CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

Naive Bayes Algorithm is a function based calculation that divided into 5 parts. Each part has its job, that has to be done, so the other function can work.

```
1. def
   NaiveBayes(data,listPenyakit,listGejala,joinPenyakitGejala):
2.   input_gejala = data
3.   listNaiveBayesScore = []
4.
5.   # Loop Penyakit
6.   for penyakit in listPenyakit:
7.         gejalaPenyakit  =  joinPenyakitGejala.filter(penyakit  =
   penyakit.id)
8.
9.     # langkah 1
10.      hasilNilaiNC                                                =
   nilaiNC(input_gejala,penyakit,gejalaPenyakit,listPenyakit,listG
   ejala)
11.
12.          # langkah 2
13.          hasilNilaiPAiVj = nilaiPAiVj(hasilNilaiNC)
14.
15.          # langkah 3
16.          hasilNilaiPVj = nilaiPVj(hasilNilaiPAiVj)
17.
18.          # Filter ke 3 penyakit tertinggi
19.          scorePenyakit = hasilNilaiPVj
20.
21.          NaiveBayesScore = {
22.            'penyakit': penyakit.penyakit,
23.            'nilai': scorePenyakit['score'],
24.            'total_gejala': gejalaPenyakit.count(),
25.            'total_ada': hasilNilaiNC['totalAda'],
26.                                    'perbandingan_gejala_dan_ada':
   hasilNilaiNC['totalAda']/gejalaPenyakit.count()
27.          }
28.
29.          listNaiveBayesScore.append(NaiveBayesScore)
30.
31.        result = filterNaiveBayes(listNaiveBayesScore)
32.        return result
```

First-line in the code above, we can see that this function receives the data with the database too. Line number 2 until 7 is preparing the data that is used.

Naive Bayes calculation is started from line 9 until 19. The code above is not about the calculation, but this function is the primary code that calling all the calculations.

```
33.     def
  nilaiNC(input_gejala,penyakit,gejalaPenyakit,listPenyakit,listG
  ejala):
34.         totalPenyakit = listPenyakit.count()
35.         totalGejala = listGejala.count()
36.
37.         listDataGejala = []
38.         for gejala in gejalaPenyakit:
39.             listDataGejala.append(gejala.gejala)
40.
41.             nilaiN = 1
42.             nilaiP = nilaiN / totalPenyakit
43.             nilaiM = totalGejala
44.             # nilaiNC = {}
45.             nilaiNC = []
46.             totalAda = 0
47.             i = 0
48.
49.             for ig in input_gejala:
50.                 if ig in listDataGejala:
51.                 nilaiNC.append(1)
52.                 totalAda += 1
53.             else:
54.                 nilaiNC.append(0)
55.
56.             i += 1
57.
58.             result = {
59.                 'nilaiN': nilaiN,
60.                 'nilaiP': nilaiP,
61.                 'nilaiM': nilaiM,
62.                 'nilaiNC': nilaiNC,
63.                 'totalAda': totalAda,
64.                 'trial': nilaiNC,
65.             }
66.             return result
```

Code above is the Nc part where it is determining wether disease have the input symptoms or not. Line 34 until 35 is declaring and preparing the data. This function is receiving list of available disease and symptoms, so in the line 38, that data is started to loop. Line 41 until 47 is declaring the needed value for calculation. The Nc is started at line 49. symptoms that user has inputed is being loop, then check if it is exists in the disease symptoms then it gets score 1 and if

not, it get score 0. Score and other declared value then push in the result and return to the primary function.

```
67.     def nilaiPAiVj(data):
68.           nilaiN = data['nilaiN']
69.           nilaiP = data['nilaiP']
70.           nilaiM = data['nilaiM']
71.           nilaiNC = data['nilaiNC']
72.           resultNilaiPAiVj = []
73.           hasil = 0
74.           # trial = []
75.
76.           i = 0
77.           for nc in nilaiNC:
78.                 hasil = 0
79.                 hasil = ((nc + nilaiM) * nilaiP) / (nilaiN +
  nilaiM)
80.                 resultNilaiPAiVj.append(hasil)
81.                 # trial.append(hasil)
82.
83.           result = {
84.                 'nilaiP': nilaiP,
85.                 'nilaiPAiVj': resultNilaiPAiVj,
86.                 # 'trial': trial
87.           }
88.
89.           return result
```

The next function is the second step of Naive Bayes. In this step, the Nc score is calculated with M, P, and N. Line 68 until 76 is declaring needed value and string. Then, Nc Score is looped. Line 78 is for making sure that var hasil always 0 when the loop is back at the top of the loop. Line 79 is where the calculation begins. All of Nc is calculated, then every Nc getting a new score between 1 and 0. Line 83 until 87 is to store the Nc result in the string so then it can be returned to the main function.

```
90.     def nilaiPVj(data):
91.           nilaiP = data['nilaiP']
92.           nilaiPAiVj = data['nilaiPAiVj']
93.           nSementara = 0
94.           trial = []
95.
96.           for n in nilaiPAiVj:
97.                 trial.append(n)
98.                 if nSementara == 0:
99.                       nSementara = n
100.                else:
101.                      nSementara *= n
102.
```

```
103.            result = {
104.                  'score': nilaiP * nSementara,
105.                  'trial': trial,
106.            }
107.
108.            return result
```

The above code is the last step. Inline 91 until 94, is where the variable and value got declared. Next line or line 96 until 101 is where the calculation begins. NilaiPAiVj is looped, and the score is stored in the trial list. Then inline 98, nSementara is checked whether the value is 0 or not, if 0, n is stored in the nSementara, and if not 0, n times with the stored value in the nSementara. It is looped until the Input User is finished. The final result is become 1 value not as much as before and it is stored in the result. Result then returned to the primary function.

5.2 Testing

This system has 323 rows of diagnosis and 1592 rows.

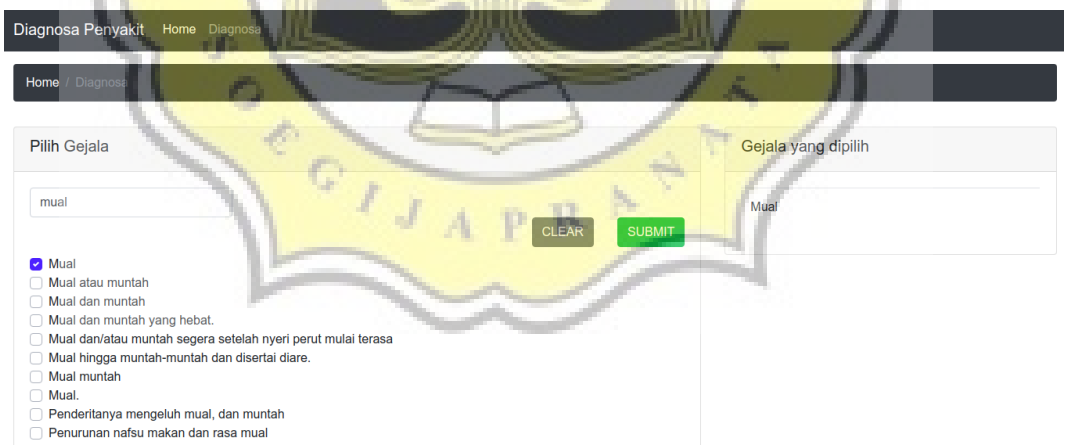

Illustration 3: symptoms Page

The symptoms page is where the user input begins. On this page, the user has to pick several symptoms. There is 2 way to input symptoms, user can look one by one on the list, or enter a keyword in the search input then the system is filtering the data only that have the keywords that user have inputted. Users can see their input in the right part of the website. And if the user is not satisfied or

wants to start from 0, the user can click the clear button, then all the checkbox is unchecked.

Illustration 4: Result diagnosis

The image above is the result page. This page is showing the user what is the recommended disease based on the user input. It is sorted by Gejala yang ada divided by Total Gejala, the result then used to sort the data. Naive Bayes score has color, green means it is the most recommended by the system. Yellow means it can be the second option. The last is grey that not displayed in the image above, grey is the last option.

Target Penyakit	Gejala	Gejala yang dipilih	Hasil			Ket.
			Posisi	Penyakit	Naive Bayes Score	
Nyeri Punggung Bawah	5	5	1	Nyeri Punggung Bawah	1.6577987650738273e-18	
			2	Listeriosis	1.036124228171142e-19	
			3	Sakit Kepala Tegang	1.036124228171142e-19	
Masuk Angin	12	5	1	Masuk Angin	1.6577987650738273e-18	
			2	Aerophobia	2.072248456342284e-19	
			3	Limfadenopati	2.072248456342284e-19	
SARS	7	5	1	Malaria	4.144496912684568e-19	Malaria karena perbandingan total penyakit dan gejala yang ada lebih banyak dari pada SARS
			2	SARS	8.288993825369136e-19	
			3	Listeriosis	2.072248456342284e-19	
Salesma	8	3	1	Salesma	1.1119867574596853e-12	
			2	Abses Payudara	2.7799668936492134e-13	
			3	Listeriosis	2.7799668936492134e-13	
Listeriosis	4	2	1	Listeriosis	9.107171543594823e-10	
			2	Abses Payudara	4.5535857717974114e-10	
			3	Chikungunya	9.107171543594823e-10	
Malaria	5	4	1	Malaria	1.3577371885954645e-15	
			2	Radang Usus	1.6971714857443306e-16	
			3	Chikungunya	3.394342971488661e-16	
Neuroma Akustik	10	9	1	Neuroma Akustik	3.6846496915556835e-30	
			2	Malaria	1.4393162857639389e-32	
			3	Aerophobia	1.4393162857639389e-32	
Demam	8	4	1	Malaria	6.788685942977322e-16	Malaria karena perbandingan total penyakit dan gejala yang ada lebih banyak dari pada Demam
			2	Demam	1.3577371885954645e-15	
			3	SARS	3.394342971488661e-16	
Dehidrasi	12	10	1	Dehidrasi	4.4989617723512625e-33	
			2	Gangguan Metabolik	8.78703471162356e-36	
			3	Whiplash	8.78703471162356e-36	
Bakteremia	9	5	1	Bakteremia	1.6577987650738273e-18	
			2	Cholangitis	1.036124228171142e-19	

Illustration 5: Trial data

Trial error is needed in programming. The image above is about the trial error and get result whether data is correct or not and try to fix it if correct. Target

Penyakit is the targeted target that should be selected in penyakit. Gejala is where the total list gejala is stored. Gejala yang dipilih is counted based on total Gejala that is inputted by the user. Posisi is the result position on the page result. Hasil penyakit shows the name of the disease and sorted by the gejala yang ada with total gejala. The next column is the Naive Bayes Score, this column shows us how high is the naive Bayes score. The last column is Keterangan. This is required when the targeted is not showing at the top position.