Task 1:

- The main idea is to prevent the user from accessing the video file directly using direct URLs.
- To achieve this, we can do the following (Assuming that only subscribed users are able to access the page):

- We can use a PHP file to read the video instead of using the real path.
  For example, instead of using something like

```
<embed src="http://www.yourwebsite.com/video/cute.mpg"

autostart="false" />
```

we can use:

```
<embed
src="http://www.yourwebsite.com/phpvideostreaming.php?ID=34fd
gf56ghghg5656534"

autostart="false" />
```

- In the phpvideostreaming.php file, the ID can be converted to the video file name either by fetching it from database or by using a naming convention for the video file names based on the ID, for example, the file name can be the result of hash_hmac('md5', ID, 'some_secret_text').

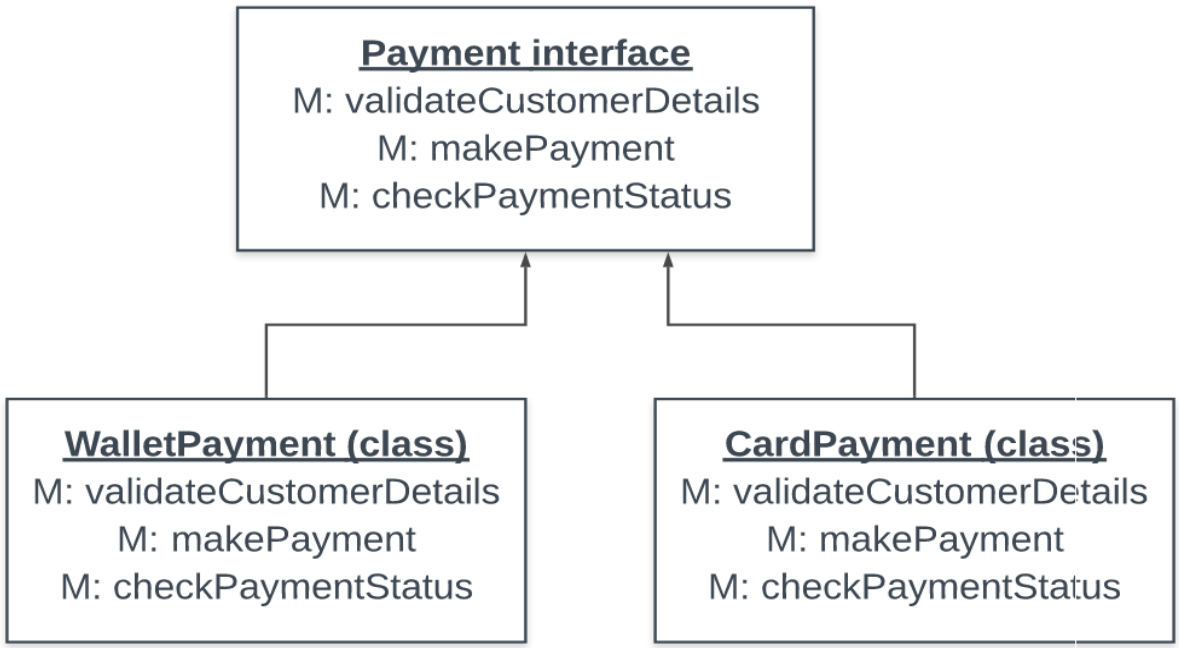
- The headers should be set to "Content-Type: video/*" and then readfile() can be used to output the stream.

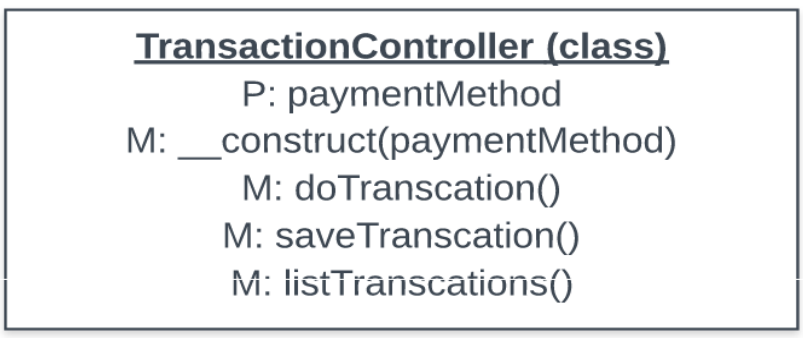
- we can use CSRF tokens and check it at the beginning of the phpvideostreaming.php file to prevent direct access to it.

- All video files should not be available for public access (eg: chmod 600).

Task 2:

- The best design pattern is **Strategy pattern** because we have the same algorithm to be done for each payment method but in different strategies.
- The algorithm shall be as follows:
  1. validate the customer details.
  2. Make the payment.
  3. Check the status of the payment.





- If we need to add another payment method for example MobilePayment, all we need to do is to create a MobilePayment strategy, without touching the already implemented strategies.

Task 3:

- First, we need to create four extra tables in the database

  - subscription: it will contain the three types of subscriptions.
  - customer_subscription: a join table that will contain the customer_id, subscription_id and expire_at.
  - customer_subscription_course: a join table that will contain the customer_subscription_id and course_id.
  - customer_subscription_topic: a join table that will contain the customer_subscription_id and topic_id.

- A method should be implemented in the Course.php model to take the customer_id as a parameter and check if the course is available to that customer or no. The method can be named isAvailableToCustomer($customer_id) and it will return true or false.

- Before showing any course to a customer, the method isAvailableToCustomer($customer_id) should be checked first.

- The method isAvailableToCustomer($customer_id) will do the following:

  - Get all subscriptions for that customer from customer_subscription table.
  - For each subscription there is three cases:

- course_subscription: A loop should be done on the courses in the customer_subscription_course, if the course is found then the method returns true.
- monthly_topic_subscription: the expire_at is checked, if not expired then a loop should be done on the topics in the customer_subscription_topic, if the course belongs to one of the topics then the method returns true.
- monthly_all_inclusive: only the expire_at is checked, if not expired then the course is available and the method returns true.

- The method should return false if it passed the previous loops.