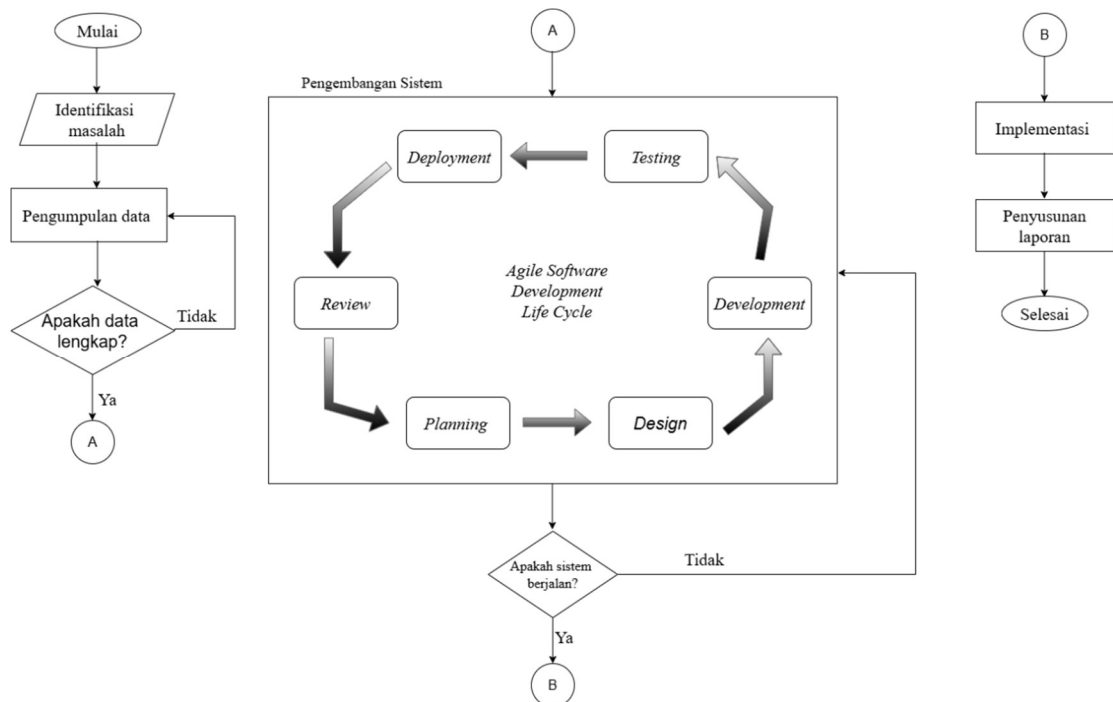


## BAB III METODOLOGI PENELITIAN

Pada bab 3 menjelaskan metode penelitian dalam merancang dan membangun Sistem deteksi penyakit pada ayam broiler menggunakan algoritma *long short term memory* berbasis *website*. Pembahasan mencakup tahapan-tahapan penelitian yang dilakukan.

### 3.1 Alur Penelitian

Alur penelitian adalah sebuah pedoman untuk membuat dan melaksanakan proses penelitian. Alur penelitian dapat dijadikan sebagai pedoman agar penelitian dapat selaras dengan tujuan penelitian. Alur penelitian yang digunakan ialah dengan model pengembangan *agile software development life cycle* (SDLC). Sebagaimana alur penelitian dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur Penelitian

Penelitian diawali dengan identifikasi masalah untuk memahami permasalahan utama yang akan diselesaikan melalui sistem. Selanjutnya dilakukan pengumpulan data melalui studi pustaka, wawancara, dan observasi. Data yang dikumpulkan dievaluasi, dan jika belum lengkap, dilakukan pengumpulan ulang hingga informasi mencukupi.

Tahap berikutnya adalah proses pengembangan sistem menggunakan model *agile* SDLC bersifat iteratif, yaitu perencanaan (*planning*), perancangan (*design*), *development*, pengujian (*testing*), *deployment*, dan *review*. Setiap tahap diulang hingga sistem sesuai dengan kebutuhan *user*. Pengujian dilakukan berkala, dan jika ditemukan kekurangan, siklus pengembangan diulang. Jika sistem telah berjalan optimal, dilakukan implementasi pada lingkungan pengguna (*user*), dilanjutkan dengan penyusunan laporan sebagai dokumentasi. Tahap akhir adalah penyelesaian penelitian yang ditandai dengan selesainya sistem dan laporan skripsi.

### **3.2 Karakteristik Objek Penelitian**

Objek utama dalam penelitian ini adalah sistem deteksi penyakit ayam broiler menggunakan algoritma *long short term memory* berbasis *website* yang dibangun untuk pendeteksian penyakit ayam broiler berdasarkan gejala yang diinputkan pada *website*. Sistem ini memberikan media yang mudah diakses *user* untuk pemeriksaan gejala penyakit ayam broiler tanpa harus menunggu tenaga ahli, dengan hasil diagnosis berbasis model kecerdasan buatan.

#### **3.2.1 Deskripsi Umum Sistem**

Sistem deteksi penyakit ayam broiler merupakan sebuah sistem yang memiliki fungsi untuk memprediksi penyakit berdasarkan gejala yang timbul pada ayam. Sistem ini dibangun menggunakan algoritma *long short term memory* yang dilatih menggunakan data historis untuk memprediksi penyakit secara akurat.

### 3.2.2 Kebutuhan *Software*

Perangkat lunak atau *software* digunakan untuk membuat dan menjalankan program sistem atau *source code*, baik pada sistem *website* maupun algoritma LSTM. Adapun perangkat lunak yang digunakan dalam penelitian ini dapat dilihat pada tabel 3.1.

Tabel 3.1 Kebutuhan *Software*

| No. | Perangkat Lunak                    | Kegunaan                                                               |
|-----|------------------------------------|------------------------------------------------------------------------|
| 1.  | Visual Studio Code                 | Untuk proses pengkodean <i>website</i> dan integrasi dengan model LSTM |
| 2.  | Google Colaboratory<br>(Collab)    | Untuk menjalankan kode python dalam proses pemodelan sistem            |
| 3.  | MySQL (Versi: 10.4.27-<br>MariaDB) | Untuk penyimpanan dan pengolahan data                                  |

### 3.2.3 Kebutuhan *Hardware*

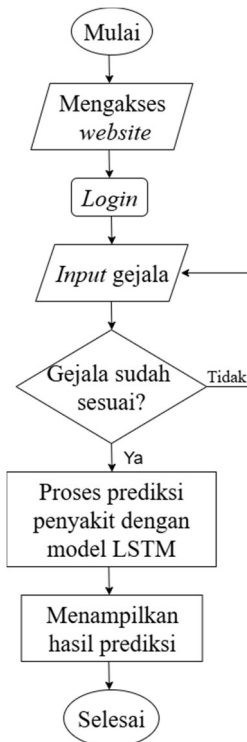
Perangkat keras atau *hardware* digunakan untuk proses pengembangan dan pengoperasian sistem. Adapun perangkat keras yang digunakan dapat dilihat pada tabel 3.2 dibawah ini.

Tabel 3.2 Kebutuhan *Hardware*

| No. | Spesifikasi Perangkat                    |
|-----|------------------------------------------|
| 1.  | <i>Operating System</i> : Windows 10 Pro |
| 2.  | Processor: AMD A4-9120 RADEON R3         |
| 3.  | RAM: 4096 MB                             |
| 4.  | Lenovo                                   |

### 3.3.1 Perancangan Sistem

Perancangan sistem digunakan untuk menggambarkan bagaimana sistem akan bekerja. Dalam penelitian ini, desain sistem akan digambarkan melalui *flowchart* sistem. Adapun *flowchart* sistem yang dimaksud dapat dilihat pada Gambar 3.2.



Gambar 3.2 Flowchart Sistem

*Flowchart* pada Gambar 3.2 menggambarkan alur sistem prediksi penyakit pada ayam broiler menggunakan model LSTM berbasis *website*. Proses dimulai dari *user* mengakses *website* dan melakukan *login*. Setelah berhasil masuk, *user* dapat menginput gejala yang dialami oleh ayam. Kemudian, sistem akan memverifikasi *input* gejala sudah sesuai atau belum. Jika belum, *user* akan diminta untuk mengoreksi gejala yang diinputkan. Jika sudah sesuai, sistem akan

melanjutkan ke proses prediksi menggunakan model LSTM. Hasil prediksi akan ditampilkan sebagai *output* dari sistem.

### 3.3.3.1 Pembuatan Website

*Website* yang dirancang untuk sistem deteksi penyakit ini, dibangun menggunakan HTML, CSS, dan JavaScript untuk *frontend*, serta penggunaan Flask untuk *backend* dan penggunaan MySQL untuk *database*. Dalam *website* yang dibuat terdapat beberapa fitur utama, yaitu:

#### 1. Form Diagnosa Penyakit

Dalam *form* diagnosa penyakit, *user* dapat memilih gejala-gejala yang muncul dengan opsi “ya” atau “tidak”. Inputan yang dimasukan kemudian dikonversi menjadi data numerik dan dikirim ke model LSTM untuk diproses.

#### 2. Menu Detail Penyakit

Dalam menu detail penyakit, *user* dapat melihat informasi mengenai penyakit yang diprediksi, termasuk deskripsi, gejala yang terjadi, dan penyebab timbulnya penyakit.

#### 3. Menu Pencegahan Penyakit

Dalam Menu pencegahan penyakit ini berisikan informasi mengenai langkah pencegahan dan pengobatan penyakit yang dapat dilakukan.

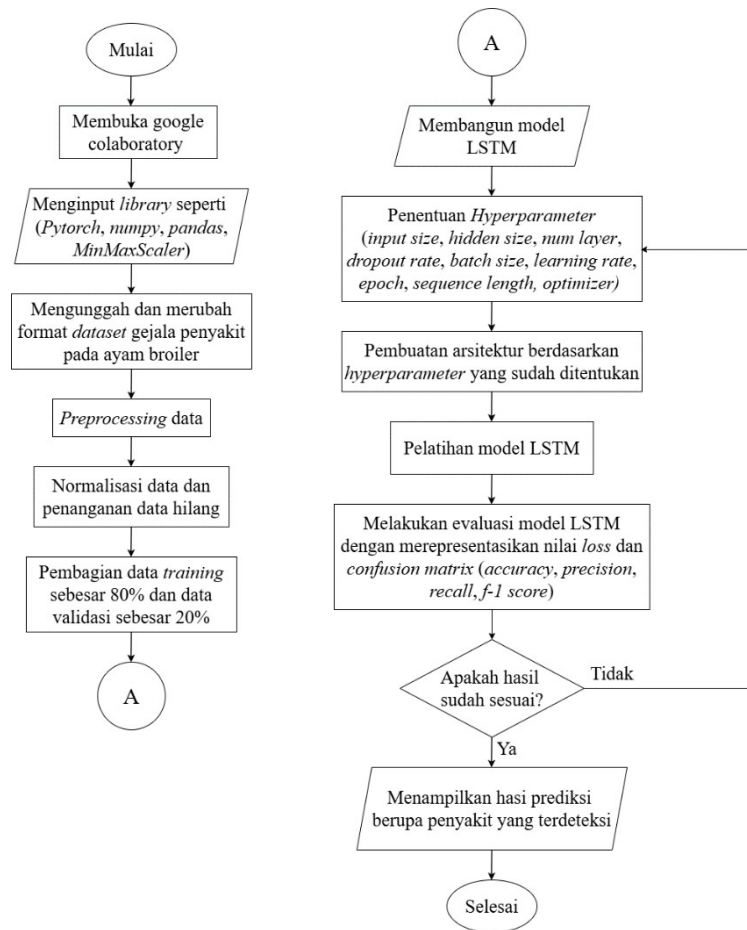
#### 4. Menu Riwayat

Dalam menu riwayat ini berisikan data penyakit yang pernah dideteksi serta terdapat grafik yang merepresentasikan seberapa banyak penyakit tertentu timbul.

### 3.3.3.2 Perancangan Model LSTM

Pada bagian ini akan dijelaskan mengenai proses pemodelan algoritma LSTM. Pemodelan LSTM digunakan dalam sistem ini untuk memprediksi penyakit pada ayam broiler berdasarkan *input* gejala. Dengan menggunakan pendekatan ini, model dapat mempelajari pola-pola kompleks dalam data gejala dan memprediksi lebih akurat.

Perancangan model LSTM ini akan direpresentasikan ke dalam sebuah *flowchart* yang dijelaskan pada Gambar 3.3.



Gambar 3.3 Perancangan model LSTM

Berikut penjelasan dari *flowchart* yang digambarkan pada Gambar 3.3:

#### 1. Membuka Google Colab

Langkah pertama dalam membuat LSTM adalah dengan membuka google colaboratory. Kemudian setelah masuk ke halaman google colab, membuat *new notebook* dan memberikan nama *file* dengan nama yang sesuai.

## 2. Menginput *Library*

*Library* merupakan kode yang berisi fungsi metode, dan kelas yang sudah dituliskan sebelumnya oleh pihak pengembang. Tujuan penggunaan *library* ini adalah untuk mempercepat dan mempermudah proses pembuatan model LSTM.

## 3. Mengunggah *Dataset*

Pada tahap ini, *dataset* yang berisi data gejala dan label penyakit dibaca terlebih dahulu. *Dataset* gejala ini menggunakan format dalam bentuk csv.

## 4. Memulai *Preprocessing* Data

Langkah selanjutnya adalah melakukan *preprocessing* data menggunakan *library MinMaxScaler*.

## 5. Normalisasi Data dan Penanganan Data Hilang

Selanjutnya adalah proses *preprocessing* data dengan normalisasi dan penanganan data hilang. Data fitur yang telah dipersiapkan dinormalisasi menggunakan metode *MinMaxScaler* agar nilai setiap fitur berada dalam rentang 0 dan 1. Normalisasi ini dilakukan untuk memastikan bahwa fitur memiliki skala yang sama dan model dapat belajar secara efektif.

## 6. Pembagian Data

Langkah selanjutnya adalah proses pembagian *dataset*, agar proses *training* model LSTM dan prediksi yang dihasilkan akurat, maka data csv yang sudah diinput dibagi menjadi dua yaitu data *training* dan data validasi dengan perbandingan 80% data *training* untuk melatih model dan 20% data validasi/*testing* untuk menguji performa model setelah pelatihan. Agar model LSTM dapat melakukan pengenalan pola serta membuat prediksi dengan baik namun tetap memperhatikan dari data gejala penyakit ayam.

## 7. Penentuan *Hyperparameter*

Langkah selanjutnya adalah menentukan batasan untuk model LSTM. Hal ini disebut dengan *hyperparameter*. Selain itu, *hyperparameter* akan bekerja dan berkaitan pada proses pembuatan model LSTM dan proses *training* model LSTM. Adapun *hyperparameter* yang digunakan dalam model LSTM pada penelitian ini adalah sebagai berikut:

### a. *Input size*

*Input size* merupakan jumlah fitur pada setiap data *input* yang dimasukkan ke dalam model pada setiap *timestep*. Dalam penelitian ini, setiap fitur mewakili satu gejala. Dalam *dataset* gejala penyakit ayam terdapat 36, maka *input size* yang digunakan adalah 36. *Input size* harus sesuai dengan dimensi data fitur agar model dapat memproses data dengan benar.

### b. *Hidden size*

*Hidden size* adalah jumlah unit yang terdapat dalam lapisan tersembunyi (*hidden layer*) dari model LSTM. Parameter ini dapat mempengaruhi kapasitas model dalam menyimpan dan mengolah informasi. *Hidden size* yang besar memungkinkan model untuk belajar pola yang lebih kompleks, namun meningkatkan kebutuhan komputasi dan resiko *overfitting*. Pada penelitian ini, *hidden size* yang digunakan adalah 64. Alasan memilih nilai 64 karena merupakan pilihan yang umum dalam literatur dan menunjukkan performa yang baik

### c. *Number of layers*

*Number of layers* menunjukkan berapa banyak lapisan LSTM yang ditumpuk secara berurutan. Semakin banyak lapisan, kompleksitas dan waktu pelatihan model akan bertambah. Pada penelitian ini, jumlah

lapisan LSTM adalah 2. Alasan memilih nilai 2 karena model dapat memproses informasi lebih dalam dibanding satu *layer* serta dapat menjaga kestabilan pelatihan.

d. *Dropout rate*

*Dropout rate* adalah proporsi *neuron* yang secara acak dimatikan selama proses pelatihan untuk mencegah model melakukan *overfitting* terhadap data *training*. *Dropout* dapat membantu model belajar pola yang lebih umum dengan mengurangi ketergantungan pada *neuron* tertentu. Pada penelitian ini, *dropout rate* yang digunakan ada sebesar 0,3. Alasan memilih nilai 0,3 adalah karena nilai tersebut cukup efektif dalam menjaga model, sehingga model lebih mampu menggeneralisasi data baru.

e. *Batch size*

*Batch size* adalah jumlah sampel data yang diproses sekaligus dalam satu iterasi pelatihan. *Batch size* ini dapat mempengaruhi stabilitas proses *training*. *Batch size* yang terlalu kecil akan membuat pelatihan tidak stabil dan lambat, sedangkan *batch size* yang terlalu besar membutuhkan memori besar dan dapat membuat pelatihan kurang efektif. Pada penelitian ini *batch size* yang digunakan adalah 32. Alasan mengambil nilai 32 adalah ukuran yang populer dan efektif dalam memberikan keseimbangan antara kecepatan komputasi dan stabilitas *training*.

f. *Learning rate*

*Learning rate* adalah besar perubahan yang diambil *optimizer* saat memperbarui bobot model berdasarkan gradien *loss function*. Nilai *learning rate* yang sesuai memainkan peran penting dalam stabilitas dan

kecepatan konvergensi pelatihan. Pada penelitian ini, *learning rate* yang digunakan adalah 0,005. Alasan memilih nilai 0,005 karena merupakan nilai standar dan umum digunakan dengan *optimizer* Adam (*Adaptive Moment Estimation*) serta nilai yang cukup ideal dalam sistem ini.

g. *Number of epoch*

*Epoch* adalah jumlah putaran penuh yang dilakukan model terhadap seluruh data *training* selama proses pelatihan. Dalam penelitian ini, model dilatih selama 200 *epoch*. Alasan memilih 200 *epoch* adalah untuk memberikan kesempatan yang cukup bagi model agar dapat belajar pola data secara optimal. Nilai *epoch* tersebut digunakan berdasarkan pengalaman yang menunjukkan stabilitas performa tanpa mengalami *overfitting* yang signifikan.

h. *Sequence of length*

*Sequence length* menunjukkan panjang urutan data yang diproses dalam satu kali *input* ke model LSTM. Dalam penelitian ini, *sequence* yang digunakan adalah 1, karena fitur mewakili kondisi gejala secara statis per observasi. Alasan memilih nilai 1 karena data yang digunakan merupakan data *snapshot* atau fitur tidak berurutan dalam bentuk *time series* panjang. Dengan nilai tersebut model dapat fokus pada fitur *input* pada satu waktu observasi.

i. *Optimizer*

*Optimizer* adalah algoritma yang memperbarui bobot model berdasarkan perhitungan *gradient*. *Optimizer* yang digunakan dalam penelitian ini adalah Adam. *Optimizer* adam ini memiliki tingkat efisiensi yang baik dalam mengatur *learning rate* secara adaptif dan membantu model konvergen lebih cepat.

#### 8. Pelatihan model LSTM

Langkah selanjutnya adalah melakukan *training* model LSTM, tujuan dilakukan *training* agar model LSTM dapat melakukan prediksi. Model dilatih menggunakan data *training* selama jumlah *epoch* yang telah ditentukan. Proses pelatihan ini melibatkan perhitungan *forward pass* untuk mendapatkan prediksi dan *backward pass* untuk memperbarui bobot model berdasarkan *loss function*.

#### 9. Evaluasi model LSTM

Setelah proses pelatihan, model diuji dengan data *testing* untuk menghitung *matrix* seperti *loss* dan akurasi. Tidak hanya itu, model dievaluasi juga dengan menghitung nilai *precision*, *recall*, dan *f-1 score*. Tahap ini menentukan seberapa baik model mampu memprediksi penyakit berdasarkan data yang belum pernah dilihat selama pelatihan.

#### 10. Menampilkan hasil prediksi

Jika model, telah menunjukkan performa yang baik, hasil prediksi akan berupa penyakit yang terdeteksi. Ini adalah *output* akhir yang akan digunakan oleh peternak untuk mengambil tindakan.

### 3.3.3.3 Perancangan Proses Deteksi oleh Model LSTM

Pada tahapan ini dijelaskan bagaimana proses sistem mendeteksi penyakit ayam broiler berdasarkan *input* gejala melalui *form* diagnosa dengan model LSTM. *User* akan mengisi sebanyak 36 gejala ke dalam bentuk pilihan biner, yaitu “Ya” (Bernilai 1) dan “Tidak” (bernilai 0). Seluruh *input* tersebut dikonversi ke dalam bentuk vektor berdimensi 36, yang merepresentasikan kondisi ayam secara keseluruhan. Proses pengisian *form* diagnosa oleh *user* ditunjukkan pada Tabel 3.3, dengan sejumlah gejala bernilai “Ya” atau “Tidak”.

Tabel 3.3 Contoh *Input* Gejala Oleh *User*

| No.            | Nama Gejala                 | Nilai  |         |
|----------------|-----------------------------|--------|---------|
| 1.             | Lesu                        | Ya     | Tidak   |
| 2.             | Nafsu Makan Turun           | Tidak  | Ya      |
| 3.             | Nafas Sesak                 | Ya     | Tidak   |
| 4.             | Nafas Ngorok                | Tidak  | Ya      |
| 5.             | Bulu Kusam                  | Ya     | Tidak   |
| 6.             | Lemas                       | Tidak  | Ya      |
| 7.             | Bersin                      | Ya     | Tidak   |
| 8.             | Batuk                       | Tidak  | Ya      |
| 9.             | Diare                       | Ya     | Tidak   |
| .....          | .....                       | .....  | .....   |
| 33.            | Bintik Merah                | Ya     | Tidak   |
| 34.            | Pucat                       | Tidak  | Ya      |
| 35.            | Bulu Sekitar Kloaka Lengket | Ya     | Tidak   |
| 36.            | Ledir Dari Mulut dan Hidung | Tidak  | Ya      |
| Hasil Prediksi |                             | Tetelo | Gumboro |

Konversi nilai *input* tersebut ke bentuk vektor biner ini menghasilkan representasi  $[1,0,1,0,1,0,1,0,1,\dots,1,0,1,0]$  dan  $[0,1,0,1,0,1,0,1,0,\dots,0,1,0,1]$ . Nilai kedua vektor tersebut kemudian diproses oleh model LSTM. Model akan membaca pola kombinasi gejala dan memetakan ke salah satu dari delapan jenis penyakit ayam broiler yang telah didefinisikan. Hasil vektor biner ke satu menghasilkan prediksi penyakit tetelo dan hasil vektor biner kedua menghasilkan prediksi penyakit gumboro.

### 3.3 Teknik Pengumpulan Data

Berikut adalah metode pengumpulan data yang dilakukan pada penelitian ini.

#### 3.4.1 Studi Literatur

Studi literatur dilakukan dengan mencari serta mengumpulkan referensi–referensi relevan yang berasal dari buku, jurnal ilmiah, artikel, dan sumber lainnya. Studi literatur dilakukan agar mendapatkan berbagai informasi yang dapat

dijadikan sebagai acuan, referensi, maupun dasar pengembangan dalam penelitian. Informasi yang diperoleh, selanjutnya dianalisis, dibandingkan, dan dimanfaatkan untuk memperkaya serta mendukung penelitian yang sedang dilakukan (Sare & Suharsono, 2023).

### 3.4.2 Wawancara

Wawancara merupakan teknik dari pengumpulan data dengan mengajukan pertanyaan – pertanyaan yang relevan pada penelitian kepada pihak terkait (Syafitri & Informatika, 2022). Wawancara ini dilakukan kepada peternak ayam broiler di peternakan Putra Bungsu. Hasil wawancara yang dilakukan dapat dilihat pada lampiran 6 di halaman lampiran.

### 3.4.3 Observasi

Observasi ini dilakukan dengan mengamati pola–pola gejala pada ayam, dimana hasil observasi ini dapat digunakan dalam meningkatkan nilai akurasi model prediksi serta dapat membantu peternak dalam mengambil tindak pencegahan dan pengobatan yang tepat pada ayam yang terkena penyakit.

### 3.4.4 Instrumen Penelitian

Instrumen Penelitian merupakan alat atau metode untuk mengumpulkan data yang diperlukan dalam sebuah penelitian. Instrumen penelitian ini berfungsi untuk mengukur dan menilai sistem yang dibangun serta tingkat penerimaan *user* terhadap sistem tersebut.

#### 3.4.4.1 *Black-box Testing*

Pengujian sistem atau *testing* merupakan tahapan untuk memvalidasi sistem sudah sesuai dengan tujuan penelitian atau belum. Pengujian dilakukan menggunakan sistem *black-box testing*, yaitu menguji fungsi-fungsi pada aplikasi sudah sesuai dengan yang diharapkan atau belum. Selain itu, dilakukan pula *User*

*Acceptance Test* (UAT), yaitu proses verifikasi untuk memastikan bahwa sistem telah memenuhi kebutuhan dan harapan pengguna akhir (Yuliana & Fazriani, 2023).

Data yang dihasilkan dari pengujian UAT dianalisis menggunakan skala *guttman* dengan menggunakan dua interval nilai yaitu “sesuai” dan tidak kemudian dikonversikan ke dalam bentuk persentase. Pada setiap *test case* pengujian akan diberikan menggunakan pendekatan biner:

- a. Jika nilai 1, fungsi sistem sesuai
- b. Nilai 0 jika output sistem tidak sesuai

Selanjutnya, penilaian dihitung persentase kelayakan menggunakan rumus

$$\text{Persentase kelayakan} = \frac{\text{Jumlah skor yang sesuai}}{\text{Jumlah total skenario}} \times 100 \quad (5)$$

Berikut persentase kelayakan dikategorikan berdasarkan kriteria yang ditunjukkan pada tabel 3.4 dibawah ini.

Tabel 3.4 Tabel Persentase Kelayakan

| No. | Pilihan Jawaban | Keterangan    | Persentase |
|-----|-----------------|---------------|------------|
| 1.  | SB              | Sangat Baik   | 81-100     |
| 2.  | B               | Baik          | 61-80      |
| 3.  | C               | Cukup         | 41-60      |
| 4.  | K               | Kurang        | 21-40      |
| 5.  | SK              | Sangat Kurang | 0-20       |