**REQUIREMENTS ENGINEERING**

**REQUIREMENTS ELICITATION:**

1. Traditional Elicitation

This involves a direct meeting with customers and stakeholders to learn their views, gather requirements and expectations from the project. The meeting can include interviews, answering surveys, hosting group discussions and major introspection sessions with the members. Predetermined questionnaires make the process more effective and results easier to study. Some of the questions in the survey could be:

1. What are the shortcomings the user faces while discovering music online?

We can first address the issues that people generally face while using music recommendation engines commonly available online. These issues are then addressed as obstacles to overcome in our software. This methodology of discussion forces people to spring up ideas by first identifying the current problems in the industry.

1. What conditions are preliminary for one to be able to access the software?

This software is purported to be built as local software that integrates the local music library with recommendations obtained from database, and fetches URLs to watch the songs. By understanding how the user wants these processes to run, and knowing the conditions of accessibility (absence of an Internet connection, less storage space, etc), the software can be necessarily modified to handle all scenarios.

1. How often should recommendations be presented?

It is imperative to know the expected frequency of recommendations for a fulfilling experience.

1. What other features would you like to be introduced in the media player?

This helps in building the software to house the most customized user experience.

1. When will the software fail?

There are several scenarios when certain features of the software can behave unexpectedly and with unwanted results, such as the Cold Start problem, poor audio quality files, incompatible file type, etc. It is also necessary to know the general interest the public will show in this software, to know if it will make a strong stand in the market. These steps are crucial in planning.

2. Observation of the existing system:

This technique is generally used when some current process is to be improved. In our project, the base aim is not something new, as music recommendation has been an increasingly popular area of research and business, ever since the web made access to music incredibly easy. Therefore, this technique would be useful.

It can also be used when the customers and/or stakeholders are not able to clearly define their requirements. They can then observe actively, or passively. Active observation is when the observer can interrupt the observant during his work and ask questions. Passive observation is when the observer only observes without interaction until the work is complete, and then asks follow-up questions. It is important in the observation phase to be free from any form of bias. Loss of bias These methods eventually help in realising the requirements of the current project.

3. Model Based Technique for Elicitation:

The validity of data collected through other means depends on the integrity and design of the structure used. These can be modelled using UML diagrams and via Use-CASE diagrams. These diagrams make the design easier to study and analyse. Also, they ease the work of implementation as there are many softwares that directly convert class diagrams into a code template. Hence, model based technique is important in building the basic structure of the software.

4. Prototype creation:

Creation of a prototype for testing and improvement is a very common process done to ensure the best quality of the product released to the general public. Prototypes created at several stages also help direct the development in certain directions and identify problems early on and avoid major software disasters.

**REQUIREMENT ANALYSIS:**

It encompasses those tasks that go into determining the needs or conditions to meet for a new or altered project, taking account of the possible conflicting requirements of the various stakeholders, analyzing and managing software or system requirements.

1. What and how:
   1. Business Requirements: This has three main key points under it –
      1. Business Process Automation(BPA): It is a strategy employed to reduce costs by automating some processes. Example would be using special third-party software for directly converting class diagrams to code template automatically.
      2. Business Process Implementation(BPI): It’s the stage wherein actual implementation of the Business Process occurs. Usage rights are obtained for each song along with other legal work, and building of the actual platform begins.
      3. Business Process Re-Engineering(BPR): It focuses on analysis and design of workflows within the organization and aims to help them rethink on how to do their work to dramatically improve customer service and reduce costs.
   2. System integration requirement: This is part of the initial process where we make use of data to divide the system into various subsystems or modules that can be separately structured and later integrated. In our project, the various subsystems could be the media player, library manager, recommender engine, etc.
   3. Architecture and Design requirement: This concerns itself with designing the entire layout of the system. It specifies the looks as well as the user experience with as much detail as possible. In our project, this would involve providing the complete blueprint for the GUI, as well as details of the various customizability features to improve UX.
   4. Technology requirement: These are the requirements of the developers of the software. It involves both software and hardware requirements as well as skills such as proficiency in certain programming languages which can make implementation an easier process.
2. Quality attributes of requirements

Quality attributes include, but are not limited to, atomicity, consistency, unambiguously, testable, scalability, reliability and usability. They are realised non-functional requirements used to evaluate the system.

Requirements analysis would also include the following stages:

1. **Problem Analysis**: Problem is analysed and feasible solution is sought.

2. **Root-cause Analysis**: The root causes of probable problems or faults are searched.

3. **Duration Analysis**: Duration of all tasks, of user and developer, is determined.

4. **Cost analysis**: Minimal cost is sought without disturbing any other requirement.

5. **Benchmarking analysis**: Comparison with other softwares is done to seek improvement.

6. **Outcome analysis**: The outcome should be of high quality and accuracy.

7. **Technical analysis**: Technical requirements are dealt with here.

8. **Activity Elimination**: Redundancies are sought and eliminated.

**SPECIFICATION:**

A software requirements specification is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do on the basis of the analysis done in the previous step.

Our proposed music recommendation software has a lot of requirements. After analysis a list of essential features is made for maximum user benefit. A few of them are:

* Music player allows for fast forward, skip, rewind, pause and play
* Feature to like/favourite certain tracks
* Library must auto update itself regularly to be able to show new downloaded content on the local machine
* It should be simple to use, with the user spending as little time as possible in learning how to use it.

The steps to be followed for software requirements specification are implemented using something called formal methods. These are described below:

Formal methods are techniques and tools based on mathematics and formal logic for the specification, development and verification of software and hardware systems. They can assume various forms and levels of rigour. Based on the rigour with which specification is done, formal methods can be classified into 3 levels:

**Level 0**: The specification is described in a free-form natural language. Due to this, there may be ambiguity and lack of organization that may lead to incompleteness and misunderstandings. This method is very cost-effective.

**Level 1**: In this level, a standardized syntax is adopted and basic consistency and completeness checks are made. But despite all this, imprecise semantics may cause other sources of error to still be present. This may be most appropriate in high-integrity systems involving safety or security.

**Level 2:** To prevent the inconsistencies that may be present even after adopting level 1, in this level a precise, perhaps mathematical form is employed. Though this method seems ideal, it is difficult to implement in practice due to high cost and the training required for implementation.

Formal methods can be applied at various points through the development process.

**1. Specification**

Formal methods may be used to give a description of the system to be developed, at whatever levels of detail desired. This formal description can be used to guide further development activities. Additionally, it can be used to verify that the requirements for the system being developed have been completely and accurately specified and also to make sure that no undocumented or unexpected assumptions are made.

**2. Development**

Once a formal specification has been produced, the specification may be used as a guide while the concrete system is developed during the design process (i.e., realized typically in software, but also potentially in hardware).

Specification is the final work product produced by the requirements engineer. It serves as the foundation for subsequent software engineering activities. It describes the function and performance of a computer-based system and the constraints that will govern its development.

**VERIFICATION:**

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfils its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle.

Verification and validation are not the same thing, although they are often confused to be. Verification is the static process of verifying documents, design, code and program. Validation on the other hand is the dynamic mechanism of testing the actual product.

Requirements verification is the process of ensuring that the system requirements are complete, correct, consistent, and clear. The primary requirements validation mechanism is the formal technical review. The review team that validates requirements includes software engineers, users and other stake-holders who examine the specification looking for errors in content or interpretation, areas where classification maybe required, missing information, inconsistencies, conflicting requirements, or unrealistic requirements. Verification ensures the requirements are complete in its specification, leaving no room for subjective judgement to be employed.

The music discovery platform will need to be verified for the following requirements –

1. The software should recommend songs to the user’s liking, or in more quantifiable terms, the recommend songs’ average rating as decided by the user must be greater than 5 on a scale of 10.
2. It needs to handle poor quality audio files well, along with tackling the problem of Cold Start, i.e., when there is insufficient initial data for the machine to learn from.
3. The design should be simplistic and appealing
4. The documentation and help guide should be extensive, while maintaining the software as easy to learn