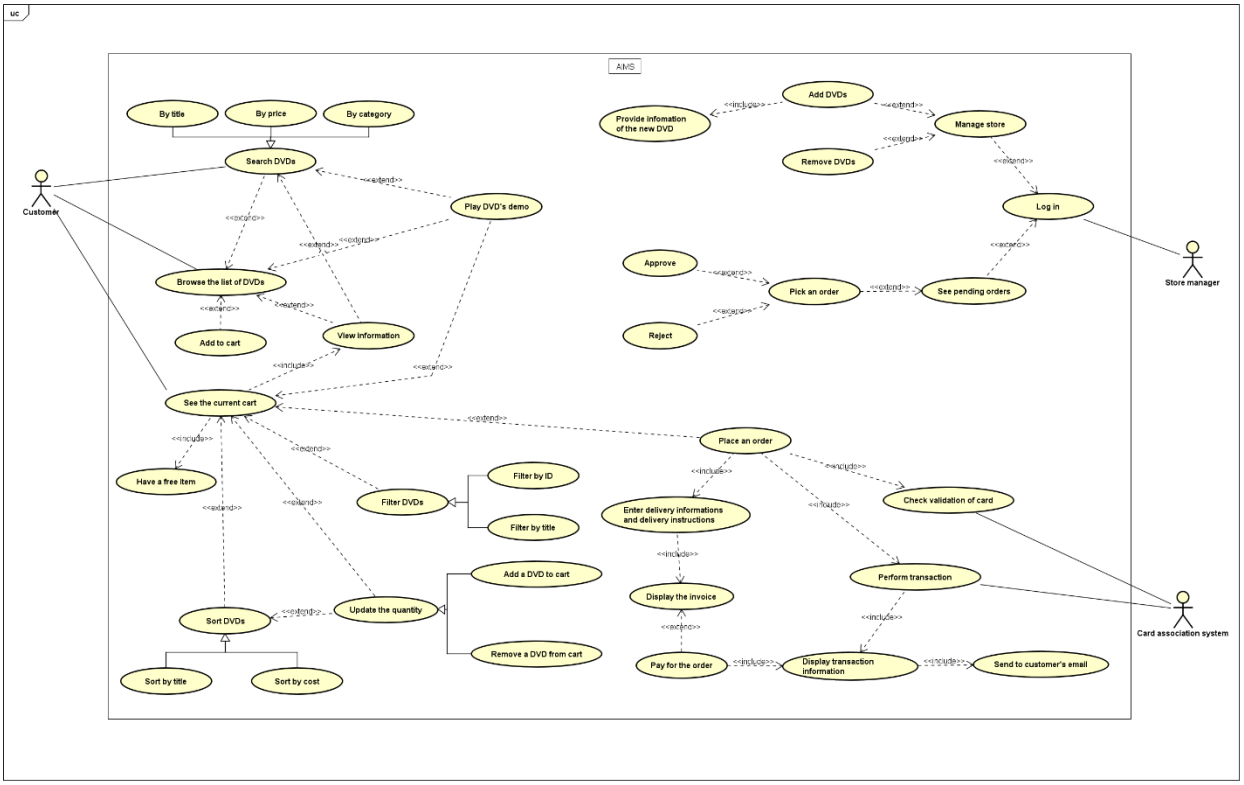
Report Lab 04

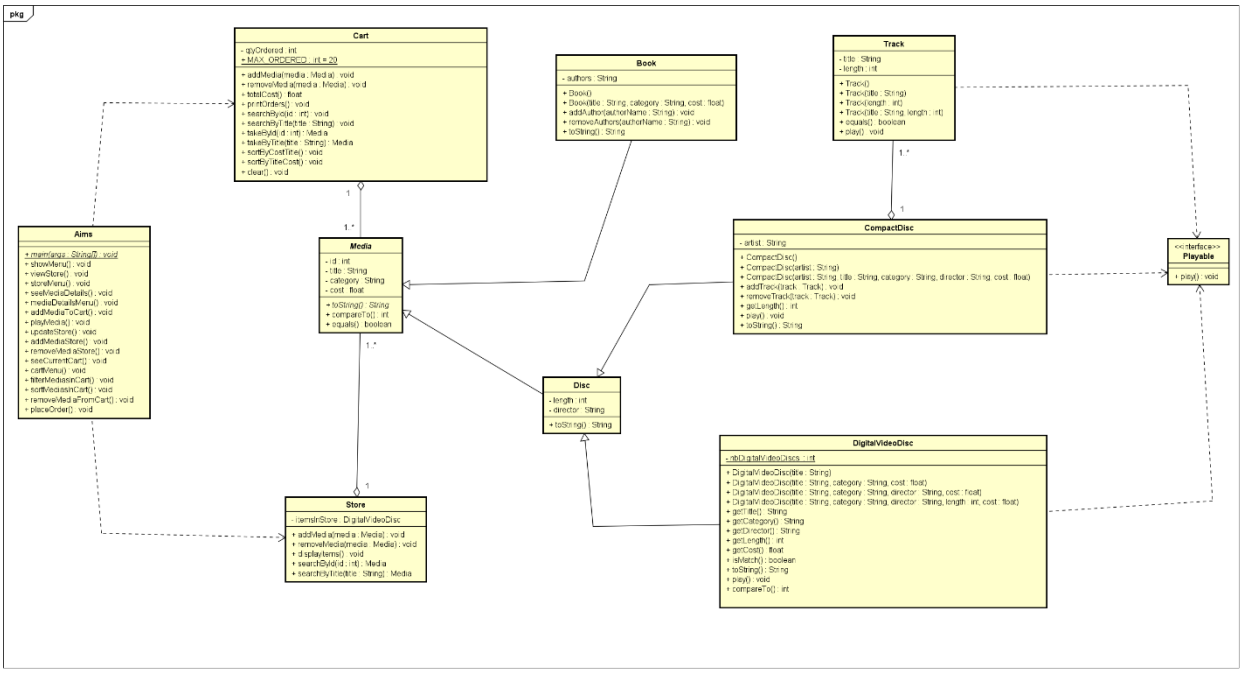
Name: Bùi Công Minh

Student ID: 20235601

1. UseCase Diagram



2. Class Diagram



3. Answering questions

- **Which classes are aggregates of other classes? Checking all constructors of whole classes if they initialize for their parts?**
    - Class Store aggregates class Media.
    - Class Cart aggregates class Media.
    - Class CompactDisc aggregates class Track.
- **Write constructors for parent and child classes. Remove redundant setter methods if any**

```
public Media(String title) {
    super();
    this.title = title;
}

public Media(String title, String category, float cost) {
    super();
    this.title = title;
    this.category = category;
    this.cost = cost;
}
```

```java
public Disc(String title, String category, String director, int length, float cost) {
    super(title, category, cost);
    this.director = director;
    this.length = length;
}

public Disc(String title, String category, float cost) {
    super(title, category, cost);
}

public Disc(String title) {
    super(title);
}
```

```java
public DigitalVideoDisc(String title) {
    super(title);
    nbDigitalVideoDiscs++;
    setId(nbDigitalVideoDiscs);
}

public DigitalVideoDisc(String title, String category, float cost) {
    super(title, category, cost);
    nbDigitalVideoDiscs++;
    setId(nbDigitalVideoDiscs);
}

public DigitalVideoDisc(String title, String category, String director, float cost) {
    super(title, category, cost);
    this.director = director;
    nbDigitalVideoDiscs++;
    setId(nbDigitalVideoDiscs);
}

public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
    super(title, category, director, length, cost);
    nbDigitalVideoDiscs++;
    setId(nbDigitalVideoDiscs);
}
```

```java
public CompactDisc(String title, String category, String artist, float cost){
    super(title, category, cost);
    this.artist = artist;
}
```

```java
public Book(String title, String category, float cost) {
    super(title, category, cost);
}
```

```java
public Track(String title, int length) {
    super();
    this.title = title;
    this.length = length;
}
```

- **Question:** Alternatively, to compare items in the cart, instead of using Comparator, we can use the Comparable interface and override the compareTo()method. You can refer to the Java docs to see the information of this interface.
  Suppose we are taking this Comparable interface approach.
- **What class should implement the Comparable interface?**

The Media class should implement the Comparable interface

- **In those classes, how should you implement the compareTo()method to reflect the ordering that we want?**

```java
public int compareTo(Media m) {
    int titleComparison = this.title.compareTo(m.title);
    if (titleComparison != 0) {
        return titleComparison;
    }
    return Double.compare(m.cost, this.cost);
}
```

- **Can we have two ordering rules of the item (by title then cost and by cost then title) if we use this Comparable interface approach?**

We cannot because the Comparable interface allows only one natural ordering for a class.

- **Suppose the DVDs has a different ordering rule from the other media types, that is by title, then decreasing length, then cost. How would you modify your code to allow this?**

We can write method compareTo() in class DigitalVideoDisc to make it overridden