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_title, subscriberCount, channel_viewCount and videoCount.
- A trainer’s workout video (**Workout_Video_Trainer**) is uniquely identified by its video_id and channel_id. Other trainers’ workout video attributes are publish_date, video_title, duration, video_viewCount, likeCount 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 publish_date, video_title, duration, video_viewCount, likeCount and workout_type.
- A recipe video (**Recipe_Video**) is uniquely identified by its video_id. Some other attributes are publish_date, video_title, duration, video_viewCount and likeCount.
- 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:

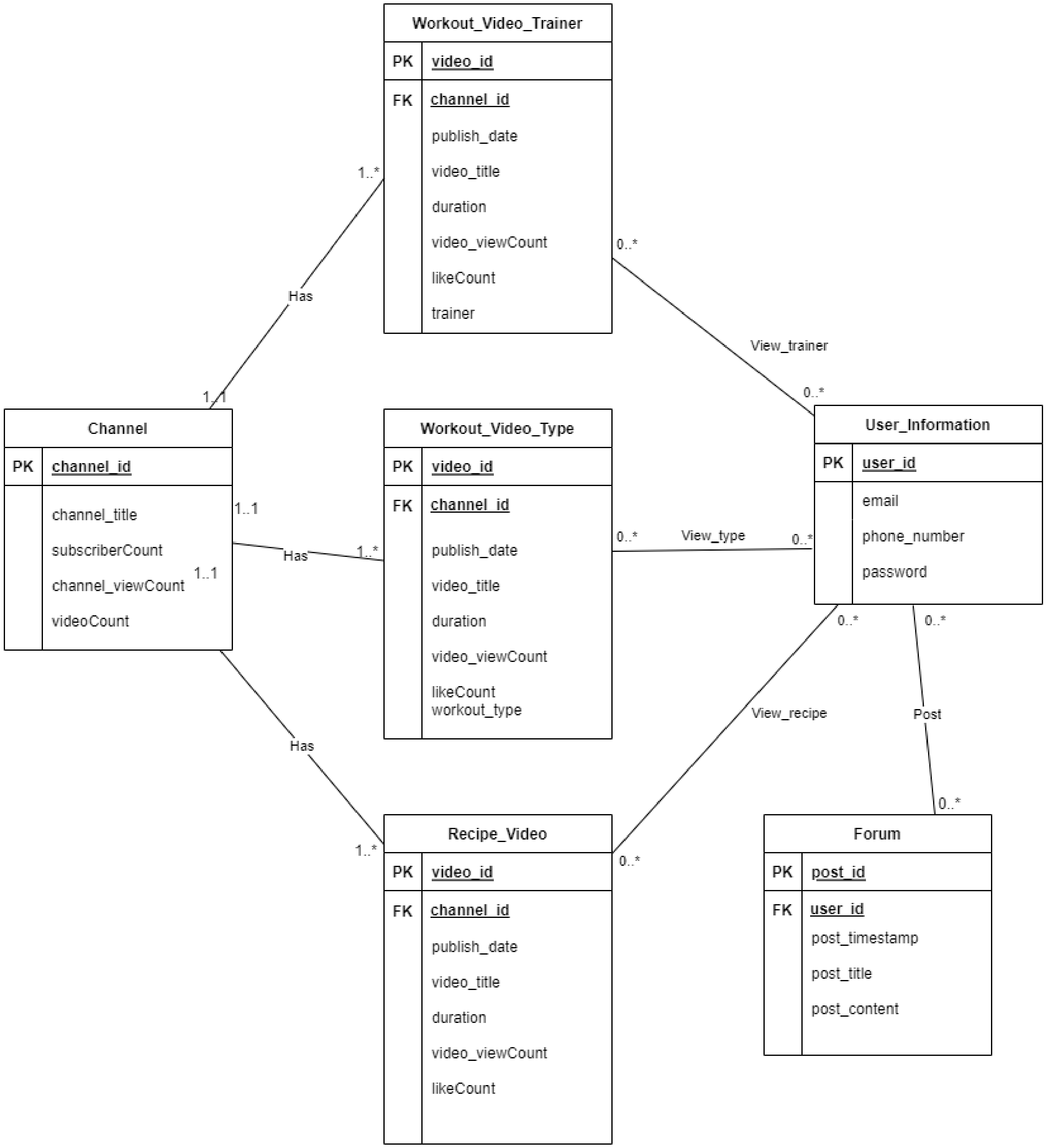
- For the relationship between **Channel** and **Workout_Video_Trainer**, each channel should have at least one workout video based on the trainer.
- For the relationship between **Channel** and **Workout_Video_Type**, each channel should have at least one workout video based on workout type.
- For the 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, the number of subscribers, the number of channel view count and the number of total videos cannot be null values.
- For **Workout_Video_Trainer,** channel id, video title, time length, the date a video was released, the number of views, the number of likes, and trainer cannot be null values.
- For **Workout_Video_Type,** channel id, video title, time length, the date a video was released, the number of views, the number of likes, and workout type cannot be null values.
- For **Recipe_Video,** channel id, video url, video title, time length, the date a video was released, the number of likes, 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).

UML Diagram


Workout_Video_Trainer
PK video_id
FK channel_id
publish_date
video_title
duration
video_viewCount
likeCount
trainer
1..*
Has
1..1
0..*
View_trainer
0..*
Channel
PK channel_id
channel_title
subscriberCount
channel_viewCount
videoCount
1..1
Has
1..*
1..1
Workout_Video_Type
PK video_id
FK channel_id
publish_date
video_title
duration
video_viewCount
likeCount
workout_type
0..*
View_type
0..*
User_Information
PK user_id
email
phone_number
password
0..*
0..*
View_recipe
Post
Has
1..*
Recipe_Video
PK video_id
FK channel_id
publish_date
video_title
duration
video_viewCount
likeCount
0..*
0..*
Forum
PK post_id
FK user_id
post_timestamp
post_title
post_content

Relational Schema:

Channel(channel_id, channel_title, subscriberCount, channel_viewCount, videoCount)
Workout_Video_Trainer(video_id, channel_id, publish_date, video_title, duration, video_viewCount, likeCount, trainer)
Workout_Video_Type(video_id, channel_id, publish_date, video_title, duration, video_viewCount, likeCount, workout_type)
Recipe_Video(video_id, channel_id, publish_date, video_title, duration, video_viewCount, likeCount)
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_title, subscriberCount, channel_viewCount, videoCount)
Workout_Video_Trainer(video_id[PK], channel_id[FK to Channel.channel_id], publish_date, video_title, duration, video_viewCount, likeCount, trainer)
Workout_Video_Type(video_id[PK], channel_id[FK to Channel.channel_id], publish_date, video_title, duration, video_viewCount, likeCount, workout_type)
Recipe_Video(video_id[PK], channel_id[FK to Channel.channel_id], publish_date, video_title, duration, video_viewCount, likeCount)
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 Channels(
        channel_id CHAR(24) NOT NULL,
        channel_title VARCHAR(255) NOT NULL,
        subscriberCount INT NOT NULL,
        channel_viewCount BIGINT NOT NULL,
        videoCount INT NOT NULL,
        PRIMARY KEY (channel_id));

CREATE TABLE Workout_Video_Trainer(
        video_id CHAR(11) NOT NULL,
        channel_id CHAR(24) NOT NULL,
        publish_date CHAR(20) NOT NULL,
        video_title VARCHAR(255) NOT NULL,
        duration VARCHAR(255) NOT NULL,
        video_viewCount INT NOT NULL,
        likeCount INT NOT NULL,
        trainer VARCHAR(255),
        PRIMARY KEY (video_id),
        FOREIGN KEY(channel_id) REFERENCES Channels(channel_id) ON DELETE
CASCADE);

CREATE TABLE Workout_Video_Type(
        video_id CHAR(11) NOT NULL,
        channel_id CHAR(24) NOT NULL,
        publish_date CHAR(20) NOT NULL,
        video_title VARCHAR(255) NOT NULL,
        duration VARCHAR(255) NOT NULL,
        video_viewCount INT NOT NULL,
        likeCount INT NOT NULL,
        workout_type VARCHAR(255) NOT NULL,
        PRIMARY KEY (video_id),
        FOREIGN KEY (channel_id) REFERENCES Channels(channel_id) ON DELETE
CASCADE);

CREATE TABLE Recipe_Video(
        video_id CHAR(11) NOT NULL,
        channel_id CHAR(24) NOT NULL,
        publish_date CHAR(20) NOT NULL,
        video_title VARCHAR(255) NOT NULL,
        duration VARCHAR(255) NOT NULL,
        video_viewCount INT NOT NULL,
        likeCount INT NOT NULL,
        PRIMARY KEY (video_id),
        FOREIGN KEY(channel_id) REFERENCES Channels(channel_id) ON DELETE
CASCADE);

CREATE TABLE User_Information(
        user_id VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL,
        phone_number VARCHAR(10) NOT NULL,
        password VARCHAR(255) NOT NULL,
        PRIMARY KEY (user_id));

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