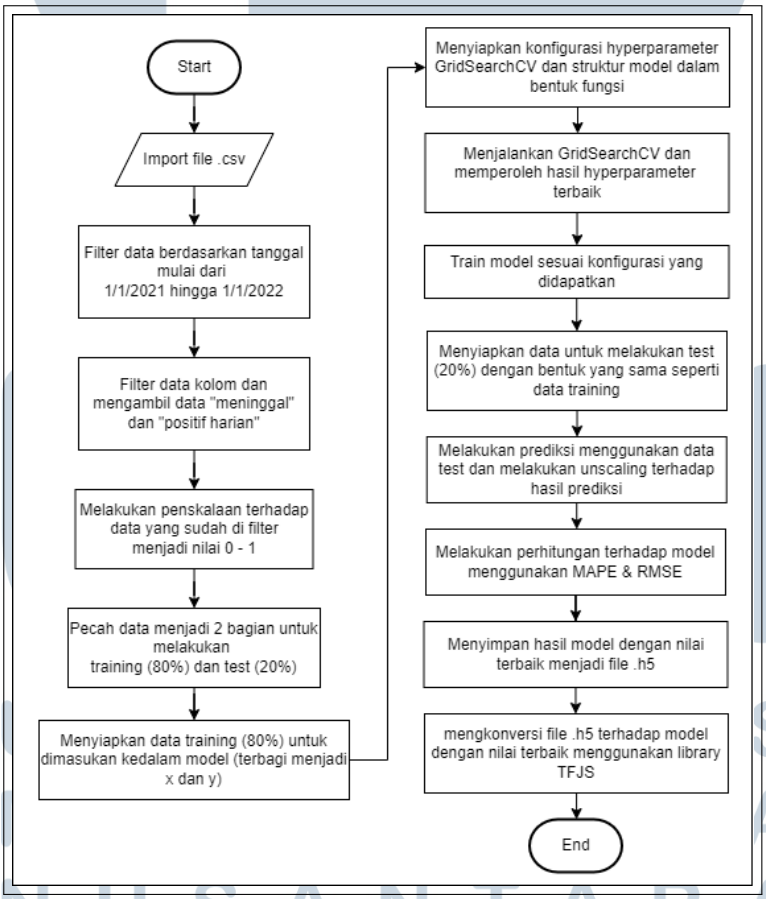
BAB 3
METODOLOGI PENELITIAN

3.1. Perancangan model prediksi

3.1.1. Flowchart Python

berikut merupakan alur dari pembuatan model menggunakan *Python*. Pembuatan diawali dengan mengumpulan dataset terlebih dahulu, kemudian melakukan normalisasi data, pembuatan model, pelatihan model, hingga pengujian model. Untuk rincian dari tahapan-tahapan yang dijalankan akan dijelaskan dari poin 3.1.2. hingga poin 3.1.5.



Gambar 3.1. Alur *flowchart* pembuatan dan pengujian model menggunakan *Python*.

3.1.2. Mengumpulkan dataset dan melakukan normalisasi data

Dataset digunakan sebagai bahan untuk melakukan prediksi terhadap kasus positif baru dan kasus kematian baru dimulai dari tanggal 1 Januari 2021 hingga 1 Januari 2022 dan terus diperbaharui setiap hari oleh pemerintah serta bisa diunduh melalui *website* resmi pemerintah secara langsung. Data yang telah disaring sesuai dengan tanggal dan kolom yang dibutuhkan akan langsung di normalisasi, tujuan dari normalisasi data adalah untuk melakukan proses penskalaan nilai atribut dari data sehingga bisa berada pada *range* tertentu [18]. Untuk melakukan normalisasi data, penulis akan menggunakan *MinMaxScaler* [19] dimulai dari skala 0 hingga 1.

Setelah itu data baru dipecah menjadi dua bagian sekitar 80% data digunakan untuk melakukan training pada model dan 20% lainnya digunakan untuk melakukan pengujian model.

Kemudian data tersebut dibagi menjadi dua bagian yaitu menjadi *x_train* dan *y_train* (x merupakan *input* variabel dan y merupakan *output* variabel yang ingin dicapai). Desain dari *input* dan *output* variabel yang ingin dibuat dapat dilihat pada gambar berikut ini.

```
# Contoh Pola yg ingin dibentuk:
#             (bagian X)                                  | (Bagian Y)
# [[[m1, p1], [m2, p2], [m3, p3], [m4, p4], [m5, p5]]] | [[m6, p6]]
# [[[m2, p2], [m3, p3], [m4, p4], [m5, p5], [m6, p6]]] | [[m7, p7]]
# [[[m3, p3], [m4, p4], [m5, p5], [m6, p6], [m7, p7]]] | [[m8, p8]]
# dst...
```

Gambar 3.2. Desain *input* dan *output* sebagai bahan *training* pada model.

3.1.3. Membuat Model

Pada tahap ini pembuatan model akan menggunakan Struktur model dengan menggunakan dua *layer* LSTM dan menggunakan 1 *dropout* yang akan dimasukan kedalam sebuah fungsi seperti contoh berikut ini.

```python
def define_model_grid (dropout_rate, first_neuron, second_neuron):
    model_gscv = Sequential()
    model_gscv.add(LSTM(first_neuron, input_shape=(x_train.shape[1], x_train.shape[2]), return_sequences=True))
    model_gscv.add(Dropout(dropout_rate))
    model_gscv.add(LSTM(second_neuron, return_sequences=False))
    model_gscv.add(Dense(y_train.shape[1]))
    model_gscv.compile(optimizer='adam', loss='mean_squared_error', metrics=['acc'])
    # model_gscv.summary()

    return model_gscv
```

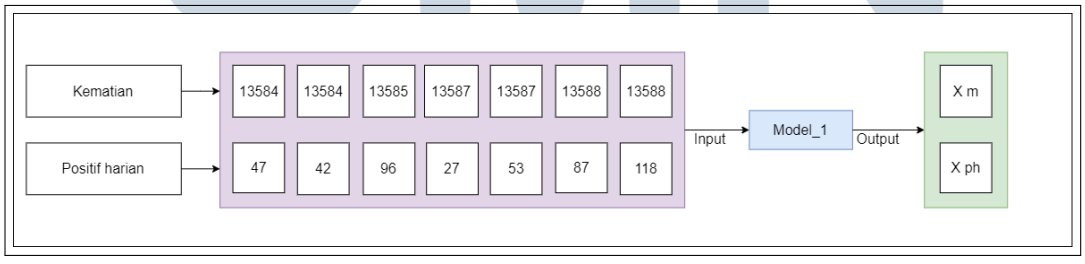
Gambar 3.3. Struktur model yang digunakan.

Selain itu terdapat konfigurasi *hyperparameter* yang sudah ditentukan dengan tujuan untuk dimasukan kedalam fungsi *GridSearchCV* yang pada dasarnya menghitung semua kombinasi dalam menemukan parameter terbaik [20]. Konfigurasinya dapat dilihat pada gambar berikut ini.

```
dropout_rate = [0.2, 0.3, 0.4]
first_neuron = [128, 64, 32]
second_neuron = [128, 64, 32]
batch_size = [10, 20, 30]
epoch = [20, 25]
```

Gambar 3.4. Konfigurasi *hyperparameter*.

Data yang akan dipelajari merupakan input *multivariate* yang berarti data yang diamati lebih dari satu data, dalam kasus kali ini data yang diamati adalah data kasus positif harian dan meninggal. Untuk ilustrasi dari model yang dibuat dapat dilihat melalui gambar berikut ini.



Gambar 3.5. Ilustrasi proses prediksi model multivariate.

Tujuan dari penggunaan *multivariate* pada prediksi ini adalah untuk membuat waktu prediksi menjadi lebih efisien dalam memproses data se-

hingga tidak banyak waktu yang terbuang untuk melakukan prediksi satu per satu.

3.1.4. Training Model

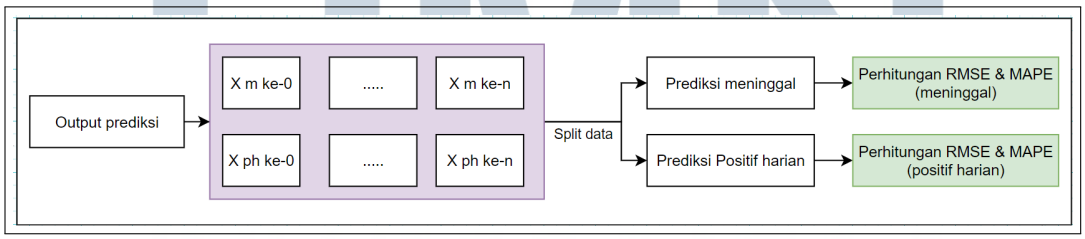
Setelah data di normalisasi dan model sudah dibentuk, berikutnya data akan digunakan untuk melakukan training model hingga selesai. Konfigurasi *batch_size* dan *epoch* akan menggunakan nilai yang didapat pada *GridSearchCV*.

3.1.5. Pengujian model

Pada tahap ini, model yang sudah melalui training akan diuji untuk dilihat keakuratannya dalam melakukan prediksi menggunakan RMSE dan MAPE.

Root Mean Square Error atau RMSE digunakan untuk memperkirakan keakuratan nilai prediksi dari model dengan nilai aktual atau yang diamati [15]. Sementara *Mean Absolute Percentage Error* atau MAPE digunakan untuk mengukur persentase kesalahan antara nilai prediksi dengan nilai sesungguhnya.

Perhitungan RMSE dan MAPE pada model akan dilakukan secara terpisah sesuai dengan kategori masing-masing (positif harian dan meninggal) agar dapat mengamati skor prediksi yang lebih rinci. Berikut ilustrasi dari perhitungan yang dilakukan.



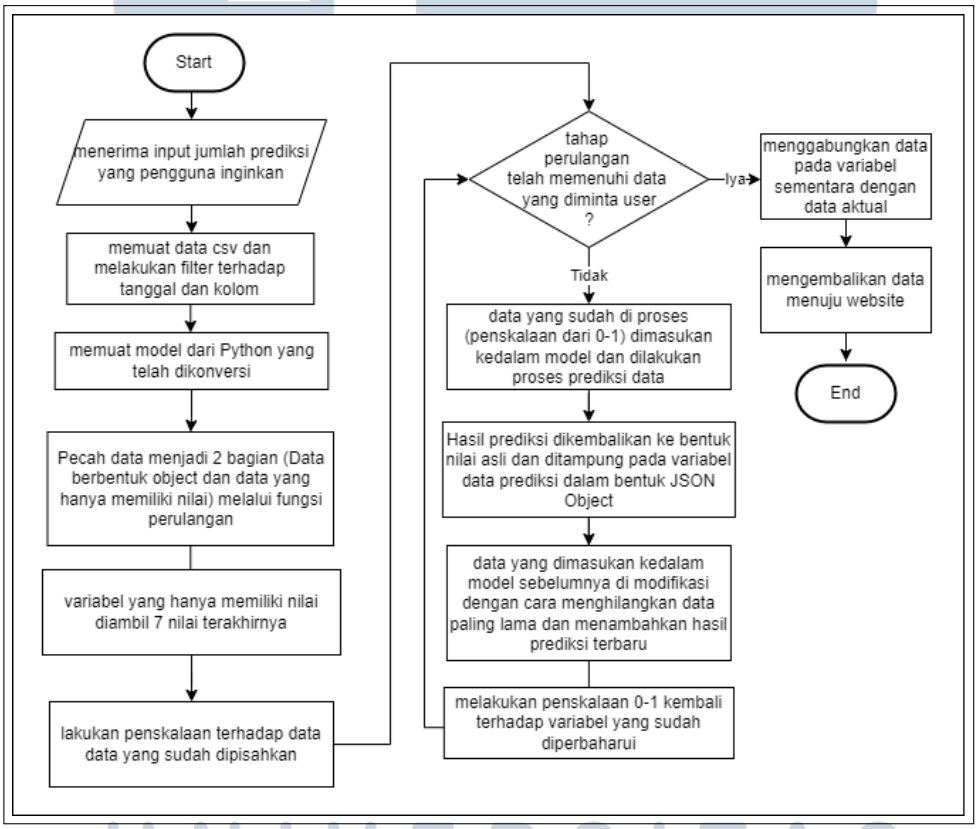
Gambar 3.6. Ilustrasi proses perhitungan RMSE dan MAPE pada model.

Pengujian model akan dilakukan sebanyak 20 kali dan akan diambil hasil terbaik dari model dan disimpan dalam bentuk .h5 dan model terbaik akan dikonversikan menjadi file .JSON dengan memanfaatkan *library TFJS* [21] agar model dapat digunakan di *Node.JS* sebagai *backend website*.

3.2. Perancangan backend website

3.2.1. Flowchart backend Node.JS

Berikut ini merupakan rancangan alur dari proses melakukan prediksi *Covid-19* yang dimulai dari memuat data *csv*, melakukan filter terhadap data berdasarkan tanggal dan kolom, memuat model, melakukan prediksi dan mengembalikan hasil akhir menuju *website*. Untuk rincian dari tahapan-tahapan yang dijalankan akan dijelaskan dari poin 3.2.2. hingga poin 3.2.5.



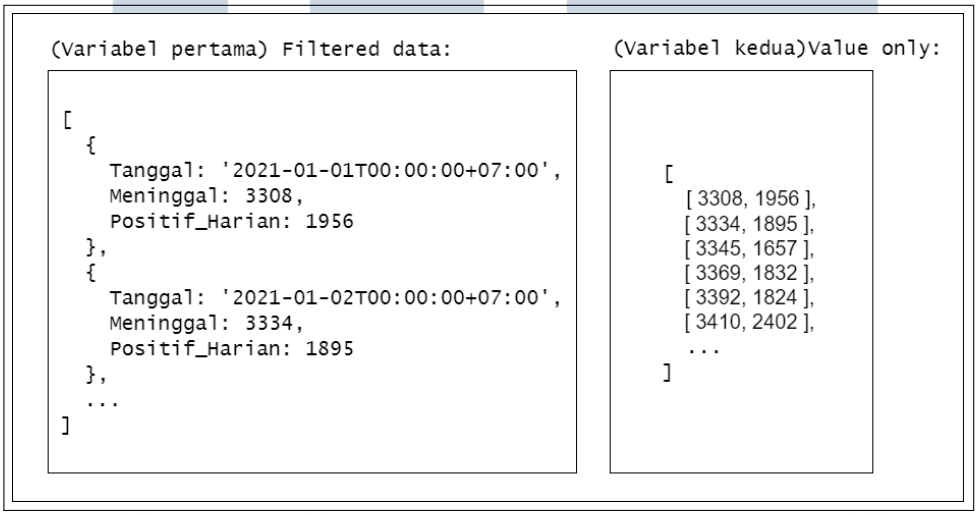
Gambar 3.7. Alur *flowchart website* prediksi *Covid-19* pada sisi *backend*.

3.2.2. Memuat data csv dan melakukan melakukan filter

Pada tahap ini program akan melakukan import terhadap data menggunakan fungsi *createReadStream* yang berfungsi untuk membuka dan membaca data *covid-19* serta menggunakan *library csv-parser* agar data csv dapat dibaca dan diubah menjadi *JSON OBject*. Setelah itu data akan di

filter berdasarkan tanggal mulai dari dari 1 januari 2021 hingga 1 januari 2022 dan melakukan filter berdasarkan kolom yang dibutuhkan yaitu kolom ”Positif Harian” dan ”Meinggal”.

Hasil filter akan dimasukan kedalam dua buah variabel, satu untuk menampung *JSON Object* dan menampung nilai positif harian dan meninggal dalam bentuk *array*. variabel pertama berfungsi sebagai tempat untuk menampung data aktual dari tanggal yang sudah ditentukan, dan variabel kedua yang hanya menampung nilai akan digunakan dalam proses melakukan prediksi. berikut ilustrasi dari kedua variabel yang ingin dibuat.



Gambar 3.8. Ilustrasi dari masing-masing variabel.

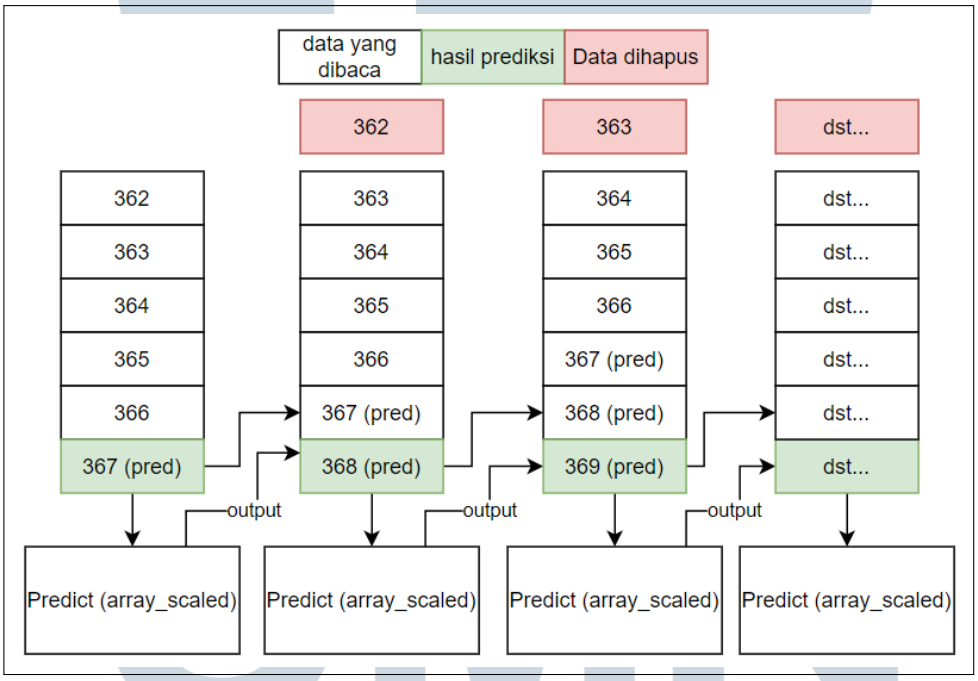
Data pada variabel kedua akan diambil 7 hari terbaru dan akan dilakukan *scaling* dalam skala nol hingga satu, setelah data *discaling* maka data akan dimasukan kedalam prediksi.

3.2.3. Memuat model prediksi

Pada tahap ini Model yang telah dikonversikan menjadi bentuk *JSON* akan dimuat kedalam javascript menggunakan bantuan *library* TFJS melalui fungsi *loadLayerModel*. Variabel yang akan dimasukan kedalam moodel perlu dikonversikan terlebih dahulu melalui TFJS dengan fungsi *tensor3d* yang berfungsi untuk membuat data tensor dalam bentuk 3 dimensi yang baru [22].

3.2.4. Proses perulangan prediksi

Fungsi prediksi melalui proses perulangan digunakan untuk melakukan prediksi di hari berikutnya. Setelah data dikonversikan dan diubah menggunakan *library* TFJS maka data akan dimasukan kedalam model. Model akan melakukan prediksi dan menampung hasilnya kedalam variabel sementara dalam bentuk *JSON Object*. Selain itu data akan kembali dimasukan kedalam variabel kedua yang kemudian akan dilakukan *scaling* kembali dengan skala nol hingga satu. Proses dilakukan berulang-ulang hingga mencapai batas tanggal yang diinginkan pengguna. Untuk gambaran dari mekanisme prediksi dapat dilihat melalui ilustrasi berikut ini.



Gambar 3.9. Mekanisme dalam melakukan prediksi pada *backend website*.

3.2.5. Mengembalikan hasil akhir keluaran

Terakhir data prediksi yang telah diubah menjadi *JSON Object* akan digabungkan dengan data aktual dan dikirimkan kembali menuju ke *frontend*. Hasil akhir dari data yang telah di prediksi akan dibentuk menjadi seperti contoh gambar berikut.

```
[
    ...
    {
      "Tanggal": "2022-01-01T00:00:00+07:00",
      "Meninggal": 13588,
      "Positif_Harian": 118,
      "key": 365,
      "unix": 1640970000000
    },
    {
      "Tanggal": "2022-01-02T00:00:00+07:00",
      "Meninggal": 13585,
      "Positif_Harian": 70,
      "key": 366,
      "unix": 1641056400000,
      "isPredicted": true
    }
]
```


Gambar 3.10. Hasil keluaran dari backend setelah melakukan prediksi.

Masing-masing properti memiliki fungsinya sendiri yang akan dimanfaatkan pada halam *frontend*. Berikut penjelasan kegunaan masing-masing properti.

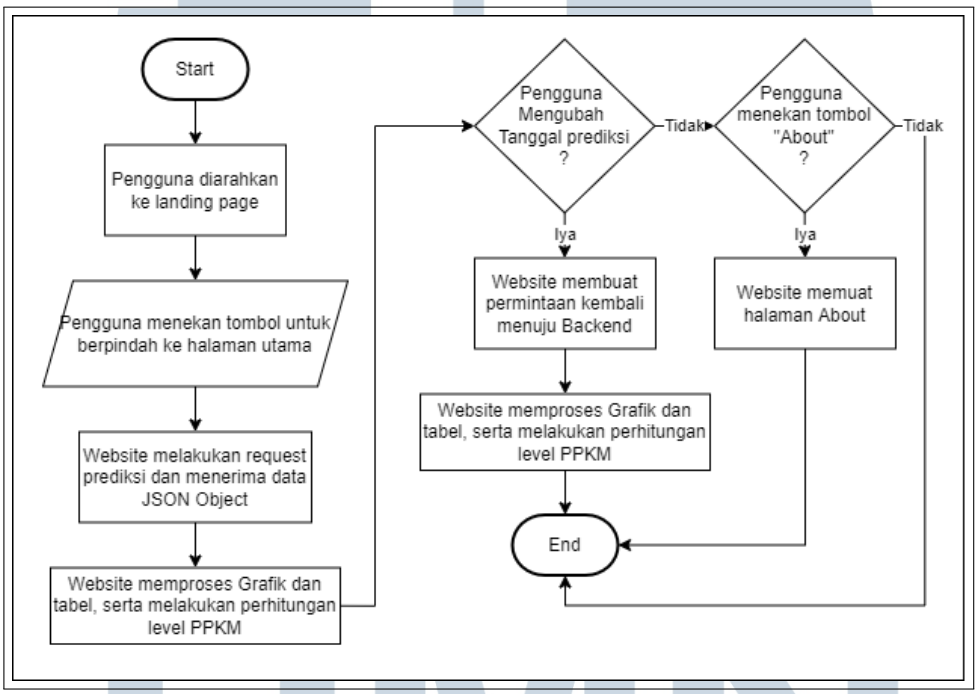
- Tanggal : digunakan pada tabel untuk menunjukan informasi tanggal kasus
- Meninggal : merupakan informasi data meninggal
- Positif_harian : merupakan informasi data positif harian
- *key* : sebagai *index* dalam tabel yang akan dimuat pada *website*
- *unix* : sebagai satuan waktu yang digunakan pada grafik *time series*
- *isPredicted* : sebagai tanda data pada *JSON Object* ini sebagai data prediksi, bukan data aktual

Data-data ini dapat dipecah dan dipilih propertinya untuk ditampilkan di halaman tertentu, sehingga data positif harian dan data meninggal dapat dipisah menjadi halaman yang berbeda.

3.3. Membangun website sistem prediksi menggunakan React.JS

3.3.1. Flowchart website model prediksi

Berikut merupakan alur dari *website* dari *landing page* hingga *about page*. Mulai dari memasuki halaman landing, proses melakukan prediksi pada *Main Page*, metode perhitungan level PPKM dan navigasi menuju halaman *About page*.

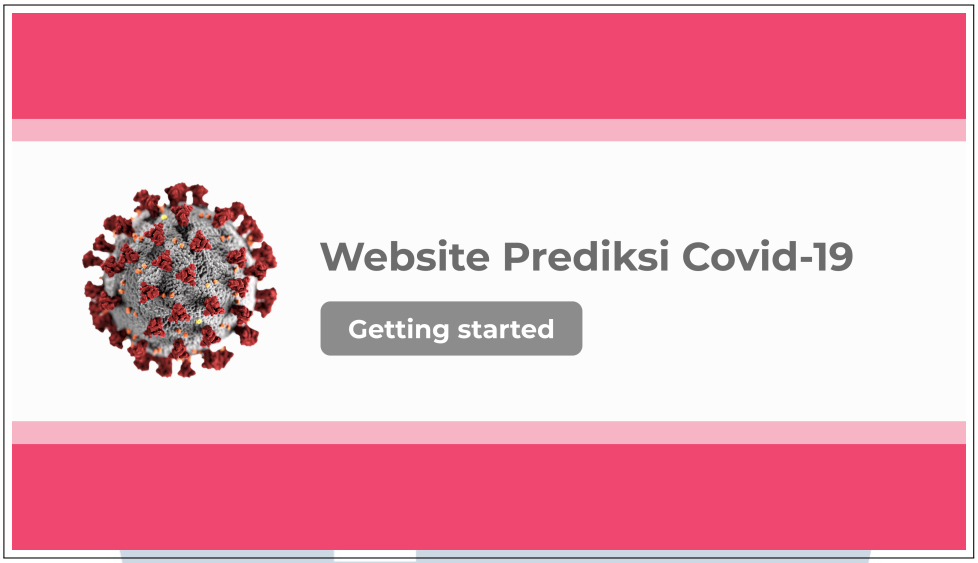


Gambar 3.11. Alur *flowchart website* prediksi *Covid-19*

3.3.2. Rancangan tampilan website

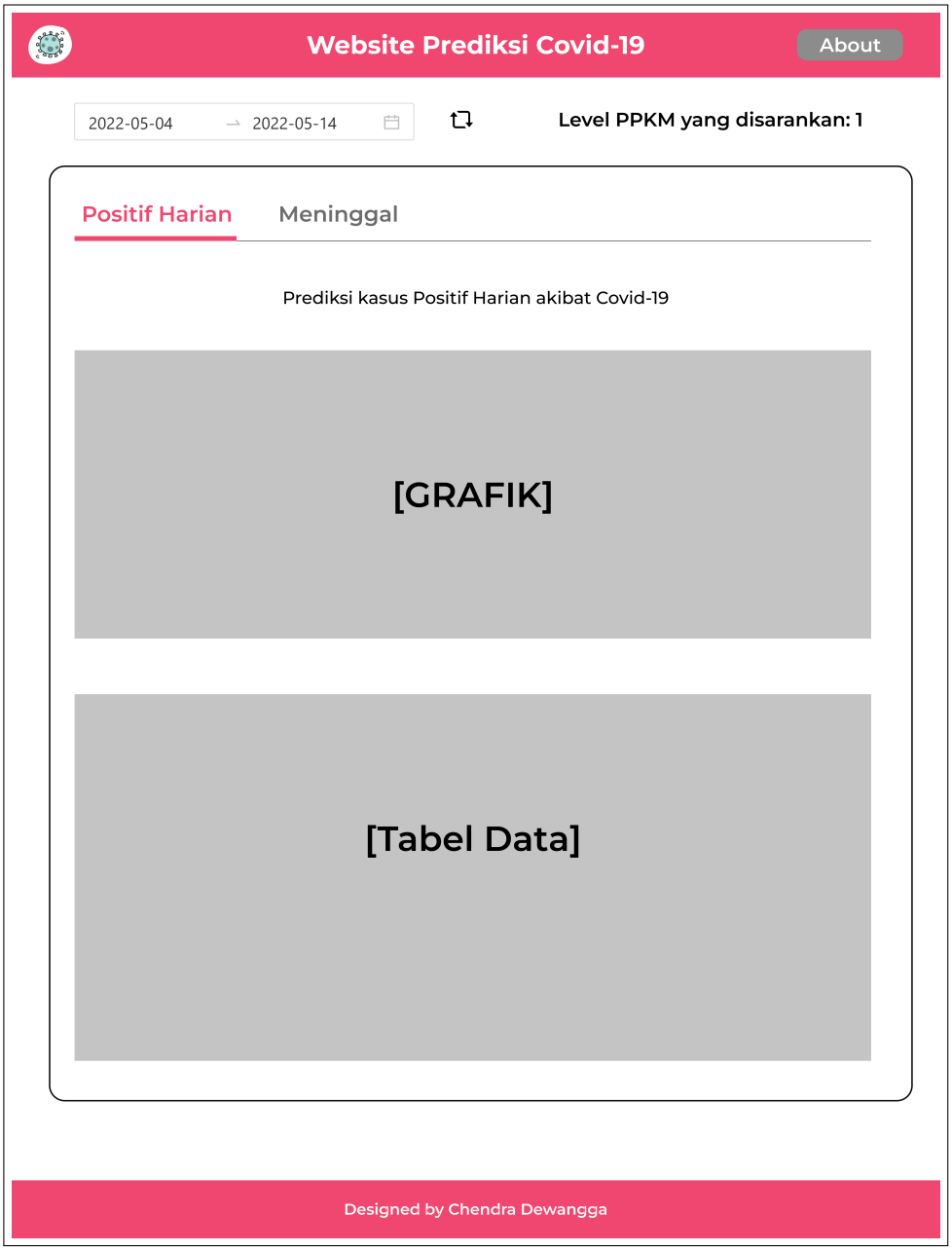
Rancangan antarmuka website akan dibagi menjadi 3 bagian yaitu *Landing page, Main page* dan *About Page*. Setiap halaman memiliki fungsi masing-masing.

Halaman *Landing page*, sebagai halaman yang pertama kali dilihat oleh pengguna ketika mengakses *website*.



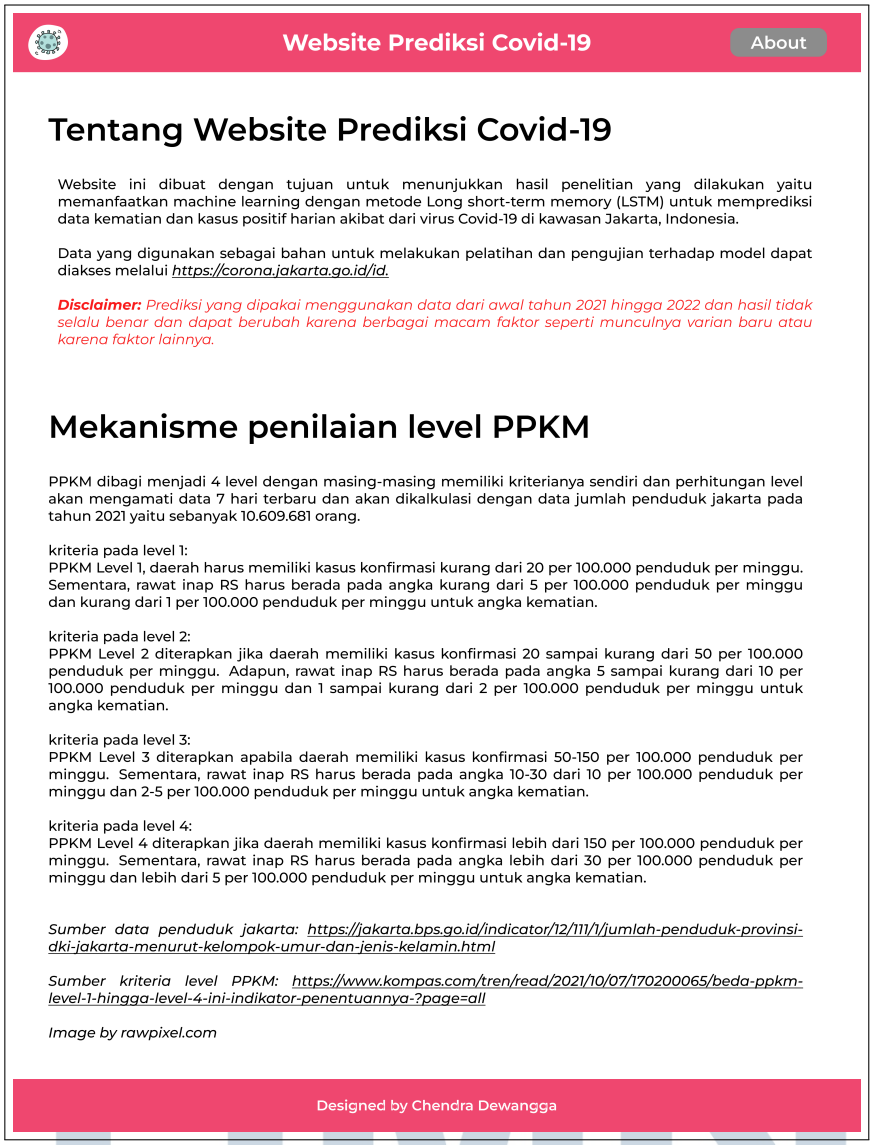
Gambar 3.12. Halaman *Landing page*.

Halaman *Main page*, sebagai halaman utama yang digunakan oleh pengguna untuk melakukan prediksi dan melihat rekomendasi level PPKM yang disarankan. Terdapat bagian khusus untuk melakukan input tanggal. Tanggal awal merupakan tanggal yang tidak dapat diubah yang memiliki tanggal 2 Januari 2022 dan tanggal akhir merupakan data tanggal yang dapat dipilih oleh pengguna. Disertai juga dengan tombol *refresh* untuk melakukan *request* menuju *backend* ketika pengguna menekan tombol tersebut. Selain itu terdapat tombol "*About*" pada kanan atas halaman untuk melakukan navigasi menuju *About page*.



Gambar 3.13. Halaman *Main page*.

Halaman *About page*, sebagai halaman yang digunakan untuk menjelaskan mekanisme dari prediksi *website* serta bagaimana cara perhitungan terhadap level PPKM.



Gambar 3.14. Halaman *About page*.

3.3.3. Metode perhitungan level PPKM

Setelah backend selesai dibuat, berikutnya adalah pembuatan *website frontend* yang akan menampilkan data prediksi *Covid-19* menggunakan *React.JS*, data yang ditampilkan akan dibagi menjadi 2 bagian yaitu data dalam bentuk grafik dan bentuk tabel serta perhitungan khusus terhadap level PPKM yang akan menghitung data kematian dan positif harian pada 7 hari terbaru dengan mengikuti persyaratan level PPKM berdasarkan sumber yang didapatkan dari kompas.com [9] untuk rumus perhitungannya berlaku setiap level dengan jumlah populasi penduduk Jakarta pada tahun 2021 yang

mencapai 10.609.681 orang [23] adalah sebagai berikut.

Rumus:

$$m = max - min \tag{3.1}$$

- m = hasil perhitungan data kasus meninggal
- max = data kasus meninggal tertinggi dalam 1 minggu
- min = data kasus meninggal terendah dalam 1 minggu

$$ph = \sum data \tag{3.2}$$

- ph = hasil perhitungan data kasus positif harian
- $\sum data$ = data positif harian selama 7 hari yang dijumlahkan

$$X_m = m/populasi * 100.000 \tag{3.3}$$

- X_m = hasil akhir perhitungan kasus meninggal
- m = merupakan data kasus meninggal
- populasi = data populasi penduduk Jakarta tahun 2021

$$X_{ph} = ph/populasi * 100.000 \tag{3.4}$$

- X_{ph} = hasil akhir perhitungan kasus positif harian
- ph = merupakan data kasus positif harian
- populasi = data populasi penduduk Jakarta tahun 2021