5.1. Implementation

This project is implemented in PHP Language. This sub-chapter, will explain how to use the program and how the program works. Before start to write the code, the YouTube API key must gotten. The YouTube API key is a “tools” that is used to access YouTube datas. The YouTube API key can gotten by following the steps in this URL: https://developers.google.com/youtube/v3/getting-started

After get the YouTube API key, the process to write the code can be start. In this project, the program has five menus there are home, history, chart, recommendation, and settings. Which is each menu have a different function and will be directed to different page too.
5.1.1. Home

Home menu is illustrated by the logo of program. Home menu is the first page which will appear when user open the site. It displays a list of popular videos. The code of Home page is shows in figure below.
In the code above, shows that the page call "get Popular()" and "popularList()" function which stayed in "kelasYoutube" class. The "getPopular()" function is used to get the data of popular video in YouTube. This function has two parameter, there are "pageToken" and "maxResult". The "pageToken" parameter is a string which is used to set paging. The "maxResult" parameter is an integer which is used to set how many the results that we want to get. But, the maximum results that allowed by
YouTube only up to 50 results for each page. While the paging up to five page. Then, this function will request the data by using Youtube API feature. The feature figured out as an URL. Then, it will give an array of json of popular videos as a return value.

/*************************************
** Function name : getPopular()
** Parameter      : stringPageToken , intMaxResult
** Return value   : arrayJSONpopularVideo
**************************************/

function getPopular($pageToken, $maxResult=NULL)
{
    /* - Kalau tdk null maxResult,ny, batas = permintaan user - */
    if(!is_null($maxResult))
    {
        if($maxResult > $this->maxResult)
        {
            $batas = $this->maxResult;
        }
        else
        {
            $batas = $maxResult;
        }
    }
    else
    {
        $batas = $this->maxResult;
    }

    /* - Akses Youtube - */
    $part = "snippet,contentDetails,statistics";

    if(is_null($pageToken))
    {
        $url = 'https://www.googleapis.com/youtube/v3/videos?key='.$this->apikey.'&part='.$part.'&chart=mostPopular&maxResults='.$batas;
    }
    else
    {
        $url = 'https://www.googleapis.com/youtube/v3/videos?key='.$this->apikey.'&part='.$part.'&chart=mostPopular&maxResults='.$batas.'&pageToken='.$pageToken;
    }

    $json = file_get_contents($url);
    $hasil = json_decode($json);
    return $hasil;
}

Figure 11. getPopular() function
Then, the result of “getPopular()” function will be used in “popularList()” function. The “popularList()” function is used to process the data to translate into html. In “popularList()”, the function named “listVideoPopuler()” will be called. It has a same function with “popularList()”, the different is “listVideoPopuler()” is used to translate the data into html of each videos, while “popularList()” is used to translate the data into html for whole videos.
function popularList($dataPopular)
{
    $html = '<ul class="media">
    foreach ($dataPopular->items as $yt)
    {
        $html .= $this->listVideoPopuler($yt);
    }
    $html .= '</ul>);
    $this->halaman = '<ul class="pager">
    if(isset($dataPopular->prevPageToken))
    {
        $this->halaman .= '<li class="previous"><a href="home.php?pageToken='.
            $dataPopular->prevPageToken.'" class="previous">Previous</a></li>
    }
    if(isset($dataPopular->nextPageToken))
    {
        $this->halaman .= '<li class="next"><a href="home.php?pageToken='.
            $dataPopular->nextPageToken.'" class="next">Next</a></li>
    }
    $html .= '</ul>;
    return $html;
} //end of func popularList($dataPopular)

function listVideoPopuler($yt)
{
    $viewCount = number_format($yt->statistics->viewCount, 0, ",", ". ");
    $channelTitle = $this->potongKata(ucwords(strtolower($yt->snippet->channelTitle)), 10);
    // Format Video
    $html = '<li class="video" id="video'.$yt->id.'">
        <div class="mediathumb">
            <a href="lihat.php?video='.$yt->id.'" title="'.($this->modifWaktu($yt->snippet->publishedAt)))."">
                <img src="'.$yt->snippet->thumbnails->medium->url.'" alt="#">
            </a>
        </div>
        <h4><a href="lihat.php?video='.$yt->id.'" title="#">'.($this->potongKata($yt->snippet->title,30))."</a></h4>
        <a class="mediaauthor" href="#">oleh $channelTitle</a>
        <span class="time">$this->modifWaktu($yt->snippet->publishedAt).'</span>
        $html = '</li>;
    return $html;
} //end of func listVideoPopuler($yt)

Figure 12. popularList() and listVideoPopuler() function
5.1.2. History

History menu will display a list of video data which has been watched by the user which is has been recorded in "seen.txt". In this page, there is a “Delete all” button. This button will delete all of video datas which has been watched by the user.
5.1.3. Chart

![Chart page interface](image)

Chart menu will display a chart of all seen videos based on categories. When “all seen videos” radio button choosed, chart will give a result from whole seen videos. When “period” radio button choosed, user must input the start date and end date. Then, chart will give a result from seen videos between start date and end date.
5.1.4. Recommendation

Recommendation menu will display the list of recommendation video which is obtained from Naive Bayes processing. But, when the program does not have a dataset of video that has been watched or when the result of Naive Bayes processing is empty, the program will display a list of popular video. The code of recommendation page is shown in figure below.

Figure 15. Recommendation page interface
Figure 16. A part of “Recommendation” page code
From the code above, the new function that never explained before is “getRecommendation()” function which is stayed in “kelasYoutube” class. This function is used to get the data of videos which recommended for user. This function has a parameter, that is “kelasBayes”. The “kelasBayes” parameter is an object of “kelasBayes” class which is used to access the “kelasBayes” class. This function give an array that has string of video’s data as a return value if the result of Naive Bayes calculation is not empty. But, if the result of Naive Bayes calculation is empty, the “getRecommendation()” function will give “0” as a return value.

```
function getRecommendation($kelasBayes)
{
    $this->getUTrTs((3/4),(1/5));
    $hsl = $kelasBayes->hitBayes(kelasFile::getPTr(),kelasFile::getPTs());
    if($hsl!=0)
    {
        kelasFile::tulisDariAwal(kelasFile::getPRe(), kelasFile::getHeaderText());
        foreach($hsl as $k => $v)
        {
            $tulis = "\n $.v;
        } /* Menulis ke recommendation.txt */
        kelasFile::tambahIsiFile(kelasFile::getPRe(),$tulis);
    }
    return $hsl;
}
else
{
    return 0;
}

//end of getRecommendation()
```

Figure 17. getRecommendation() function

The figure above is shows about the code of “getRecommendation()” function. From the code above, we know that the “getRecommendation()” function call “getUTrTs()” function in its class. This function is used to get Unseen, Training, and Testing data. The first parameter in “getUTrTs()” function are used to set the ratio between video
that has been watched and video that never been watched. While the second parameter is used to set the ratio between the number of trainning dataset and the number of testing dataset.

After that, the “getRecommendation()” function call “hitBayes()” function which is stayed in “kelasBayes” class. This function is used to implement the Naive Bayes calculation. The “hitBayes()” function has two parameter, there are “tr” and “ts”. The “ts” parameter is a string of trainning.txt path, while “ts” parameter is a string of testing.txt path.

```php
//******************************
** Function name : hitBayes()
** Parameter   : stringPathTrainning, stringPathTesting
** Return value : arrayHasilBayes[intIndex] = stringRowFile atau 0
******************************
function hitBayes($tr, $ts)
{
    $hasil=$this->gruping($tr);
    $phasil=$this->hitProb($hasil);
    $prior = $this->hitPrior($tr);
    $sum=0;
    foreach($prior as $k => $p)
    {
        $sum = $sum+$p;
    }
    $txt = file_get_contents($ts);
    $rows = explode("\n ", $txt);
    array_shift($rows);
    $headers = explode(" ||| ", $rows[0]);
    $a=0;
    $jmlHasilBayes=0;
    $currFeature = $this->getCurrFeature();
    $currBobot = $this->getCurrBobot();
    foreach($rows as $key => $val)
    {
        $isi = explode(" ||| ", $val);
        $ly=$ly+1;$lys="";
        $ln=$ln+1; $lns="";
        [insert code here]
```
foreach($isi as $k => $v)
{
    if(in_array($k, $currFeature))
    {
        $like = $this->getLikehood($headers[$k], $v, $phasisl);
        $grupTs[$key][$headers[$k]] = $v;
        if(strcmp($like, "xxx") == 0)
        {
            $ly = "$ly;
            $lys = "$lys." . $like."(\$ly.) x ";
            $ln = "$ln;
            $lns = "$lns." . $like."(\$ln.) x ";
        }
        else
        {
            if($currBobot[$k] > 0)
            {
                $hy = 1;
                $hn = 1;
                $lys = "$lys."; 
                $lns = "$lns."; 
                for($n = 0; $n < $currBobot[$k]; $n++)
                {
                    $hy = $hy * $phasisl[$headers[$k]][$like][]['pSeen'];
                    $lys = "$lys." . $phasisl[$headers[$k]][$like][]['pSeen']." x ";
                    $hn = $hn * $phasisl[$headers[$k]][$like][]['pUnseen'];
                    $lns = "$lns." . $phasisl[$headers[$k]][$like][]['pUnseen']."x";
                }
                $ly = "$ly * $hy;
                $lys = "$lys." x ";
                $ln = "$ln + $hn;
                $lns = "$lns." x ";
            }
        }
    }
}$hsly = $prior['seen'] * $ly;
$hsln = $prior['unseen'] * $ln;
$grupTs[$key]['1hsYes'] = "$lys;
$grupTs[$key]['1hsNo'] = "$lns;
$grupTs[$key]['likehoodYes'] = "$ly;
$grupTs[$key]['likehoodNo'] = "$ln;
$grupTs[$key]['hasilYes'] = "$hsly;
$grupTs[$key]['hasilNo'] = "$hsln;
/** MAP **/
if($hsly > $hsln)
{
    $hasilBayes[$jmlHslBayes] = "isi" => "$val;
    $hasilBayes[$jmlHslBayes] = "$hsly;
    $jmlHslBayes++;
    $hsBayes = 1;
} else
{
    $hsBayes = 0;
}$grupTs[$key]['hasilBayes'] = "$hsBayes;
In “hitBayes()” function, the data from training data will be grouped for each feature. Then, the likelihood probability will be calculated by using “hitProb()” function. The posteriori probability and MAP calculated in this function manually.

If the result of “hitBayes()” is not 0, the “getRecommendation()” function will write the result in “recommendation.txt”. Then the “getRecommendation()” function will give the result of “hitBayes()” as a return value.
5.1.5. Settings

The settings menu will display a form to set the settings of Naive Bayes processing. The setting is about which attributes that will be used and its weight. This menu has a button named “Set to Default”. This button is to set the settings into the default settings (each attribute will be used and its weight is 1).
5.2. Testing

To test the program, user must have a dataset of videos that has been watched. To get the data, user only need to watch some videos.

5.2.1. First Test

In the first test, researcher use 30 seen data with random data of videos.

![Figure 20. Data of seen.txt in first testing](image-url)
Then, the program gives 42 data of videos as a recommendation. The video recommendation from the program have two videos which are the same with YouTube’s video recommendation.

Figure 21. Recommendation result of first testing
5.2.2. Second Test

In the first test, researcher use 10 seen data that have same Channel ID.

Then, the program give 35 data of videos as a recommendation. The video recommendation from the program have ten video which is same with YouTube's video recommendation.
Figure 23. Recommendation result of second testing