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.

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 “Wishlisted” 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, Wishlisted 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. You can search for other users by name with the inbuilt search feature.

Lastly, the application features a calendar. There is a global events list, which contains updates 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.

**Software Requirements Specification**

For

Online Entertainment   
Management Application

Version 1.0

**Prepared by:**

**Abhishek Divekar**

**Reg No. 131080051**

**And**

**Sahaj Gandhi**

**Reg No. 131080048**

VJTI, Mumbai.  
29th January, 2015

Table of Contents

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 2

1.5 References 2

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Features 3

2.3 User Classes and Characteristics 3

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 5

2.6 User Documentation 5

2.7 Assumptions and Dependencies 5

3. System Features and Functional Specifications 6

4. External Interface Requirements 8

4.1 User Interfaces 8

4.2 Hardware Interfaces 11

4.3 Software Interfaces 11

4.4 Communications Interfaces 11

5. Other Nonfunctional Requirements 11

5.1 Performance Requirements 11

5.2 Safety Requirements 11

5.3 Security Requirements 12

5.4 Software Quality Attributes 12

**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 wishlist 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 “Wishlisted” 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. “Wishlist”, “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 for the internet and developers assigned to this task should be sure to visit not only the associated section (x.xx), but the JSoup documentation as well (at ).
* **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 (y.y1, y.y2, y.y3). Special attention should be paid to (z.zz) 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 (listed in (x.x1)).
* **Users Documentation writers:**Section that are crucial for User Documentation writers are the hardware requirements (k.k1) and software requirements (k.k2). 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:**

**Purpose:**

**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/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

* Registration of learning license.
* Registration of permanent license
* Registration of Vehicle.
* Maintaining records of License holder information.
* Maintaining records of vehicle registration and its owner information.
* Maintaining records of offence done by driver.

**2.3 User Classes and Characteristics**

As this is system developed as RTO registration application the users are broadly classified as following:

**Administrator:**

The administrator manage authorized user accounts (create users, modify users information and delete users), manage authentication (user login and log out).

**Registrar:**

Registrar will register the learning license, permanent license, vehicle registration, renew the license and updating the vehicle registration as per applicants application.

**Traffic Police:**

Check the details of suspected/ offender driver and the vehicle, by using license no, registration no. Enter the offence details if he/she break the rules.

**Applicant:**

An applicant can be a agent or common people. They can apply for new driving license or renew existing license or vehicle registration applicants can apply online also.

**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.

**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).

**Hardware Requirements:-**

1. Client-side:
   1. Processor with clock frequency of 1.66GHz or higher.
   2. 2 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 with clock frequency of 2.66GHz or higher.
   2. 32 GB RAM or higher (more is better).
   3. 200 GB 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 “Wishlist”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**

**3.1**

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 |
| System  Authentication | |  | | --- | | Login | | Modify Profile | | Logout | | |  | | --- | | Login into system | | Change details of profile | | Logout from system | |
| Administrator | |  | | --- | | Create Authorized  Profile | | Modify Authorized  Profile | | Delete Authorized  Profile | | |  | | --- | | Create new profile, assign default privileges | | Change details of authorized profile | | Remove the authorized profile | |
| Registrar | |  | | --- | | Register the learning  license | | Register the permanent license | | Renew the Driving license | | Register the Vehicle | | |  | | --- | | Get the data required document check,scan and store registration details to database server | | Get to he data required document,verify registration no and store to server | | Renew the license of applicants after checking all required documents | | Get the data required documents check, scan and register the applicants and store the registration details to the database server | |
| Traffic Police | |  | | --- | | Check the details of License holder | | Enter offence details | | |  | | --- | | Check the details of suspected/offender driver, by using license no | | Enter the offence details | |
| Applicant | Apply for the various registration | The agent/normal people can apply for various registration to RTO office |
|  |  |  |

**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 RTO registration system.

**Login GUI**

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

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

**Learning License Registration Form**

The details of the applicant is entered such as name, DOB, gender , address, phone number, password, duration of stay at the present address in month, city, pin code, state, nationality, e-mail etc.

**Permanent License Registration Form**

The details of the applicant is entered such that name, gender, address, type of license, PAN card number, Passport number etc.

**License Renew Form**

This form will have all the details of applicant. Register will extend the validity.

**Vehicle Registration Form**

On this form registrar have to enter all the details of vehicle as well as owner details.

1

**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. Traffic police access to the server database and filling of information about the offence.

**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 c#.net as the front end and MS-SQL server as the back end. 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. For internet communication protocols the traffic police and other users can easily download forms and apply online.

**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:

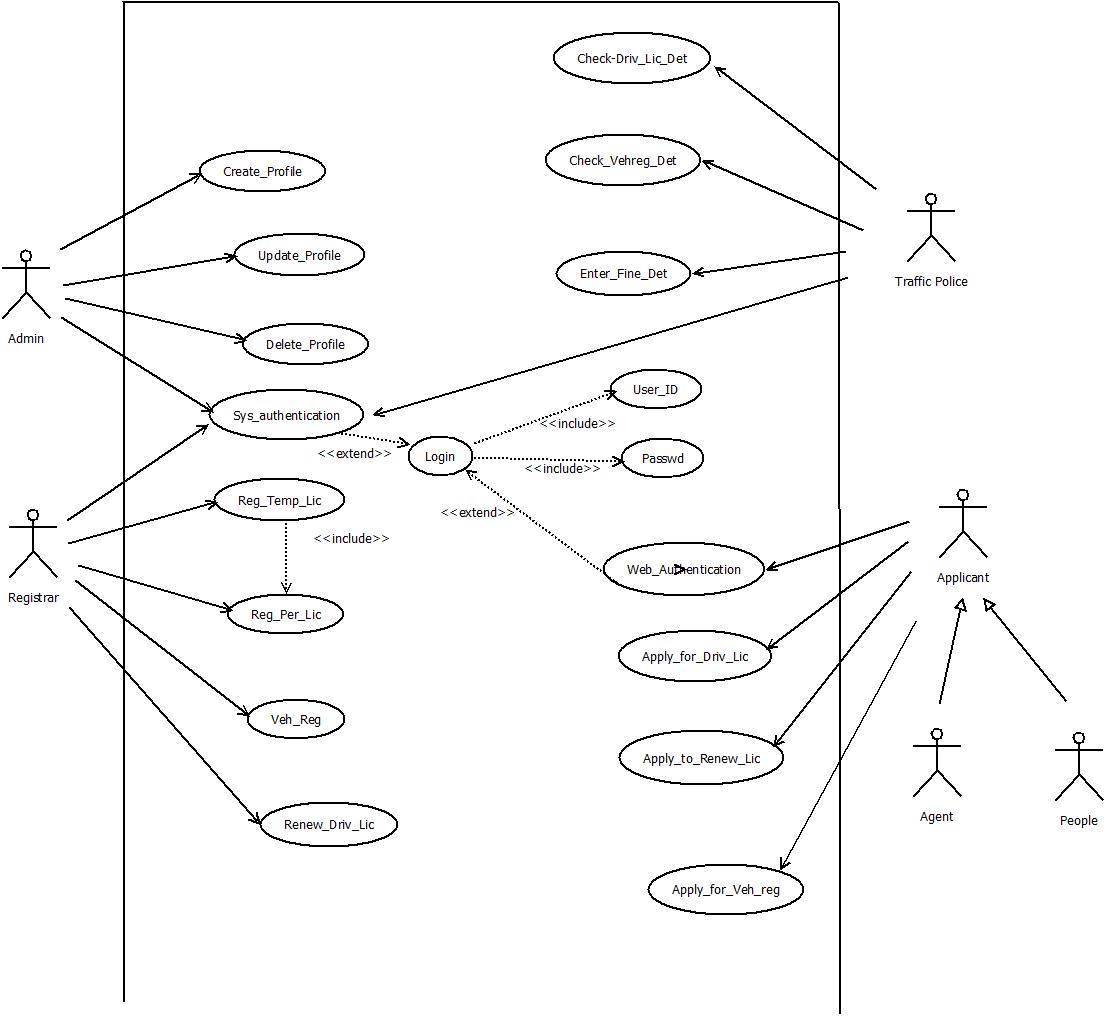
* SQL injection should be explicitly detected and action should be taken against users attempting to engage in such activity.
* All system data must be backed up every 24 hours and the backup copies stored in another server at different building or location for disaster recovery.
* Users sign up from 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 “Wishlist”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.

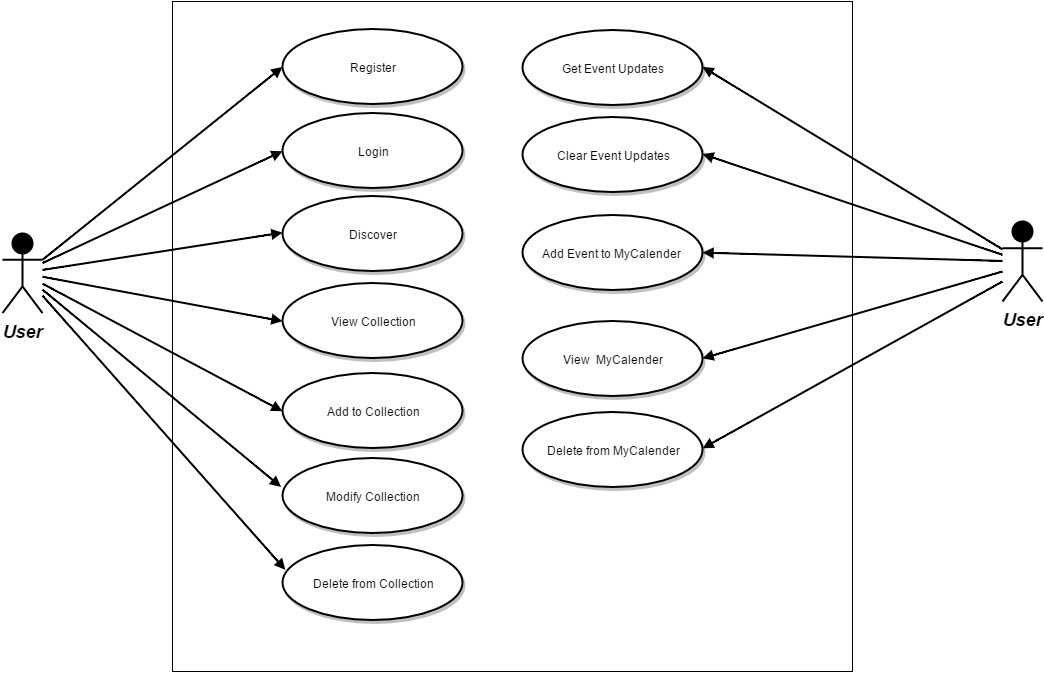
**Use Case Diagram**

Use Case Diagram for RTO Registration Application

****

**USE CASES DIAGRAM FOR ENTERTAINMENT MANAGEMENT:**

Use cases:



1. Register:
   1. Subsystem: User
   2. Description: Allows the user to register to use the system. He/she must provide authentication and then fill in his/her relevant details.
   3. Actors: User
   4. Preconditions:
      1. The user must not already be registered on the system.
      2. The user has a valid email address.
      3. The user is connected to the internet via a supported browser.
   5. Basic Flow/Scenario:
      1. User: Enter full name (FName, MName, LName)
      2. User: Enter current residential address
      3. User: Enter phone number
      4. User: Enter address
      5. User: Enter age
      6. User: Enter gender
      7. User: Enter password
      8. System: If password is not taken, accept.
      9. User: Re-enter password
      10. System: If password is same as earlier, accept.
      11. User: Click ‘Confirm’
      12. System:
          1. Confirmed, storing in database
          2. Error, could not store in database
   6. Exceptions:
      1. System overload: User must wait until system load has decreased.
      2. Username-password already taken: User must contact choose another username/password.
      3. Password does not satisfy safety standards: User must choose another password.
      4. Re-entered password not same as original: User must re-enter that password.
      5. Details box left unmarked: User must go back and mark it.
      6. System error: database not found.
      7. System error: network timeout/lapse.
   7. Post-condition:
      1. User’s details saved in system.
      2. User is automatically logged in.
2. Login:
   1. Subsystem: User
   2. Description: The User, when s/he wants to use the application, must log in.
   3. Actors: User
   4. Preconditions:
      1. The User has registered.
   5. Basic Flow/Scenario:
      1. User: Click on Login button
      2. System: load login page
      3. User: Enter email
      4. User: enter website password
      5. System: validate username and password.
      6. System: load User’s home page.
      7. User: (optional) click “forgot password” button. System: send password re-set link via email.
   6. Exceptions:
      1. Invalid username or password: ask User to enter both again.
      2. Forgot password: resend password on email.
   7. Post-condition:
      1. User is logged in and the system displays his/her profile page.

1. Discover:
   1. Subsystem: User
   2. Description: Allows the user to browse the database to find media which may be to their liking.
   3. Actors: User
   4. Preconditions:
      1. The User has registered and logged in.
   5. Basic Flow/Scenario:
      1. User: click on the “Discover” button on his profile page.
      2. System: load the Discover page to the introduction tab.
      3. User: click on one of the tabs in Discover (Movies, TV shows, Music, Books and News)
      4. System: load the first page of listings (contains 20 listings) corresponding to that tab.
      5. User: perform one of the following actions:
         1. Sort by a particular field.
         2. Search by a particular field.
         3. Load next page of listings.
         4. Click on the ‘+’ marker next to any particular listing.
         5. Switch to a different tab.
      6. System: perform the corresponding action (in case of 4, redirect to ‘Add to my Collection’ use case).
   6. Exceptions:
      1. User clicks on ‘+’ button: redirect to ‘Add to my Collection’ use case.
      2. User quits the browser tab: system exits the session and logs him  
         out.
   7. Post-Condition:
      1. The user can browse the media listings.
2. Add to Collection:
   1. Subsystem: User
   2. Description: Allows the user to add a listing to his collection as either a “Owned” or “Wishlisted” item.
   3. Actors: User
   4. Preconditions:
      1. The user has registered and logged in.
      2. The user is on the discover page
      3. The user has clicked on a ‘+’ button on a listing on the discover page.
   5. Basic Flow/Scenario:
      1. System: show dialog, asking the user:
         1. If they are sure they want to add the item to their collection (“Add to your collection”)
         2. A radio button which allows the user to select whether the item should be marked as “Wishlisted” or “Owned”.
         3. 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].
         4. 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].
      2. User: fill in the required and additional fields.
      3. User: click on the “Save” button to add the item to the Collection or “Exit” button to stop the process.
      4. System: show a suitable message informing the user of the action taken.
   6. Exceptions:
      1. User quits the browser tab: system exits the session and logs him  
         out.
      2. User clicks on the “Exit” button on the dialog: return to “discover” use case.
   7. Post-Condition:
      1. The system stores the details in its database.
      2. The system has closed the dialog and is now on the “Discover” page.

1. Modify Collection:
   1. Subsystem: User
   2. Description: Allows the user to make changes to the items in his collection.
   3. Actors: User
   4. Preconditions:
      1. The user has registered and is logged in.
      2. The user is on their Collection page
      3. The user has clicked on the ‘Modify’ button on a listing on the Collection page.
   5. Basic Flow/Scenario:
      1. System: show dialog, asking the user to make changes:
         1. A radio button which allows the user to select whether the item should be marked as “Wishlisted” or “Owned”.
         2. 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].
         3. 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. User: fill in the required and additional fields.
      3. User: click on the “Save” button to save the changes to the Collection or “Exit” button to stop the process.
      4. System: show a suitable message informing the user of the action taken.
   6. Exceptions:
      1. User quits the browser tab: system exits the session and logs him  
         out.
      2. User clicks on the “Exit” button on the dialog: return to “View my collection” use case.
   7. Post-Condition:
      1. The system stores the details in its database.
      2. The system has closed the dialog and is now displaying the “Collection” page.
2. Get Event Updates:
   1. Subsystem: User
   2. Description: Allows the user to refresh updates on the ‘Events’ page.
   3. Actors: User
   4. Preconditions:
      1. The user has registered and is logged in.
      2. The user is on their profile page
   5. Basic Flow/Scenario:
      1. User: click on “Get Event Updates” button.
      2. System: Load the “Events list” page
      3. System: load list of all the new events that the user has not yet viewed.
   6. Exceptions:
      1. User quits the browser tab: system exits the session and logs him  
         out.
   7. Post-Conditions:
      1. The user is able to view the various events that may be relevant to them.
3. Add Event to Calendar:
   1. Subsystem: User
   2. Description: Allows the user to bookmark an event that they are interested in attending.
   3. Actors: User
   4. Preconditions:
      1. The user has registered and is logged in.
      2. The user has clicked the “Get Event Updates” button on their profile page.
      3. The system has loaded the “Events List” page.
      4. The user has clicked on the ‘+’ button next to an event listing.
   5. Basic Flow/Scenario:
      1. System: load dialog, asking the user:
         1. If he wants to add the event to his or her calendar.
         2. Offers a warning if the event overlaps with another event already on his calendar.
   6. Exceptions:
      1. User quits the browser tab: system exits the session and logs him  
         out.
      2. User clicks on the ‘Cancel’ button: redirect to the “Events list” page.
   7. Post-Conditions:
      1. The system has added the event listing to the user’s calendar.

1. Use case:- Update\_Profile

Actor:- Administrator

Purpose:- It is used for updation of Authorized profile

Overview:- It allows the Administrator to modify the details of authorized profiles.

1. Use case:- Delete\_Profile

Actor:- Administrtor

Purpose:- It is used to deletion of Authorized profile

Overview:- It allows administrator to delete the authorized profiles.

1. Use case:- Check\_App\_Det

Actor:- Registrator

Purpose;- It is used to check the details of the applicant with the existing database.

Overview:- It allows the Registrar to check the details like required document of applicant.

1. Use case:- Reg\_Temp\_Lic

Actor:- Registrar

Purpose:- It is used to Temperory License Registration

Overview:- It allows the Registrar to temporary license in the applicant if the name of the applicant d oes not exists in the database.

1. Use case:- Reg\_Per\_Lic

Actor:- Registrar

Purpose:- It is used to Permanent License Registration

Overview:- It allows the Registrar to permanant license in the applicant if the name of the applicant does not exists in the database.

1. Use case:- Renew\_Driv\_Lic

Actor:- Registrar

Purpose:- It is used to Renew driving license

Overview:- It allows the Registrar to renew the driving license.

1. Use case:- Veh\_Reg

Actor:- Registrar

Purpose:- It is used to Vehicle registration

Overview:- It allows the Registrar to registration of vehicle in the name of applicant if the applicant is not exist in the database.

1. Use case:- Check\_Driv\_Lic

Actor:- Traffic police

Purpose:- It is used to check the details of driving license

Overview:- It allows the Registrar to check the details of driving license like validity, type etc.

Precondition

Basic flow of interaction

Alternate flows of interaction

Post condition

Special condition

**Non-Functional requirements**