CPSC304\_Project Proposal

**1.Project description: what is the domain, what aspects of the domain will be modeled by the database.**

The domain that we will model is going to be on Japanese animated series. We will cover information such as characters, voice actors of the anime and so on, manga that the anime originated from and an anime forum that are accessed by three different kinds of users(Admin, guest and member).

Animated series has a title as its identifier. It’s the superclass of manga and anime. The combination of title from animated series and season number defines the key of an anime. Similarly, the combination of animated series title and manga author serves as key to a manga. The manga entity is optional as not all animated series originate from manga. Each manga and anime has a rating to show how popular it is.

Voice actors voice characters in anime; therefore, we have a ternary relationship “voice” that connects anime, character and voice actors together. Voice actors are identified by their name and birthday, their biography and gender are also part of the entity. A character in an anime has name, description (biography) and rating of how popular the character is.

Each forum contains date and title, and it’s identified by forum ID. The three different types of users of the forum are admin, regular member and guest user. Each user is identified by their unique email address. Admin and regular member have username, real name, and password. Admin is allowed to create the forum, its subtopics and write comment on the forum whereas regular members are only allowed to create subtopics of the forum and comment. A guest user is allowed to read the forum only.

**2.An E/R diagram for the database the application will use. (I suggest using Microsoft Visio or the online (Attached at the back)**

**3.What are the application specifications? (i.e., what functionality will the system provide)**

### 1.10 Add Anime

**User:** Admin User

**Input:** Title,genre,rating, description,season,status, start date, studio, #of episode

Opition: Profile picture

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether an anime already exists, by checking the primary key of database enity: title and season. If the anime is not in the database, a new anime entry is added to the database and “OK” is returned. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If anime is in the database, adding is aborted and "Cancelled" is returned.
* If either of the title or season is empty, “Cancelled” is returned.

### 1.11 Remove Anime

**User:** Admin User

**Input:** Table name of anime and specific attribute of anime we want to remove

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether an anime already exists, by checking the input from the admin user. If the anime is in the database then this anime should be deleted and returned “OK” otherwise an exception “Cancelled” is returned. Since character is an entity depending on the anime enity, when an anime is deleted, the character is deleted as well.

**Exceptions**:

* If anime is not in the database, removing is aborted and "Cancelled" is returned.
* If an anime is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 1.12 Update Anime

**User:** Admin User

**Input:** Table name of anime and specific attribute of anime we want to update

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether an anime already exists, by checking the input from the admin user. If the anime is in the database then this anime should be updated and returned “OK”, otherwise an exception “Cancelled” is returned. On update the system will update the same attribute changed wherever it appears in other tables.

**Exceptions**:

* If anime is not in the database, update is not done and "Cancelled" is returned.
* If an anime is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

### 1.13 Search by Anime

**User:** Admin User,Member

**Input:** Name of anime we want to search

**Output:** "OK" or "Cancelled" and Table with results

**Basic Case**:

The system will look into the database and determine whether the searched anime exists, by checking the input from the user. If the anime is in the database then this anime should be returned, otherwise an exception “Cancelled” is returned..

**Exceptions**:

* If anime is not in the database, search is not done and "Cancelled" is returned.

### 1.20 Add Character

**User:** Admin User

**Input:** Name, anime, description, rating

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a character already exists, by checking the primary key of this enity: Name and anime. If the character is not in the database, a new character entry is added to the database and returned “OK”. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If a character is in the database, adding is aborted and "Cancelled" is returned.
* If either of the name or anime is empty, “Cancelled” is returned.

### 

### 1.21 Remove Character

**User:** Admin User

**Input:** Table name of character and specific attribute of character of whom we want to remove.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a character already exsists, by checking the input from the admin user. If the character is in the database, then this character should be deleted and returned “OK” otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If character is not in the database, removing is aborted and "Cancelled" is returned.
* If a character is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 

### 

### 1.22 Update Character

**User:** Admin User

**Input:** Table name of character and specific attribute of character of whom we want to update.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a character already exists, by checking the input from the admin user. If the character is in the database, then this character should be updated and returned “OK” otherwise an exception “Cancelled” is returned. On update the system will also update the same attribute in other tables as well.

**Exceptions**:

* If character is not in the database, updating is aborted and "Cancelled" is returned.
* If a character is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

### 1.23 Search by Character

**User:** Admin User,Member

**Input:** Name of Character we want to search

**Output:** "OK" or "Cancelled" and Table with results

**Basic Case**:

The system will look into the database and determine whether the searched character exists, by checking the input from the user. If the character is in the database then this Character should be returned, otherwise an exception “Cancelled” is returned..

**Exceptions**:

* If character is not in the database, search is not done and "Cancelled" is returned.

### 1.30 Add Voice Actor

**User:** Admin User

**Input:** Name, birthday, gender, bio

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a voice actor already exsists, by checking the primary key of this enity: Name and birthday. If the voice actor is not in the database, a new voice actor entry is added to the database and returned “OK”. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If an voice actor is in the database, adding is aborted and "Cancelled" is returned.
* If either of the name or birthday is empty, “Cancelled” is returned.

### 1.31 Remove Voice Actor

**User:** Admin User

**Input:** Table name of voice actor and voice actor with specific attributes we want to remove.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a voice actor already exists, by checking the input from the admin user. If the voice actor is in the database then this character should be deleted and returned “OK” otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If the voice actor is not in the database, removing is aborted and "Cancelled" is returned.
* If a voice actor is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 1.32 Update Voice Actor

**User:** Admin User

**Input:** Table name of voice actor and voice actor with specific attributes we want to update.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a voice actor already exists, by checking the input from the admin user. If the voice actor is in the database then this actor should be updated and returned “OK”, otherwise an exception “Cancelled” is returned. On update, the system will update the same attribute in other tables too.

**Exceptions**:

* If the voice actor is not in the database, updating is aborted and "Cancelled" is returned.
* If a voice actor is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

### 1.33 Search by Voice Actor

**User:** Admin User,Member

**Input:** Name of Voice Actor we want to search

**Output:** "OK" or "Cancelled" and Table with results

**Basic Case**:

The system will look into the database and determine whether the searched Voice Actor exists, by checking the input from the user. If the Voice Actor is in the database then this Voice Actor should be returned, otherwise an exception “Cancelled” is returned..

**Exceptions**:

* If Voice is not in the database, search is not done and "Cancelled" is returned.

### 1.40 Add Manga

**User:** Admin User

**Input:** Title,genre,rating, description,author, description,volume, published date,.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a manga already exsists, by checking the primary key of this enity: Title and author .If the manga is not in the database, a new manga entry is added to the database and returned “OK”. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If the manga is in the database, adding is aborted and "Cancelled" is returned.
* If either Title or author is empty, “Cancelled” is returned.

### 1.41 Remove Manga

**User:** Admin User

**Input:** Table name of mange and manga with specific attributes we want to remove.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a manga already exsists, by checking the input from the admin user. If the manga is in the database then this character should be deleted and returned “OK” otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If the manga is not in the database, removing is aborted and "Cancelled" is returned.
* If manga is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 1.42 Update Manga

**User:** Admin User

**Input:** Table name of mange and manga with specific attributes we want to update.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a manga already exists, by checking the input from the admin user. If the manga is in the database then this manga should be updated and return “OK”, otherwise an exception “Cancelled” is returned. On update the system also updates the same attribute in other tables too.

**Exceptions**:

* If the manga is not in the database, updating is aborted and "Cancelled" is returned.
* If manga is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

### 1.43 Search by Manga

**User:** Admin User,Member

**Input:** Name of Manga we want to search

**Output:** "OK" or "Cancelled" and Table with results

**Basic Case**:

The system will look into the database and determine whether the searched Manga exists, by checking the input from the user. If the Manga is in the database then this Manga should be returned, otherwise an exception “Cancelled” is returned..

**Exceptions**:

* If Manga is not in the database, search is not done and "Cancelled" is returned.

### 1.50 Add Forum

**User:** Admin User

**Input:** Forum ID, name, date

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a forum is already exists, by checking the primary key of this enity: Name and Forum ID. If the forum is not in the database, a new forum entry is added to the database and returned “OK”. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If a forum is in the database, adding is aborted and "Cancelled" is returned.
* If either of the name or forum ID is empty, “Cancelled” is returned.

### 

### 1.51 Remove Forum

**User:** Admin User

**Input:** Table name of forum and forum with specific attribute that we want to remove.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a forum already exsists, by checking the input from the admin user. If the forum is in the database, then this forum should be deleted and returned “OK” otherwise an exception “Cancelled” is returned. Also, since Subtopic and Comment are weak enities that depend on Forum, if a specifc forum is deleted then all the Subtopic and Comment that rely on this forum should be deleted.

**Exceptions**:

* If Forum is not in the database, removing is aborted and "Cancelled" is returned.
* If a Forum is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 1.52 Update Forum

**User:** Admin User

**Input:** Table name of forum and forum with specific attribute that we want to update.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a forum already exists, by checking the input from the admin user. If the forum is in the database, then this forum should be updated and “OK” is returned otherwise an exception “Cancelled” is returned. Also, since Subtopic and Comment are weak enities that depend on Forum, if a specific forum primary key is updated then all the Subtopic and Comment that rely on this forum should be updated.

**Exceptions**:

* If Forum is not in the database, updating is aborted and "Cancelled" is returned.
* If a Forum is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

### 1.60 Add Comment

**User:** Admin User,Member

**Input:** Forum ID, msg, date

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a comment is already exists, by checking the primary key of this enity: Name and Forum ID. If the forum is not in the database, a new forum entry is added to the database and returned “OK”. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If a forum is in the database, adding is aborted and "Cancelled" is returned.
* If either of the name or forum ID is empty, “Cancelled” is returned.

### 

### 1.61 Remove Comment

**User:** Admin User,Member

**Input:** Table name of comment and comment with specific attribute that we want to remove.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a comment already exsists, by checking the input from the admin user. If the comment is in the database, then this comment should be deleted and returned “OK” otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If Comment is not in the database, removing is aborted and "Cancelled" is returned.
* If a Comment is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 1.62 Update Comment

**User:** Admin User,Member

**Input:** Table name of comment and comment with specific attribute that we want to update.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a comment already exists, by checking the input from the admin or member. If the comment is in the database, then this comment should be editted and returned “OK” otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If Comment is not in the database, updating is aborted and "Cancelled" is returned.
* If a Comment is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

### 1.63 Search by Comment

**User:** Admin User,Member

**Input:** Name of comment we want to search

**Output:** "OK" or "Cancelled" and Table with results

**Basic Case**:

The system will look into the database and determine whether the searched Comment exists, by checking the input from the user. If the Comment is in the database then this Comment should be returned, otherwise an exception “Cancelled” is returned..

**Exceptions**:

* If Comment is not in the database, search is not done and "Cancelled" is returned.

### 1.70 Add Subtopic

**User:** Admin User, Member

**Input:** Forum ID, date created, title

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a Subtopic already exists, by checking the primary key of this enity: Forum ID, Date created and title. If the Subtopic is not in the database, a new topic entry is added to the database and returned “OK”. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If a subtopic is in the database, adding is aborted and "Cancelled" is returned.
* If forum ID, date create or title is empty, “Cancelled” is returned.

### 

### 1.71 Remove Subtopic

**User:** Admin User,Member

**Input:** Table name of topic and topic with specific attribute that we want to remove.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a topic already exsists, by checking the input from the admin user. If the topic is in the database, then this topic should be deleted and returned “OK” otherwise an exception “Cancelled” is returned. Also, since Comment is a weak enities that depends on Subtopic, if a specifc Subtopic is deleted then all the Comments that rely on this forum should be deleted.

**Exceptions**:

* If Subtopic is not in the database, removing is aborted and "Cancelled" is returned.
* If a Subtopic is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 1.72 Update Subtopic

**User:** Admin User,Member

**Input:** Table name of topic and topic with specific attribute that we want to update.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a topic already exists, by checking the input from the admin or member. If the topic is in the database, then this topic should be updated and returned “OK” otherwise an exception “Cancelled” is returned. Also, since Comment is a weak enities that depends on Subtopic, if a specifc Subtopic primary key is updated then all the Comments that rely on this forum should be updated.

**Exceptions**:

* If Subtopic is not in the database, updating is aborted and "Cancelled" is returned.
* If a Subtopic is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

### 1.73 Search by Subtopic

**User:** Admin User,Member

**Input:** Name of Subtopic we want to search

**Output:** "OK" or "Cancelled" and Table with results

**Basic Case**:

The system will look into the database and determine whether the searched Subtopic exists, by checking the input from the user. If the Subtopic is in the database then this Subtopic should be returned, otherwise an exception “Cancelled” is returned..

**Exceptions**:

* If Subtopic is not in the database, search is not done and "Cancelled" is returned.

### 1.80 Finding most loved character(Aggregation query)

**User:** All User

**Input:** The Character table and the rating field.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the Characetr table in database and return the character with the MAX(rating).

**Exceptions**:

* If all the characters have zero rating then “Cancelled” is returned.
* If more than one character have the same high rating then all the characters with the same rating would return.

### 1.90 Search for Manga that has been adapted into Anime(Join query)

**User:** Admin User,Member

**Input:** Anime table, Manga table

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a Manga has an Anime adaptation. If the joined query is successful and in the database, then this query should return “OK” and a table of results, otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If no adpatations exist then return “Cancelled”
* if multiple adaptations exist return all adapations.

### 2.00 Search for authors that created more than five manga(Nested aggregation with group-by)

**User:** User,Member

**Input:** Manga table

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the Manga table in database and return the authors with COUNT(manga) > 5 . If certain authors meet the requirement then “OK” is returned. If none of the authors create more than five manga then “cancelled ” is returned.

**Exceptions**:

* If no authors exist then return “Cancelled”
* if multiple authors exist return all adapations.

### 2.10 Search for authors that created more than five manga(Nested aggregation with group-by)

**User:** User,Member

**Input:** Manga table

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the Manga table in database and return the authors with COUNT(manga) > 5 . If certain authors meet the requirement then “OK” is returned. If none of the authors create more than five manga then “cancelled ” is returned.

**Exceptions**:

* If no authors exist then return “Cancelled”
* if multiple authors exist return all adapations.

### 2.20 Add User

**User:** Admin User

**Input:** Email, name, username, password

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a user already exists, by checking the primary key of this enity: Email. If the User is not in the database, a new User entry is added to the database and returned “OK”. Otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If a user is in the database, adding is aborted and "Cancelled" is returned.
* If email is empty, “Cancelled” is returned.

### 

### 2.21 Remove User

**User:** Admin User

**Input:** Table name of user and user with specific attribute that we want to remove.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a user already exsists, by checking the input from the admin user. If the user is in the database, then this user should be deleted and returned “OK” otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If User is not in the database, removing is aborted and "Cancelled" is returned.
* If a User is in the database but an incorrect attribute combination is provided, removing is aborted and "Cancelled" is returned.

### 2.22 Update User

**User:** Admin User

**Input:** Table name of User and User with specific attribute that we want to update.

**Output:** "OK" or "Cancelled"

**Basic Case**:

The system will look into the database and determine whether a user already exists, by checking the input from the admin user. If the user is in the database, then this forum should be updated and “OK” is returned otherwise an exception “Cancelled” is returned.

**Exceptions**:

* If Forum is not in the database, updating is aborted and "Cancelled" is returned.
* If a Forum is in the database but an incorrect attribute combination is provided, updating is aborted and "Cancelled" is returned.

**2.30 Extra features1 (Let the member user could comment in our forum)**

**2.31 Extra features2 (Let the guest user login into our website with facebook account)**

**4.What platforms do you plan for the final project be on (i.e., Oracle, Java).**

We will be using CS department’s oracle database, PHP for our website’s interactive part with database, css for styles and fonts, and HTML5 and maybe Twitter bootstrap for layout of the website. We may also use Facebook API that allow Facebook user to access our web application. We are not planning on using any special hardware or software..