# BAB V. IMPLEMENTASI DAN PENGUJIAN

## 4.1 Implementasi Database

Implementasi database yang telah dibuat sebelumnya dan databaset pada aplikasi ini diberi nama '*skripsi_aircraft'*. Berikut ini adalah hasil implementasi dari perancangan sebelumnya:

### 4.1.1 Implementasi Tabel Jenis Sayap

Tabel *jenis_sayap* yang sebelumnya dirancang pada Tabel 4.3 yang diimplementasikan pada Gambar 5.1.

| | # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | id | int(11) | | | No | *None* | | AUTO_INCREMENT | Change Drop More |
| ☐ | 2 | features | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| ☐ | 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| ☐ | 4 | image | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |

Gambar 5.1. *Screenshot* Tabel jenis_sayap

### 4.1.2 Implementasi Tabel Penempatan Sayap

Tabel *penempatan_sayap* yang sebelumnya dirancang pada Tabel 4.4 yang diimplementasikan pada Gambar 5.2.

| | # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | id | int(11) | | | No | *None* | | AUTO_INCREMENT | Change Drop More |
| ☐ | 2 | features | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| ☐ | 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| ☐ | 4 | image | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |

Gambar 5.2. *Screenshot* Tabel penempatan_sayap

### 4.1.3 Implementasi Tabel Jumlah Sayap

Tabel *jumlah_sayap* yang sebelumnya dirancang pada Tabel 4.5 yang diimplementasikan pada Gambar 5.3.

| | # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | id | int(11) | | | No | *None* | | AUTO_INCREMENT | Change Drop More |
| ☐ | 2 | features | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| ☐ | 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |
| ☐ | 4 | image | varchar(255) | utf8mb4_general_ci | | No | *None* | | | Change Drop More |

Gambar 5.3. *Screenshot* Tabel jumlah_sayap

### 4.1.4 Implementasi Tabel Arah Sayap

Tabel *arah_sayap* yang sebelumnya dirancang pada Tabel 4.6 yang diimplementasikan pada Gambar 5.4.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.4. *Screenshot* Tabel arah_sayap

### 4.1.5 Implementasi Tabel Jenis Mesin

Tabel *jenis_mesin* yang sebelumnya dirancang pada Tabel 4.7 yang diimplementasikan pada Gambar 5.5.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.5. *Screenshot* Tabel jenis_mesin

### 4.1.6 Implementasi Tabel Jumlah Mesin

Tabel *jumlah_mesin* yang sebelumnya dirancang pada Tabel 4.8 yang diimplementasikan pada Gambar 5.6.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.6. *Screenshot* Tabel jumlah_mesin

### 4.1.7 Implementasi Tabel Posisi Mesin

Tabel *posisi_mesin* yang sebelumnya dirancang pada Tabel 4.9 yang diimplementasikan pada Gambar 5.7.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.7. *Screenshot* Tabel jumlah_mesin

### 4.1.8 Implementasi Tabel Badan Pesawat

Tabel *badan_pesawat* yang sebelumnya dirancang pada Tabel 4.10 yang diimplementasikan pada Gambar 5.8.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.8. *Screenshot* Tabel badan_pesawat

### 4.1.9 Implementasi Tabel Bentuk Ekor

Tabel *bentuk_ekor* yang sebelumnya dirancang pada Tabel 4.11 yang diimplementasikan pada Gambar 5.9.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.9. *Screenshot* Tabel bentuk_ekor

### 4.1.10 Implementasi Tabel *Landing Gear*

Tabel *landing_gear* yang sebelumnya dirancang pada Tabel 4.12 yang diimplementasikan pada Gambar 5.10.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.10. *Screenshot* Tabel landing_gear

### 4.1.11 Implementasi Tabel *Canard*

Tabel *canard* yang sebelumnya dirancang pada Tabel 4.13 yang diimplementasikan pada Gambar 5.11.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.11. *Screenshot* Table canard

### 4.1.12 Implementasi Tabel Persenjataan

Tabel *persenjataan* yang sebelumnya dirancang pada Tabel 4.14 yang diimplementasikan pada Gambar 5.12

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.12. *Screenshot* Tabel persenjataan

### 4.1.13 ImplementasiTabel Warna Pesawat

Tabel *warna_pesawat* yang sebelumnya dirancang pada Tabel 4.15 yang diimplementasikan pada Gambar 5.13

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 3 | binary_fusi | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 4 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.13. *Screenshot* Tabel warna_pesawat

### 4.1.14 Implementasi Tabel *Dataset*

Tabel *dataset* yang sebelumnya dirancang pada Tabel 4.16 yang diimplementasikan pada Gambar 5.14.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | jenis_sayap | int(255) | | | No | None | | | Change Drop More |
| 3 | penempatan_sayap | int(255) | | | No | None | | | Change Drop More |
| 4 | jumlah_sayap | int(255) | | | No | None | | | Change Drop More |
| 5 | arah_sayap | int(255) | | | No | None | | | Change Drop More |
| 6 | jenis_mesin | int(255) | | | No | None | | | Change Drop More |
| 7 | jumlah_mesin | int(255) | | | No | None | | | Change Drop More |
| 8 | posisi_mesin | int(255) | | | No | None | | | Change Drop More |
| 9 | badan_pesawat | int(255) | | | No | None | | | Change Drop More |
| 10 | jenis_ekor | int(255) | | | No | None | | | Change Drop More |
| 11 | landing_gear | int(255) | | | No | None | | | Change Drop More |
| 12 | canard | int(255) | | | No | None | | | Change Drop More |
| 13 | persenjataan | int(255) | | | No | None | | | Change Drop More |
| 14 | warna | int(255) | | | No | None | | | Change Drop More |
| 15 | jenis_pesawat | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| 16 | nama_pesawat | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.14. *Screenshot* Tabel dataset

### 4.1.15 Implementasi Tabel *Separate Fusi Dataset*

Tabel *separate_fusi_dataset* yang sebelumnya dirancang pada Tabel 4.17 yang diimplementasikan pada Gambar 5.15.

| | # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| ☐ | 2 | class | int(11) | | | No | None | | | Change Drop More |
| ☐ | 3 | wings_features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| ☐ | 4 | engine_features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| ☐ | 5 | fuselage_features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| ☐ | 6 | tail_features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| ☐ | 7 | add_features | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| ☐ | 8 | tipe_pesawat | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| ☐ | 9 | image | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |
| ☐ | 10 | buatan | varchar(255) | utf8mb4_general_ci | | No | None | | | Change Drop More |

Gambar 5.15. *Screenshot* Tabel separate_fusi_dataset

## 4.2 Implementasi *Design Interface*

Implementasi *design interface* dari aplikasi ini yang telah dirancang sebelumnya. Berikut adalah masing - masing implementasi dari *mockup design interface* yang telah dibuat sebelumnya:

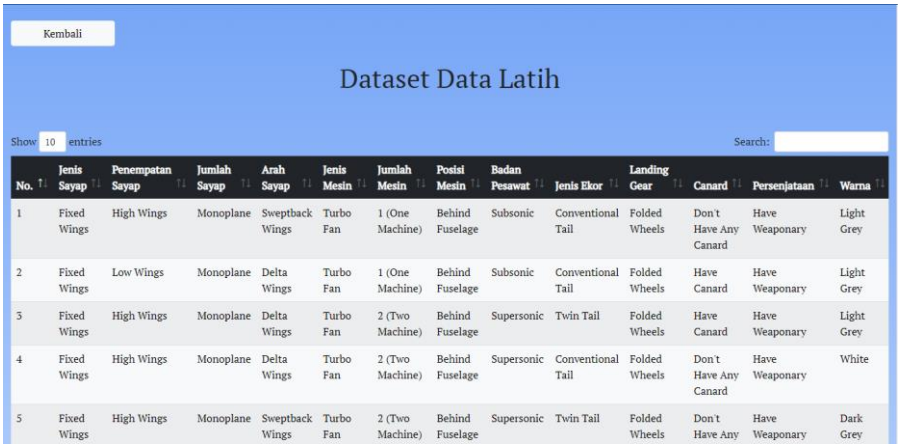
### 4.2.1 *Interface* Halaman Utama

Implementasi *mockup* yang telah dirancang sebelumnya pada Gambar 4.17 dan dapat dilihat pada Gambar 5.16.


Gambar 5.16. Hasil *Interface* Halaman Utama

### 4.2.2 *Interface* Halaman Dataset

Implementasi *mockup* yang telah dirancang sebelumnya pada Gambar 4.18 dan dapat dilihat pada Gambar 5.17.




Gambar 5.17. Hasil *Interface* Halaman Dataset

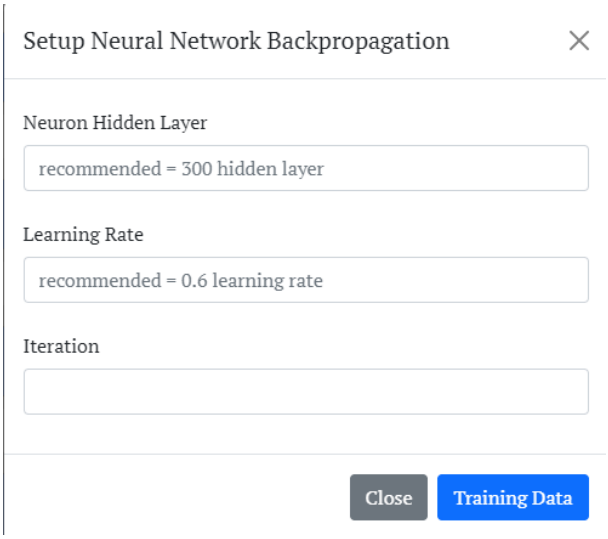
### 4.2.3 *Interface* Halaman Identifikasi

Implementasi *mockup* yang telah dirancang sebelumnya pada Gambar 4.19 dan dapat dilihat pada Gambar 5.18.


Gambar 5.18. Hasil *Interface* Halaman Identifikasi

### 4.2.4 Interface Halaman Setup Metode Backpropagation

Implementasi *mockup* yang telah dirancang sebelumnya pada Gambar 4.20 dan dapat dilihat pada Gambar 5.19.




Gambar 5.19. Hasil *Interface* Halaman *Setup* Pada Metode *Backpropagation*

## 4.3 Implementasi Proses Sistem

Implementasi proses sistem pada aplikasi pengenalan dan identifikasi pesawat udara militer terdiri dari beberapa potongan kode program sebagai berikut.

### 4.3.1 Kode Program Fusi Informasi Dengan Gerbang Logika *X-OR*

Pada potongan kode berikut merupakan bagian dari alur program pengenalan dan identifikasi pesawat udara militer yang berfungsi untuk menggabungan beberapa fitur menjadi fitur utama yang digunakan sebagai data *input*. Potongan kode program dapat dilihat pada Gambar 5.20.

```
# XOR gerbang logika
def XOR(x, y):
    if x != y:
        return '1'
    else:
        return '0'
# ciri di fusikan menjadi 1 binary
def fusi_informasi(x):
    for i in range(len(x) - 1):
        temp = []
        for j in range(len(x[0])):
            if i == 0:
                temp += XOR(x[i][j], x[i + 1][j])
            else:
```

```
                temp += XOR(information_fusion[0][j], x[i + 1][j])
        information_fusion = ''.join(temp).split()
    return information_fusion
```

Gambar 5.20. Potongan Kode Fusi Informasi Dengan Gerbang Logika X-OR

### 4.3.2 Kode Program *Preprocessing*

Pada potongan kode berikut merupakan bagian dari alur program pengenalan dan identifikasi pesawat udara militer yang berfungsi untuk menormalisasi hasil fusi informasi yang masih berupa biner menjadi bilangan desimal yang lebih kecil. Potongan kode program dapat dilihat pada Gambar 5.21.

```
# menambahkan comma setelah 2 angka
def add_dot_separator(binary):
    if len(binary) <= 1:
        for all_values in range(len(binary)):
            new_value = list(binary[all_values])
        new_value.insert(2, '.')
        new_value = ''.join(new_value).split()
    elif len(binary) > 1:
        new_value = list(binary)
        new_value.insert(2, '.')
        new_value = ''.join(new_value).split()
    return new_value

# convert comma binary menjadi angka desimal
def convert_decimal(binary):
    decimal = ''
    positif = negatif = 0
    value = 0.0
    for i in range(len(binary)):
        decimal = binary[0]
    decimal = list(decimal)
    index_dot = decimal.index('.')
    size_dot = decimal.index('.')
    for i in range(len(decimal)):
        if i < index_dot:
            positif = positif + 1
            if decimal[i] == '1':
```

```
                value = value + pow(2, (size_dot - 1))
        elif i > index_dot:
            negatif = negatif + 1
            if decimal[i] == '1':
                value = value + pow(2, -negatif)
        size_dot -= 1
return value
```

Gambar 5.21. Potongan Kode *Preprocessing*

### 4.3.3 Kode Program *Hamming Distance*

Pada potongan kode berikut merupakan bagian dari alur program pengenalan dan identifikasi pesawat udara militer yang berfungsi untuk mengenali pesawat udara dengan kondisi yang mendekati dengan jenis pesawat yang ada pada dataset atau jarak bilangan biner paling kecil. Potongan kode program dapat dilihat pada Gambar 5.22.

```
def hammingDistance(value1, value2):
    distance = 0
    for i in range(len(value1)):
        for j in range(len(value1[0])):
            if value1[i][j] != value2[i][j]:
                distance += 1
    return distance

def get_min_hd(value1, value2):
    min_distance = 0
    value_min_distance = ''
    current_min_distance = hammingDistance(value1, value2[0])
    for i in range(len(value2) - 1):
        for j in range(len(value2[i])):
            for k in range(len(value2[i][j])):
                if value1[j][k] != value2[i + 1][j][k]:
                    min_distance += 1
            if min_distance < current_min_distance:
                current_min_distance = min_distance
                value_min_distance = value2[i + 1][j]
            min_distance = 0
    return value_min_distance
```

Gambar 5.22. Potongan Kode *Hamming Distance*

### 4.3.4 Kode Program Metode *Backpropagation*

Pada potongan kode berikut merupakan bagian dari alur program pengenalan dan identifikasi pesawat udara militer yang berfungsi untuk mengidentifikasi *class* pesawat udara. Potongan kode program dapat dilihat pada Gambar 5.23.

```
import numpy as np
import time

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def deriv_sigmoid(x):
    return x * (1.0 - x)

class Neural_Network:
    def __init__(self, input, hidden, output, learningRate):
        self.n_input = input
        self.n_hidden = hidden
        self.n_output = output
        self.learning_rate = learningRate
        self.bias_hidden = np.ones((hidden, 1))
        self.bias_output = np.ones((output, 1))

        self.weight_hidden = np.random.uniform(low=-0.2,
high=0.2, size=(self.n_hidden, self.n_input))
        self.weight_output = np.random.uniform(low=-0.2,
high=0.2, size=(self.n_output, self.n_hidden))

    def forward(self, input_list):
        inputs = np.array(input_list, ndmin=2).T
        self.hiddenLayer = np.dot(self.weight_hidden, inputs) +
self.bias_hidden
        self.Net_hiddenLayer = sigmoid(self.hiddenLayer)

        self.outputLayer = np.dot(self.weight_output,
self.Net_hiddenLayer) + self.bias_output
        self.Net_outputLayer = sigmoid(self.outputLayer)
        return self.Net_outputLayer
```

```
    def calc_loss(self, target_list):
        targets = np.array(target_list, ndmin=2).T
        m = targets.shape[0]
        cost = 0
        cost = cost + ((self.Net_outputLayer - targets) **
2).sum() / m
        return cost

    def backprop(self, input_list, target_list):
        inputs = np.array(input_list, ndmin=2).T
        targets = np.array(target_list, ndmin=2).T
        m = targets.shape[0]

        self.output_error = self.Net_outputLayer - targets
        self.output_error_weight = np.dot(self.output_error *
deriv_sigmoid(self.Net_outputLayer),
                                          self.Net_hiddenLayer.T)
* (1 / m)
        self.output_error_bias = np.sum(self.output_error,
axis=1, keepdims=True) * (1 / m)

        self.hidden_error = np.dot(self.weight_output.T,
self.output_error)
        self.hidden_error_weight = np.dot(self.hidden_error *
deriv_sigmoid(self.Net_hiddenLayer), inputs.T) * (1 / m)
        self.hidden_error_bias = np.sum(self.hidden_error,
axis=1, keepdims=True) * (1 / m)

    def update_params(self):
        self.weight_output = self.weight_output -
self.learning_rate * self.output_error_weight
        self.weight_hidden = self.weight_hidden -
self.learning_rate * self.hidden_error_weight
        self.bias_output = sigmoid(self.bias_output -
self.learning_rate * self.output_error_bias)
        self.bias_hidden = sigmoid(self.bias_hidden -
self.learning_rate * self.hidden_error_bias)
```

```
    def predict(self, input_list, weight_hidden, weight_output,
bias_hidden, bias_output):
        inputs = np.array(input_list, ndmin=2).T
        hiddenLayer = np.dot(weight_hidden, inputs) + bias_hidden
        Net_hiddenLayer = sigmoid(hiddenLayer)

        outputLayer = np.dot(weight_output,Net_hiddenLayer) +
bias_output
        self.Net_outputLayer = sigmoid(outputLayer)
        return self.Net_outputLayer

    def train_data(self, data_train, epochs):
        accu_train = []
        all_label = []
        for i in range(epochs):
            loss = 0
            startTime = time.time()
            for record in data_train:
                all_values = record.split(',')
                inputs = np.asfarray(all_values[1:])
                targets = np.zeros(self.n_output) + 0.01
                targets[int(all_values[0])] = 0.99
                correct_label = int(all_values[0])
                self.forward(inputs)
                loss = self.calc_loss(targets)
                self.backprop(inputs, targets)
                self.update_params()
                outputs = self.predict(inputs,
self.weight_hidden, self.weight_output, self.bias_hidden,
self.bias_output)
                label = np.argmax(outputs)
                all_label.append(label)
                if (label == correct_label):
                    accu_train.append(1)
                else:
                    accu_train.append(0)
            if loss < 0.0001:
                break
                print("Epochs = ", i)
```

```
                print("Loss =", loss)
            if i % 10 == 0:
                print("Epochs = ", i)
                print("Loss =", loss)
        accu_array = np.asarray(accu_train)
        self.accu_train = accu_array.sum() / accu_array.size *
100
        self.final_loss = loss
        print("Accu Train: ", self.accu_train, "%")
        print('The script took {0} second !'.format(time.time() -
startTime))
```

Gambar 5.23. Potongan Kode Program Metode *Backpropagation*

## 4.4 Pengujian Fungsional

Pengujian Fungsional digunakan untuk menguji setiap fitur yang ada pada aplikasi dan melihat kecocokan hasil dengan hasil yang diharapkan.

### 4.4.1 Pengujian Sistem

Pengujian sistem dilakukan dengan cara menguji setiap fitur yang ada pada aplikasi yang bertujuan untuk mengetahui setiap fitur berjalan dengan baik atau tidak berjalan sesuai harapan. Berikut adalah beberapa pengujian fungsional pada aplikasi.

Tabel 5.1. Pengujian Fungsional Halaman Utama

| No. | Fitur | Hasil yang Diharapkan | Hasil yang Diperoleh | Hasil |
|---|---|---|---|---|
| 1 | Halaman utama aplikasi | Menampilkan halaman utama aplikasi | Halaman utama aplikasi berhasil tampil | Berhasil |
| 2 | Tombol “Dataset *Aircraft*” | Beralih ke halaman dataset ketika tombol ditekan | Berhasil beralih ke halaman dataset ketika tombol ditekan | Berhasil |
| 3 | Tombol “Mulai Identifikasi” | Beralih ke halaman identifikasi ketika tombol ditekan | Berhasil beralih ke halaman identifikasi ketika tombol ditekan | Berhasil |

Tabel 5.2. Pengujian Fungsional Halaman Dataset

| No. | Fitur | Hasil yang Diharapkan | Hasil yang Diperoleh | Hasil |
|---|---|---|---|---|
| 1 | Halaman dataset aplikasi | Menampilkan halaman dataset aplikasi | Halaman dataset aplikasi berhasil tampil | Berhasil |
| 2 | Tabel dataset *aircraft* | Menampilkan seluruh dataset pesawat udara yang digunakan | Dataset pesawat udara militer dapat ditampilkan pada tabel dengan seluruh total dataset yaitu 155 | Berhasil |

Tabel 5.3. Pengujian Fungsional Halaman Identifikasi

| No. | Fitur | Hasil yang Diharapkan | Hasil yang Diperoleh | Hasil |
|---|---|---|---|---|
| 1 | Halaman identifikasi aplikasi | Menampilkan halaman identifikasi aplikasi | Halaman identifikasi aplikasi berhasil tampil | Berhasil |
| 2 | *Form input* data fitur pesawat | *Form input* dapat menerima masukkan dari user | *Form input* berhasil menerima masukkan dari user yang sudah ditetapkan | Berhasil |
| 3 | Tombol “Training Data” | Memunculkan *form setup* metode *backpropagation* | *Form setup* metode *backpropagatiom* berhasil muncul ketika tombol ditekan | Berhasil |
| 4 | *Form setup* metode *backpropagation* | *Form setup* dapat menerima *input* dari user | *Form setup* berhasil menerima data *input* untuk metode *backpropagation* | Berhasil |
| 5 | Tombol “Identifikasi Pesawat Udara” | Tombol identifikasi dapat mengidentifikasi dan halaman identifikasi dapat menampilkan hasil identifikasi | Hasil identifikasi pesawat udara berhasil muncul beserta akurasi, nama pesawat dan asal negara buatan ketika tombol identifikasi ditekan | Berhasil |

4.4.2 Pengujian Metode

Pengujian metode yang digunakan dimaksud untuk mengetahui ketepatan dari penggunaan metode *Backpropagation* dan fusi informasi dalam mengenali dan mengidentifikasi pesawat udara militer yang menggunakan data teks sebagai data *input*. Pada pengujian metode disini terdapat 3 skenario pengujian yaitu:

1. Mencari kombinasi yang optimal antara *learning rate* dan jumlah *node hidden layer* untuk mencari akurasi tertinggi serta melakukan pengujian dengan kondisi 80% data latih dan 20% data uji
2. Melakukan pengujian dengan data pesawat yang baru atau data yang tidak tercantum pada dataset (diambil secara random)
3. Melakukan penggunaan dengan menggunakan data pesawat baru yang diambil dari buku

4.4.2.1 Pengujian Metode Skenario Pertama

Pada skenario pertama akan dilakukan pencarian kombinasi jumlah *node hidden layer* dan *learning rate* yang paling optimal dengan jumlah dataset 155 pesawat udara.

Tabel 5.4. Tabel Perhitungan Mencari Kombinasi Paling Optimal

| **No.** | **Jumlah Node *Hidden Layer*** | ***Learning Rate*** | **Akurasi** | **MSE** |
|---|---|---|---|---|
| 1 | 50 | 0,1 | 77,56 % | 0,0055671 |
| 2 | 50 | 0,3 | 88,92 % | 0,0005331 |
| 3 | 50 | 0,5 | 87,97 % | 0,001619 |
| 4 | 50 | 0,6 | 89,31 % | 0,002860 |
| 5 | 100 | 0,1 | 77,56 % | 0,011504 |
| 6 | 100 | 0,3 | 90,54 % | 0,001953 |
| 7 | 100 | 0,5 | 92,09 % | 0,002092 |
| 8 | 100 | 0,6 | 93,85 % | 0,000186 |
| 9 | 200 | 0,1 | 82,88 % | 0,008420 |
| 10 | 200 | 0,3 | 92,78 % | 0,000411 |
| 11 | 200 | 0,5 | 93,79 % | 0,00088 |
| 12 | 200 | 0,6 | 94,07 % | 0,00237 |
| 13 | 300 | 0,1 | 83,69 % | 0,012354 |
| 14 | 300 | 0,3 | 92,59 % | 0,001771 |
| 15 | 300 | 0,5 | 93,6 % | 0,00238 |
| 16 | 300 | 0,6 | 95,33 % | 4,3746e-05 |

| | | | | |
|---|---|---|---|---|
| 17 | 400 | 0,1 | 86,33 % | 0,00737 |
| 18 | 400 | 0,3 | 92,41 % | 0,00127 |
| 19 | 400 | 0,5 | 94% | 0,00022 |
| 20 | 400 | 0,6 | 94,6 % | 0,0022 |

Jumlah *node hidden layer* dan *learning rate* sangat berpengaruh terhadap hasil pelatihan *backpropagation* tetapi pelatihan memiliki batasan jumlah *node hidden layer* dan *learning rate* untuk mendapatkan hasil yang tertinggi. Pada Tabel 5.4 merupakan hasil dari skenario pengujian yang pertama dan menghasilkan kombinasi jumlah *node hidden layer* dan *learning rate* yaitu 300 *node hidden layer* dan *learning rate* 0.6. Untuk kombinasi yang optimal mendapatkan akurasi 95.33% dan mendapatkan *error* 4.3746e-05.

Selanjutnya setelah dilakukan pencarian kombinasi *learning rate* dan jumlah *node hidden layer* yang optimal maka akan dilakukan pengujian dataset yang telah dibagi menjadi 80% data pelatihan dan 20% data pengujian. Berikut adalah hasil pengujian menggunakan 80% data pelatihan dan 20% data pengujian.

Tabel 5.5. Hasil Pengujian 31 Data Pesawat Udara

| No. | Data Aktual | Prediksi | Status | Akurasi |
|---|---|---|---|---|
| 1 | 0 | 0 | Benar | Akurasi 87 % |
| 2 | 0 | 0 | Benar | |
| 3 | 0 | 0 | Benar | |
| 4 | 0 | 0 | Benar | |
| 5 | 0 | 0 | Benar | |
| 6 | 0 | 0 | Benar | |
| 7 | 0 | 0 | Benar | |
| 8 | 0 | 0 | Benar | |
| 9 | 0 | 0 | Benar | |
| 10 | 1 | 1 | Benar | |
| 11 | 1 | 1 | Benar | |
| 12 | 1 | 1 | Benar | |
| 13 | 1 | 1 | Benar | |
| 14 | 1 | 1 | Benar | |
| 15 | 1 | 1 | Benar | |
| 16 | 2 | 3 | Salah | |
| 17 | 2 | 3 | Salah | |
| 18 | 2 | 2 | Benar | |

| | | | | |
|---|---|---|---|---|
| 19 | 2 | 2 | Benar | |
| 20 | 2 | 2 | Benar | |
| 21 | 2 | 3 | Salah | |
| 22 | 2 | 2 | Benar | |
| 23 | 3 | 2 | Salah | |
| 24 | 3 | 3 | Benar | |
| 25 | 3 | 3 | Benar | |
| 26 | 3 | 3 | Benar | |
| 27 | 3 | 3 | Benar | |
| 28 | 3 | 3 | Benar | |
| 29 | 3 | 3 | Benar | |
| 30 | 3 | 3 | Benar | |
| 31 | 3 | 3 | Benar | |

Tabel 5.5 merupakan hasil dari pengujian dengan 31 data uji dan menghasilkan akurasi yang tinggi yaitu 87 %. Data dengan angka 0 sampai 3 merupakan label dari data yang diujikan dengan ketentuan angka 0 merupakan pesawat militer, angka 1 pesawat sipil, angka 2 helikopter militer dan angka 3 helikopter sipil. Selanjutnya akan dilakukan validasi menggunakan *confusion matrix* pada hasil pengujian Tabel 5.5.

Tabel 5.6. Hasil *Confusin Matrix* Pada Pengujian Tabel 5.5

| *Class* | | Data Sebenarnya | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 |
| Data Prediksi | 0 | 9 | 0 | 0 | 0 |
| | 1 | 0 | 6 | 0 | 0 |
| | 2 | 0 | 0 | 4 | 3 |
| | 3 | 0 | 0 | 1 | 8 |

Tabel 5.7. Tabel Penentuan *True Positif, True Negatif, False Positif* dan *False Negatif* Pada Pengujian Tabel 5.5

| *Class* | *True Positif* | *True Negatif* | *False Positif* | *False Negatif* |
|---|---|---|---|---|
| 0 | 9 | 22 | 0 | 0 |
| 1 | 6 | 25 | 0 | 0 |
| 2 | 4 | 23 | 1 | 3 |

| | | | | |
|---|---|---|---|---|
| 3 | 8 | 19 | 3 | 1 |

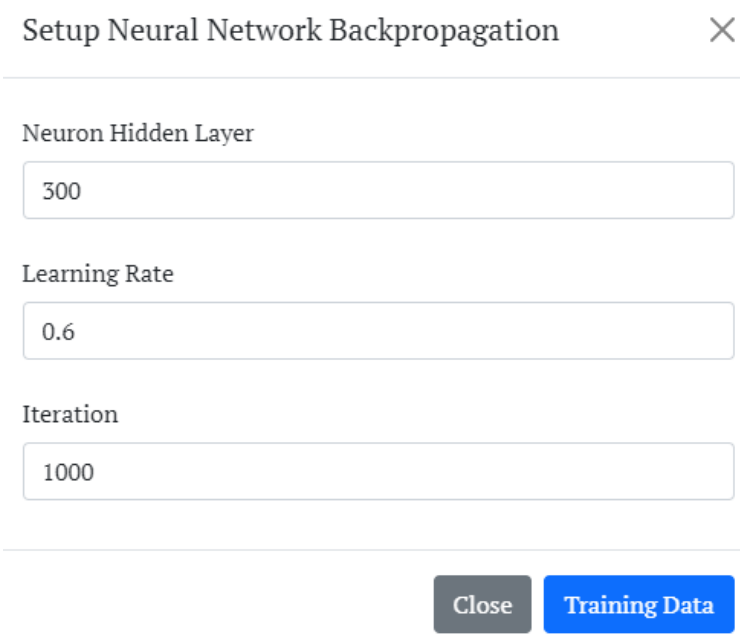
Tabel 5.8. Hasil Perhitungan *Precision, Recall, F1-Score* dan *Accuracy* Pada Pengujian Tabel 5.5

| *Class* | *Precision* | *Recall* | *F1-Score* | *Accuracy* |
|---|---|---|---|---|
| 0 | 9/(9+0) = 1 | 9/(9+0) = 1 | (2X9)/(2X9)+0+0 = 1 | (8+22)/(8+22+0+0) = 1 |
| 1 | 6/(6+0) = 1 | 6/(6+0) = 1 | (2X6)/(2X6)+0+0 = 1 | (6+25)/(6+25+0+0) = 1 |
| 2 | 4/(4+1) = 0,80 | 4/(4+3) = 0,57 | (2X4)/(2X4)+1+3 = 0,67 | (4+23)/(4+23+1+3) = 0,87 |
| 3 | 8/(8+3) = 0,73 | 8/(8+1) = 0,89 | (2X8)/(2X8)+3+1 = 0,80 | (8+19)/(8+19+3+1) = 0,87 |

Tabel 5.6 merupakan hasil *confusion matrix* pada Tabel 5.5 dan Tabel 5.7 merupakan hasil penentuan untuk mencari *True Positif, True Negatif, False Positif* dan *False Negatif*. Tabel 5.8 merupakan perhitungan *Precision, Recall, F1-Score* dan *Accuracy* untuk setiap label yang ada pada dataset yang mendapatkan nilai bagus.

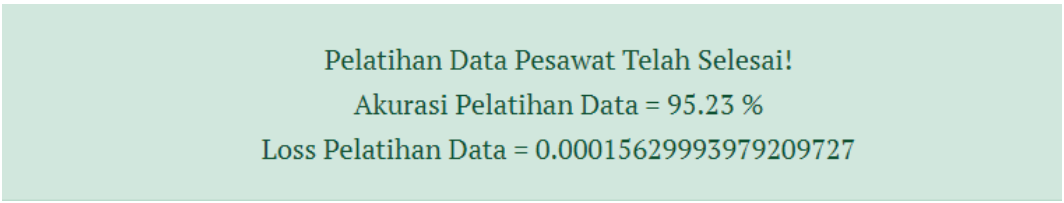
4.4.2.2 Pengujian Metode Skenario Kedua

Pada skenario kedua akan dilakukan pengujian data baru yang tidak ada pada dataset pada aplikasi yang telah di-*deploy* pada website. Berikut adalah langkah-langkah untuk melakukan pengujian. Yang pertama adalah melakukan *setup* untuk melakukan pelatihan data. Contoh *setup* dapat dilihat pada Gambar 5.24.



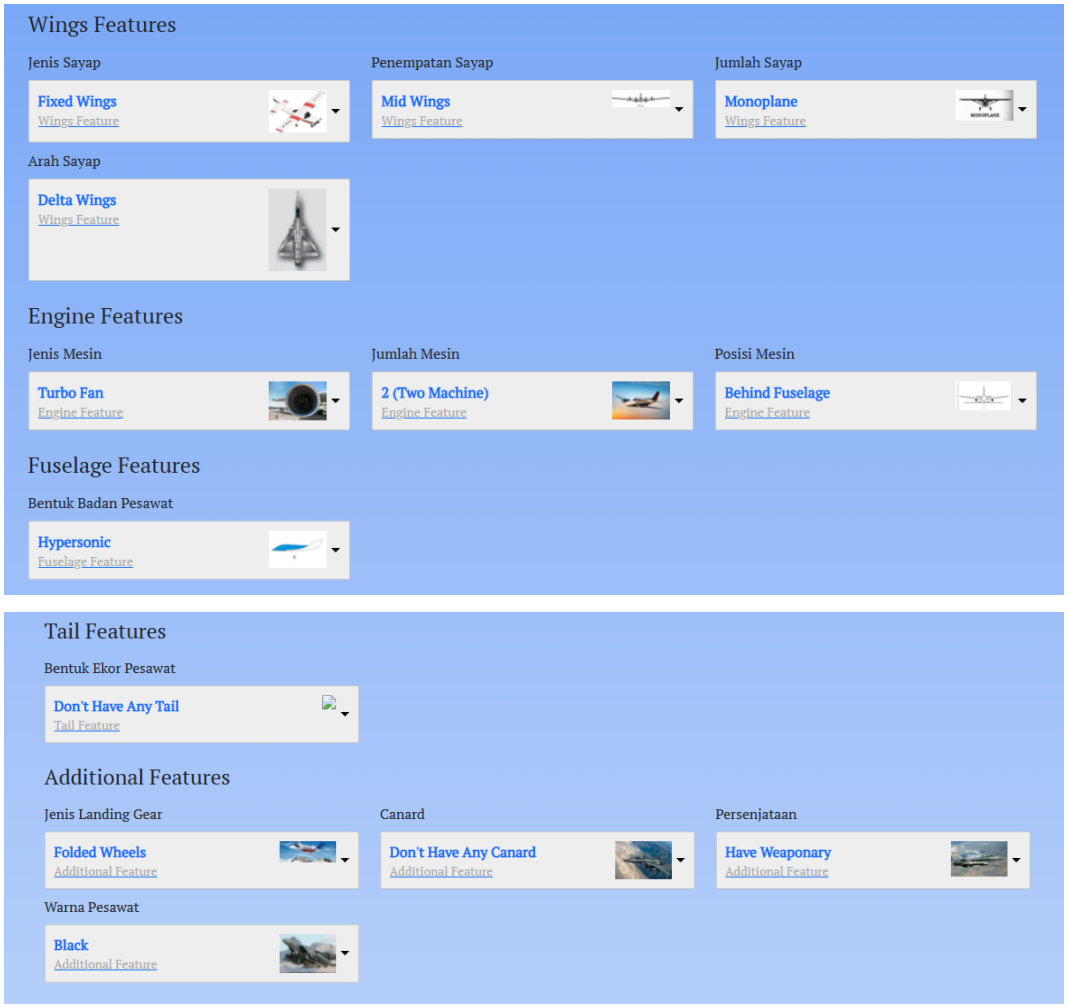

Gambar 5.24. Gambar *form* ketika melakukan pelatihan data pada aplikasi

Pada proses pelatihan, website akan melakukan *loading* pada tab di browser yang menandakan bahwa aplikasi sedang melakukan pelatihan data dan membutuhkan waktu untuk menyelesaikan pelatihan data.




Gambar 5.25. Gambar ketika aplikasi telah melakukan pelatihan data

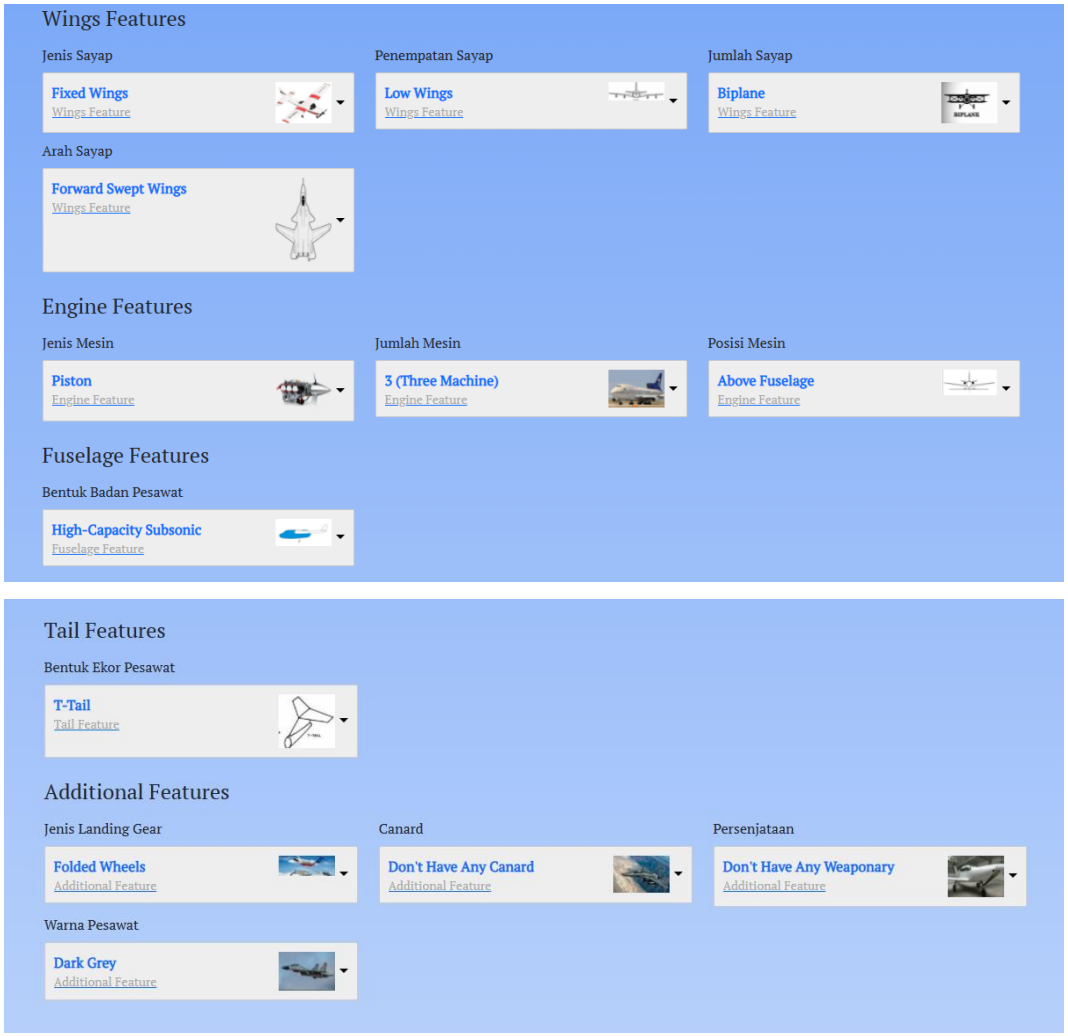
Setelah melakukan pelatihan data maka dapat mengisikan fitur pesawat yang telah tersedia di aplikasi. Contohnya pada Gambar 5.26 yang mengisi fitur pesawat Xi'an H-20 dan hasil dari identifikasi bisa dilihat pada Gambar 5.27.



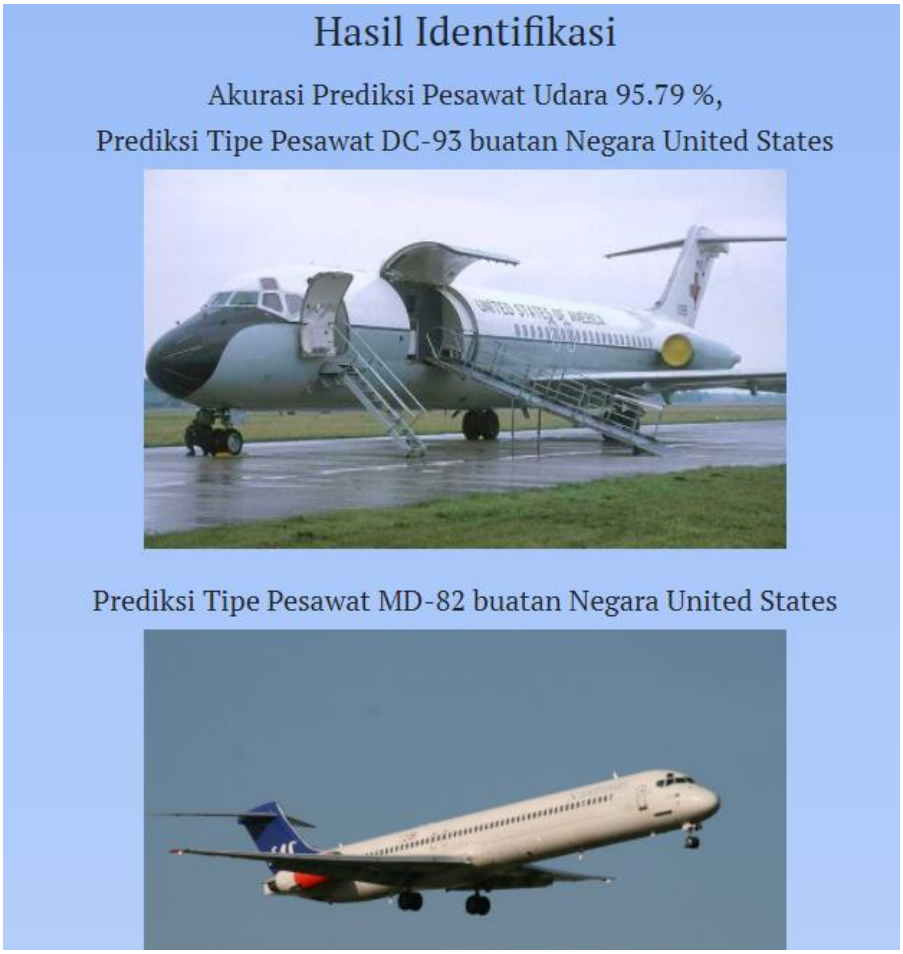

Gambar 5.26. Gambar Ketika Mengisikan Fitur Pesawat Yang Akan Diuji


Gambar 5.27. Hasil Identifikasi Pesawat Yang Ada Pada Dataset

Selanjutnya akan dilakukan pengujian data fitur pesawat yang diambil secara acak atau data fitur pesawat yang tidak ada pada dataset. Gambar 5.28 merupakan data fitur yang diambil secara acak dan Gambar 5.29 merupakan hasil identifikasi pesawat dari data fitur yang diambil secara acak dan aplikasi akan mencari jenis pesawat yang mendekati dengan menggunakan *hamming distance*.



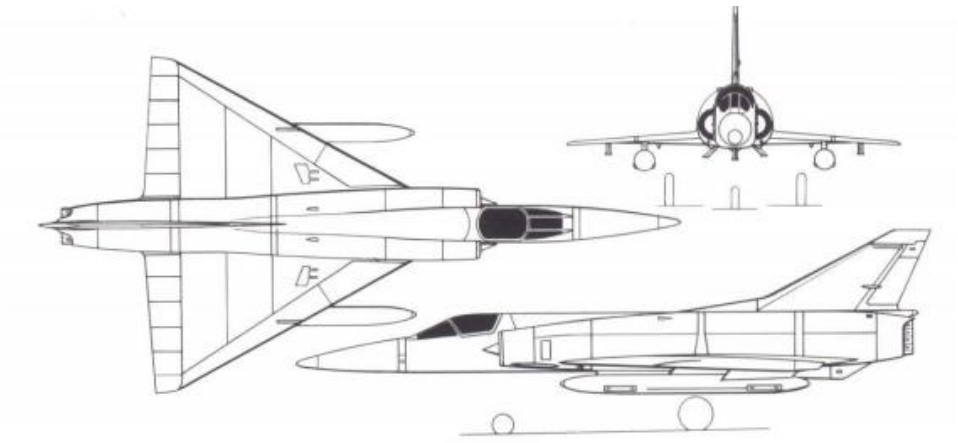

Gambar 5.28. Gambar Form Yang Telah Diisi Dengan Data Acak



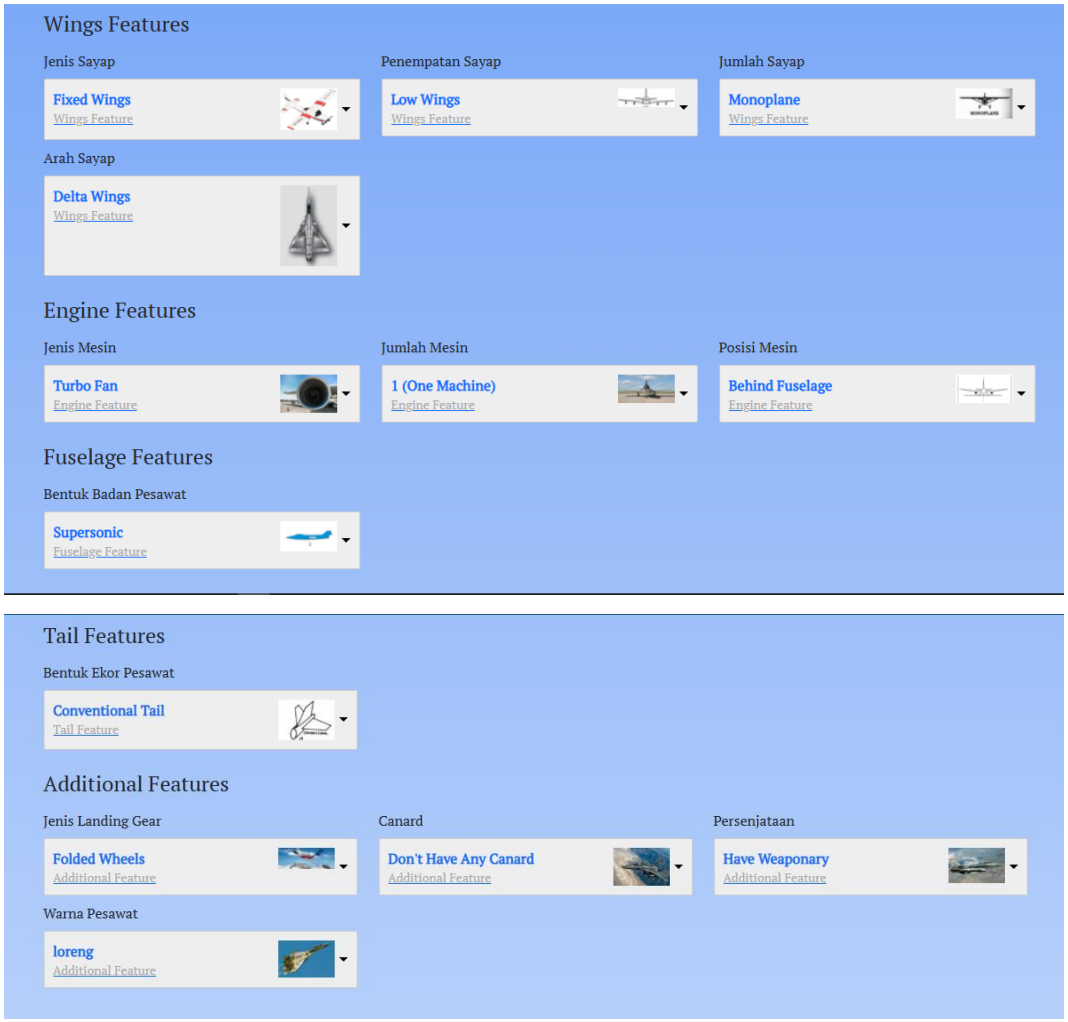

Gambar 5.29. Hasil Pengujian Dari Data Fitur Yang Dipilih Secara Acak

Pada hasil pengujian di Gambar 5.29 muncul 2 hasil identifikasi pesawat, hal tersebut bisa saja terjadi karena pada dataset terdapat pesawat yang memiliki fitur yang sama atau hasil fusi informasi fitur yang sama karena hal itu maka muncul 2 atau lebih ketika identifikasi pesawat.

4.4.2.3 Pengujian Metode Skenario Ketiga



Gambar 5.30. *Prototype* Pesawat Dassault Mirage III




Gambar 5.31. Form yang telah diisi dengan fitur pesawat Dassault Mirage III


Gambar 5.32. Hasil identifikasi dari data pesawat Dassault Mirage III

Pada skenario ke tiga, data pesawat diambil dari buku (Gunston, 1980) yaitu pesawat Dassault Mirage III. Gambar 5.30 merupakan gambar arsitektur dari pesawat Dassault Mirage III. Pesawat Dassault Mirage III tidak tercantum pada dataset. Pada form identifikasi, fitur dari pesawat Dassault Mirage III dimasukkan ke dalam aplikasi untuk dilakukan identifikasi pesawat, dapat dilihat pada Gambar 5.31. Pada Gambar 5.32 merupakan hasil identifikasi dari pesawat Dassault Mirage III yaitu sistem menunjukan kemiripan pesawat dengan Dassault Mirage 2000, pada fitur terdapat beberapa perbedaan dari Dassault Mirage 2000 yaitu pada bentuk badan pesawat.