so far is PyCharm, which is an IDE for python code. We have also chosen to include Atom (general text editor), and Visual Studio Code (IDE) which can write in a vast amount of languages, even though we will mostly be using python to develop.

[Music] For storing out documentation, and projects, we use a collaboration platform for sharing files and working in groups called GitHub. In GitHub we can upload files and keep them synchronized with the rest of the group.

In addition, the songs for the service will be stored in an online database or cloud service.

[Music] For communication within our project, we use two different programs for maintaining digital communication for our project when we are not working physically on campus. Two programs we use are Discord, and Facebook Messenger.