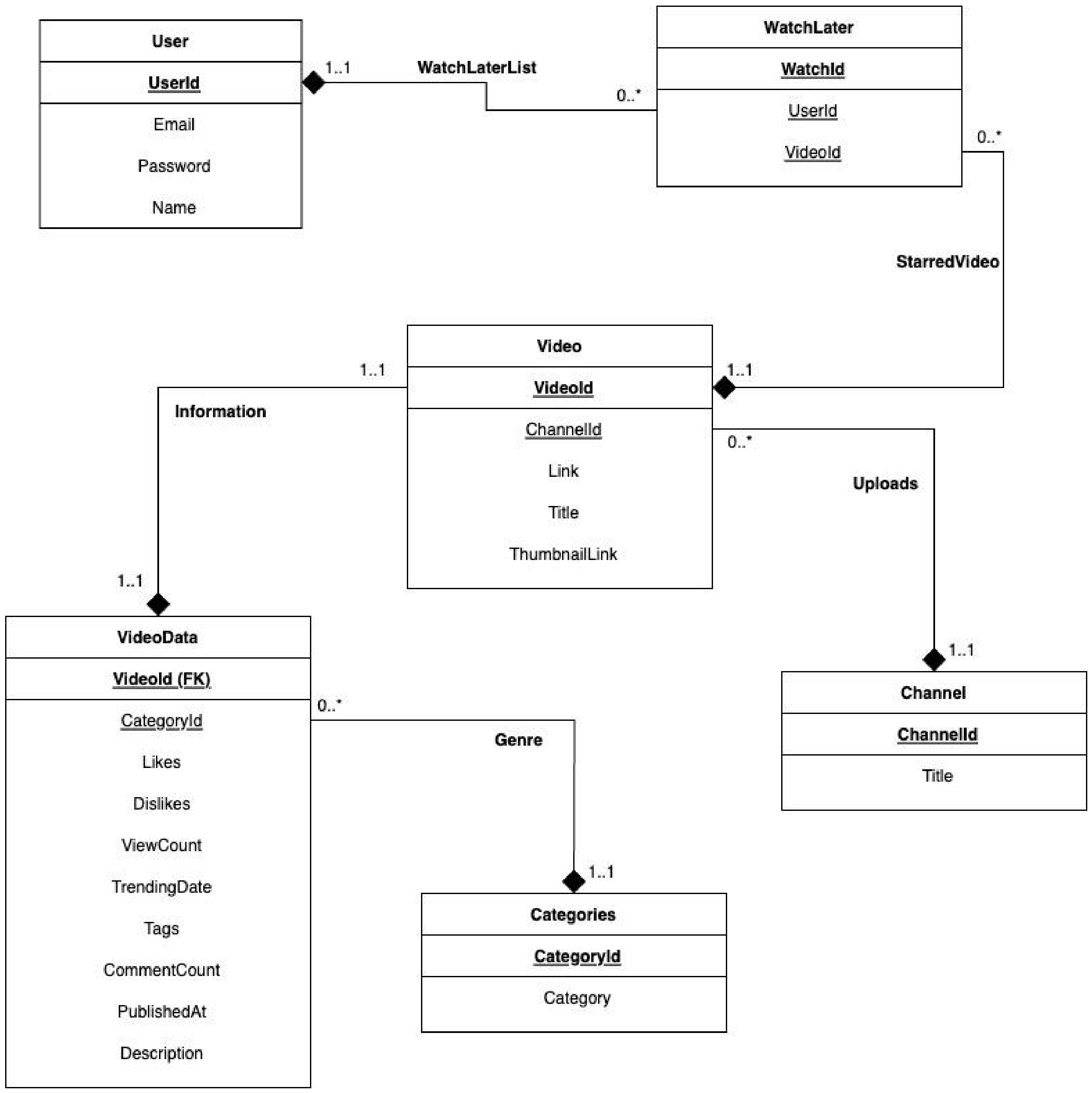
UML Diagram:


User
UserId
Email
Password
Name
1..1
WatchLaterList
0..*
WatchLater
WatchId
UserId
VideoId
0..*
StarredVideo
Video
VideoId
ChannelId
Link
Title
ThumbnailLink
1..1
1..1
Information
0..*
Uploads
1..1
VideoData
VideoId (FK)
CategoryId
Likes
Dislikes
ViewCount
TrendingDate
Tags
CommentCount
PublishedAt
Description
0..*
Genre
1..1
Categories
CategoryId
Category
1..1
Channel
ChannelId
Title

Assumptions:

- Each Video should have one set of VideoData, and each set of VideoData 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 WatchLater lists, and each WatchLater should have one to many Videos.
- Each VideoId in WatchLater corresponds to one VideoId in Videos.
- Each WatchLater should have one corresponding User, and each User should have zero to many WatchLater lists
- Each VideoData should have one Category, and each Category should have zero to many sets of VideoData

Normalization form:

All relational schema already adheres to BCNF form. Because we want to prioritize ensuring valid VideoData 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

WatchLater (WatchId, UserId, VideoId)
FD1 - WatchId -> UserId, VideoId

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

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

VideoData (VideoId, CategoryId, Likes, Dislikes, ViewCount, TrendingDate, Tags, CommentCount, PublishedAt, Description)
FD1 - VideoId -> CategoryId, Likes, Dislikes, ViewCount, TrendingDate, Tags, CommentCount, PublishedAt, Description

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 VideoData both use VideoId as their superkey, but this is not a redundancy. Though they could be combined so that VideoData'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. WatchLater

```
        WatchLater (
        WatchId:INT [PK]
        UserId:VARCHAR(20) [FK to User.UserId]
        VideoId:VARCHAR(20)[FK to Video.VideoId]
        )
```

3. Video

```
        Video (
        VideoId:VARCHAR(20) [PK]
        ChannelId:VARCHAR(30) [FK to Channel.ChannelId]
        Link:VARCHAR(255)
        Title:VARCHAR(255)
        ThumbnailLink:VARCHAR(255)
        )
```

4. Channel

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

5. VideoData

```
        VideoData (
        VideoId:VARCHAR(20) [PK, FK to Video.VideoId]
        CategoryId:INT [FK to Categories.CategoryId]
        Likes:INT
        Dislikes:INT
        ViewCount:INT
        TrendingDate:DATE
        Tags:TEXT
        CommentCount:INT
        PublishedAt:DATE
        Description:TEXT
        )
```

6. Categories

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