

## Project Stage 2

The database should store information about **channel**, **workout videos based on trainer**, **workout videos based on workout type**, **recipe videos**, **user information** and **forum**.

- A **Channel** is uniquely identified by its `channel_id`. Other channel attributes are `channel_name` and `subscriber_num`.
- A trainer's workout video (**Workout\_Video\_Trainer**) is uniquely identified by its `video_id` and `channel_id`. Other trainers' workout video attributes are `video_url`, `video_title`, `time_length`, `release_date`, `video_views` and `trainer`.
- A workout video based on type (**Workout\_Video\_Type**) is uniquely identified by `video_id` and `channel_id`. Other workout videos based on type attributes are `video_url`, `video_title`, `time_length`, `release_date`, `video_views` and `workout_type`.
- A recipe video (**Recipe\_Video**) is uniquely identified by its `video_id`. Some other attributes are `video_url`, `video_title`, `time_length`, `release_date` and `video_views`.
- A **User** is uniquely identified by their `user_id`. Some other attributes are `email`, `phone_number` and `password`.
- A **Forum** post is uniquely identified by its `post_id` and `user_id`. Some other attributes are `post_timestamp`, `post_title` and `post_content`.

Description of relationship:

- A channel can have multiple workout videos based on the trainer, and each workout video based on the trainer belongs to only one channel.
- A channel can have multiple workout videos based on type, and each workout video based on type belongs to only one channel.
- A channel can have multiple recipe videos, and each recipe video belongs to only one channel.
- A workout video based on the trainer can be watched by any user and a user can watch any workout video(s) based on the trainer.
- A workout video based on type can be watched by any user and a user can watch any workout video(s) based on type.
- A recipe video can be watched by any user and a user can watch any recipe video(s).
- A user can post any number of posts on the forum and the forum can have any number of users posting.

Assumptions:

- For the relationship between **Channel** and **Workout\_Video\_Trainer**, each channel should have at least one workout video based on the trainer.
- For the relationship between **Channel** and **Workout\_Video\_Type**, each channel should have at least one workout video based on workout type.
- For the relationship between **Channel** and **Recipe\_Video**, each channel should have at least one recipe video.
- For the relationship between **User\_Information** and **Forum**, a user can be registered but have no post on the forum.

- For the relationship between **User\_Information** and **Workout\_Video\_Trainer**, a user can be registered but has not watched any workout video based on the trainer.
- For the relationship between **User\_Information** and **Workout\_Video\_Type**, a user can be registered but has not watched any workout video based on type.
- For the relationship between **User\_Information** and **Recipe\_Video**, a user can be registered but has not watched any recipe video.
- For **Channel**, the channel name and the number of subscribers cannot be null values.
- For **Workout\_Video\_Trainer**, channel id, video url, video title, time length, the date a video was released, the number of views, and trainer cannot be null values.
- For **Workout\_Video\_Type**, channel id, video url, video title, time length, the date a video was released, the number of views, and workout type cannot be null values.
- For **Recipe\_Video**, channel id, video url, video title, time length, the date a video was released, and the number of views cannot be null values.
- For **User\_Information**, password and email cannot be null values, but phone number can be null values.
- For **Forum**, post content, post timestamp, and post title cannot be null values.

\*Note: Two categories of videos can be found on the website: one based on trainers and the other based on workout types. For videos based on the trainer, only channels with the trainer's name included in the channel name are selected (e.g. Channel names of both *LilyYoga* and *Lily* are valid to be chosen).

## Relational Schema:

Channel(channel\_id, channel\_name, subscriber\_num)

Workout\_Video\_Trainer(video\_id, channel\_id , video\_url, video\_title, time\_length, release\_date, video\_views, trainer)

Workout\_Video\_Type(video\_id, channel\_id , video\_url, video\_title, time\_length, release\_date, video\_views, workout\_type)

Recipe\_Video(video\_id, channel\_id , video\_url, video\_title, time\_length, release\_date, video\_views)

User\_Information(user\_id, email, phone\_number, password)

Forum(post\_id, user\_id, post\_timestamp, post\_title, post\_content)

Or:

Channel(channel\_id[PK], channel\_name, subscriber\_num)

Workout\_Video\_Trainer(video\_id[PK], channel\_id[FK to Channel.channel\_id], video\_url, video\_title, time\_length, release\_date, video\_views, trainer)

Workout\_Video\_Type(video\_id[PK], channel\_id[FK to Channel.channel\_id], video\_url, video\_title, time\_length, release\_date, video\_views, workout\_type)

Recipe\_Video(video\_id[PK], channel\_id[FK to Channel.channel\_id], video\_url, video\_title, time\_length, release\_date, video\_views)

User\_Information(user\_id[PK], email, phone\_number, password)

Forum(post\_id[PK], user\_id[FK to User\_information.user\_id], post\_timestamp, post\_title, post\_content)

```
CREATE TABLE Channel(  
    channel_id INT NOT NULL,  
    channel_name VARCHAR(255) NOT NULL,  
    subscriber_num INT NOT NULL,  
    PRIMARY KEY (channel_id) )
```

```
CREATE TABLE Workout_Video_Trainer(  
    video_id INT NOT NULL,  
    channel_id INT NOT NULL,  
    video_url VARCHAR(255) NOT NULL,  
    video_title VARCHAR(255) NOT NULL,  
    time_length VARCHAR(255) NOT NULL,  
    release_date DATE NOT NULL,  
    video_views INT NOT NULL,  
    trainer VARCHAR(255),  
    PRIMARY KEY (video_id, channel_id),  
    FOREIGN KEY(channel_id) REFERENCES Channel(channel_id) ON DELETE  
    CASCADE )
```

```
CREATE TABLE Workout_Video_Type(  
    video_id INT NOT NULL,  
    channel_id INT NOT NULL,  
    video_url VARCHAR(255) NOT NULL,  
    video_title VARCHAR(255) NOT NULL,  
    time_length VARCHAR(255) NOT NULL,  
    release_date DATE NOT NULL,  
    video_views INT NOT NULL,  
    workout_type VARCHAR(255) NOT NULL,  
    PRIMARY KEY (video_id, channel_id),  
    FOREIGN KEY (channel_id) REFERENCES Channel (channel_id) ON DELETE  
    CASCADE )
```

```
CREATE TABLE Recipe_Video(  
    video_id INT NOT NULL,  
    channel_id INT NOT NULL,  
    video_url VARCHAR(255) NOT NULL,  
    video_title VARCHAR(255) NOT NULL,  
    time_length VARCHAR(255) NOT NULL,  
    release_date DATE NOT NULL,  
    video_views INT NOT NULL,  
    PRIMARY KEY (video_id, channel_id),  
    FOREIGN KEY(channel_id) REFERENCES Channel(channel_id) ON DELETE  
    CASCADE )
```

```
CREATE TABLE User_Information(  
    user_id INT NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    phone_number VARCHAR(255),  
    password VARCHAR(255) NOT NULL,  
    PRIMARY KEY (user_id) )
```

```
CREATE TABLE Forum(  
    post_id INT NOT NULL,  
    user_id VARCHAR(255) NOT NULL,  
    post_timestamp VARCHAR(255) NOT NULL,  
    post_title VARCHAR(255) NOT NULL,  
    post_content VARCHAR(5000) NOT NULL,  
    PRIMARY KEY (post_id , user_id),  
    FOREIGN KEY(user_id) REFERENCES User_Information(user_id) ON DELETE  
    CASCADE )
```

UML Diagram:

