**Title**  
**NowScreening TV (NT)** is designed to maintain details about all the TV Shows and notify the users about new episodes as soon as they get released. The website also maintains a user profile and recommends new shows to watch based on user interests for all the registered users.  
  
  
**Authors**  
1. Sai Krishna Karanam - [karanam.s@husky.neu.edu](file:///C:\Users\Amrutha\Downloads\karanam.s@husky.neu.edu)  
2. Anurag Obulampalli - [obulampalli.a@husky.neu.edu](mailto:obulampalli.a@husky.neu.edu)  
  
  
**1.Abstract**

This project deals with maintaining details about all the TV Shows. The idea is to present users with an easier way to register for updates on their favorite shows and maintaining their registration status for multiple TV shows. Along with providing with the registration functionality, NowScreening TV also has other exciting functionalities related to admin and employees. It is an easy to use website which can be improved constantly based on the need.  
  
  
**2.Introduction**

The purpose of this website is to enable users to carry out various functionalities related to subscription for notifications. It maintains record of all TV Shows, seasons and episodes and the information related to users registered and subscribed for all those shows. The primary users of this website can be anyone who wants to subscribe to their favorite shows to receive notification on release of new episodes of that show or an employee who maintains the website for constant updates on user subscriptions and other transactions. The goals achieved through this website includes user login, user registration status, maintaining payment transactions of users, subscriptions, updating tv shows, seasons, episodes and their ratings, etc.  
  
  
**3.Requirements  
Actors**

1. Admin: The authority that manages the registrations of the NT. One important responsibility of the admin is the creation of new admins and employees. The admin is the top-level authority that can manage the employees, users and other admins.
2. Employee: The employee is an authority who manages the data of NT. An employee adds new TV shows and episodes once they are released to the NT so the users can watch their favorite TV shows. The employee also manages the alerts to the user based on the user subscription.
3. User: The user is a participant of the NT. A user can create an account by signing up for the NT and can access various features like subscribing for TV shows, mark watched shows, episodes and maintain a list of his favorite shows.

**Use Cases**

### **3.1 Add Employee** **[add\_employee]**

|  |
| --- |
| **Description:** The admin adds an employee in the system to manage employee activities. |
| **Step-by-step Description:**   1. [#Admin] – An admin enters the employee credentials to the system through the add employee page. 2. [#NT] - The system queues the request, sets the credentials and sends an email to the employee with the credentials. |

### **3.2 Remove User** **[remove\_user]**

|  |
| --- |
| **Description:** The Admin and Employee access this use case to remove users in the system. |
| **Step-by-step Description:**   1. [#Admin/#Employee] – An admin or employee notifies the user of removal. 2. [#User] - The user checks the notes and responds accordingly. 3. [#Admin/#Employee] – The admin or employee requests the system to remove the user from the system if he is not satisfied with the user response. 4. [#NT] - The system queues the request and removes the user from the system. |

### **3.3 Remove Employee** **[remove\_employee]**

|  |
| --- |
| **Description:** The Admin access this use case to remove an employee in the system. |
| **Step-by-step Description:**   1. [#Admin] – An admin notifies the employee of removal. 2. [#Admin] – The admin requests the system to remove the employee. 3. [#NT] - The system queues the request and removes the employee from the system. |

### **3.4 Add Admin** **[add\_admin ]**

|  |
| --- |
| **Description:** The admin adds another admin in the system to manage employee activities. |
| **Step-by-step Description:**   1. [#Admin] – An admin enters the new admin credentials to the system through the add admin page. 2. [#NT] - The system queues the request, sets the credentials and sends an email to the admin with the credentials. |

### **3.5 Remove Admin** **[remove\_admin]**

|  |
| --- |
| **Description:** The Admin access this use case to remove an admin in the system. |
| **Step-by-step Description:**   1. [#Admin] – An admin notifies the admin of removal. 2. [#Admin] – The admin requests the system to remove the admin. 3. [#NT] - The system queues the request and removes the admin from the system. |

### **3.6 Add TV Show** **[add\_tv\_show]**

|  |
| --- |
| **Description:** The Employee access this use case to add a new TV Show to the system. |
| **Step-by-step Description:**   1. [#Employee] – Requests the system to add a new TV show through Add TV Show page. 2. [#NT] - The system queues the request and adds the TV Show to the system. |

### **3.7 Add New Episode** **[add\_new\_episode]**

|  |
| --- |
| **Description:** The Employee access this use case to add a new episode of a TV Show to the system. |
| **Step-by-step Description:**   1. [#Employee] – Requests the system to add a new episode through Add New Episode page. 2. [#NT] - The system queues the request and adds the new episode to the system. |

### **3.8** **Pause Alerts for User** **[pause\_alerts\_for\_user]**

|  |
| --- |
| **Description:** The Employee access this use case to pause all the alerts for a user in the system. |
| **Step-by-step Description:**   1. [#Employee] – Requests the system to pause all alerts for a user in the system through Pause Alerts for User page. 2. [#NT] - The system queues the request and removes the user from notification list in the system. |

### **3.9 Un-Pause Alerts for User** **[un-pause\_alerts\_for\_user]**

|  |
| --- |
| **Description:** The Employee access this use case to un-pause all the alerts for a user in the system after the user pays his bill. |
| **Step-by-step Description:**   1. [#Employee] – Requests the system to un-pause all alerts for a user in the system through Un-Pause Alerts for User page after the user pays his bill. 2. [#NT] - The system queues the request and adds the user from notification list in the system. |

### **3.10 Subscribe for a TV Show [subscribe\_for\_a\_tv\_show]**

|  |
| --- |
| **Description:** The user wants to subscribe for a TV Show. |
| **Step-by-step Description:**   1. [#User] – A user requests that a new subscription be made for a TV Show. 2. [#NT] - The system queues the request and accepts the subscription and notifies the user that the subscription was accepted. 3. [#User] - The user receives the notification. |

### **3.11 Mark Watched Season [mark\_watched\_season]**

|  |
| --- |
| **Description:** The user wants to mark a watched season. |
| **Step-by-step Description:**   1. [#User] – The user submits an item identifier to mark a watched season of a TV Show. 2. [#NT] - The system confirms the submission of the item identifier. |

### **3.12 Mark Watched Episode [mark\_watched\_episode]**

|  |
| --- |
| **Description:** The user wants to mark a watched episode. |
| **Step-by-step Description:**   1. [#User] – The user submits an item identifier to mark a watched episode of a TV Show. 2. [#NT] - The system confirms the submission of the item identifier. |

### **3.13 Retrieve Watched [retrieve\_watched]**

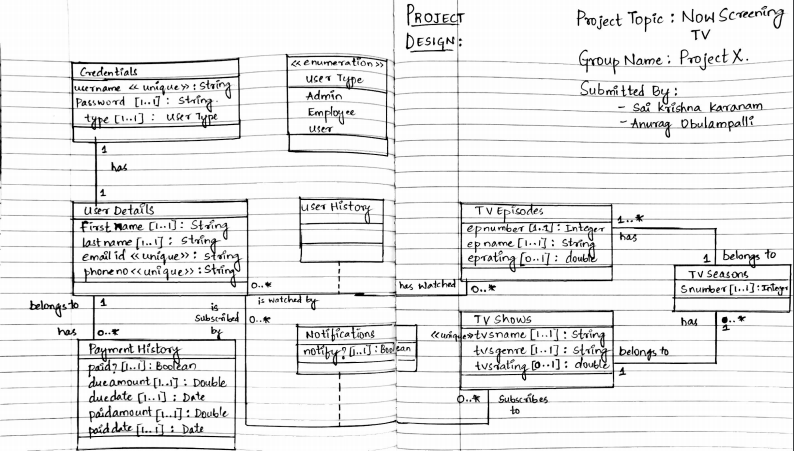
|  |
| --- |
| **Description:** The user wants to view list of watched seasons and episodes. The system retrieves the list of watched seasons and episodes by that user and displays it to the user. |
| **Step-by-step Description:**   1. [#User] – The user submits an item identifier to view list of watched seasons and episodes. 2. [#NT] - The system returns the list of watched seasons and episodes of the user. 3. [#User] – The user views the list of watched seasons and episodes returned by the system. |

### **3.14 Pay Bill [pay\_bill]**

|  |
| --- |
| **Description:** The user wants to pay the bill for a given billing cycle. |
| **Step-by-step Description:**   1. [#User] – The user submits a request to pay the bill. 2. [#NT] - The system queues the request and redirects to the payment options. 3. [#User] – The user selects the payment mode and makes the payment. 4. [#NT] - The system receives the payment and notifies the user. 5. [#User] – The user receives the notification of the successful transaction. |

**4.Design**

**UML Diagram:**



**5.Implementation**

The following were used to develop the project.

**Programming Language: Java**

**IDE: Eclipse**

**D.B.M.S: MySQL 5.7**

The software is run from the command line interface. It first gives the option to choose from Existing User, New User, Sponsor and to Exit the application.



Figure 5: Login options

1. Introduction: User explores all the TV Shows and can subscribe to his favorite TV shows.

Details: The user is logged into NT to view all the available TV shows. Once he is logged in, he can select option ‘2’ which is related to the use case of exploring the TV shows with all the other details about the show. After that, the user can subscribe for any TV show that he likes.

Database Tables:  
 Table Name: Tvshows  
 Table Fields: id int primary key auto\_increment,

tvsname varchar(255) not null,

  tvsgenre varchar(255) not null,

  tvrating double

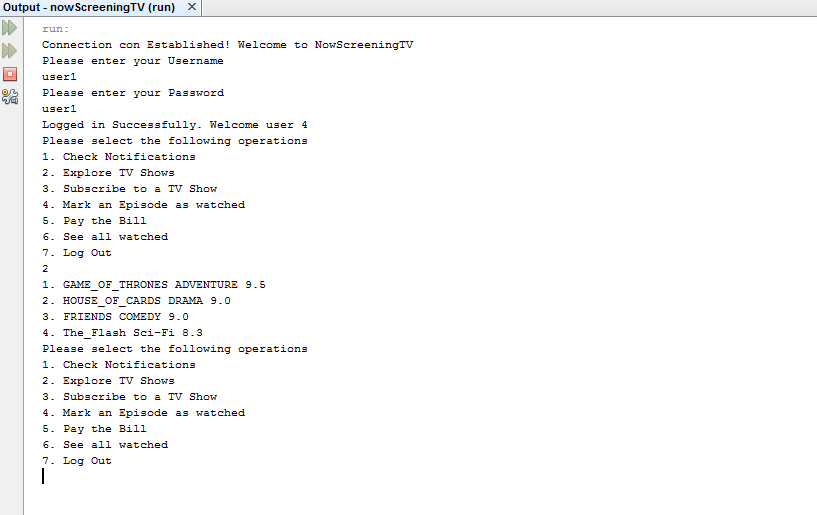


Figure 5.1 User explores all the TV Shows

2. Introduction: User checks notifications for the subscribed TV Shows and views the list of watched.  
  
Details: The user is logged into ET to check his notifications and to view the list of tv shows he has watched. For this, user can select option ‘6’ which is related to the use case of retrieve watched. The user can select option ‘1’ to check his notifications for the tv shows he has subscribed.

Database Tables:  
 Table Name: Userhistory  
 Table Fields: isWatchedBy int not null,

  foreign key(isWatchedBy) references Userdetails(id)

  on update cascade on delete cascade,

  hasWatched int not null,

  foreign key(hasWatched) references tvEpisodes(id)

  on update cascade on delete cascade,

  primary key(isWatchedBy,hasWatched)

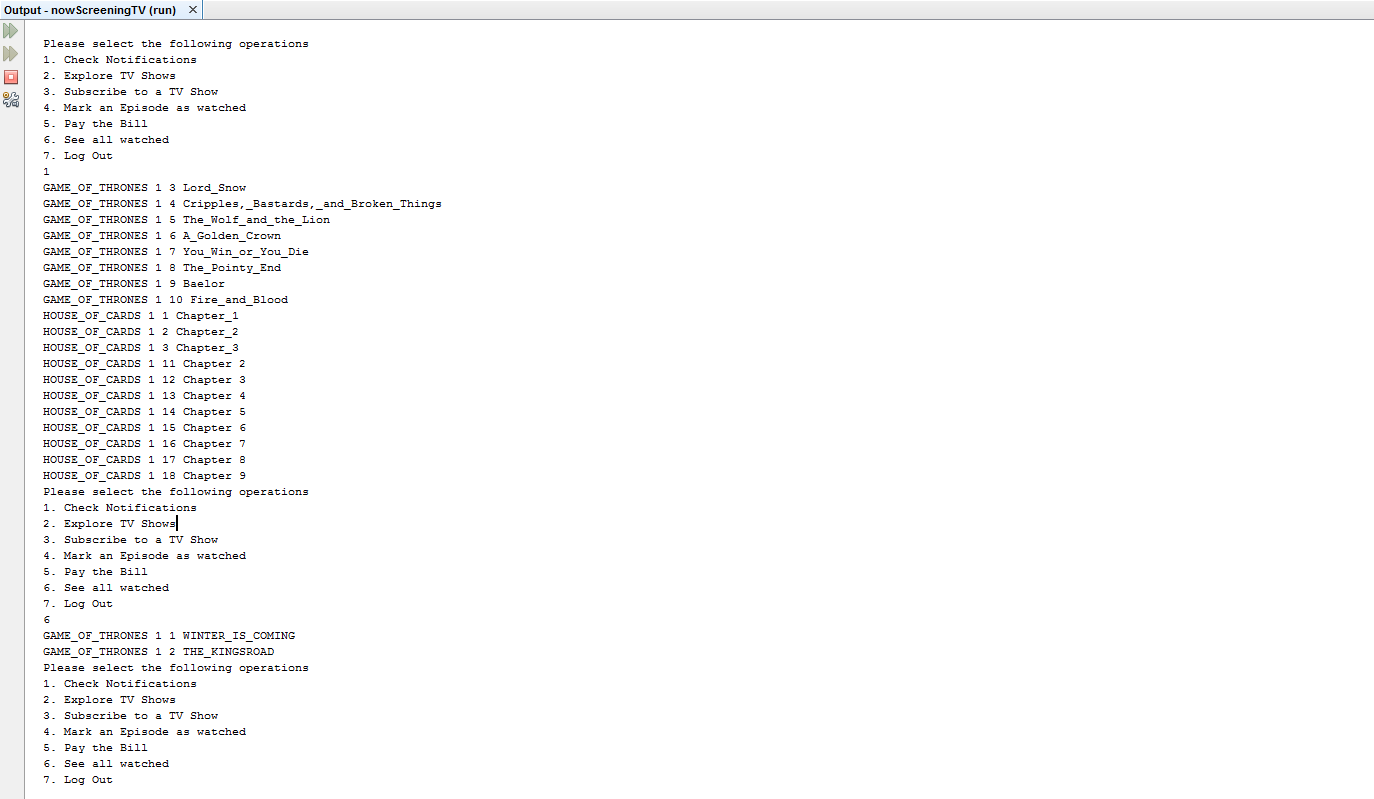


Figure 5.2.1 Check Notifications and See all watched.

Error Handling: Errors could occur if the user tries to mark an episode that doesn’t exist as watched. This case would print the respective error messages to the user.

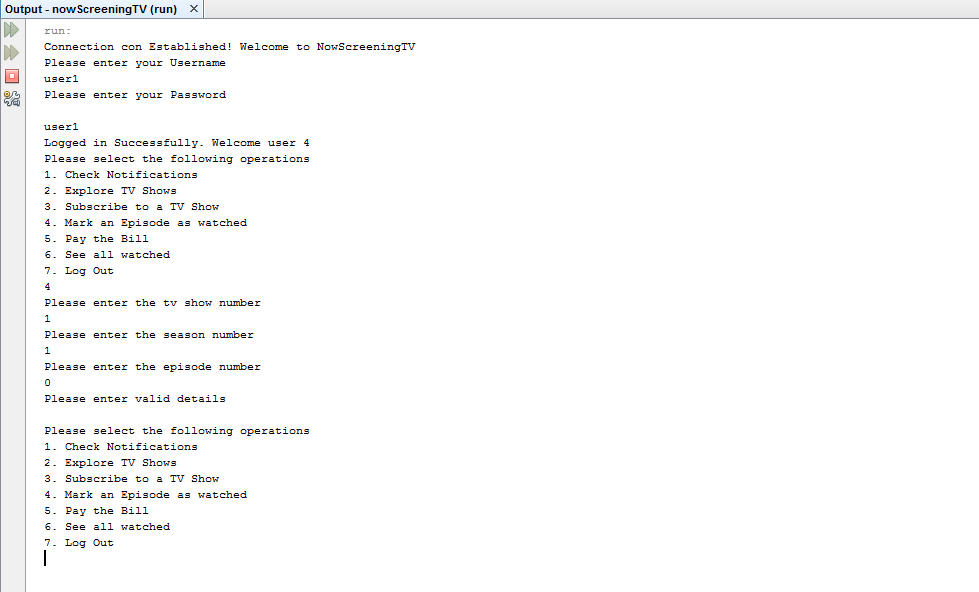


Figure 5.2.2: User tries to mark an episode that doesn’t exist as watched. Error thrown: “Please enter valid details”

3. Introduction: Employee adds a new TV show to NowScreening TV.

Details: The employee is logged into ET to add a new TV show. He can select option ‘1’ for this purpose which is related to the use case add new tv show.

Database Tables:  
 Table Name: Tvshow  
 Table Fields: id int primary key auto\_increment,

tvsname varchar(255) not null,

  tvsgenre varchar(255) not null,

  tvrating double

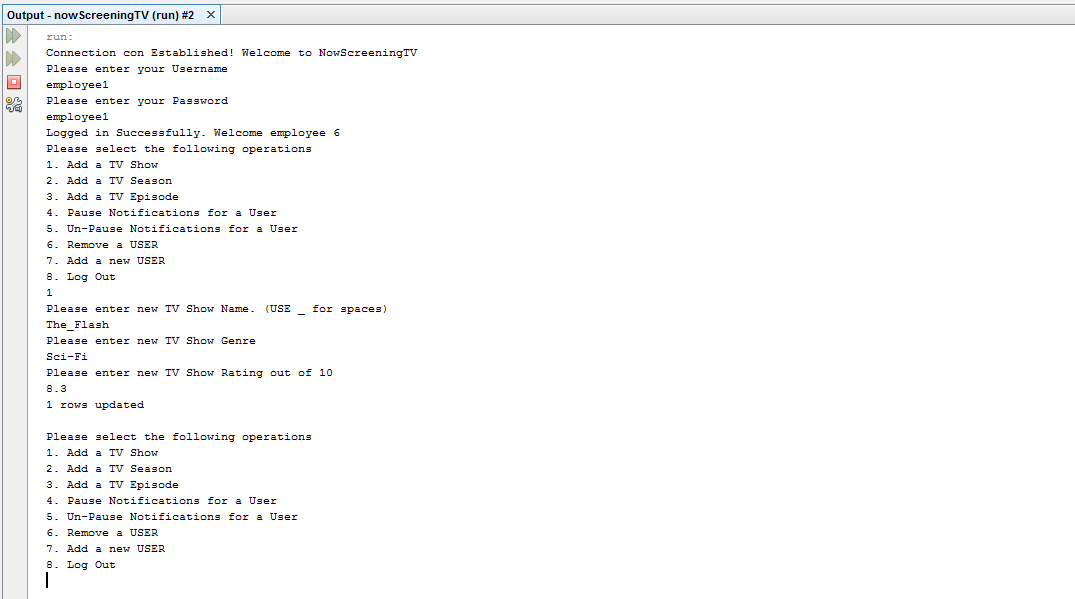


Figure 5.2.3: Employee adds a new TV Show to NowScreening TV.

**6.Discussion**

The following results were achieved in this project:

* Searching capabilities:

The software can be used by the user to register for an event of his choice. For this, the software provides a search option using which he can search for the upcoming events. There are numerous ways in which he can filter his search:

* View all the upcoming events in ascending order.
* View the events filtered by the category/type.For example, Music, Sports,etc.
* View the events in ascending order of the price of the ticket.
* Event Registration:

Once the user decides upon the event for which he wants to book a ticket for, he needs to type the name of the event to register for it View all the upcoming events in ascending order.

* View Registration Details:

The user can view the details of his registration and his account anytime. He must be logged in for this purpose.

* Cancellation of a ticket:

The software also allows him to cancel his ticket. The user must enter the name of the event he wishes to cancel. Once the ticket is cancelled, his/her seat now becomes available for another user to book.

* Deleting the account:

If the user does not want to be a part of the EMS, he can cancel delete his account.

* Cancellation of a ticket:

The software also allows him to cancel his ticket. The user must enter the name of the event he wishes to cancel. Once the ticket is cancelled, his/her seat now becomes available for another user to book.

* Help for the user:

The software is user friendly. It provides help to the users in multiple ways:

* If the user has any queries regarding using the software, he can view the FAQs(Frequently Asked Questions).
* He can also retrieve the contact details of the event manager in case he has any queries regarding an event.
* He can also view the contact details of the EMS team in case of any general queries regarding the software.

**7. Conclusion**

The project successfully implements the all actions pertaining to event management such as user account creation, user account deletion, event registration, searching events and cancellation of ticket. The user can also subscribe to mailing list and view contact details of the concerned authority for assistance. In addition, the project incorporates features which enable a sponsor to sponsor for an event.

**Further work suggested:**

The current implementation uses command line interface for user inputs. The project could be developed into a Graphical User Interface to make it more user friendly.  
  
**References**

# 1. <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/> . Java JDBC API

2.<http://www.tutorialspoint.com/jdbc/> . TutorialsPoint

#### 3. <http://dev.mysql.com/doc/refman/5.7/en/create-table-foreign-keys.html> [MySQL 5.7 Reference Manual](http://dev.mysql.com/doc/refman/5.7/en/)/[SQL Statement Syntax](http://dev.mysql.com/doc/refman/5.7/en/sql-syntax.html)/[Data Definition Statements](http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-data-definition.html)/[CREATE TABLE Syntax](http://dev.mysql.com/doc/refman/5.7/en/create-table.html)/Using FOREIGN KEY Constraints Using FOREIGN KEY Constraints

4. Baclawski , Kenneth,CS5200 Database Management Translating to Relational ,<https://piazza.com/northeastern/fall2016/cs5200/resources> L02.pdf