**Software Requirements Specification**

For

Online Entertainment   
Management Application

Version 1.1

**Prepared by:**

**Abhishek Divekar**

**Reg No. 131080051**

**And**

**Sahaj Gandhi**

**Reg No. 131080048**

VJTI, Mumbai.

DETAILED PROBLEM STATEMENT FOR ONLINE ENTERTAINMENT MANAGEMENT APPLICATION:

In recent years, there has been a sharp rise in the amount of media we consume from online sources. Several billion-dollar companies including YouTube, Netflix, Pandora, Spotify, iTunes and Amazon, offer licensed content which is available for free or a small fee. It is getting increasingly difficult for the average user to keep track of the numerous new Movies, TV shows, Music, Books and News that are being released, as well as the ones we have already viewed or own. This application offers the user a single point of reference for all their media (movies, TV, music, books and news), with the option to explore other media that they may enjoy.

Each user of the application must sign up via the website, and login in order to access the data. All transactions are assumed to happen via the online website.

A core feature of this application is a very large database of movie, TV, music, books and news article listings. The user marks a subset of movies, TV shows, Music and News, as his or her own ‘Collection’. This collection is maintained online so that the user may access it at their convenience via a browser.

The application automatically downloads updates of the listings from a variety of sources (due to restrictions in storage space and licenses, the actual media itself may not be downloaded, only the listings). This ensures that the user is upto date with the latest movies, music, TV shows, books and news.

An important feature of the application is the “Discover” option, which allows the user to browse more media which may be according to his or her liking. For example, if the user likes Hindi Classical music, he or she may discover new artists or songs by browsing under the “Hindi Classical” genre, under Music, under Discovery. You can search in the Discover option.

While browsing, the user may decide to add a song or movie to their Collection if they enjoy it. They might add it as a “Wish-listed” item if they wish to listen to it in the future, or as an “Owned” item if they already own the piece. Thus, the user’s collection grows. Later, Wish-listed items may be converted to Owned items (with the assumption that the user has purchased them elsewhere). Entries from the collection may also be deleted by the user. Users may give ratings to entries which have been marked ‘Owned’.

The user’s profile displays basic statistics that may be important to other users, such as the number of movies watched, favorite genre, most highly rated movies, most poorly rated movies, etc. These statistics are updated from the user’s personal media collection.

Lastly, the application features a calendar. There is a global events list, which containsupdates of all the new events in the near future. e.g. movies or TV show episodes that are being released this week, upcoming concerts, breaking news, etc. Each user has their own ‘personal calendar’, on which he/she bookmarks the various events from the global events list. The calendar feature helps the user stay organized, and ensures that the user does not double-book events in the same time slot.

**Table of Contents**:

[1. Introduction 5](#_Toc447142968)

[1.1 Purpose 5](#_Toc447142969)

[1.2 Document Conventions 5](#_Toc447142970)

[1.3 Intended Audience and Reading Suggestions 6](#_Toc447142971)

[1.4 Project Scope 8](#_Toc447142972)

[1.5 References 8](#_Toc447142973)

[2. Overall Description 9](#_Toc447142974)

[2.1 Product Perspective 9](#_Toc447142975)

[2.2 Product Functions 10](#_Toc447142976)

[2.3 User Classes and Characteristics 11](#_Toc447142977)

[2.4 Operating Environment 11](#_Toc447142978)

[2.4.1 Software Requirements:- 11](#_Toc447142979)

[2.4.2 Hardware Requirements:- 13](#_Toc447142980)

[2.5 Design and Implementation Constraints: 13](#_Toc447142981)

[2.6 User Documentation 15](#_Toc447142982)

[2.7 Assumptions and Dependencies: 15](#_Toc447142983)

[3. System Features 17](#_Toc447142984)

[4. External interface requirement 18](#_Toc447142985)

[4.1 User Interfaces 18](#_Toc447142986)

[4.2 Hardware interfaces 19](#_Toc447142987)

[4.3 Software interfaces 19](#_Toc447142988)

[4.4 Communication interfaces 19](#_Toc447142989)

[5. Nonfunctional Requirements 20](#_Toc447142990)

[5.1 Performance Requirements: 20](#_Toc447142991)

[5.2 Safety Requirements: 20](#_Toc447142992)

[5.3 Security Requirements: 21](#_Toc447142993)

[5.4 Software Quality Attributes: 22](#_Toc447142994)

[6. Use-case Diagram and specification: 24](#_Toc447142995)

[6.1 Use Case Diagram: 24](#_Toc447142996)

[6.2 Use-case specifications: 25](#_Toc447142997)

[6.2.1 Register: 25](#_Toc447142998)

[6.2.2 Login: 26](#_Toc447142999)

[6.2.3 Discover: 26](#_Toc447143000)

[6.2.4 View Collection: 27](#_Toc447143001)

[6.2.5 Add to Collection: 28](#_Toc447143002)

[6.2.6 Modify Collection: 29](#_Toc447143003)

[6.2.7 Delete from Collection: 31](#_Toc447143004)

[6.2.8 View Calendar and Events List: 31](#_Toc447143005)

[6.2.9 Get Event Updates: 32](#_Toc447143006)

[6.2.10 Clear Event Updates: 33](#_Toc447143007)

[6.2.11 Add Event to Calendar: 33](#_Toc447143008)

[6.2.12 Delete from Calendar: 34](#_Toc447143009)

[6.2.13 View Profile: 35](#_Toc447143010)

[6.2.14 Update Database: 35](#_Toc447143011)

[Appendix A: 37](#_Toc447143012)

[Glossary 37](#_Toc447143013)

# 1. Introduction

## 1.1 Purpose

The system aims to be the central hub for all the user’s entertainment needs. It allow users to organize their media, get live updates on new music, movies, TV shows, books, and news bookmark events which they may be interested in attending, and build a wish-list of future purchases.

Concretely, the main objectives of the system are as follows:

1. Registration & login to get access to the large database of media such as movies, music, TV shows, books and news articles.
2. Allow user to build a personal collection of such media, which they may segregate into “Wish-listed” for items they may wish to own in the future, or “Owned”, for items they already own.
3. Access their own collection of such media via an online portal.
4. Allow the user to find more media that may be to their liking via the “Discover” option.
5. List the user’s tastes and preferences on their online profile.
6. Allow users to view profiles of other users, to get an idea of what other media they may enjoy, based on other people with similar tastes.
7. Provide live daily updates of all real-life events which the user may be interested in; these include concerts, TV and movie releases, book releases, and breaking news.
8. Allow the user to bookmark the events mentioned in the updates section, thus adding them to their own personal calendar.

## 1.2 Document Conventions

To develop the document the font type is Times New Roman and the fonts are sized such that the hierarchy of the topics is easily understood and followed as in the case any conventional documentation practice as follows:

IEEE 830 Standard Software Requirement Specification Format has been used for this document.

Standard conventions specification (IEEE 380):

* Heading:  
  Font Style: Times New Roman  
  Size: 16
* Subheading:  
  Font Style: Times New Roman  
  Size: 14
* Content:  
  Font Style: Times New Roman  
  Size: 12

Other conventions used in this document:

* Line spacing of content is Multiple at 1.25.
* All single words in double quotes are assumed to be features or properties with the same name, which are visible to the application user, and which may find usage inside the application, e.g. “Wish-list”, “Owner”, “Discover” etc.
* All multilevel lists have at least one line of space between list items.
* The start of paragraphs which are about a certain topic may have the name of the topic in bold (e.g. see section 1.3).

## 1.3 Intended Audience and Reading Suggestions

This document is intended for readers that include:

* **Developers:**Developers will find this document a full-featured and complete listing of the various functionality that the user shall see and the intended flow to be followed while using the application; the flow is best understood by following the use-case specifications.   
  The document also contains an ER diagram of the database which is meant to aid development work.   
  The application makes use of an external library JSoup to collect data from the internet. Developers assigned to this task should be sure to visit the JSoup documentation (at http://jsoup.org/apidocs/).
* **Project managers:**This document is a wholesome list of requirements which are expected to be fulfilled by the stipulated deadline. A suggested reading which would spawn an intuitive project plan would be a run-through of the core features, listed in the “Purpose”. Special attention should be paid to the extraction code in particular, as it requires a large upfront bandwidth cost and then smaller, less frequent bandwidth costs for updates.   
  Since this project uses external libraries, please consult their documentation and terms of usage.
* **Users Documentationwriters:**Section that are crucial for User Documentation writers are the hardware requirements (2.4.2) and software requirements (2.4.1). Though most user machines should be able to handle this application, outdated machines may find it a bit heavy.  
  This document should otherwise provide a brief view of the various actions that the application is capable of; however, as the application is GUI-based, visiting the website and implementing features is probably the best path to understanding the functionality. The use-cases may provide some information, but this document is largely for developers and project managers. Please only consult it to ensure that no features are missing from the User Documentation.
* **Marketing Personnel:**The relevant parts of this document are the Purpose (1.1), Product Scope (1.4), Product Functions (2.2), User Classes and Characteristics (2.3), and Operating Environment (2.4). A good understanding of the flow can be gleaned from the user manual.
* **Users:**Quite simply, this document is not intended for user consumption; it is filled with technical jargon (UML, ER diagrams etc.) and descriptions of system components which are hidden from the user. A much better approach for user wishing to learn the system would be the User Documentation, or actually trying the system in the Beta release.

We will now describe what features are in the scope of the software and what are not in the scope of the software to be developed.

## 1.4 Project Scope

In Scope:

* Login to the system to access your very own calendar which maps your selected movies, TV show timings and music launch dates.
* Anytime access to your very-own personalized planner.
* Access all the latest music and movie ratings and launch dates and timings
* Discover feature allows users to access music, TV shows, news and movies in accordance to those previously liked and rated well off.
* Daily updates provided wherein latest news is flashed to the users. These updates may even give details about the stars of the user’s favorite movie stars or tv actors or even their favorite musicians.

Benefits Objectives & Goals:

1. Easy insertion, updating and retrieval of data using GUI are achieved through this system.

2. Input validation can be performed.

3. Access to system is provided with password & registration no.

The system is also minimizes the work load by allowing traffic inspector or concerned authority to send a request to the main server and receiving information within a span of minutes.

## 1.5 References

* Standard IEEE 830 Recommended practice for Software Requirement Specification is referred.

# 2. Overall Description

## 2.1 Product Perspective

The online entertainment manager is a means to organize all your entertainment media and discover new media. It is meant to be an integrated system that replaces the dozens of accounts that one accumulates by using services like Amazon Bookstore and iTunes, which fulfill a similar, but lesser, purpose. While this application does not re-make the wheel, it takes existing software and repurposes it into a smoother, more intuitive product.

The product is not self-contained; due to its current position as a start-up, the application draws data and event updates from several accessory sites to store in its database. The user also requires an internet connection to use it; access is provided via a web browser (supported browsers are Google Chrome, Mozilla Firefox, Internet Explorer and Microsoft Edge).

As it stands (as of 29th January 2016), there are no DRM requirements which must be fulfilled; the product is not intended to be a payment gateway for the media, simply a tool for discovery, organization and updates. This area holds scope for expansion for future versions of the application.

The application tries to be all-expansive, and contains listings for thousands of movies, books, TV shows, songs and news articles in its listing (along with other users’ ratings and reviews). The search feature on the “Discover” page allows ease of access to the listings.

Existing system Limitations:

* Most users maintain their data across several accounts, which adds the overhead of switching between accounts and dealing with different interfaces.
* There are few applications which provide listings along with the social aspect of viewing other people’s listings.
* Discovery of new media is not as convenient; one must look at several sources to try and find media that suits their tastes. The centralized database simplifies this process to a great extent.
* Other applications similar in nature are occasionally locked to a particular device or set of devices (most prominent among the offenders are the Apple iTunes store and the Google Play store). The Entertainment Manager faces none of these issues and is reachable from any device on any platform, including both mobile and desktop, provided that the user has a compatible web browser (which constitutes a large majority of the users in this Internet-age).
* There are occasionally DRM licensing issues that are faced by similar products which disallow content to become available to users due to the restrictions placed by the country in which the DRM license was issued (prime examples of this are Netflix, which was unavailable in India until lately, and Spotify and Pandora, which are still only available in the United States due to DRM licensing).   
  As this product does not conduct online monetary transactions and is only related to providing listings which can be used for users to collect and explore new movies/music/TV shows/books/news, it faces no issues with licensing, and is always able to provide the listings to the users. [Side note: this clause may be subject to change in the future if the software requirements were to change to reflect a purchasing aspect].
* As of now, there are no applications which provide the users with the feature of live updates to new real-life events such as movie releases, breaking news, TV airings, books releases and concert updates.   
  As a result of being the only application which offers this feature, the online Entertainment Manager also has the honor of being the only application which offers the “Calendar” feature, which allows users to bookmark events to their personal calendar.

## 2.2 Product Functions

Main features available in this project

* Liking a particular TV show, movie, music or news, thus adding it into your ‘Owned’ library
* Wish-listing a particular TV show, movie, music or news
* Mapping of the details of the favorited movies, TV shows and music launches
* Maintaining database of every user containing all his/her owned items
* Maintaining database of every user containing all his/her wish-listed items
* Rating of movies, TV shows and music

## 2.3 User Classes and Characteristics

**User:**

This is a single end-user system and the application interacts with the user and users do not interact with each other.

**Admin:**

The admin has options to update the various database tables from a screen with options on it. The back-end code uses web-scraping from various websites to get the data and inserts it into the database.

## 2.4 Operating Environment

Since this is an online application that is hosted via a website, the user (who is on the client-side) interacts with the application via a web browser, sending and receiving webpages using the HTTP ‘GET’ and ‘POST’ methods. The operating system underlying the browser on the client-side is immaterial.

On the server-side, the backend code is run using one of the existing frameworks which allows the server to deliver webpages containing HTML, CSS and JavaScript to the client browser. For testing purposes prior to deployment, a simulated server is used, working on the assumption that a real-time server will work in a similar fashion.

### 2.4.1 Software Requirements:-

1. Client-side requirements:
   1. Operating system (any of the following):  
      The only requirement of the OS is that it has browser support.  
      Supported OSes:
      1. Windows (XP upwards)
      2. Mac OSX
      3. Any of the \*-nix varieties (including Ubuntu and its variants, FreeBSD, etc) are supported.
      4. Android mobile operating system
      5. iOS mobile operating system
   2. Browser:  
      The only requirements of the browser is that it is able to send HTTP GET and POST messages to a remote server, and t is able to correctly interpret code written in HTML 4 upwards, CSS 2 upwards and JavaScript.  
      Supported browsers:
      1. Google Chrome / Chromium
      2. Mozilla Firefox
      3. Safari (on Mac OSX or on iOS)
      4. Android browser
      5. Opera
      6. Internet Explorer 8 upwards
      7. Microsoft Edge
2. Server-side requirements:
   1. Any OS of the following:
      1. Windows (XP upwards)
      2. Mac OSX
      3. Any of the \*-nix varieties (including Ubuntu and its variants, FreeBSD, etc) are supported.
   2. GitHub Windows client and GitSmart Git client for Ubuntu: collaborative development and version control of source code (necessary as the project has multiple developers working in parallel on the same codebase).
   3. Java version 7 upwards for executing source code.
   4. Oracle database connector module (connects to an SQL database).
   5. JSoup external library: for HTML parsing in Java.
   6. Bootstrap external library: for CSS handling and rescaling for mobile devices on client-side.
   7. Any of the following browsers, for testing of code:
      1. Google Chrome / Chromium
      2. Mozilla Firefox
      3. Safari (on Mac OSX or on iOS)
      4. Android browser
      5. Opera
      6. Internet Explorer 8 upwards
      7. Microsoft Edge
   8. Apache Tomcat 7 upwards (for testing of service handling on local computers).
   9. Any JSP-support application (for actual deployment).

### 2.4.2 Hardware Requirements:-

1. Client-side:
   1. Processor: Intel Core i3/i5/i7 or AMD, with clock frequency of 1.66GHz or higher.
   2. 1 GB RAM or higher (more is better).
   3. 200 MB of HDD or higher.
   4. Standard monitor, keyboard and mouse.
2. Server-side:
   1. Processor: Intel Core i7 with clock frequency of 2.66GHz or higher.
   2. 32 GB RAM or higher (more is better).
   3. 200 GB of HDD or higher.
3. Database server:
   1. Processor: Intel Core i7 with clock frequency of 2.66GHz or higher.
   2. 16 GB RAM or higher (more is better).
   3. 2 TB of HDD or higher.

## 2.5 Design and Implementation Constraints:

* User passwords must be stored in a secure way using an appropriate hashing algorithm; actual passwords should not be stored in the database, only the hashes should be stored.
* Users should not be allowed to log in on the same account multiple times (this is usually indicative of account tampering).
* User ratings and reviews should be accessible to all other users; however, user “Wish-list”s and “Owner” collections should not. This is in tune with the user only revealing his own thoughts about specific books/movies/etc, but not his or her entire collection.
* Only the system administrator can receive reports on the user login sessions.
* The database should use SQLite standards in the development stages for high portability; later, if necessary, it may be shifted to a more robust MySQL or Oracle database for deployment.
* Since the application makes use of the external JSoup module, it must adhere to the standard and guidelines provided by the makers of that module. The JSoup documentation can be found here: <http://jsoup.org/apidocs/>
* Since the application makes use of the external Bootstrap module, it must adhere to the standard and guidelines provided by the makers of that module. The Bootstrap documentation can be found here: <http://getbootstrap.com/getting-started/>
* It should be noted that the hardware requirements require that not more than 200 GB of disk space is used; this may cause a problem as the database is large in size and contains several pictures. The database developer should ensure that redundancy is reduced, especially when dealing with images. Perhaps a compression algorithm can be applied to the images before storing them in the database.
* Each page should be fetched from the server and loaded in less than 20 seconds, complete with any associated images, on a 512 kbps internet connection.
* While in its current iteration the software does not include the facility of buying movies/music/TV shows/books or subscribing to newspapers, there should be the option to expand into this sector in the future. This may not necessarily include the facility of actually viewing such media in the application. This is very appealing to the consumer, as it means they only give their credit information to one application, rather than several, as is the current scenario.

## 2.6 User Documentation

The proposed system is in development stage and requires a complete model for necessary documentation. However once the Beta prototype is ready, any user can access the application.

The software is to be well equipped with user manuals to be read before using the software and signing up to use the application. To efficiently use the software, or user can find help online in the form of tutorials in the home page.

## 2.7 Assumptions and Dependencies:

Assumption are the requirements which we assume before development product. In this section we assume that we are going to state assumptions other than those of the basic need for the project and the project’s feasibility.

Dependencies means the product development is depend on our assumption which is important to project. So assumption and dependencies are important for development of that project.

The project developers make the following assumptions:

* The user must maintain a connection to the internet while loading new pages generated by the application.
* Two entities are connected every communication: one on the user side and the other on the system side. This connection need only be constant while loading new pages (i.e. delivering new content); several other features such as sorting and searching on pages, etc. may be implemented using JavaScript code running in the user’s browser.
* The application gathers data based on a web-scraping model. One fundamental assumption is that it is legal to do this, as the data is in the public domain. As of now, none of the data sources prohibit the scraping of data or updates, and it is expected to remain that way for the foreseeable future.
* Aside from the legality of building the database and receiving updates, it is assumed that the content is available whenever the server may request it (as the sources are mostly large websites run by multi-billion dollar corporations, this is a fairly safe assumption).
* The providers of the two external libraries (Bootstrap and JSoup) will continue to allow their product (or at lease existing versions of their product) to be used in an open-source manner, without any royalty to be paid for their usage.
* Each account must have a registered user email address; the confirmation message for the Entertainment Manager account is sent to that email address.
* While logging in, the user may use either his email address or login username in the “Username” field.
* There is only one administrator who has access to all accounts and may view details and suspend accounts in case of misuse.
* The system is prone to downtime if there are excess users, in case of a security breach or if the assumptions change.
* The user agrees to abide by the system’s terms of service and will not intentionally cause harm to the system or other users.

The system has the following dependencies:

* JSoup: an external HTML parsing module. The system maintains a dependency on any working version with the same API as the latest one as of 29th January, 2016.
* BootStrap: an external CSS manipulating library. The system maintains a dependency on any working version with the same API as the latest one as of 29th January, 2016.

# 3. System Features

Applicant can apply for driving license via Agent or personally. He/ she will fill the form and submit require documents like Id proof example PAN card, Voter ID card and address proof light bill etc.

        Registrar will check all required document before register the license. Some of the applicants are trying to issue new license after suspending previous license from another station. In order to avoid this registrar will enter all details and verified with existing database. If it is not match with any document then the registrar will first learning license which is valid for six months. After the one month applicant is eligible to give the test drive. One senior inspector will take this test. The applicant has to pass this test then the registrar will issue the proper license in the name of applicant.

        In similar way registrar will register the vehicle also. If there some changes like change in the name, address, change vehicle owner, no objection certificate (N.O.C.) registrar will update that information in the database as per the applicants request. License has its expiry date. After the expiry date license has to renew the license. Registrar will renew the license after receiving the application from license holder.

The functional requirements specification of system are described as follows:

|  |  |  |
| --- | --- | --- |
| User Class | Use Case | Description |
| User | |  | | --- | | Register | | Login | | Discover | | View Collection | | Add To Collection | | Modify Collection | | Delete From Collection | | Get Event updates | | Clear Event updates | | View My Calendar | | Add Event to Calendar | | Delete Event from Calendar | | |  | | --- | | Sign up with a valid email. | | Login into system | | Browse the database of media listings | | Displays the subset of the database the user has tagged | | Add selected listing to user’s collection. | | Change listing properties or ratings. | | Remove listing from collection | | Push unseen updates to user’s account | | Clear updates from user’s cache. | | Shows the user’s bookmarked events | | Bookmark and event | | Remove a bookmarked event from the user’s calendar | |

# 4. External interface requirement

## 4.1 User Interfaces

These interfaces to be implemented are designed to be user friendly and up to the standard so that it allows the actors to achieve their goals. These includes self-explanatory tools/icons.

This specification describes the user interface look operation and feel of

**Registration GUI**

Registration for user includes the filling of details of the user like name, email id, gender and date of birth along with details like username and password.

In case the name, email id, username or password are not in accordance the given rules, the user will be asked to fill in those details again, following those rules.

**Login GUI**

Login for user includes submit and cancel button the user need to type valid username and password.

In case user sets invalid as Username and Password then the communication is incomplete by sending error message as invalid under or password.

**The Discover Feature**

The Discover feature allows users to search and stream Movies, TV shows, Music and News. This feature allows them to access these shows, movies and music of their previously liked artists as well. The news feature brings about the latest details and news of their favorite actors, musicians and even comedians.

**Wish-list**

This feature allows the users to add their entertainment content to wish-list, as the name suggests. This allows the user to later access those shows or movies anytime later and allows them to even add these to their own database by owning it.

**Adding to “Owned”**

This feature allows the users to like and rate the various albums and thus map it to their very own personalizedcalendar. This calendar then offers the user an overview of all the tv shows, movies and music favorites by them and displays various other details of the same.

**Calendar**

This calendar is indeed a personalized one. It displays the basic details of the user’s favorite shows, Music and movie launches and when it would aired live and other details similar to that.

**Rating of the movies, shows and music**

This feature allows the user's to rate a particular show or movie and thus allow them to add it either to their wish-list or to their favorites. These rating form the base for every users next updates as the updates will feature posts or movies or music depending on these ratings of the user.

## 4.2 Hardware interfaces

The server has direct access the authorized user on complimentary can access the database on his login credentials and thus access the available data as RDBMS.

## 4.3 Software interfaces

Proposed system is user friendly as well as powerful because of its use of RDBMS which is powerful database management tool to work upon. The system proposed,Company Management system uses HTML and CSS as the front end and SQL server as the back end. Jsoup is an external HTML parser. The proposed system does the same work as the current system but in a more efficient way.

## 4.4 Communication interfaces

The system will be required to communicate the database using standard protocol.

# 5. Nonfunctional Requirements

## 5.1 Performance Requirements:

* The hardware requirements require that not more than 200 GB of disk space is used; this may cause a problem as the database is large in size and contains several pictures. The database developer should ensure that redundancy is reduced, especially when dealing with images. Additionally, perhaps a compression algorithm can be applied to the images before storing them in the database.
* The system should always have at least 2 times the number of servers and processing capacity that is has at average load.
* Each page should be fetched from the server and loaded in less than 20 seconds, complete with any associated images, on a 512 kbps internet connection.
* Loading the user’s profile once they log in does not take more than 10 seconds on a 512kbps internet connection.

## 5.2 Safety Requirements:

Given that this application encourages sharing of ratings, reviews and media collections among users, there are not too many safety requirements when it comes to user profiles:

* If the user has forgotten their password, it is possible to mail them a link wherein they may reset it. There is no other way to reset a password, and if the user cannot access that particular email account, then the application cannot be operated by that user.
* Deleting entries from the “Collections” page cannot be undone quickly (i.e. there is no Trash feature); the user must find that entry again using the “Discover” feature and then re-add it to his collections.
* Event updates once cleared cannot be restored at all for that user.
* Reviews or ratings given by the users are at their own discretion and not any bias that the website has included. The website is also not responsible for any possible negative consequences that a user’s comments may have.

## 5.3 Security Requirements:

As with any application that is hosted on the internet, there are a few constraints which should be maintained to protect the website from unauthorized access or loss of data:

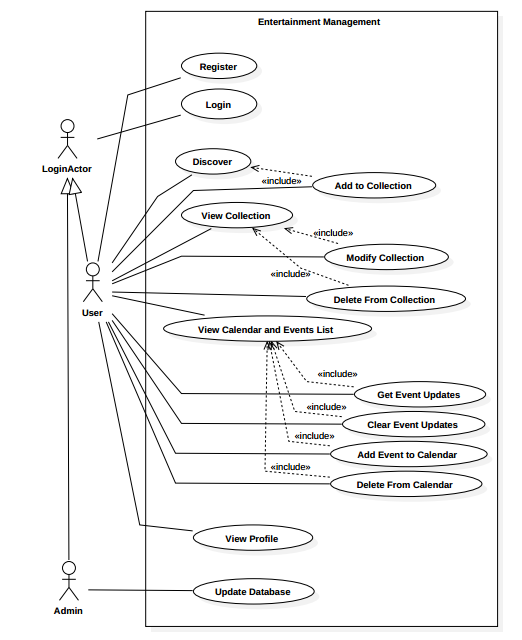
* SQL injection should be explicitly detected and action should be taken against users attempting to engage in such activity.
* All system data must be backed up every 24 hours and the backup copies stored in another server at different building or location for disaster recovery.
* Users sign up from their email account and any forgotten passwords redirects are sent to the same email account; if the user loses access to that email account, the user loses access to their Entertainment Manager account as well.
* User passwords must be stored in a secure way using an appropriate hashing algorithm; actual passwords should not be stored in the database, only the hashes should be stored.   
  The MD5 algorithm is a suitable hashing algorithm, as the hash-strings it generates are of a fixed length, and unique for the large majority of passwords.
* To ensure that passwords achieve the high entropy (and are thus difficult to crack), the user’s password should have at least one digit, one uppercase letter, one special character (special characters are: !@#$%^&\*><?| ), and should be at least 8 characters long. Spaces are not allowed.
* Users should not be allowed to log in on the same account multiple times (this is usually indicative of account tampering).
* User ratings and reviews should be accessible to all other users; however, user “Wish-list’s and “Owner” collections should not. This is in tune with the user only revealing his own thoughts about specific books/movies/etc, but not his or her entire collection.
* Only the system administrator can receive reports on the user login sessions.
* As this is a smaller system and has yet to grow, it is highly vulnerable to DDoS-type attacks. To prevent these, the system should store the user’s IP addresses and block those which are making queries at an anomalous rate which is unlike what an actual user would make.

## 5.4 Software Quality Attributes:

* **Availability of source**:   
  The source code will, for now, be made freely available at the address <https://github.com/ARDivekar/Entertainment_management>. Both the address and the source code availability may change in the future at the discretion of the management team. The database shall not be made available due to its size and the importance it has to the application.
* **Availability of system**:   
  The system is available for use 24/7, having no timing restriction for accessing the application. The users may also stay logged in as long as they want.
* **Robustness**:   
  Updates are made to the user’s collections only in discrete intervals. If the connection is broken halfway through, the changes are rolled back.
* **Consistency**:   
  The "Online Entertainment Management" system automatically updates all the transactions data performed by user to their collection, so the most recent data is fetched in response to the query.
* **Adaptability**:   
  While the ER diagram of the database (x.x1) reflects the major tables that are required to be implemented, the relationships between the tables it is not rigid, and the tables may be normalized or de-normalized for performance reasons and reduction of redundancy.   
  Furthermore, the various media tables (Movie, TV, Music, Book, NewsArticle), as well as the Events table, have attribute sets which are likely to change (expanded or reduced) based on the availability of the data from the online sources. Make the presumption that the source may change later in the future (which is highly possible given the nature of the web), and insert appropriate NULL values into the tuple values.
* **Usability:**This application is aimed at those users who consume a sizeable amount of media daily and want to remain in touch with the latest and best media. Thus, the application is aimed at users who find the existing processes or databases lacking. Such a user will most likely welcome the appearance of a large body of entries, rather than shun it.  
  At the same time, the set of people who consume lots of media and those who understand complex user interfaces in very finite. Thus, the interface should hopefully be as simple as possible, providing the user a small number of options on each screen. The user should be allowed to “Discover” by searching in the search bar, which finds the correct result or at least an approximation of it.
* **Testability:**The application should be built in modules which represent different features of the application; some which are the core and some which add accessory features.  
  New features should be tested first on dummy data and then with the core features before being integrated with the rest of the application.
* **Portability:**It is important that the source code is written in Java, as it is one of the most easily portable object-oriented languages available. Other languages such as Python may also be used provided the two can be used automatically and in conjunction. Languages such as C and C++ should be avoided as they are not as easily portable there is less module support for them, and the application does not perform time-critical computations.

# 6. Use-case Diagram and specification:

## 6.1 Use Case Diagram:



## 6.2 Use-case specifications:

### 6.2.1 Register:

* + Subsystem: User
  + Description: Allows the user to register to use the system. He/she must provide authentication and then fill in his/her relevant details.
  + Actors: User
  + Preconditions:
    - The user must not already be registered on the system.
    - The user has a valid email address.
    - The user is connected to the internet via a supported browser.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Enter full name (FName, LName) |  |  |
| 2 | Enter email ID |  |  |
| 3 | Enter Date of Birth |  |  |
| 4 | Enter gender |  |  |
| 5 | Enter username |  |  |
| 6 | Enter password |  |  |
|  |  | 7 | If password is not taken, accept. |
| 8 | Re-enter password |  |  |
|  |  | 9 | If password is same as earlier, accept. |
| 10 | Click ‘Confirm’ |  |  |
|  |  | 11 | Confirmed, storing in database |

* + Exceptions:
    - 1.1] System overload: User must wait until system load has decreased.
    - 1.2] Network error: contact network administrator
    - 10.1] Username already taken: User must contact choose another username.
    - 6.1] Password does not satisfy safety standards: User must choose another password.
    - 8.1] Re-entered password not same as original: User must re-enter that password.
    - 11.1] Details left unfilled: User must go back and fill them.
  + Post-condition:
    - User’s details saved in system.
    - User is automatically logged in.

### 6.2.2 Login:

* + Subsystem: LoginActor
  + Description: The LoginActor, when s/he wants to use the application, must log in.
  + Actors: LoginActor
  + Preconditions:
    - The LoginActor has registered.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **LoginActor** |  | **System** |
| 1 | Click on Login button |  |  |
|  |  | 2 | Load login page |
| 3 | Enter email or username |  |  |
| 4 | Enter password |  |  |
|  |  | 5 | Validate username/email and password |
|  |  | 6 | Log into system |

* + Exceptions:
    - 5.1] Invalid username or password: ask User to enter both again.
    - 1.1] Click on ‘forgot password’: resend password to email.
  + Post-condition: User is logged in and the system displays his/her profile page.

### 6.2.3 Discover:

* + Subsystem: User
  + Description: Allows the user to browse the database to find media which may be to their liking.
  + Actors: User
  + Preconditions:
    - The User has registered and logged in.
    - The User has clicked on the “Discover” button on his profile page.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on the “Discover” button on his profile page. |  |  |
|  |  | 2 | Load the Discover page to the introduction tab. |
| 3 | Click on one of the tabs in Discover (Movies, TV shows, Music, Books and News) |  |  |
|  |  | 4 | Load the first page of listings corresponding to that tab. |
| 5 | Click on Next page button |  |  |
|  |  | 6 | Go to next page |
| 7 | Click on + button on any listing |  |  |
|  |  | 8 | Go to ‘Add To Collection’ use-case |

* + Exceptions:
    - 2.1] Database is not functional: display error message to user.
    - 6.1] Database is not functional: display error message to user.
    - 6.2] Reached last page: show remaining listings.
    - 8.1] User clicks on ‘+’ button: redirect to ‘Add to my Collection’ use case.
  + Post-Condition: The user can browse the media listings.

### 6.2.4 View Collection:

* + Subsystem: User
  + Description: Allows the user to view his collection. He/she adds various modes of entertainment to his favorites and thus can view it through this function
  + Actors: User
  + Preconditions:
    - The user must be logged into the system.
    - The user must have added something into his favorites to view it.
    - The user is on their home page.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on View Collection button |  |  |
|  |  | 2 | Load user’s collection page, split into various formats – Music, TV shows, Movies and News |
| 3 | Clicks on corresponding tab |  |  |
|  |  | 4 | Shows the user's owned data by default. |
|  |  | 5 | Shows the user's wish-listed data under wish-list tab. |
| 6 | Click on Next page button |  |  |
|  |  | 7 | Go to next page |
| 8 | Click on modify button on any listing |  |  |
|  |  | 9 | Go to ‘Modify Collection’ use-case |
| 10 | Click on tab of ‘Wishlisted’/’Owned’ |  |  |
|  |  | 11 | Toggle between Wishlisted and Owned sections. |

* + Exceptions:
    - 2.1] Database is not functional: display error message to user.
    - 4.1] User has no listings in Collection: show empty screen and button for Wishlisted only.
    - 7.1] Database is not functional: display error message to user.
    - 7.2] Reached last page: show remaining listings.
    - 9.1] User clicks on ‘Modify’ button: go to ‘Modify Collection’ use-case.
  + Post-condition:
    - User views his owned and wish-listed content.

### 6.2.5 Add to Collection:

* + Subsystem: User
  + Description: Allows the user to add a listing to his collection as either an “Owned” or “Wish-listed” item; also rate and review items if they are owned.
  + Actors: User
  + Preconditions:
    - The user has registered and logged in.
    - The user is on the discover page
    - The user has clicked on a ‘+’ button on a listing on the discover page.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on + button next to listing on Discover page |  |  |
|  |  | 2 | Show dialog, asking the user: |
|  |  | 2a | If they are sure they want to add the item to their collection (“Add to your collection”) |
|  |  | 2b | A radio button which allows the user to select whether the item should be marked as “Wish-listed” or “Owned”. |
|  |  | 2c | An optional star-rating (out of 5) that the user can click to give a rating to the item [only applicable if the “Owned” radio button has been selected]. |
|  |  | 2d | An optional text box that the user can fill to give a Review to the item [only applicable if the “Owned” radio button has been selected]. |
| 3 | Fill in the required and additional fields. |  |  |
| 4 | Click on the “Save” button to add the item to the Collection. |  |  |
|  |  | 5 | Save changes made to collection in database. |

* + Exceptions:
    - 4.1] User click “Exit” button instead of “Save”: close “Add to Collection” and go back to “Discover” use-case.
    - 5.1] Database is not functional: display error message to user and rollback operation.
  + Post-Condition:
    - The system stores the details in its database.
    - The system has closed the dialog and is now on the “Discover” page.

### 6.2.6 Modify Collection:

* + Subsystem: User
  + Description: Allows the user to make changes to the items in his collection.
  + Actors: User
  + Preconditions:
    - The user has registered and is logged in.
    - The user is on their Collection page
    - The user has clicked on the ‘Modify’ button on a listing on the Collection page.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
|  |  | 1 | Show dialog, asking the user to make changes: |
|  |  | 1a | A radio button which allows the user to select whether the item should be marked as “Wish-listed” or “Owned”. |
|  |  | 1b | An optional star-rating (out of 5) that the user can click to modify their rating or give a rating to the item [only applicable if the “Owned” radio button has been selected]. |
|  |  | 1c | An optional text box that the user can fill to modify their review, or give a Review to the item [only applicable if the “Owned” radio button has been selected]. |
| 2 | Fill in the required and additional fields. |  |  |
| 3 | Click on the “Save Changes” button to save the changes to the Collection. |  |  |
|  |  | 4 | Save changes made to database |

* + Exceptions:
    - 3.1] User click “Exit” button instead of “Save Changes”: close “Modify Collection” and go back to “View Collection” use-case.
    - 4.1] Database is not functional: display error message to user and rollback operation.
  + Post-Condition:
    - The system stores the details in its database.
    - The system has closed the dialog and is now displaying the “View Collection” use-case.

### 6.2.7 Delete from Collection:

* + Subsystem: User
  + Description: Allows the user to delete something from his collection. He/she can delete any of his owned data possibly added by mistake or for other reasons.
  + Actors: User
  + Preconditions:
    - The user must be logged into the system.
    - The user must have added something into his favorites to delete it.
    - The user is connected to the internet via a supported browser.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on Delete from Collection button on any listing |  |  |
|  |  | 2 | Shows buttons and confirmation dialog. |
| 3 | Click on “Delete” button. |  |  |
|  |  | 4 | Remove data from database |

* + Exceptions:
    - 3.1] User click “Exit” button instead of “Delete”: close “Delete From Collection” and go back to “View Collection” use-case.
    - 4.1] Database is not functional: display error message to user and rollback operation.
  + Post-condition:
    - User deletes one or more from his owned content.

### 6.2.8 View Calendar and Events List:

* + - Subsystem: User
    - Description: Allows the user to view his calendar. He/she adds various modes of entertainment to his favorites and thus can view the brief details of upcoming events through this calendar.
    - Actors: User
    - Preconditions:
    - The user must be logged into the system.
    - The user must add something into his favorites to view it in the calendar.
    - The user is connected to the internet via a supported browser.
    - Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on Calendar button on profile page |  |  |
|  |  | 2 | Shows the Events list (i.e. the upcoming events like concerts etc.) and user’s Calendar, which has bookmarked upcoming events |

* + - Exceptions:
    - 2.1] Database is not functional: display error message to user.
    - Post-condition: User views his owned data details on his Calendar.

### 6.2.9 Get Event Updates:

* + Subsystem: User
  + Description: Allows the user to refresh updates on the ‘Events’ page.
  + Actors: User
  + Preconditions:
    - The user has registered and is logged in.
    - The user is on their profile page
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on “Get Event Updates” button. |  |  |
|  |  | 2 | Load list of all the new events that the user has not yet viewed. |

* + Exceptions:
    - 2.1] Database is not functional: display error message to user.
  + Post-Conditions:
    - The user is able to view the various events that may be relevant to them.

### 6.2.10 Clear Event Updates:

* + - Subsystem: User
    - Description: Allows the user to clear events from the events page after reading/browsing through them.
    - Actors: User
    - Preconditions:
    - The user must be logged into the system.
    - The user is connected to the internet via a supported browser.
    - Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on Clear Events button to clear events. |  |  |
| 2 | Click “Clear” button |  |  |
|  |  | 3 | Events are cleared from the page and user’s database. |

* + - Exceptions:
    - 2.1] User click “Cancel” button instead of “Clear”: close “Clear Event Updates” and go back to “View Calendar and Events List” use-case.
    - 3.1] Database is not functional: display error message to user and rollback operations.
    - Post-condition: User clears the events off the Events List.

### 6.2.11 Add Event to Calendar:

* + Subsystem: User
  + Description: Allows the user to bookmark an event that they are interested in attending.
  + Actors: User
  + Preconditions:
    - The user has registered and is logged in.
    - The system has loaded the “Calendar and Events List” page.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | The user has clicked on the ‘+’ button next to an event listing. |  |  |
|  |  | 2 | Load dialog, asking the user for confirmation: |
| 3 | Click ‘Confirm’ button. |  |  |
|  |  | 4 | Event is added to the Calendar and this is saved in database for future use. |

* + Exceptions:
    - 3.1] User click “Cancel” button instead of “Clear”: close “Add Event To Calendar” and go back to “View Calendar and Events List” use-case.
    - 3.2] If the event overlaps with another event already on his calendar: Offers a warning and let user continue with ‘Confirm’ or ‘Cancel’.
    - 4.1] Database is not functional: display error message to user and rollback operations.
  + Post-Conditions:
    - The system has added the event listing to the user’s calendar.

### 6.2.12 Delete from Calendar:

* + Subsystem: User
  + Description: Allows the user to delete something from his calendar. He/she can delete any of his owned data possibly added by mistake or for other reasons for the same.
  + Actors: User
  + Preconditions:
    - The user must be logged into the system.
    - The user must have added something into his owned items as adding it there would mean adding it to the calendar.
    - The user is connected to the internet via a supported browser.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Click on Calendar item to delete an event. |  |  |
|  |  | 2 | Offer confirm button |
| 3 | Clicks Confirm button |  |  |
|  |  | 4 | Event is removed from the Calendar and this change is saved in database for future use. |

* + Exceptions:
    - 3.1] User click “Cancel” button instead of “Clear”: close “Delete Event From Calendar” and go back to “View Calendar and Events List” use-case.
    - 4.1] Database is not functional: display error message to user and rollback operations.
  + Post-condition: User deletes an event from his calendar.

### 6.2.13 View Profile:

* + Subsystem: User
  + Description: Allows the user to delete something from his calendar. He/she can delete any of his owned data possibly added by mistake or for other reasons for the same.
  + Actors: User
  + Preconditions:
    - The user must be logged into the system.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Log in, or click on ‘View Profile’ button. |  |  |
|  |  | 2 | Display user’s profile information and buttons actions to him. |

* + Exceptions:
    - 2.1] Database is not functional: display error message to user and rollback operations.
  + Post-condition: User can view his or her profile data.

### 6.2.14 Update Database:

* + Subsystem: Admin
  + Description: Allows the user to delete something from his calendar. He/she can delete any of his owned data possibly added by mistake or for other reasons for the same.
  + Actors: User
  + Preconditions:
    - The admin must be logged into the system.
  + Basic Flow/Scenario:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** |  | **System** |
| 1 | Log in or click ‘Update Database’ button. |  |  |
|  |  | 2 | Display admin page with buttons to update various tables of database: Music, Movies, TVShow, Book and Events. |
| 3 | Make selection on which data to update, and how many items to fetch. |  |  |
|  |  | 4 | Extract relevant data from web and how many items were inserted into the database. |

* + Exceptions:
    - 4.1] Database is not functional: display error message to user and rollback operations.
    - 4.2] Network error: server cannot connect to other websites.
    - 4.3] All items are extracted from that website.
    - 4.4] Admin clicks “Cancel” button, terminating the execution.
  + Post-condition:
    - Data has been added to the database, visible to users in the ‘Discover’ section.
    - The admin has been informed on how much data was extracted.

# Appendix A:

## Glossary

* **User:** Any member of the Entertainment Manager System with login credentials.
* **Discover:** A function which shows a smart but random selection of movies, TV shows and music in terms of the taste of the user.
* **Owned:** A movie, TV show or song which is liked by the owner of the account.
* **Jsoup:** It is a Java library for working with real-world HTML.
* **Collection:** It is a database of the all the ‘owned’ set of items by the owner.
* **Wish-list:** Some item which the owner wishes to add to his ‘Collection’ sometime in the future.
* **Calendar:** It is a personalized calendar for every user such that each user can store their event details on it.
* **MD5:** A hashing algorithm used to store passwords.
* **Apache Tomcat:** It is an open source software implementation of Java Servlet, Java Server, Java Expression Language and Java Web Socket technologies
* **DRM:** A Digital Rights Management Schemes which are used to restrict the propriety hardware and copyrighted works.
* **Bootstrap:** A popular HTML, CSS and JavaScript framework for making a good responsive website.