A state design pattern is used for a video player application with ‘Play’, ‘Rewind’ and ‘Forward’ functionalities.

We describe 4 states here:

- Play: The video is playing
- Forward: Video is forwarded
- Rewind: Video is backwarded

First of all, there is a Context interface which will connect with client and refer to the state. Reference to stop state is not given. Only play, forward and rewind reference is possible.

Pseudocode for Context interface is given below:

```
interface VideoState //interface for different states
{
   public void alert(AlertStateContext ctx);
}

class VideoStateContext
{
        private VideoState currentState //VideoState is described later
        private time //to keep track of time for forward and rewind

        public VideoStateContext()
        {
                currentState = new Play() //constructor; video plays when an object is created
        }

        public void setState(VideoState state)
        {
               currentState = state; //updating state
        }

        public void setTime(this, add)
        {
               time = time + add; //updating time
        }
```

```
        public void alert()
                {
                currentState.alert(this); //to do the codes of current state
        }

        }
```

Then the implementation of different states is done.

```
class Play implements VideoState
{
   @Override
   public void alert(VideoStateContext ctx)
   {
      //play the video
   }

}


class Forward implements VideoState
{
   @Override
   public void setTime(this, addedTime)
   {
      //time of video is added with minimum of summation value and end value
   }

}

class Rewind implements VideoState
{
   @Override
   public void setTime(this, addedTime)
   {
      //time of video is subtracted with maximum of subtracted value and start value
   }

}
```

Thus, the functionalities can be implemented.