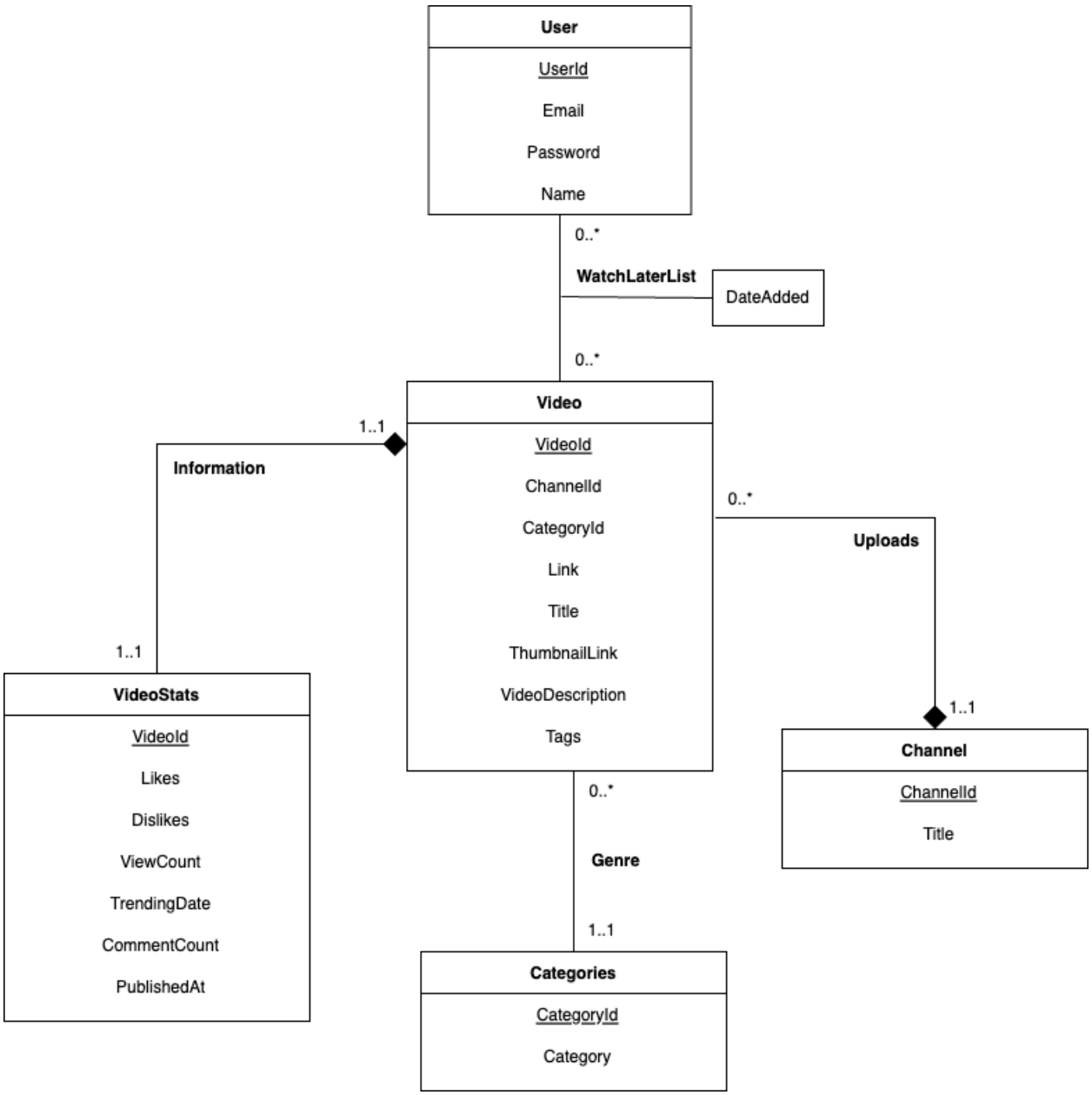
UML Diagram:


User
UserId
Email
Password
Name
0..*
WatchLaterList
DateAdded
0..*
Video
VideoId
ChannelId
CategoryId
Link
Title
ThumbnailLink
VideoDescription
Tags
1..1
Information
1..1
VideoStats
VideoId
Likes
Dislikes
ViewCount
TrendingDate
CommentCount
PublishedAt
0..*
Uploads
1..1
Channel
ChannelId
Title
0..*
Genre
1..1
Categories
CategoryId
Category

Relationship Description:

- Each User can have multiple WatchLater lists (via WatchLaterList), where it adds Videos that it wants to watch later.
- Each Video has associated VideoStats (via Information), which contains information about statistics regarding the Video.
- Each Video also has an associated Channel (via Uploads), based on the Channel that uploaded the Video.
- Within Video, each set of data has a unique Category for the video (via Genre).

Assumptions:

- Each Video should have one set of VideoStats, and each set of VideoStats should correspond to one video.
- Each Video should have one corresponding Channel, and each Channel can have zero to many Videos.
- Each Video should have zero to many User being added into the WatchLater lists.
- Each User can add zero to many Video into the WatchLaterLists.
- Each Video should have one Category, and each Category should have zero to many sets of Video.

Normalization form:

All relational schema already adheres to BCNF form. Because we want to prioritize ensuring valid VideoStats is correctly associated with each Video and additionally, our database may require frequent updates with currently trending videos, as well as frequent insert, delete, and update operations via WatchLater, we wanted to choose a form that would ensure a high level of data integrity and reduce redundancies. Because BCNF is stricter than 3NF, we chose BCNF to achieve our data integrity goals.

User (UserId, Email, Password, Name)
FD1 - UserId -> Email, Password, Name

WatchLaterList (UserId, VideoId, DateAdded)
FD1 - UserId, VideoId -> DateAdded

Video (VideoId, ChannelId, CategoryId, Link, Title, ThumbnailLink, VideoDescription, Tags)
FD1 -> VideoId -> ChannelId, CategoryId, Link, Title, ThumbnailLink, VideoDescription, Tags

Channel (ChannelId, Title)
FD1 - ChannelId -> Title

VideoStats (VideoId, Likes, Dislikes, ViewCount, TrendingDate, CommentCount, PublishedAt)
FD1 - VideoId -> Likes, Dislikes, ViewCount, TrendingDate, CommentCount, PublishedAt

Categories (CategoryId, Category)
FD1 - CategoryId -> Category

Our relational schema satisfied the requirements for BCNF because every functional dependency X->Y has X as a superkey. Each entity has an Id that allows it to be identified and access other attributes. There are no redundancies. Video and VideoStats both use VideoId as their superkey, but this is not a redundancy. Though they could be combined so that VideoStat's attributes are under Video, with VideoId as a superkey, they are separated for convenience in the database, as Video's attributes will be used to display key information about the video, such as its thumbnail and title.

Relational Schema:

```
Table-Name(Column1:Domain [PK], Column2:Domain [FK], Column3:Domain [FK]...)
```

1. User

```
User (
UserId:VARCHAR(20) [PK]
Email:VARCHAR(255)
Password:VARCHAR(255)
Name:VARCHAR(255)
)
```

2. WatchLaterList

```
WatchLaterList (
UserId:VARCHAR(20) [PK, FK to User.UserId]
VideoId:VARCHAR(20)[PK, FK to Video.VideoId]
DateAdded:DATE
)
```

3. Video

```
Video (
VideoId:VARCHAR(20) [PK]
ChannelId:VARCHAR(30) [FK to Channel.ChannelId]
CategoryId:INT [FK to Categories.CategoryId]
Link:VARCHAR(255)
Title:VARCHAR(255)
ThumbnailLink:VARCHAR(255)
VideoDescription:TEXT
Tags:TEXT
)
```

4. Channel

```
Channel (
ChannelId:VARCHAR(30) [PK]
Title:VARCHAR(255)
)
```

5. VideoStats

```
VideoStats (
VideoId:VARCHAR(20) [PK, FK to Video.VideoId]
Likes:INT
Dislikes:INT
ViewCount:INT
TrendingDate:DATE
CommentCount:INT
PublishedAt:DATE
)
```

6. Categories

```
Categories (
CategoryId:INT [PK]
Category:VARCHAR(30)
)
```