## **Step 2 : Requirements Breakdown and Prioritization**

**1) Break Down Requirements**

**Functional Requirements:**

The following essential features will be present in the Virtual Art Gallery application:

● By entering their username, email address, and password, users ought to be able to register.

● With their login credentials, users must be able to safely log in and out.

● By entering information like the title, artist, year, style, and description, users should be able to add new artworks to the system.

● All artworks ought to be accessible to users, with the option to search or filter them according to particular standards.

● Artworks added by users can be updated or removed.

● It should be possible for each user to make their own gallery and give it a title or name.

● It must be possible for users to add and remove artwork from their own galleries.

● A SQL-based database should be used to store and retrieve all data, including users, artwork, and galleries.

● Custom error messages and exception handling, like alerting the user when an invalid operation takes place, should be part of the system.

● Users will be able to navigate features using keyboard input as the entire application operates as a menu-driven console interface.

**Non-Functional Requirements:**

The application must meet a number of non-functional requirements in addition to functionality:

● To guarantee modularity and reusability, the code should adhere to object-oriented principles and arrange logic into classes and methods.

● To enable modifications without modifying the code, database configuration settings ought to be kept in a distinct.ini file.

● The program should be made to function locally and offline without requiring the internet.

● To guarantee a seamless user experience and avoid crashes, the application must have appropriate exception handling throughout.

● Python's unittest module should be used when writing unit tests to verify significant codebase components.

● For ease of use and platform independence, the application will be implemented as a text-based command-line interface.

## **2) Prioritize Requirements**

### **High Priority:**

These features must be finished first because they are essential to the system's operation

● Login and user registration.

● Adding and looking at artworks.

● Storing and retrieving all data in a database.

● Implementing the basic framework for menu-driven user interaction.

● Utilizing appropriate exception handling and object-oriented design.

● Ensuring that the Python application runs locally.

### **Medium Priority:**

### **These features are important but can be implemented after the core functionalities:**

● Updating and deleting artworks.

● Creating personal galleries and managing their contents.

● Reading database settings from a configuration file.

● Writing unit tests for essential components.

### **Low Priority:**

These features enhance the user experience but are not essential for the first working version:

● Custom exception classes with detailed error descriptions.

● Advanced CLI navigation or visual enhancements (e.g., text formatting).

● Additional validation logic or future-ready enhancements like export options.