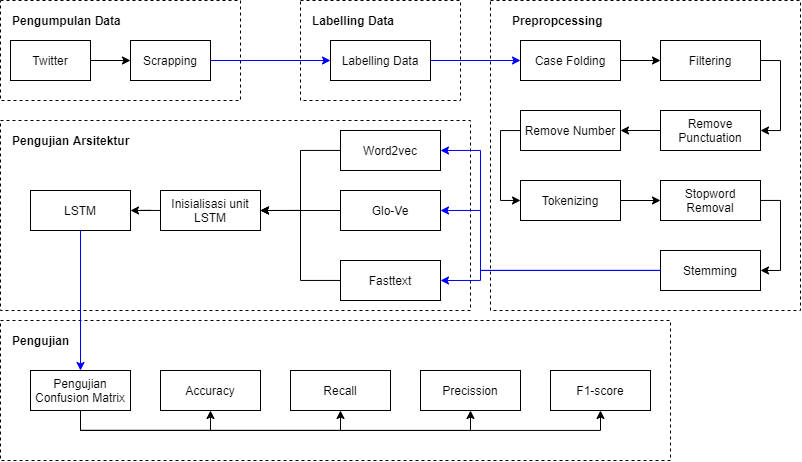
**BAB III METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM**

**3.1 Metodologi Penelitian**

Pada bab ini akan membahas mengenai metodologi penelitian yang digunakan pada penelitian ini. Penelitian ini termasuk penelitian non-implementatif berupa hasil analisis pengujian *word embedding* dan arsitektur LSTM. Metodologi penelitian yang akan digunakan pada penelitian ini dapat dilihat pada gambar 3.1.



**Gambar 3.1 Metodologi Penelitian**

**3.1.1 Pengumpulan Data**

Data yang akan digunakan pada penelitian ini adalah data sekunder yang bersumber dari media sosial *twitter* menggunakan teknik *web scrapping*, data yang diambil untuk data penelitian berupa *tweet* dari beberapa *influencer* sebagai data utama dan data *trending* selama 1 minggu sebagai data pendukung. *Influencer* dipilih karena untuk menjadi *influencer* dibutuhkan proses pengungkapan diri (*self disclosure*) dimedia sosial, dimana salah satu bentuk *self disclosure* adalah mengekspresikan emosi (Saifulloh dan Siregar, 2019). *Web scrapping* dilakukan menggunakan bahasa pemprograman *python* dengan *library selenium*/*tweepy* dan menghasilkan dataset dengan format csv/xlsx.

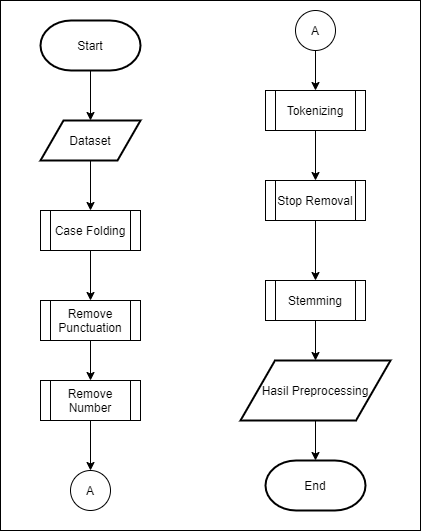
**3.1.2 Labelling Data**

Data yang didapatkan dari hasil *web scrapping* kemudian diberikan label emosi berdasarkan model Ekman, yaitu bahagia, sedih, takut, jijik, marah, terkejut. Labelling akan dilakukan oleh beberapa mahasiswa yang cukup sering bermain twitter, hal ini dilakukan agar data yang diperoleh untuk training model tidak bersifat subjektif. Ilustrasi labelling data dapat dilihat pada tabel 3.1.

**Tabel 3.1 Ilustrasi Labelling**

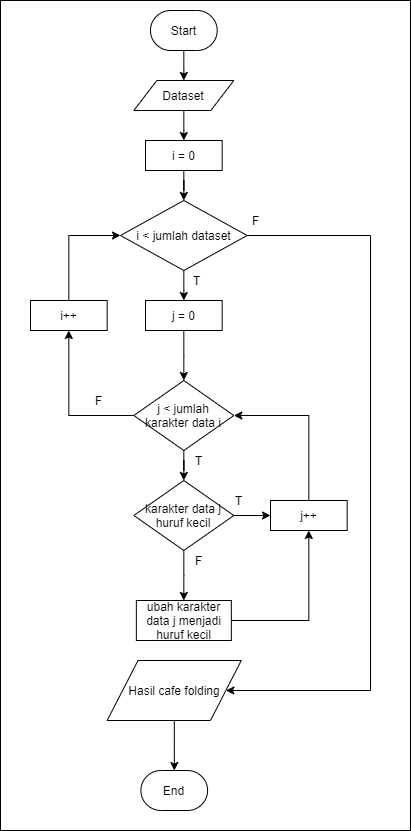
|  |  |
| --- | --- |
| **Teks** | **Emosi** |
| emang cocok banget mukanya senyum terus bahagia selalu | Bahagia |
| Sedih banget pas pamitan ke bapak kos | Sedih |
| Kalau ketemu anjing kabur aja, takut | Takut |
| “GaSuKa gElAy” oposih jijik dewe aku mahan | Jijik |
| dIAJAK FOTO MALAH BERANTEM | Marah |
| Terkejut bila tengok Berita tadi | Terkejut |

**3.1.3 Preprocessing**

 Data yang sudah diperoleh dari proses *web scrapping* dan sudah diberi label, selanjutnya masuk pada proses *preprocessing*. *Preprocessing* berguna untuk membersihkan data dari derau dan bagian tidak informatif yang tidak dibutuhkan, sehingga teks siap untuk diklasifikasi (Haddi et al., 2013). *Preprocessing* akan menjadikan data bersih sehingga proses pembuatan vektor kata dan klasifikasi lebih akurat (Nurrohmat dan SN, 2019). Tahapan *preprocessing* yang akan digunakan pada penelitian ini adalah *case folding*, *remove punction*, *remove number*, *tokenizing*, *stop removal*, *stemming*. *Flowchart* dari proses *preprocessing* dapat dilihat pada gambar 3.2.

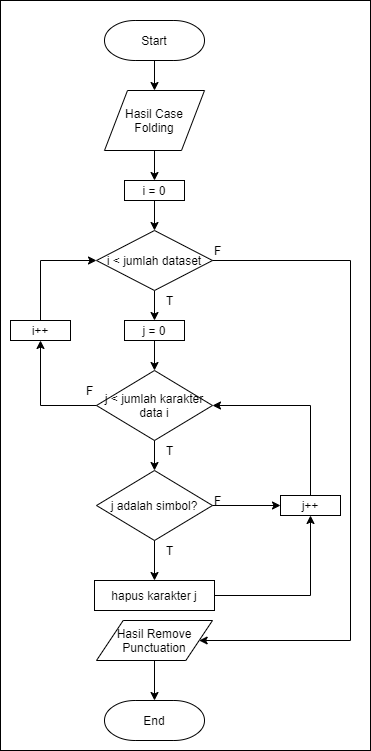
**Gambar 3.2 *Flowchart Preprocessing***

1. ***Case folding***

Langkah pertama pada proses *preprocessing* adalah *case folding*. *Case folding* merupakan proses mengubah seluruh huruf dalam teks tersebut menjadi *lower case* atau huruf kecil semua (Salam et al., 2018), dengan huruf yang diterima dari “a” sampai “z”. *Case folding* bertujuan agar terdapat standarisasi dalam penulisan. *Flowchart* dari proses *case folding* dapat dilihat pada gambar 3.3.

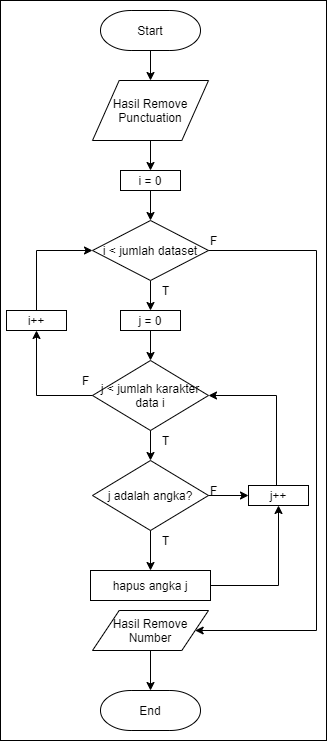
**Gambar 3.3 *Flowchart case folding***

1. ***Remove punctuation***

 Data yang telah melalui proses *case folding*, selanjutnya data akan diproses pada *remove punctuation*. *Remove punctuation* merupakan proses menghapus tanda baca pada teks dengan tujuan mengurangi beban pemprosesan klasifikasi karena dianggap tidak penting dan termasuk *delimiter*, contoh tanda baca yang yang dihapus titik (.), koma(,), tanda tanya (?), *slash* (/), *hastag* (#), tanda seru (!) dan lain-lain. *Flowchart* dari proses *remove punctuation* dapat dilihat pada gambar 3.4.

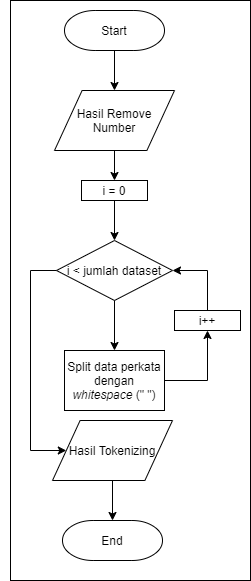
**Gambar 3.4 *Flowchart remove punctuation***

1. ***Remove number***

 Data hasil dari proses *remove punctuation* kemudian diproses ke proses *remove number*. *Remove number* merupakan proses menghapus angka pada suatu teks, penghapusan ini sebab angka dianggap tidak memiliki arti dan termasuk *delimiter*, mirip seperti *remove punctuation* hanya berbeda pada objek yang dihapus. *Flowchart* proses *remove number* dapat dilihat pada gambar 3.5.

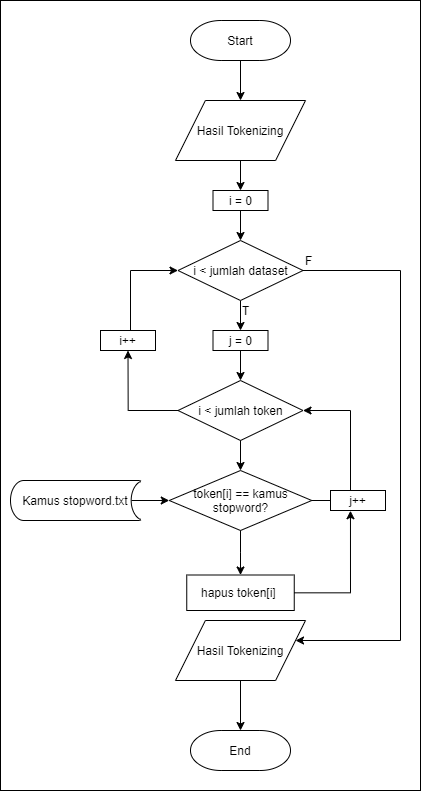
**Gambar 3.5 *Flowchart remove number***

1. ***Tokenizing***

 Data hasil dari proses *remove number* kemudian diproses ke proses *tokenizing*. *Tokenizing* merupakan pemotongan kalimat berdasarkan tiap-tiap kata penyusunnya, biasanya pemotongan berdasarkan *whitespace* seperti spasi, tab, dan enter. Tiap-tiap kata hasil *tokenizing* disebut token (Juwiantho et al., 2020). *Flowchart* dari proses *tokenizing* dapat dilihat pada gambar 3.6.

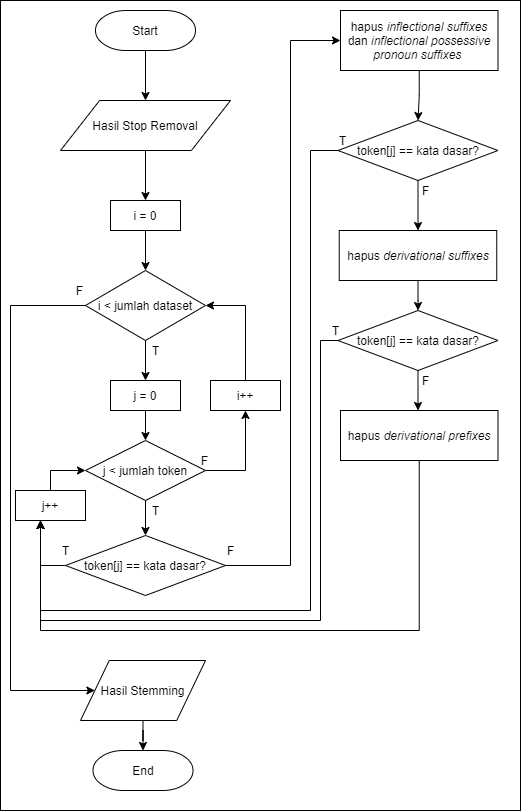
**Gambar 3.6 *Flowchart tokenizing***

1. ***Stopword Removal***

 Data hasil dari proses *tokenizing* kemudian diproses ke proses *stopword removal*. *Stopword Removal* merupakan proses untuk menghilangkan kata-kata yang tidak memiliki arti atau makna, namun tidak akan mengubah makna dari komentar tersebut (Juwiantho et al., 2020). *Stopword removal* akan mengurangi ukuran indeks, waktu pemprosesan klasifikasi, dan *noise* dari suatu data. Biasanya *stopword* berupa kata ganti orang dan kata hubung, seperti “aku”, “kamu”, “kita”, “dan”, “atau”. *Flowchart* dari proses *stopword removal* dapat dilihat pada gambar 3.7.

**Gambar 3.7 *Flowchart stopword removal***

1. ***Stemming***

 Data hasil dari proses *stopword removal* kemudian masuk ke proses *stemming*. *Stemming* merupakan proses untuk mentransformasi kata kerja yang berimbuhan pada suatu dokumen menjadi kata dasar (*root word*), dengan menghapus awalan, akhiran maupun sisipan . *Stemming* bertujuan untuk mengurangi varian kata dengan makna yang hampir sama pada suatu dokumen, serta meningkatkan performa pada tahap *information retrieval*. Pada penelitian ini akan menggunakan *library* sastrawi, karena data pada penelitian ini menggunakan bahasa Indonesia. *Flowchart* proses *stemming* dapat dilihat pada gambar 3.8.

**Gambar 3.8 *Flowchart stemming***

Contoh data sebelum dan sesudah *preprocessing* dapat dilihat pada tabel 3.2

**Tabel 3.2 Contoh *preprocessing***

|  |  |
| --- | --- |
| Teks asli (sebelum *preprocessing)* | “Hore, akhirnya timnas Indonesia menang dengan skor 2-0 !!” |
| Teks setelah *case folding* | “hore, akhirnya timnas indonesia menang dengan skor 2-0 !!” |
| Teks setelah *remove punctuation* | “hore, akhirnya timnas indonesia menang dengan skor 20” |
| Teks setelah *remove number* | “hore akhirnya timnas indonesia menang dengan skor” |
| Teks setelah *tokenizing* | [“hore”, “akhirnya”, “timnas”, “indonesia”, “menang”, “dengan”, “skor”] |
| Teks setelah *stopword removal* | [“hore”, “akhirnya”, “timnas”, “indonesia”, “menang”, “skor”] |
| Teks setelah *stemming* | [“hore”, “akhir”, “timnas”, “indonesia”, “menang”, “skor”] |
| Teks setelah *preprocessing* | [“hore”, “akhir”, “timnas”, “indonesia”, “menang”, “skor”] |

**3.1.4 Pengujian Arsitektur**

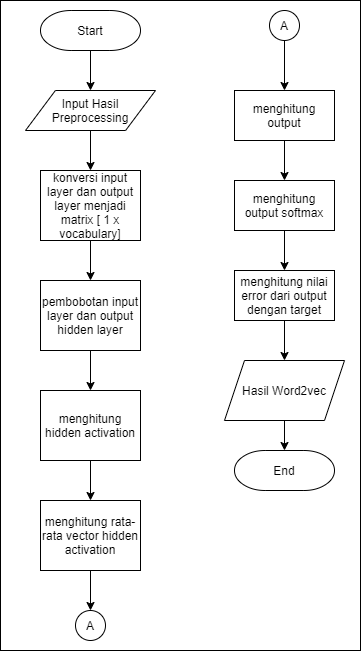
Pengujian arsitektur bertujuan untuk menemukan arsitektur terbaik sehingga mendapatkan hasil *accuracy*, *recall*, *precission* yang maksimal. Pengujian arsitektur menggunakan beberapa parameter, diantaranya jumlah *unit* / *neuron*, fungsi aktivasi, optimasi, *epoch*, jumlah *dropout*. Parameter yang diuji adalah jumlah jumlah *unit* / *neuron* dan jumlah *dropout*. Penelitian ini akan menggunakan jumlah *unit* / *neuron* sebanyak 50 *unit* sampai 200 *unit*, sedangkan untuk *dropout* akan menggunakan 20, 30, dan 50. *Dropout* berguna untuk memutus sebagian koneksi antar unit secara random sehingga mengurangi *overfitting* secara signifikan. Fungsi aktivasi pada penelitian ini menggunakan fungsi aktivasi *Rectifier Linear Unit* (ReLU). Optimasi yang digunakan pada penelitian ini adalah *Adaptive Moment Estimation* (ADAM). Jumlah *epoch* yang digunakan pada penelitian ini sebanyak 50 *epoch*. Penelitian yang dilakukan oleh Sari (2017) menunjukkan model LSTM menghasilkan akurasi terbaik 96,38% menggunakan fungsi aktivasi *Rectifier Linear Unit* (ReLU) dan optimasi *Adaptive Moment Estimation* (ADAM) dengan jumlah *epoch* yang digunakan sebanyak50 *epoch* (Sari et al., 2017). Fungsi *loss* yang digunakan adalah fungsi *Loss Categorical Cross Entropy*, karena pada penelitian ini terdapat lebih dari 2 kelas. Pembagian antara data *training* dan data *testing* pada penelitian ini menggunakan 70% data *training* dan 30% data *testing* dengan, hal ini dikarenakan jumlah pembagian 70:30 dan 80:20 dapat menghasilkan model terbaik (Gholamy et al., 2018).

**3.1.5 Word embedding**

Bagian ini akan menjelaskan tahapan dari proses *word embedding*, hasil dari *preprocessing* kemudian masuk ke proses *word embedding*. *Word embedding* merupakan suatu teknik memetakan kata-kata berdasarkan suatu kamus yang sudah ada sehingga menghasilkan vektor-vektor angka yang berisi angka rill (Halim et al., 2020), hal ini dilakukan karena *deep learning* tidak dapat memproses data *string* atau teks, oleh karena itu data harus diubah dari data *string* atau teks menjadi vektor angka menggunakan *word embedding*. Penggunaan *word embedding* berpengaruh terhadap akurasi yang dihasilkan oleh model. Penelitian ini akan membandingkan *word embedding word2vec*, *Glo-Ve*, *fasttext* untuk mencapai tujuan penelitian.

1. ***Word2vec***

*Word embedding word2vec* banyak digunakan karena kelebihannya dalam mengenali kata-kata yang mirip berdasarkan sematiknya (Nurdin et al., 2020). *Word embedding word2vec* menggunakan *unsupervised neural network* yang terdiri dari sebuah *hidden layer* dan *fully connected layer* (Nurdin et al., 2020). Hasil dari *preprocessing* kemudian masuk ke proses *word2vec*. Model *word2vec* yang akan digunakan pada penelitian ini adalah model *Continous bag of words* (CBOW). *Flowchart* proses *word2vec* dapat dilihat pada gambar 3.9.

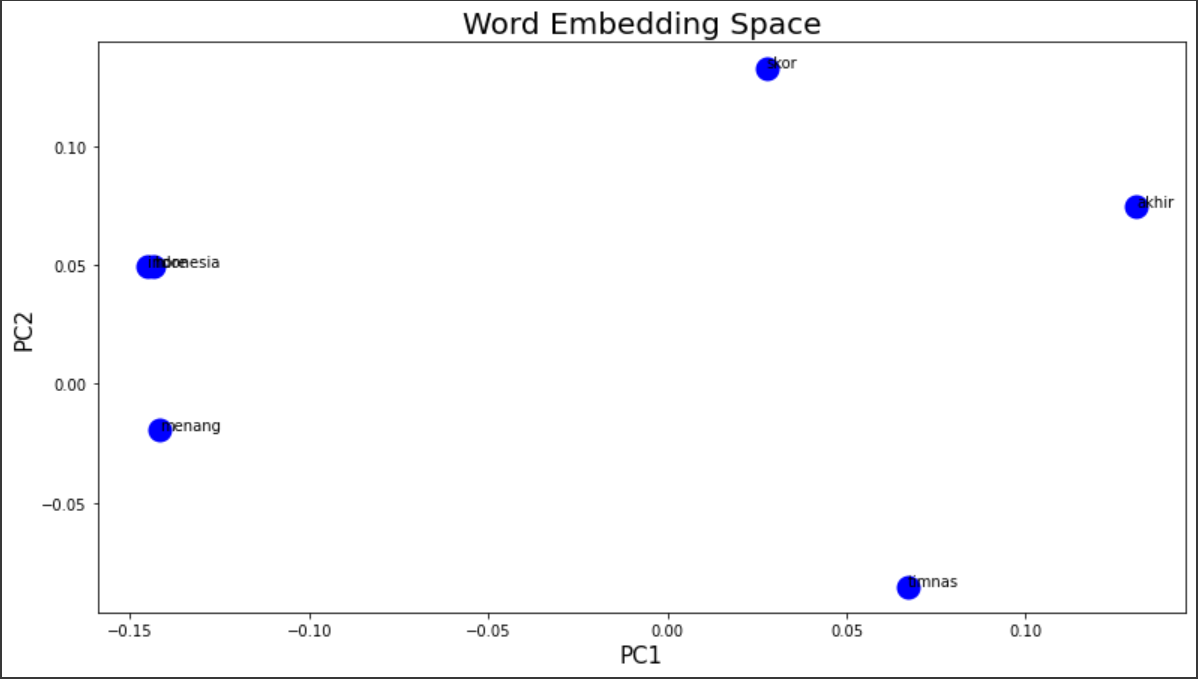
**Gambar 3.9 *Flowchart word2vec***

Contoh hasil *word2vec* dengan vektor berdimensi 5 dengan kalimat yang sama dengan hasil proses *preprocessing*, contoh dapat dilihat pada tabel 3.2.

**Tabel 3.3 Contoh Hasil *Word2vec***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kata | Dimensi | | | | |
| D1 | D2 | D3 | D4 | D5 |
| hore | 0.084777 | 0.057706 | -0.082217 | -0.054878 | -0.096413 |
| akhir | -0.094585 | 0.068689 | 0.090819 | -0.059799 | 0.088703 |
| timnas | -0.076862 | -0.039967 | 0.068271 | 0.027309 | -0.035721 |
| indonesia | 0.095575 | -0.025707 | -0.084364 | -0.094829 | -0.006755 |
| menang | 0.085284 | -0.021515 | -0.033860 | -0.067008 | -0.096384 |
| skor | -0.022820 | 0.086830 | -0.074412 | -0.014862 | 0.085389 |

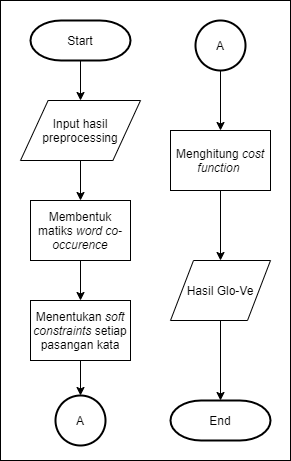
Vektor vektor hasil proses *word embedding word2vec* dapat divisualisasikan secara 2 dimensi yang menggambarkan hubungan antar kata. Visualisasi vektor vektor hasil *word2vec* dapat dilihat pada gambar 3.10.



**Gambar 3.10 Visualisasi *word2vec***

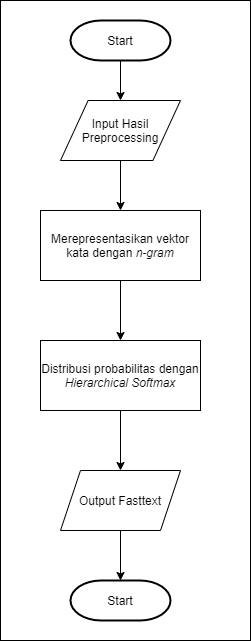
1. ***Glo-Ve***

Hasil dari proses *preprocessing* kemudian masuk ke *word embedding Glo-Ve* untuk dipetakan menjadi vektor angka. *Global vectors for word representation* (Glo-Ve)merupakan salah satu metode *word embedding* yang mengandalkan *co-occurrence* kata atau statistik kemunculan kata dalam kumpulan kata atau korpus yang ditangkap langsung oleh model untuk memperoleh hubungan semantik antar kata dalam korpus. Glo-Ve menggunakan metode *global matrix factorization* yang mewakili jumlah kemunculan atau frekuensi dalam suatu korpus (Pennington et al., 2014). *Flowchart word embedding Glo-Ve* dapat dilihat pada gambar 3.10.

**Gambar 3.11 *Flowchart Glo-Ve***

1. ***Fasttext***

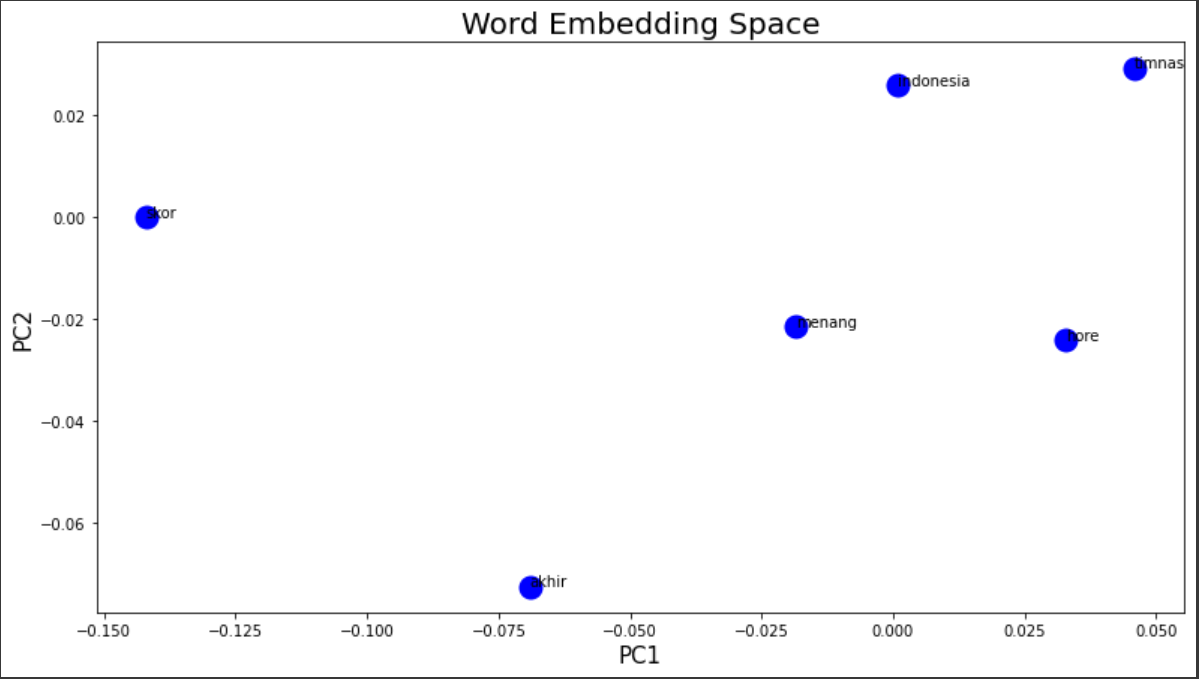
Hasil dari proses *preprocessing* kemudian masuk ke proses *word embedding fasttext* untuk mengubahnya ke vektor angka. *Fasttext* merupakan metode *word embedding* pengembangan dari *word2vec*. *Fasttext* memliki kelebihan yaitu dapat menangani masalah *out of vocabulary* dengan memperhatikan informasi *subword* menggunakan *n-gram* ke dalam model skipgram(Bojanowski et al., 2017). *Flowchart* proses *word embedding fasttext* dapat dilihat pada gambar 3.11.

**Gambar 3.12 *Flowchart fasttext***

Contoh hasil *fasttext* dengan vektor berdimensi 5 dengan kalimat yang sama dengan hasil proses *preprocessing*, contoh dapat dilihat pada tabel 3.3. Vektor vektor hasil proses *word embedding word2vec* dapat divisualisasikan secara 2 dimensi yang menggambarkan hubungan antar kata. Visualisasi vektor vektor hasil *word2vec* dapat dilihat pada gambar 3.11.

**Tabel 3.4 Contoh hasil *fasttext***

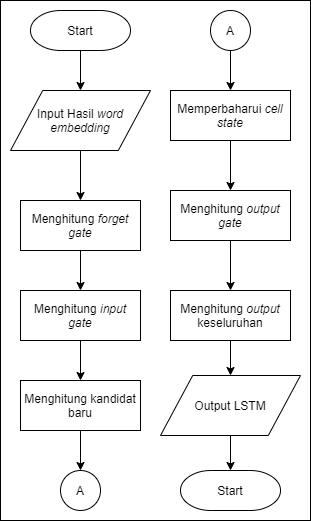
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kata | Dimensi | | | | |
| D1 | D2 | D3 | D4 | D5 |
| hore | -0.015188 | 0.006831 | -0.000868 | -0.033525 | -0.036368 |
| akhir | 0.026232 | -0.020445 | 0.043318 | 0.081208 | -0.070539 |
| timnas | 0.041364 | 0.038119 | -0.043051 | -0.017365 | -0.014987 |
| indonesia | 0.026756 | -0.005694 | -0.003622 | -0.020880 | 0.005214 |
| menang | -0.005905 | 0.024995 | 0.015284 | 0.050226 | 0.000018 |
| skor | 0.041438 | -0.059298 | 0.087633 | 0.081472 | 0.033918 |



**Gambar 3.13 Visualisasi *fasttext***

**3.1.6 LSTM**

Hasil *output word embedding* kemudian masuk pada proses LSTM. LSTM memiliki *memory cell* dan *gate units* (*input gate, forget gate, output gate*) untuk menangani masalah masalah *vanishing gradient*. *Forget gate* berfungsi untuk menentukan informasi mana yang dihapus dari *cell*. *Input gate* berfungsi untuk menentukan nilai dari *input* untuk diperbaharui pada *state memory*. *Output gate* berfungsi untuk menentukan *output* berdasarkan *input* dan memori pada *cell* (Utomo, 2020). *Flowchart* proses LSTM dapat dilihat pada gambar 3.12.

**Gambar 3.14 *Flowchart* LSTM**

Analisis mengenai *Long Short Term Memory* (LSTM) lebih lanjut dibahas dalam contoh perhitungan algoritma LSTM dengan data yang didapatkan dari hasil proses *preprocessing* dan hasil *word embedding fasttext* pada tabel 3.4. Data hasil *preprocessing* dan *word embedding fasttext* terdapat 6 kata, yaitu “hore”, “akhir”, “timnas”, “indonesia”, “menang”, “skor”, dimana kata-kata tersebut akan menjadi *input* dari LSTM secara berurutan. Contohnya hasil *output* dari proses “hore” akan menjadi input dari kata “akhir”. Contoh perhitungan LSTM yang digunakan sebanyak 5 *neuron*. *Neuron* berguna dalam pembentukan *weight* tiap gerbang (*W*), *recurrent weight* tiap gerbang(*U*), dan bias tiap gerbang (*b*). Proses perhitungan *input* kata “hore” dengan 5 *neuron*, yaitu :

1. Menghitung *forget gate* (*ft*)

Proses menghitung *forget gate* menggunakan rumus pada persamaan 2.6. Variabel *weight forget gate* (*Wf*), *recurrent weight forget gate* (*Uf*), *bias forget gate* (*bf*) di inisialisasi awal dengan nilai *random*, sedangkan untuk variabel *output* proses sebelumnya (*ht-1*) merupakan *output* dari proses sebelumnya, namun karena pada contoh ini adalah kata awal maka nilai *ht-1* adalah matriks dengan nilai 0, dan *xt* merupakan input yang didapatkan dari hasil *word embedding fasttext*. Sehingga masing masing variabel dijabarkan sebagai berikut :

Nilai *weight forget gate* (*Wf*)kemudian dikalikan dengan *input* (*xt*), hasil dari perkalian tersebut dejabarkan sebagai berikut :

Selanjutnya, nilai *recurrent weight forget gate* (*Uf*)dikalikan dengan *output* dari proses sebelumnya (*ht-1*), karena *output* dari proses sebelumnya (*ht-1*) tidak ada maka nilainya 0, sehingga hasil *recurrent weight forget gate* (*Ui*)dikalikan dengan *output* dari proses sebelumnya (*ht-1*) adalah 0, hasil tersebut dijabarkan sebagai berikut :

Hasil proses dan kemudian ditambahkan dengan nilai *bias* (*bi*), dijabarkan sebagai berikut :

Nilai *forget gate* (*ft*)didapatkan dengan menghitung hasil dengan fungsi aktivasi *sigmoid*. Fungsi aktivasi *sigmoid* akan menghasilkan nilai antara 0 dan 1, proses menghitung *forget gate* (*ft*)dijabarkan sebagai berikut :

1. Menghitung *input gate* (*it*)

Proses selanjutnya menghitung *input gate* menggunakan rumus pada persamaan 2.7, sama seperti *forget gate* ada beberapa variabel yang dibutuhkan yaitu *weight input gate* (*Wi*), *recurrent weight input gate* (*Ui*), *input* (*xt*), *output* dari proses sebelumnya (*ht-1*), dan *bias input gate* (*bi*). Nilai *Wi, Ui, bi* untuk insialisasi awal bernilai *random*, nilai variabel *Wi, Ui, bi* dijabarkan sebagai berikut :

Langkah selanjutnya mengalikan *input*(*xt*)dengan *weight input gate* (*Wi*), langkah ini dijabarkan sebagai berikut :

Selanjutnya, nilai *recurrent weight input gate* (*Ui*)dikalikan dengan *output* dari proses sebelumnya (*ht-1*), karena *output* dari proses sebelumnya (*ht-1*) tidak ada maka nilainya 0, sehingga hasil *recurrent weight input gate* (*Ui*)dikalikan dengan *output* dari proses sebelumnya (*ht-1*) adalah 0, hasil tersebut dijabarkan sebagai berikut :

Nilai hasil dan kemudian ditambah dengan nilai *bias input gate* (*bi*), proses ini dijabarkan sebagai berikut :

Nilai *input gate* (*it*) didapatkan dengan menghitung nilai dengan fungsi aktivasi *sigmoid*. Fungsi aktivasi *sigmoid* akan menghasilkan nilai antara 0 dan 1, proses perhitungan *input gate* (*it*) dijabarkan sebagai berikut :

1. Menghitung nilai kandidat baru ()

Proses selanjutnya menghitung nilai kandidat baru () menggunakan rumus pada persamaan 2.8, ada beberapa variabel yang digunakan dalam proses ini yaitu *weight* kandidat konteks (*Wc*), *recurrent weight* kandidat konteks (*Uc*), *input* (*xt*), *output* proses sebelumnya (*ht-1*), dan nilai *bias* kandidat konteks (*bc*). Nilai *Wc*, *Uc*, *bi* untuk inisialisasi awal adalah *random*. Nilai *Wc*, *Uc*, *bi* dijabarkan sebagai berikut :

Nilai kandidat baru didapatkan dengan menghitung nilai dengan fungsi aktivasi *tanh*, fungsi aktivasi *tanh* menghasilkan nilai antara 1 dan -1. Pertama mengalikan nilai *input* (*xt*) dengan nilai *weight* kandidat konteks (*Wc*), proses ini dijabarkan sebagai berikut :

Proses selanjutnya, nilai *recurrent weight* kandidat konteks (*Uc*)dikalikan dengan *output* dari proses sebelumnya (*ht-1*), karena *output* dari proses sebelumnya (*ht-1*) tidak ada maka nilainya 0, sehingga hasil *recurrent weight* kandidat konteks(*Uc*)dikalikan dengan *output* dari proses sebelumnya (*ht-1*) adalah 0, hasil tersebut dijabarkan sebagai berikut :

Proses selanjutnya, hasil dari proses dan ditambahkan dengan nilai *bias* kandidat konteks (), penjabaran proses ini dijabarkan sebagai berikut :

Hasil dari proses kemudian dihitung menggunakan fungsi aktivasi *tanh* untuk menghasilkan nilai kandidat baru (). Penjabaran perhitungan nilai kandidat baru () sebagai berikut :

1. Menghitung *cell state*

Langkah selanjutnya adalah memperbaharui *cell state* lama *Ct-1* menjadi *cell state* baru *Ct*. Nilai *cell state* baru dihitung menggunakan persamaan 2.9, karena nilai *cell state* sebelumnya belum ada, maka nilainya 0, nilai *cell state* lama dijabarkan sebagai berikut :

Langkah selanjutnya mengalikan nilai *cell state* lama dengan nilai *forget gate*, langkah ini dijabarkan sebagai berikut :

Nilai *input gate* dikalikan dengan nilai kadidat baru yang diperoleh pada proses sebelumnya, langkah ini dijabarkan sebagai berikut :

Nilai *cell state* dihitung dengan menjumlahkan hasil dari proses dan , penjabaran proses ini dijabarkan sebagai berikut :

1. Menghitung *output gate* (*Ot*)

Langkah selanjutnya adalah menghitung *output gate*, langkah ini dihitung menggunakan persamaan 2.10. Variabel yang dibutuhkan pada langkah ini yaitu *weight ouput gate (Wo*), *recurrent weight output gate* (*Uo*), *input* (*xt*), *output* dari proses sebelumnya (*ht-1*), dan *bias output gate* (*bo*). Nilai *Wo*, *Uo*, *bo* pada inisialisasi awal bernilai *random* yang dijabarkan sebagai berikut :

Langkah selanjutnya mengalikan *input* (*xt*) dengan *weight output gate* (*Wo*), penjabaran langkah ini sebagai berikut :

Selanjutnya mengalikan *output* proses sebelumnya dengan *recurrent weight output gate*, penjabaran proses pengalian sebagai berikut :

Selanjutnya nilai dari hasil proses dan ditambahkan dengan nilai *bias output gate*, penjabaran proses ini sebagai berikut :

Menghitung *output gate* didapatkan dengan menghitung dengan fungsi aktivasi *sigmoid*, dimana fungsi aktivasi ini akan menghasilkan nilai antara 0 sampai 1. Proses menghitung *output gate* dijabarkan sebagai berikut :

1. Menghitung *output final* (*ht*)

Langkah terakhir adalah menghitung *output final* (*ht*) menggunakan persamaan 2.11, pertama menghitung nilai *tanh*(Ct) yang dijabarkan sebagai berikut :

Selanjutnya, menghitung *output gate* dengan mengalikan *output gate* dengan , proses ini dijabarkan sebagai berikut :

Proses *input* kata “hore” menghasilkan *output final* , *output final* ini akan digunakan untuk proses *input* kata selanjutnya, yaitu kata “akhir”, hasil *output final* kata “akhir” akan digunakan pada proses *input* kata “timnas” dan begitu seterusnya. Hasil keseluruhan proses dari awal hingga akhir untuk kata [“hore”, “akhir”, “timnas”, “indonesia”, “menang”, “skor”] dapat dilihat pada tabel 3.5, tabel 3.6, tabel 3.7, tabel 3.8, tabel 3.9.

**Tabel 3.5 Hasil *forget gate* (*ft*) tiap orde**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Orde(*t*)** | **Input(*xt*)** | ***Forget gate* (*ft*)** | | | | |
| 0 | hore | 0,508226 | 0,491192 | 0,494533 | 0,500566 | 0,504109 |
| 1 | akhir | 0,539047 | 0,48413 | 0,497744 | 0,511704 | 0,498023 |
| 2 | timnas | 0,435014 | 0,479691 | 0,602584 | 0,496556 | 0,511039 |
| 3 | indonesia | 0,52241 | 0,511705 | 0,518073 | 0,530234 | 0,505841 |
| 4 | menang | 0,160368 | 0,54972 | 0,524729 | 0,532748 | 0,525823 |
| 5 | skor | 0,537819 | 0,557364 | 0.503936 | 0,567526 | 0,534936 |

**Tabel 3.6 Hasil *input gate* (*it*) tiap orde**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Orde(*t*)** | **Input(*xt*)** | ***Input gate* (*it*)** | | | | |
| 0 | hore | 0,499075 | 0,507994 | 0,484948 | 0,504131 | 0,500975 |
| 1 | akhir | 0,682631 | 0,531195 | 0,478373 | 0,499388 | 0,505449 |
| 2 | timnas | 0,474113 | 0,596975 | 0,494946 | 0,557532 | 0,484612 |
| 3 | indonesia | 0,486321 | 0,516592 | 0,504831 | 0,510159 | 0,504823 |
| 4 | menang | 0,541772 | 0,555008 | 0,513857 | 0,533343 | 0,535446 |
| 5 | skor | 0,163149 | 0,544335 | 0,079544 | 0,550523 | 0,54317 |

**Tabel 3.7 Hasil Kandidat baru () tiap orde**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Orde(*t*)** | **Input(*xt*)** | **Kandidat baru()** | | | | |
| 0 | hore | 0,469511 | 0,035555 | 0,021698 | 0,025733 | 0,013274 |
| 1 | akhir | 0,168775 | 0,475897 | -0,25554 | -0,081948 | 0,615148 |
| 2 | timnas | 0,051175 | 0,008288 | 0,178389 | 0,106098 | 0,065152 |
| 3 | indonesia | -0,05184 | 0,0211608 | 0,038442 | 0,111437 | 0,035598 |
| 4 | menang | 0,0905463 | 0,134515 | 0,0510476 | 0,159938 | 0,0945864 |
| 5 | skor | 0,004678 | 0,141765 | 0,125673 | 0,124838 | 0,157632 |

**Tabel 3.8 Hasil *Cell state* (*Ct*) tiap orde**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Orde(*t*)** | **Input(*xt*)** | **Kandidat baru()** | | | | |
| 0 | hore | 0,234321 | 0,018062 | 0,010522 | 0,012973 | 0,00665 |
| 1 | akhir | 0,241521 | 0,261538 | -0,117006 | 0,301301 | 0,526422 |
| 2 | timnas | 0,129328 | 0,130405 | 0,017903 | 0,208766 | 0,300595 |
| 3 | indonesia | 0,042351 | 0,07766 | 0,028682 | 0,167545 | 0,170024 |
| 4 | menang | 0,055847 | 0,117348 | 0,041281 | 0,174561 | 0,1400484 |
| 5 | skor | 0,030799 | 0,142573 | 0,030799 | 0,156694 | 0,160538 |

**Tabel 3.9 Hasil *Output gate* (*Ot*) tiap orde**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Orde(*t*)** | **Input(*xt*)** | ***Output gate* (*Ot*)** | | | | |
| 0 | hore | 0,50845 | 0,497704 | 0,499944 | 0,508844 | 0,483251 |
| 1 | akhir | 0,340574 | 0,514624 | 0,513813 | 0,390576 | 0,462021 |
| 2 | timnas | 0,52287 | 0,462506 | 0,293463 | 0,521102 | 0,518733 |
| 3 | indonesia | 0,522442 | 0,506951 | 0,529952 | 0,510473 | 0,495368 |
| 4 | menang | 0,535088 | 0,527644 | 0,544744 | 0,510951 | 0,502957 |
| 5 | skor | 0,508765 | 0,526352 | 0,511101 | 0,544413 | 0,51488 |

**Tabel 3.10 Hasil *output final* (*ht*) tiap orde**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Orde(*t*)** | **Input(*xt*)** | ***Output final* (*ht*)** | | | | |
| 0 | hore | 0,117579 | 0,008961 | 0,00526 | 0,006601 | 0,003213 |
| 1 | akhir | 0,080693 | 0,131606 | -0,059846 | 0,114244 | 0,232165 |
| 2 | timnas | 0,026734 | 0,003833 | 0,051802 | 0,055081 | 0,033749 |
| 3 | indonesia | 0,022112 | 0,039291 | 0,015196 | 0,084736 | 0,083422 |
| 4 | menang | 0,029852 | 0,061635 | 0,022475 | 0,088297 | 0,069981 |
| 5 | skor | 0,000948 | 0,074539 | 0,015736 | 0,084615 | 0,081954 |

Setelah mendapatkan *output final* (*ht*) langkah selanjutnya masuk ke *dense layer* untuk menentukan hasil klasifikasi emosi. *Dense layer* mendapatkan *input* dari hasil *output final* (*ht*) orde terakhir. Persamaan *dense layer* menurut Hochreiter dan Schmidhber (1997) dapat dilihat pada persamaan 3.1.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .(3.1)

Keterangan :  
 Y = *dense layer  
 ht* = *output final*  
 *Wy* = *weight dense layer*  
 *by* = *bias dense layer*

Nilai *weight dense layer* (*Wy*) dan nilai *bias* (*by*)dinisialisasi dengan nilai *random*, *weight dense layer* (*Wy*)dan *bias* (*by*)memiliki 6 *neuron* sesuai dengan jumlah target emosi. Nilai *weight dense layer* (*Wy*) dan nilai *bias* (*by*)dijabarkan sebagai berikut :

Perhitungan *dense layer* (*Y*) menggunakan persamaan 3.1, penjabaran proses perhitungan *dense layer* sebagai berikut :

Hasil *dense layer* kemudian dicari nilai *probabilitas* tertinggi, dari hasil *dense layer* nilai *probabilitas* tertinggi didapatkan pada index ke 0, sehingga pada kata “Hore, akhirnya timnas Indonesia menang dengan skor 2-0 !!” memiliki emosi bahagia.

**3.1.7 Pengujian**

Tahap terakhir adalah pengujian sesuai dengan perancangan arsitektur yang dibangun dan *word embedding* yang dibandingkan, pengujian akan menggunakan *confusion matrix* untuk mendapatkan nilai *accuracy*, *precision*, *recall*,dan *f1-score*. Arsitektur terbaik dari masing-masing *word embedding* akan dibandingkan untuk mengetahui tujuan penelitian tercapai atau tidak serta menganalisis hasil dari penelitian ini, tabel pengujian perancangan arsitektur dan *word embedding* dapat dilihat pada tabel 3.11.

**Tabel 3.11 Pengujian**

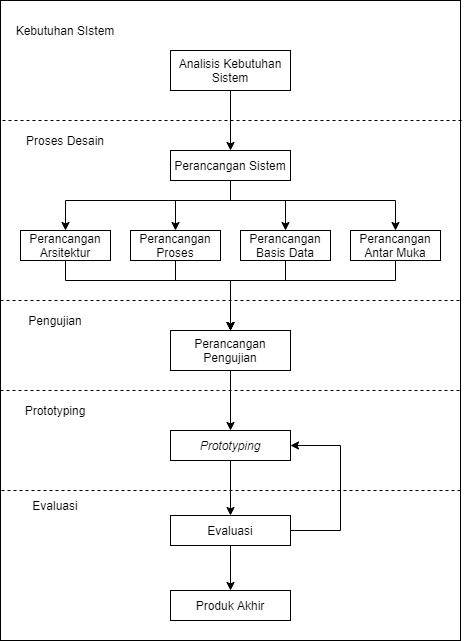
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Word embedding* | *Unit* | *Dropout* | *accuracy* | *precission* | *recall* | *f1-score* |
| *Word2vec* | 50 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 100 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 150 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 200 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| Glo-Ve | 50 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 100 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 150 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 200 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |

**Tabel 3.12 Pengujian (lanjutan)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Word embedding* | *Unit* | *Dropout* | *accuracy* | *precision* | *recall* | *f1-score* |
| *Fasttext* | 50 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 100 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 150 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |
| 200 | 20 |  |  |  |  |
| 30 |  |  |  |  |
| 50 |  |  |  |  |

**3.2 Metodologi Pengembangan Sistem**

Metode pengembangan sistem yang digunakan pada penelitian ini adalah *prototyping*. *Prototyping* digunakan pada penelitian ini untuk mengembangkan model menjadi final, sehingga sistem yang dikembangkan lebih cepat dan lebih hemat dibandingkan metode pengembangan sistem lainnya. Metodologi pengembangan sistem yang digunakan pada penelitian ini dapat dilihat pada gambar 3.15.

**Gambar 3.15 Metodologi Pengembangan Sistem**

**3.2.1 Analisis Kebutuhan Sistem**

Analisis kebutuhan sistem mengidentifikasi kebutuhan yang dibutuhkan sistem untuk mengembangkan sistem dengan cara mendefinisikan sistem kedalam bagian-bagian komponen yang lebih kecil. Analisis kebutuhan sistem dibagi menjadi 2, yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

**3.2.2 Kebutuhan Fungsional**

Kebutuhan yang berkaitan langsung dengan sistem dinamakan kebutuhan fungsional, kebutuhan fungsionalitas ini memuat gambaran umum yang dapat dilakukan oleh sistem, gambaran umum sistem yang akan dibuat pada penelitian ini sebagai berikut :

1. Sistem dapat memproses teks yang diinputkan dengan *Long Short Term Memory*(LSTM) dan *word embedding fasttext*.
2. Sistem dapat menghasilkan emosi yang cukup akurat.
3. Sistem dapat melakukan proses *login* dan *register*.
4. Sistem dapat menyimpan teks dan hasil emosi.
5. Sistem dapat menampilkan hasil rekap emosi.

**3.2.3 Kebutuhan Non-fungsional**

Kebutuhan yang tidak berkaitan langsung dengan sistem dan menggambarkan properti yang dimiliki oleh sistem dinamakan kebutuhan non-fungsional. Kebutuhan non-fungsional terdiri dari kebutuhan perangkat keras (*hardware*) dan kebutuhan perangkat lunak (*software*) :

1. Kebutuhan perangkat keras (*hardware*)

Kebutuhan perangkat keras (*hardware*) yang dibutuhkan dan spesifikasi untuk pengembangan sistem pada penelitian ini dapat dilihat pada tabel 3.13.

**Tabel 3.13 Kebutuhan perangkat keras (*hardware*)**

|  |  |  |
| --- | --- | --- |
| No | Perangkat keras (*hardware*) | Keterangan |
| 1. | *Processor* | Intel Core i5-8265U 8th Gen |
| 2. | *RAM* | 4 GB DDR4 |
| 3. | *Storage* | Harddisk 500GB |

1. Kebutuhan perangkat lunak (*software*)  
   Kebutuhan perangkat lunak (*software*) yang dibutuhkan dan spesifikas untuk pengembangan sistem pada penelitian ini dapat dilihat pada tabel 3.14.

**Tabel 3.14 Kebutuhan perangkat lunak (*software*)**

|  |  |  |
| --- | --- | --- |
| No | Perangkat lunak (*software*) | Keterangan |
| 1. | OS *system* | Windows |
| 2. | *Google colab* | *Notebook cloud* |
| 3. | *Chrome* | Web browser |
| 4. | *python* | Bahasa pemprograman |
| 5. | *XAMPP* | DBMS |

**3.2.4 Proses Desain**

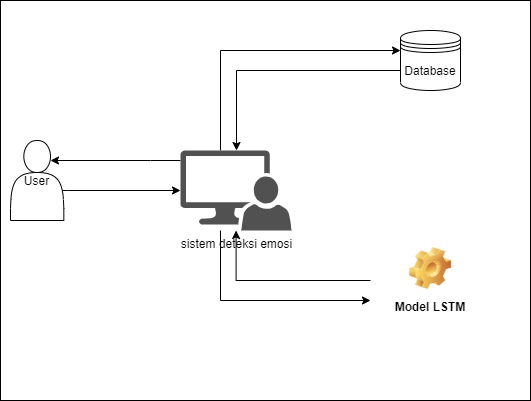
Proses desain akan menjelaskan proses perancangan sistem yang berguna untuk memberikan gambaran umum sisem sehingga mempermudah dalam membangun sistem. Proses ini terdiri dari perancangan sistem dan perancangan pengujian.

**3.2.5 Perancangan Sistem**

Perancangan sistem membahas mengenai rancangan sistem yang akan dibangun pada penelitian ini. Perancangan sistem terdiri dari perancangan arsitektur, perancangan proses, perancangan basis data, dan perancangan antar muka. Penjelasan lebih detail mengenai perancangan arsitektur, perancangan proses, perancangan basis data, dan perancangan antar muka sebagai berikut :

1. **Perancangan Arsitektur**

Arsitektur sistem yang akan dibuat terdiri dari *user*, *database*, model LSTM, dan sistem deteksi emosi. *User* merupakan aktor yang dapat menginput teks pada sistem untuk mendeteksi emosi. Sistem deteksi emosi memproses teks yang diinput oleh *user* menggunakan model LSTM dan menampilkan hasil kepada *user*. Hasil deteksi emosi akan disimpan didalam *database*. Ilustrasi arsitektur sistem yang akan dibuat dapat dilihat pada gambar 3.16.



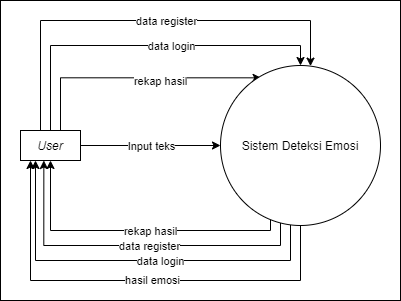
**Gambar 3.16 Arsitektur sistem**

1. **Perancangan proses**

Proses perancangan sistem pada penelitian ini menggunakan *Data Flow Diagram* (DFD) untuk menggambarkan aliran data pada sistem. DFD yang digunakan pada sistem ini sebagai berikut :

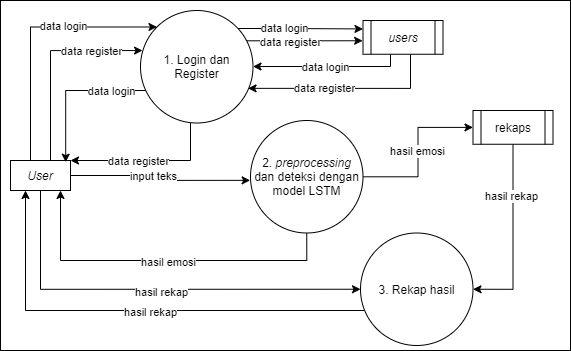
1. DFD level 0 (*Diagram Context*)

DFD level 0 atau *Diagram Context* menggambarkan sistem secara *basic* dan aliran *input* sampai *output* saja, oleh karena itu tidak ada *database*. Sistem ini hanya memiliki 1 entitas saja yaitu *user*. DFD level 0 pada sistem ini dapat dilihat pada gambar 3.17.

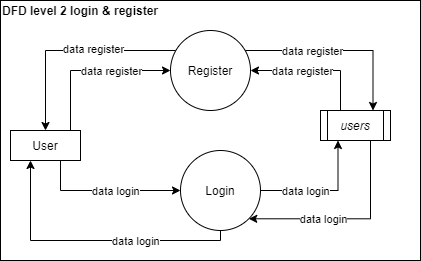
**Gambar 3.17 DFD level 0**

1. DFD level 1

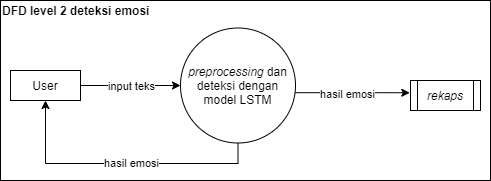
DFD level 1 menggambarkan aliran data yang lebih kompleks dari pada DFD level 1, dengan memecah beberapa proses pada DFD level 0. Sistem ini memilik 3 proses yaitu *login* dan *register*, *preprocessing* dan deteksi emosi dengan model LSTM, dan rekap hasil. DFD level 1 juga menggambarkan *database* yang akan digunakan sistem, pada sistem ini menggunakan 2 tabel *database* yaitu, *users* dan *rekaps*. DFD level 1 pada sistem ini dapat dilihat pada gambar 3.18.

**Gambar 3.18 DFD level 1**

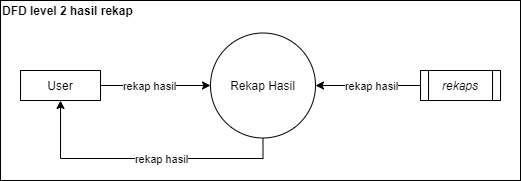
1. DFD level 2

DFD level 2 menguraikan proses-proses pada DFD level 1 menjadi lebih rinci, terdapat 3 proses yang diuraikan pada DFD level 2 sistem ini, yaitu proses *login* dan *register*, proses *preprocessing* dan deteksi dengan model LSTM, dan proses rekap hasil. Proses *login* dan *register* akan *store* data *login* dan data *register*  ke *data base users*. DFD level 2 proses *login* dan *register* dapat dilihat pada gambar 3.19.

**Gambar 3.19 DFD level 2 proses *login* dan *register***

DFD level 2 proses *preprocessing* dan deteksi emosi dengan model LSTM menggambarkan proses *preprocessing* dan deteksi emosi dengan model LSTM kemudian hasil emosi di *store* ke database *rekaps*. DFD level 2 proses *preprocessing* dan deteksi emosi dengan model LSTM dapat dilihat pada gambar 3.20.

**Gambar 3.20 DFD level 2 proses *preprocessing* dan deteksi emosi dengan model LSTM**

DFD level 2 proses rekap hasil menggambarkan proses *user request* rekap hasil, kemudian sistem mengambil data pada *data base rekaps* dan menampilkan ke *user*. DFD level 2 proses rekap hasil dapat dilihat pada gambar 3.21.

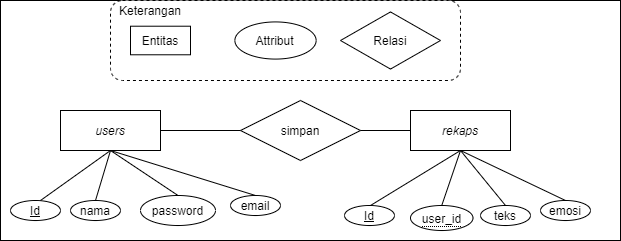
**Gambar 3.21 DFD level 2 proses rekap hasil**

1. **Perancangan *data base***

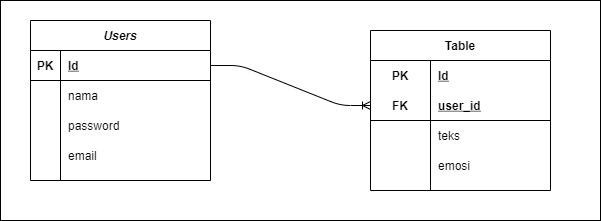
Perancangan *data base* beguna untuk menjabarkan *data base* yang digunakan pada sistem, sistem ini akan menggunakan *web server Apache* dan *Data base Management System* (DBMS) menggunakan *mySQL*. Perancangan *data base* menggunakan *Entity Relationship Diagram* (), Relasi Antar Tabel (RAT), dan Struktur Tabel. Penjabaran perancangan *data base* sebagai berikut :

1. *Entity Relationship Diagram* (ERD)

*Entity Relationship Diagram* (ERD) memvisualisasikan basis data saling terhubung yang dilambangkan dengan simbol-simbol. Simbol-simbol tersebut adalah entitas, atribut, dan relasi. Entitas pada perancangan *data base* sistem ini terdapat 2, yaitu *users* dan *rekaps*. Penjabaran ERD yang akan diterapkan pada sistem ini dapat dilihat pada gambar 3.22.

**Gambar 3.22 *Entity Relationship Diagram* (ERD)**

1. Relasi Antar Tabel (RAT)

Relasi Antar Tabel (RAT) merepresentasikan hubungan tabel satu ke tabel yang lain dengan menghubungkan *PRIMARY KEY* dengan *FOREIGN KEY*, RAT berguna untuk mengatur operasi pada suatu *data base*. RAT pada sistem yang akan dibangun dapat dilihat pada gambar 3.23.

**Gambar 3.23 Relasi Antar Tabel**

1. Struktur Tabel

Struktur tabel yang akan digunakan pada sistem ini sebagai berikut :

1. Tabel *Users*

Tabel *users* memiliki 4 kolom yaitu id, nama, password, email. Penjabaran mengenai tabel *users* dapat dilihat pada tabel 3.15.

**Tabel 3.15 Tabel *users***

|  |  |  |  |
| --- | --- | --- | --- |
| **Kolom** | **Tipe data** | ***Constraints*** | **Keterangan** |
| Id | *Integer*(11) | *Primary key* , *Auto Increment* | Id *user* |
| nama | *varchar*(191) |  | nama *user* |
| password | *varchar*(191) |  | Password *user* |
| email | *varchar*(191) |  | email *user* |

1. Tabel *Rekaps*

Tabel *rekaps* memiliki 4 kolom yaitu id, user\_id, teks, emosi. Penjabaran mengenai tabel *rekaps* dapat dilihat pada tabel 3.16.

**Tabel 3.16 Tabel *rekaps***

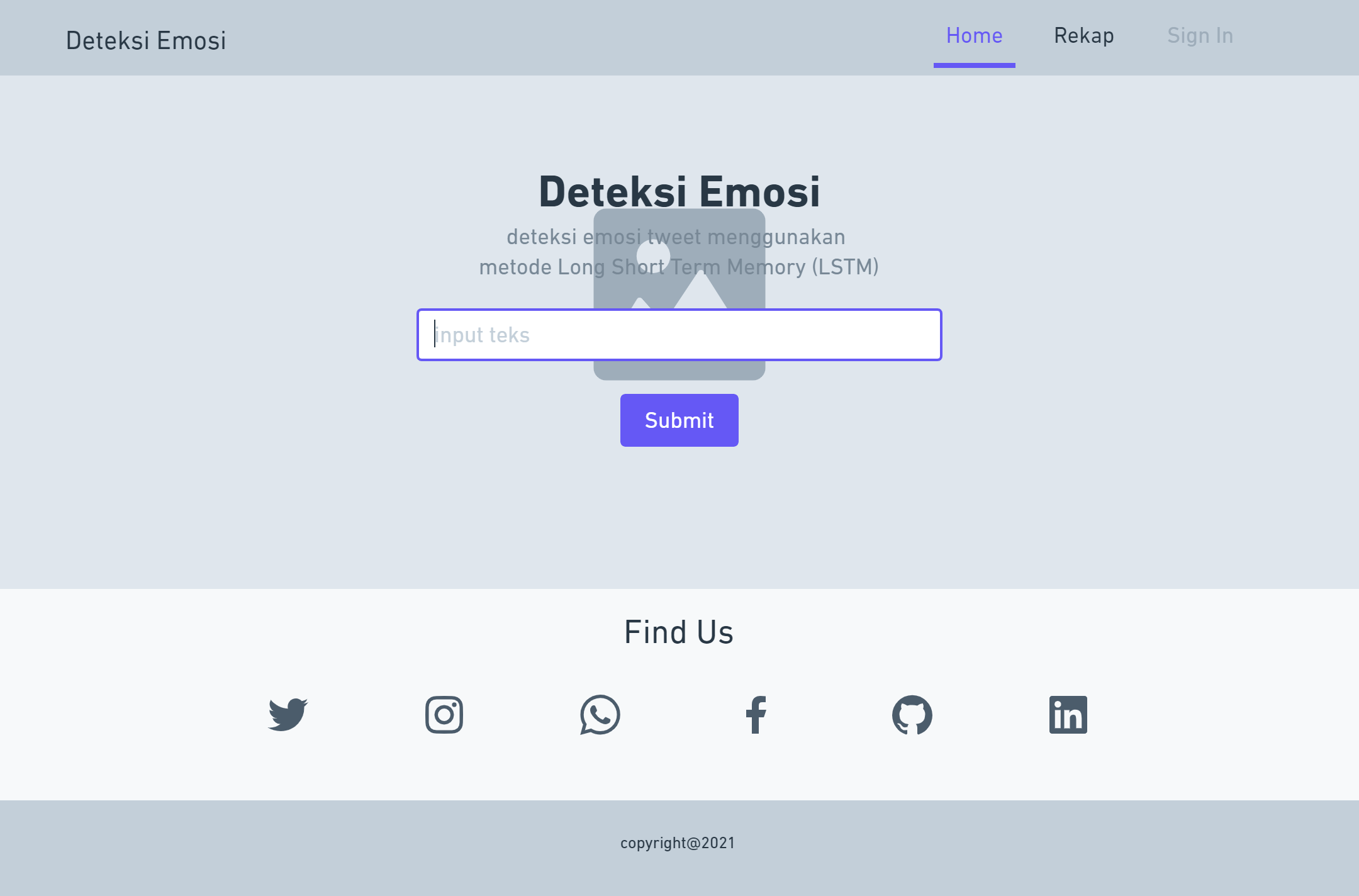
|  |  |  |  |
| --- | --- | --- | --- |
| **Kolom** | **Tipe data** | ***Constraints*** | **Keterangan** |
| Id | *Integer*(11) | *Primary key* , *Auto Increment* | Id *rekaps* |
| user\_id | *Integer*(11) | *Foreign key* | Id *user* |
| teks | *varchar*(191) |  | teks *rekaps* |
| emosi | *varchar*(191) |  | emosi *rekaps* |

1. **Perancangan antar muka**

Perancangan antar muka merupakan mekanisme komunikasi antara user dengan sistem. Perancangan antar muka yang akan diterapkan pada sistem ini terdiri dari beberapa bagian sebagai berikut :

* 1. Halaman Home

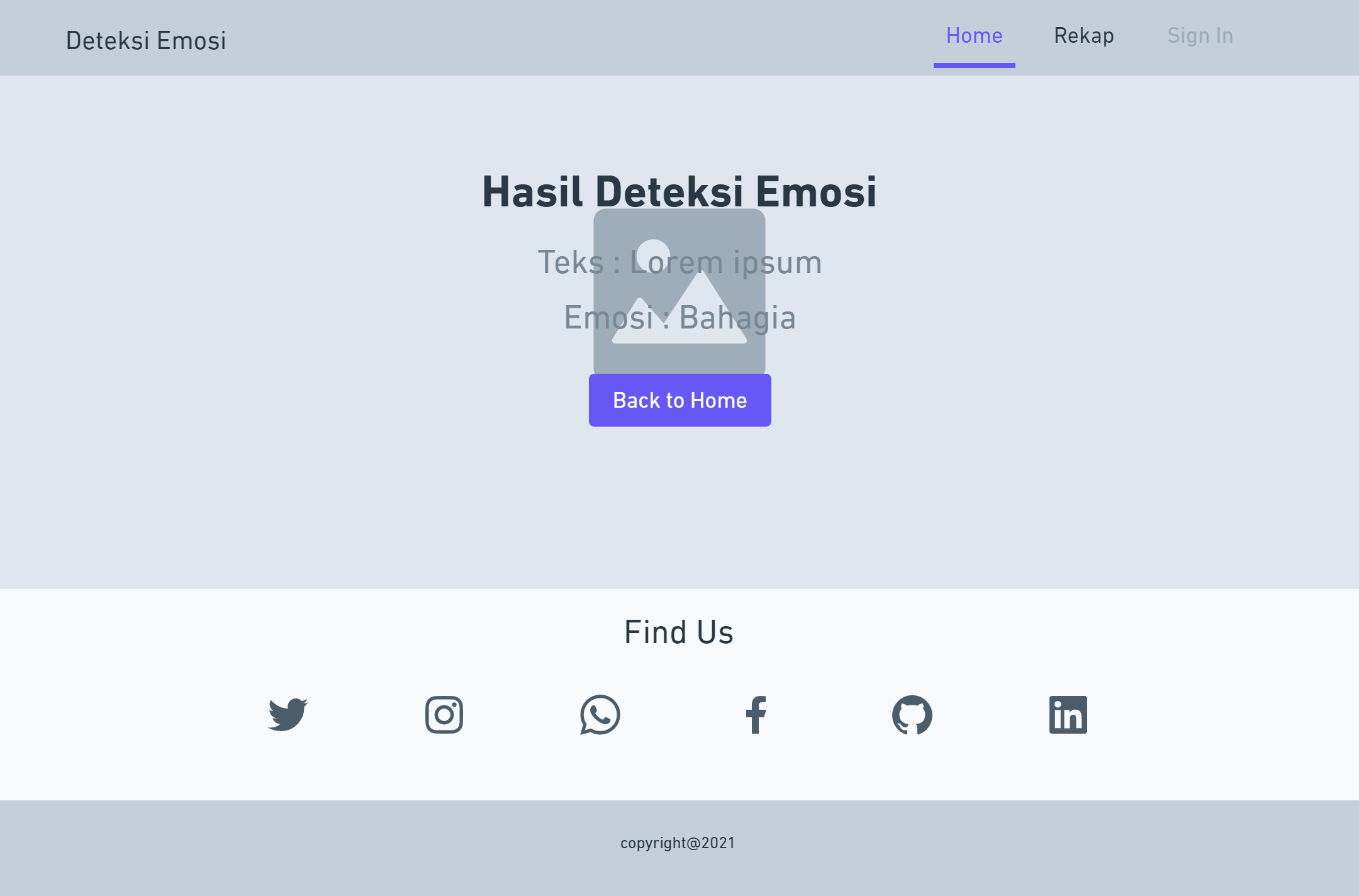
Halaman ini merupakan halaman yang akan diakses pertama kali oleh user, terdiri dari form untuk menginput teks dan button untuk mengetahui hasil emosi.



**Gambar 3.24 Halaman Home**

* 1. Halaman Hasil

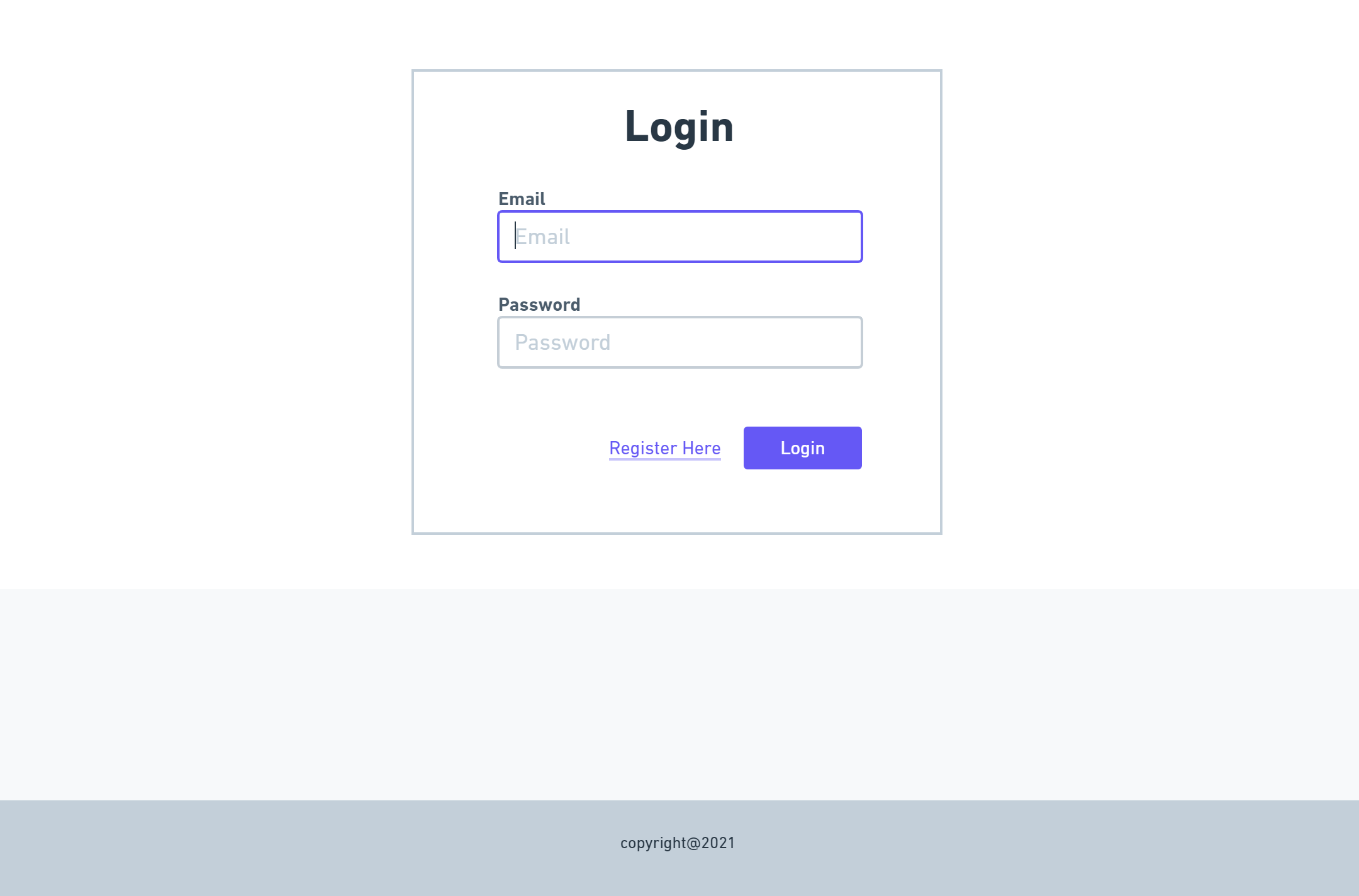
Halaman ini akan menampilkan teks yang diinput pada halaman home serta hasil emosi bedasarkan model LSTM.



**Gambar 3.25 Halaman Hasil**

* 1. Halaman Login

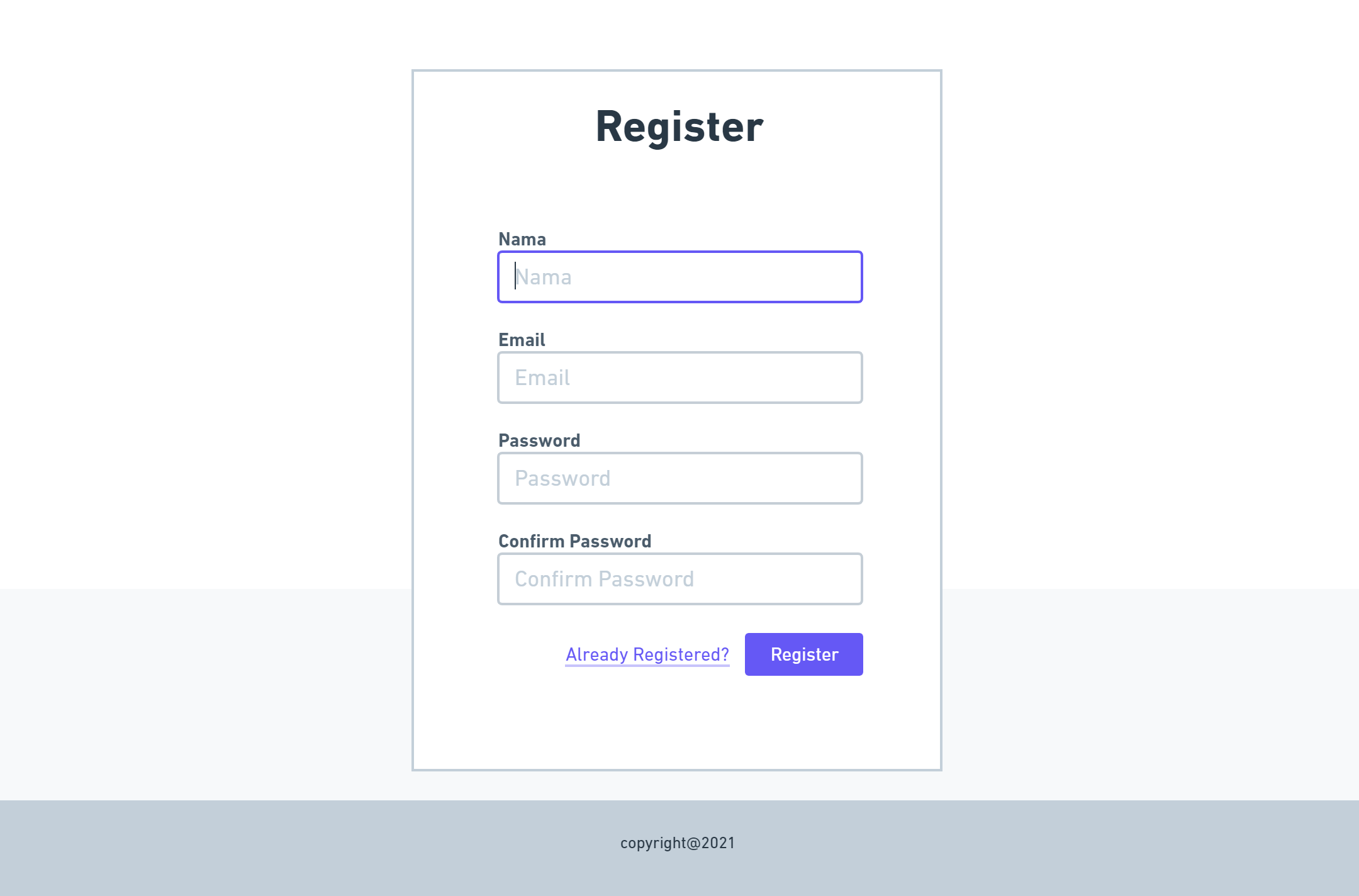
User dapat melakukan login jika sudah memiliki akun melalui halaman login dengan menginput email dan password dengan benar.



**Gambar 3.26 Halaman Login**

* 1. Halaman Register

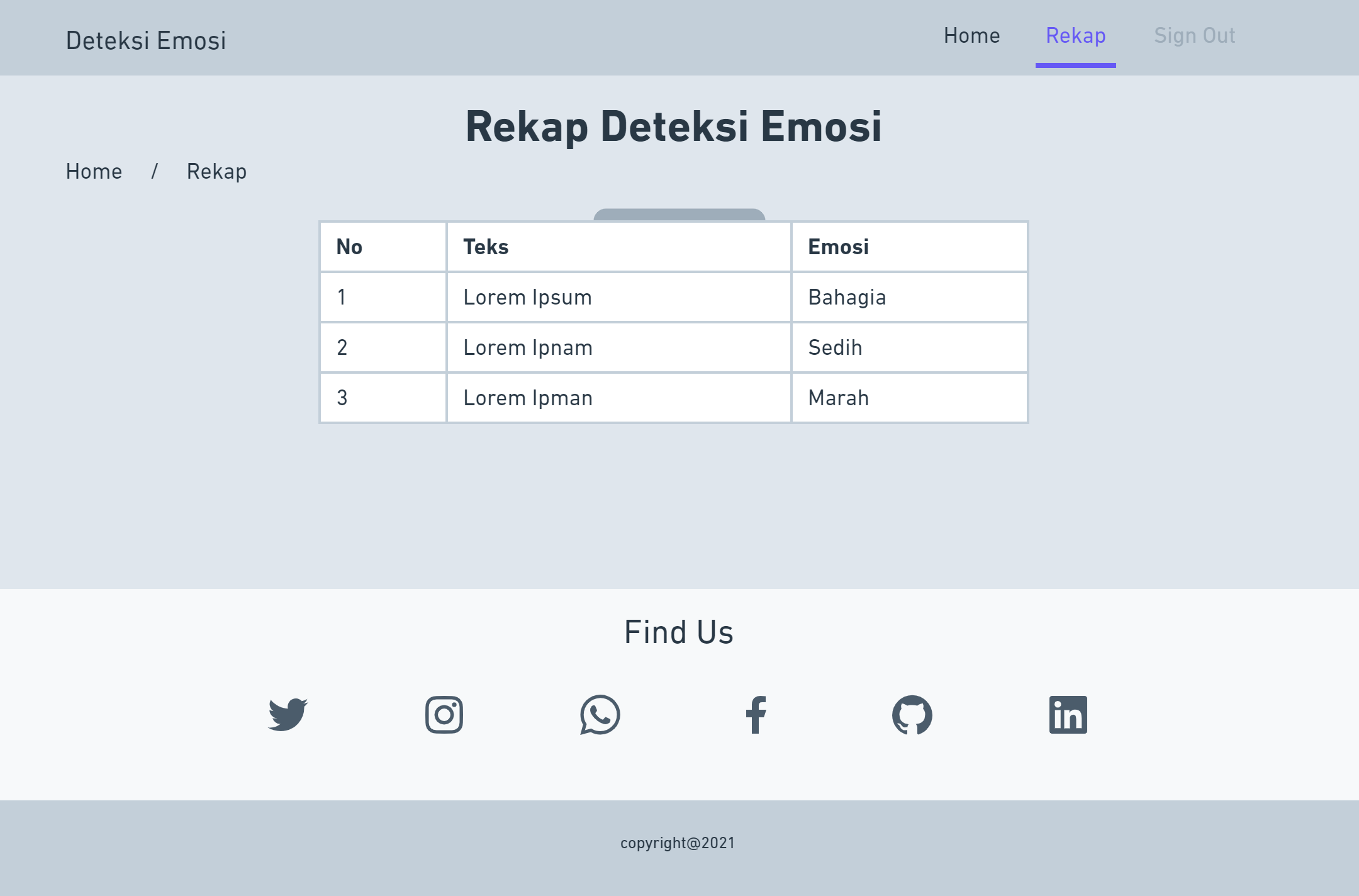
User yang belum memiliki akun dapat mendaftarkan akun terlebih dahulu melalui halaman register dengan menginput nama, email, password, dan konfirmasi password.



**Gambar 3.27 Halaman Register**

* 1. Halaman Rekap Hasil

User dapat melihat hasil rekap terhadap teks yang telah diinput dengan mengakses halaman rekap hasil, rekap hasil disajikan dalam bentuk tabel.



**Gambar 3.28 Halaman Hasil**

**3.2.6 Perancangan Pengujian**

Perancangan pengujian berguna untuk menguji sistem terhadap kesesuaian rancangan yang telah dibuat. Pengujian sistem yang digunakan pada penelitian ini adalah metode *black box testing*, yaitu pengujian yang menekankan fungsionalitas tanpa mengetahui *coding* dari sistem tersebut. *Black box testing* bertujuan untuk mengukur kinerja dari sistem yang telah dibangun. Detail perancangan pengujian sistem dapat dilihat pada tabel 3.17.

**Tabel 3.17 Detail perancangan pengujian sistem**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Halaman** | **Pengujian** | **Hasil** | |
| **Berhasil** | **Gagal** |
| 1. | Halaman utama | *User* dapat menginput teks |  |  |
| Sistem dapat melakukan validasi input |  |  |
| 2. | Halaman hasil | Sistem dapat menampilkan hasil deteksi emosi |  |  |
| 3. | Halaman login | *User* dapat melakukan login |  |  |
| Sistem dapat melakukan validasi input |  |  |
| 4. | Halaman Register | *User* dapat melakukan register |  |  |
| Sistem dapat melakukan validasi input |  |  |
| 5. | Halaman Rekap | Sistem dapat menampilkan hasil rekap |  |  |

**3.3 Jadwal Penelitian**

Jadwal penelitian yang dibuat sesuai dengan metodologi penelitian dan metodologi pengembangan sistem, jadwal penelitian lebih detail dapat dilihat pada tabel 3.18 dan tabel 3.19.

**Tabel 3.18 Jadwal penelitian**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Kegiatan** | **April** | | | | **Mei** | | | | **Juni** | | | | **Juli** | | | | **Agustus** | | | | **September** | | | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| 1 | **Pengumpulan data** | | | | | | | | | | | | | | | | | | | | | | | | |
| Scrapping *twitter* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | **Labelling** | | | | | | | | | | | | | | | | | | | | | | | | |
| Labelling Data |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | ***Preprocessing*** | | | | | | | | | | | | | | | | | | | | | | | | |
| *Case Folding* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Remove Punctuation* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Remove Number* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Tokenizing* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Stopword Removal* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Stemming* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Tabel 3.19 Jadwal Penelitian (lanjutan)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | **Pengujian Arsitektur** | | | | | | | | | | | | | | | | | | | | | | | | |
| *Word Embedding* (*word2vec*, *Glo-Ve*, *fasttext*) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Inisialiasi Unit |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LSTM |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | **Pengujian** | | | | | | | | | | | | | | | | | | | | | | | | |
| Pengujian *Confussion Matrix* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | **Analisis Kebutuhan Sistem** | | | | | | | | | | | | | | | | | | | | | | | | |
| Analisis kebutuhan sistem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | **Proses Desain** | | | | | | | | | | | | | | | | | | | | | | | | |
| Perancangan Arsitektur |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Perancangan Proses |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Perancangan Antar Muka |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | **Pengujian Sistem** | | | | | | | | | | | | | | | | | | | | | | | | |
| Perancangan Pengujian |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | ***Prototyping*** | | | | | | | | | | | | | | | | | | | | | | | | |
| *Prototyping* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | **Evaluasi** | | | | | | | | | | | | | | | | | | | | | | | | |
| Evaluasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Produk Akhir |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |