

## **BAB V**

### **IMPLEMENTASI DAN PENGUJIAN SISTEM**

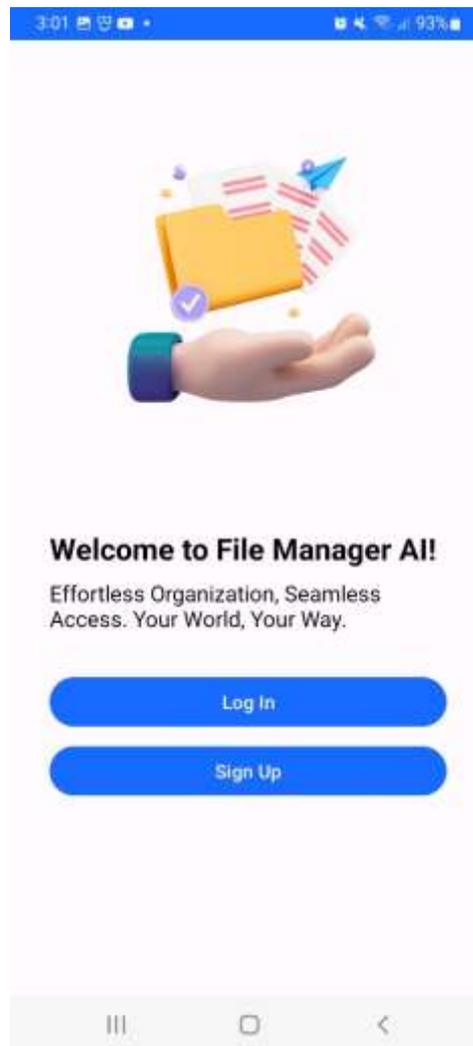
#### **5.1 HASIL IMPLEMENTASI**

Hasil Implementasi dari pengembangan perangkat lunak aplikasi FileManager.AI adalah sebuah aplikasi MVP (*Minimum Viable Product*) di mana MVP merupakan versi produk aplikasi yang berisi fitur inti sesuai dengan *project plan* yang di buat sebelumnya.

Pada Implementasinya Model *Machine Learning* “*Picture*” dan “*Document*” di jalankan pada Aplikasi REST-API (*Backend*) dan *backend* tersebut di konsumsi oleh aplikasi Android. Berikut merupakan rancangan UI dari aplikasi android.

### 5.1.1 Halaman Welcome Page

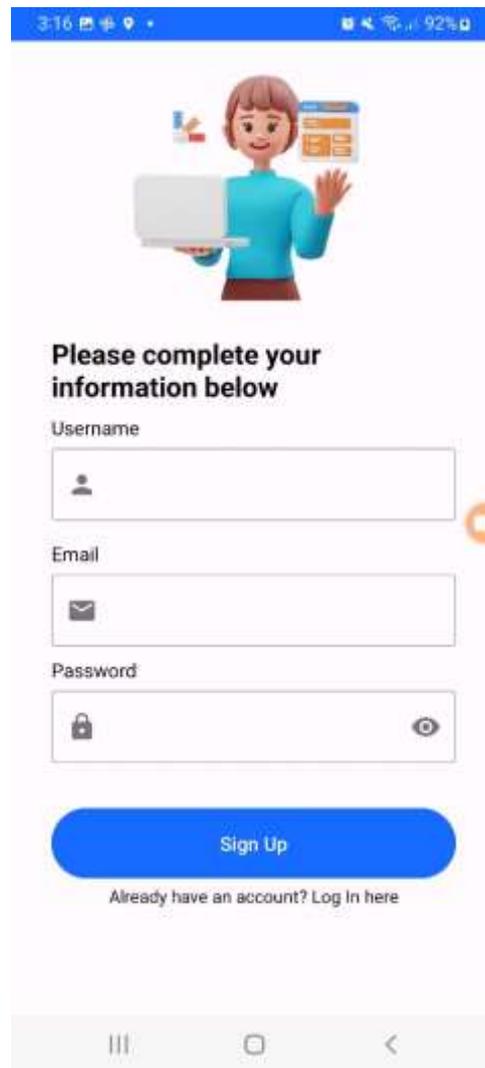
Halaman *Welcome Page* merupakan halaman yang pertama pengguna lihat ketika pertama membuka aplikasi. Pada halaman ini pengguna disambut dengan penjelasan singkat mengenai fitur utama aplikasi. Selain itu pada halaman ini terdapat juga tombol untuk melakukan *login* atau *register*.



**Gambar 5. 1** Halaman Welcome Page

### 5.1.2 Halaman Register

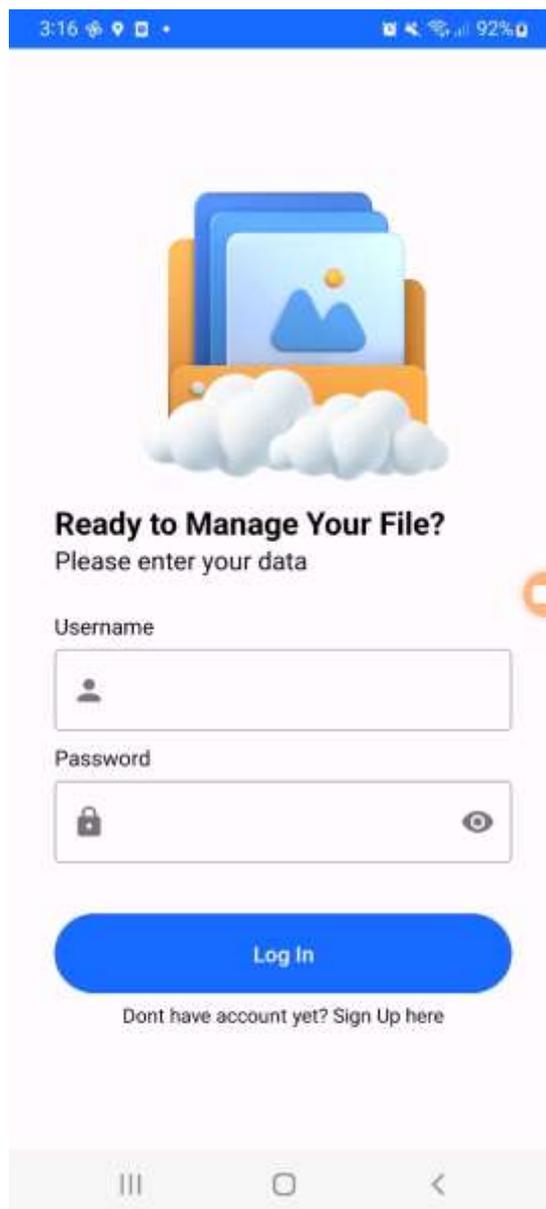
Pada halaman ini pengguna dapat melakukan registrasi. Untuk menggunakan aplikasi FileManger.AI pengguna wajib memiliki sebuah akun. Di dalam halaman registrasi pengguna wajib mengisi *username*, *email*, dan *password*.



**Gambar 5. 2 Halaman Register**

### 5.1.3 Halaman Login

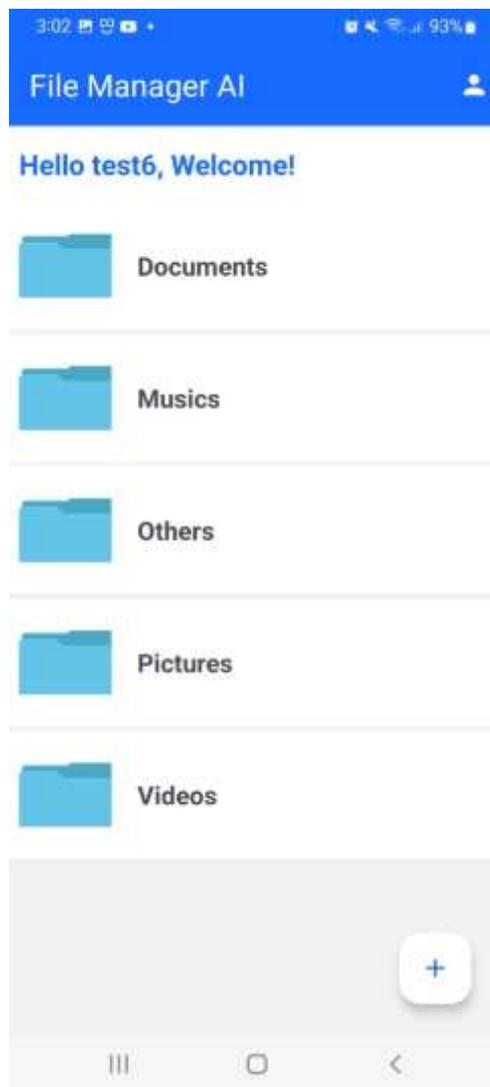
Setelah pengguna melakukan registrasi, selanjutnya pengguna dapat melakukan *login*. Pada halaman *login* pengguna hanya perlu memasukkan *username* dan *password*.



**Gambar 5. 3 Halaman Login**

#### 5.1.4 Halaman Home

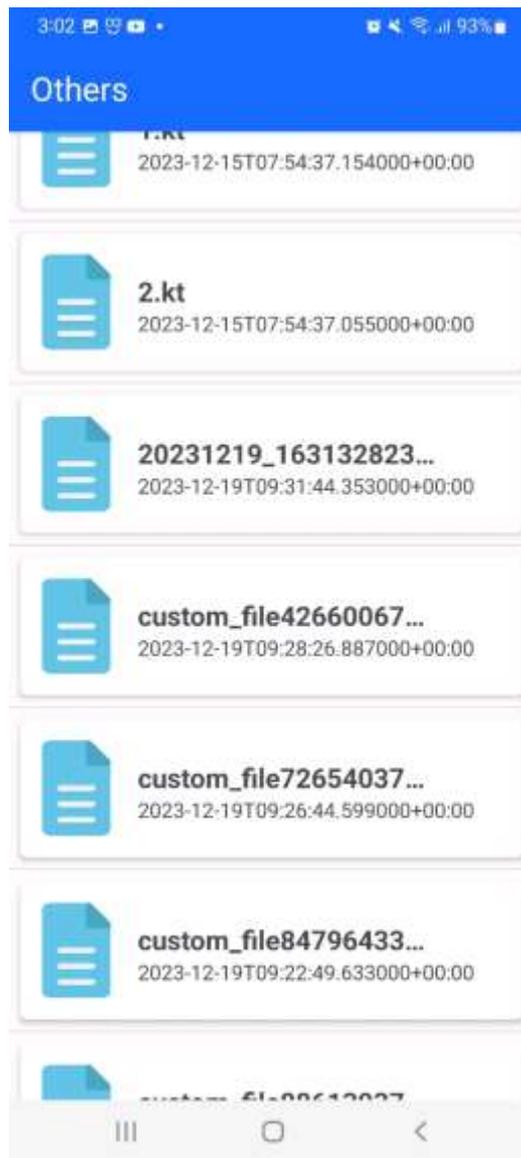
Setelah pengguna melakukan *login* maka pengguna akan melihat halaman *home*. Pada halaman ini terdapat beberapa folder yaitu *Documents*, *Musics*, *Others*, *Pictures* dan *Videos*. Pada halaman ini pengguna dapat menjelajahi folder dan *file* yang di simpan oleh pengguna. Pada halaman ini pengguna juga dapat menekan tombol “+” pada kanan bawah layar untuk mengunggah *file* ke *backend*. *file* akan di simpan pada *backend* dan bertindak sebagai *backup file* pengguna.



**Gambar 5. 4 Halaman Home**

### 5.1.5 Halaman List File

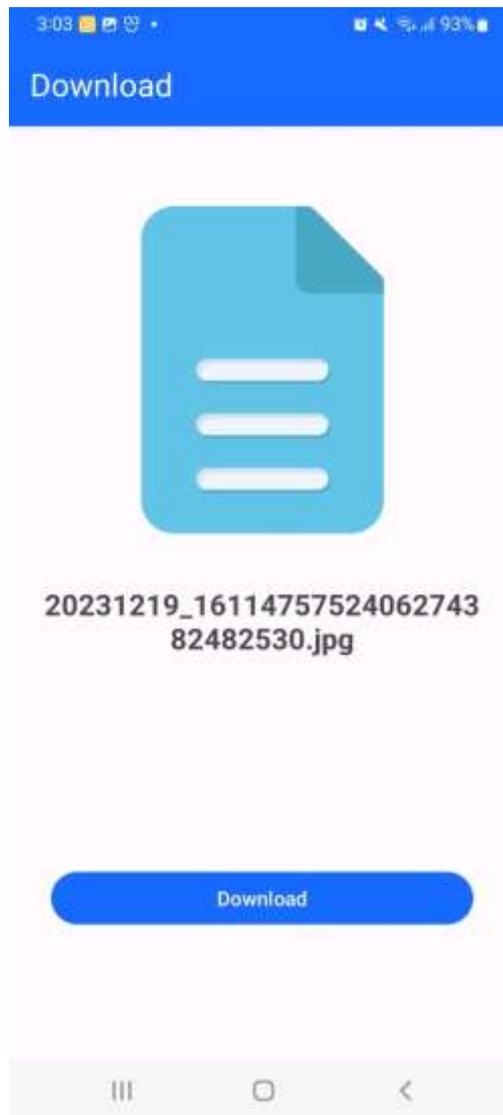
Halaman ini menyajikan daftar *file* yang tersimpan dalam folder-folder pengguna. Dalam halaman ini setiap *file* yang ditampilkan dengan informasi yang relevan seperti nama *file* dan modifikasi *file* terakhir.



**Gambar 5. 5 Halaman List File**

### 5.1.6 Halaman Download File

Ketika pengguna menekan salah satu *file* pada halaman list file, pengguna akan melihat halaman download file. Pada halaman ini pengguna dapat melakukan *download* kembali *file* yang telah di unggah ke *backend* dan disimpan pada penyimpanan lokal pengguna.



**Gambar 5. 6 Halaman Download File**

## 5.2 PENGUJIAN SISTEM DAN PERANGKAT LUNAK

### 5.2.1 Pengujian Akurasi Machine Learning

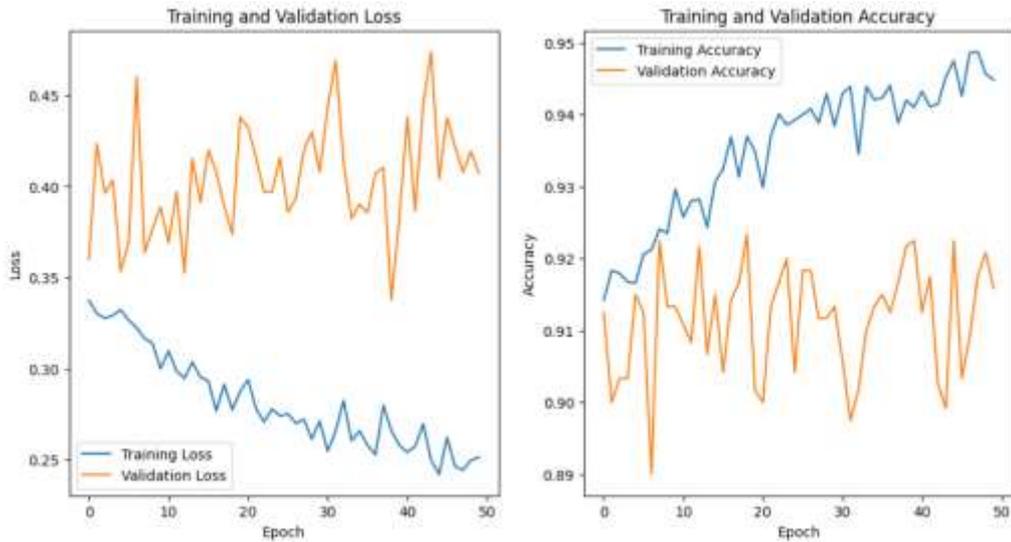
#### 5.2.1.1 Model “Picture”

*Model Machine Learning “Picture”* di *training* sebanyak 50 *epoch*. Dari hasil *training* tersebut didapatkan akurasi pada *data training* 94% dan akurasi pada *data testing* 91%.

```
Epoch 1/50  
188/188 [#####] - 117s 622ms/step - loss: 0.3373 - accuracy: 0.9142 - val_loss: 0.3606 - val_accuracy: 0.9125  
Epoch 2/50  
188/188 [#####] - 139s 739ms/step - loss: 0.3302 - accuracy: 0.9183 - val_loss: 0.4234 - val_accuracy: 0.9088  
Epoch 3/50  
188/188 [#####] - 128s 681ms/step - loss: 0.3276 - accuracy: 0.9179 - val_loss: 0.3962 - val_accuracy: 0.9033  
Epoch 4/50  
188/188 [#####] - 129s 688ms/step - loss: 0.3291 - accuracy: 0.9168 - val_loss: 0.4633 - val_accuracy: 0.9033  
Epoch 5/50  
188/188 [#####] - 128s 638ms/step - loss: 0.3323 - accuracy: 0.9166 - val_loss: 0.3533 - val_accuracy: 0.9150  
Epoch 6/50  
188/188 [#####] - 119s 632ms/step - loss: 0.3265 - accuracy: 0.9206 - val_loss: 0.3695 - val_accuracy: 0.9125  
Epoch 7/50  
188/188 [#####] - 119s 632ms/step - loss: 0.3223 - accuracy: 0.9212 - val_loss: 0.4601 - val_accuracy: 0.8988  
Epoch 8/50  
188/188 [#####] - 118s 628ms/step - loss: 0.3165 - accuracy: 0.9241 - val_loss: 0.3636 - val_accuracy: 0.9225  
Epoch 9/50  
188/188 [#####] - 118s 630ms/step - loss: 0.3140 - accuracy: 0.9235 - val_loss: 0.3767 - val_accuracy: 0.9133  
Epoch 10/50  
188/188 [#####] - 118s 627ms/step - loss: 0.2999 - accuracy: 0.9297 - val_loss: 0.3887 - val_accuracy: 0.9133  
Epoch 11/50  
188/188 [#####] - 119s 634ms/step - loss: 0.3096 - accuracy: 0.9258 - val_loss: 0.3691 - val_accuracy: 0.9188  
Epoch 12/50  
188/188 [#####] - 120s 639ms/step - loss: 0.2987 - accuracy: 0.9288 - val_loss: 0.3978 - val_accuracy: 0.9083  
Epoch 13/50  
...  
Epoch 49/50  
188/188 [#####] - 122s 651ms/step - loss: 0.2698 - accuracy: 0.9457 - val_loss: 0.4192 - val_accuracy: 0.9208  
Epoch 50/50  
188/188 [#####] - 129s 686ms/step - loss: 0.2514 - accuracy: 0.9448 - val_loss: 0.4071 - val_accuracy: 0.9150  
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings.
```

**Gambar 5. 7 Training Model “Picture”**

Hasil akurasi pada data *training* dan *validation* sebanyak 50 *epoch* dapat divisualisasikan dengan grafik berikut:



**Gambar 5. 8 Grafik Training Model “Picture”**

### 5.2.1.2 Model “Document”

*Model Machine Learning “Document” di training sebanyak 30 epoch.*

Dari hasil *training* tersebut didapatkan akurasi pada *data training* 96% dan akurasi pada *data testing* 95%.

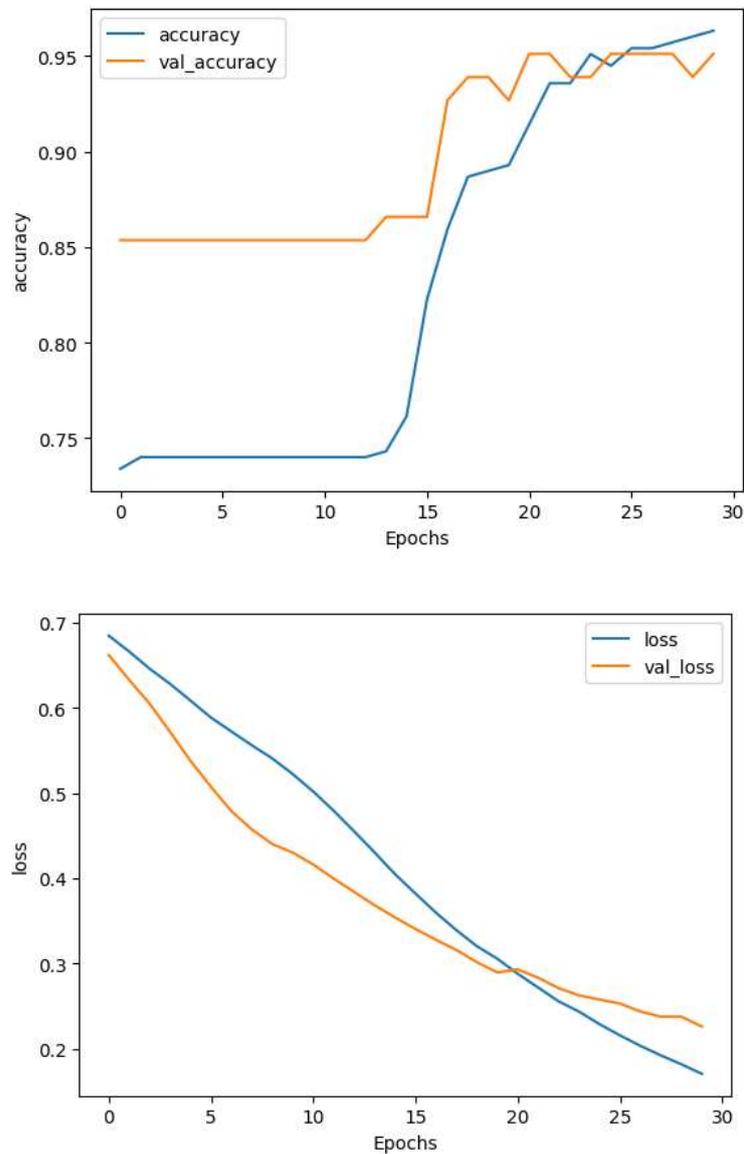
```

Epoch 1/30
11/11 [=====] - 1s 25ms/step - loss: 0.6847 - accuracy: 0.7339 - val_loss: 0.6619 - val_accuracy: 0.8537
Epoch 2/30
11/11 [=====] - 8s 5ms/step - loss: 0.6668 - accuracy: 0.7481 - val_loss: 0.6327 - val_accuracy: 0.8537
Epoch 3/30
11/11 [=====] - 8s 5ms/step - loss: 0.6458 - accuracy: 0.7481 - val_loss: 0.6848 - val_accuracy: 0.8537
Epoch 4/30
11/11 [=====] - 8s 8ms/step - loss: 0.6281 - accuracy: 0.7481 - val_loss: 0.5716 - val_accuracy: 0.8837
Epoch 5/30
11/11 [=====] - 8s 7ms/step - loss: 0.6885 - accuracy: 0.7481 - val_loss: 0.5377 - val_accuracy: 0.8537
Epoch 6/30
11/11 [=====] - 8s 7ms/step - loss: 0.5885 - accuracy: 0.7481 - val_loss: 0.5673 - val_accuracy: 0.8537
Epoch 7/30
11/11 [=====] - 8s 5ms/step - loss: 0.5722 - accuracy: 0.7481 - val_loss: 0.4787 - val_accuracy: 0.8537
Epoch 8/30
11/11 [=====] - 8s 5ms/step - loss: 0.5568 - accuracy: 0.7481 - val_loss: 0.4571 - val_accuracy: 0.8537
Epoch 9/30
11/11 [=====] - 8s 5ms/step - loss: 0.5486 - accuracy: 0.7481 - val_loss: 0.4468 - val_accuracy: 0.8537
Epoch 10/30
11/11 [=====] - 8s 7ms/step - loss: 0.5222 - accuracy: 0.7481 - val_loss: 0.4308 - val_accuracy: 0.8537
Epoch 11/30
11/11 [=====] - 8s 6ms/step - loss: 0.5918 - accuracy: 0.7481 - val_loss: 0.4161 - val_accuracy: 0.8537
Epoch 12/30
11/11 [=====] - 8s 5ms/step - loss: 0.4791 - accuracy: 0.7481 - val_loss: 0.3996 - val_accuracy: 0.8537
Epoch 13/30
...
Epoch 29/30
11/11 [=====] - 8s 7ms/step - loss: 0.1816 - accuracy: 0.9602 - val_loss: 0.2374 - val_accuracy: 0.9396
Epoch 30/30
11/11 [=====] - 8s 5ms/step - loss: 0.1784 - accuracy: 0.9633 - val_loss: 0.2268 - val_accuracy: 0.9512
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings.

```

**Gambar 5. 9 Training Model “Picture”**

Hasil akurasi pada data *training* dan *validation* sebanyak 50 *epoch* dapat divisualisasikan dengan grafik berikut:



**Gambar 5. 10 Grafik Training Model “Document”**

### 5.2.2 Pengujian Aplikasi REST-API (Backend)

Secara umum pengujian Aplikasi REST-API (*Backend*) dilakukan dengan menggunakan *tools* postman. Berikut merupakan *route* yang di miliki oleh

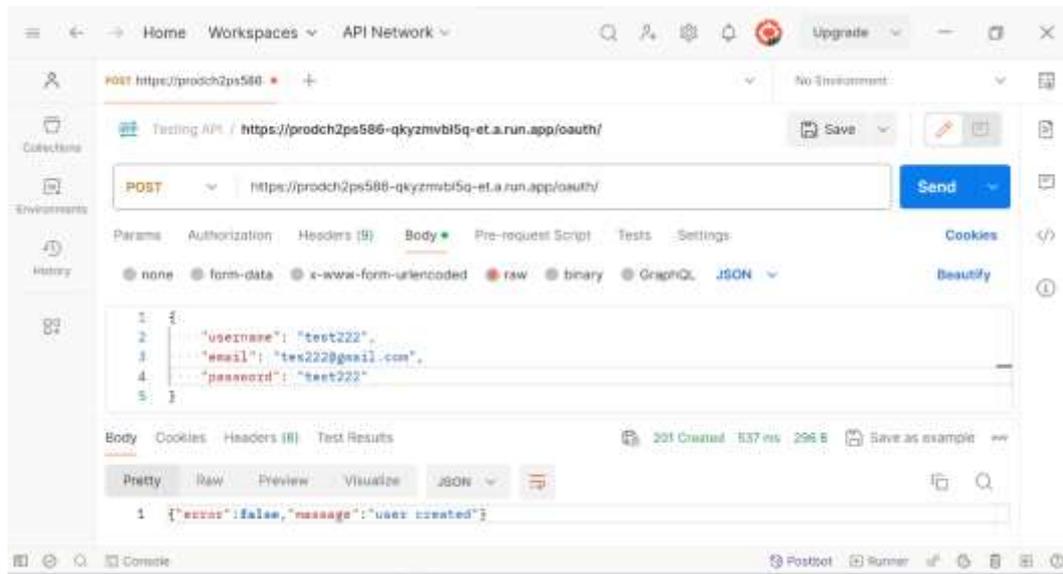
Aplikasi REST-API (*Backend*) yang dapat dilihat dengan menggunakan *swagger-ui*.



**Gambar 5. 11 List Route**

### 5.2.2.1 Testing route /oauth

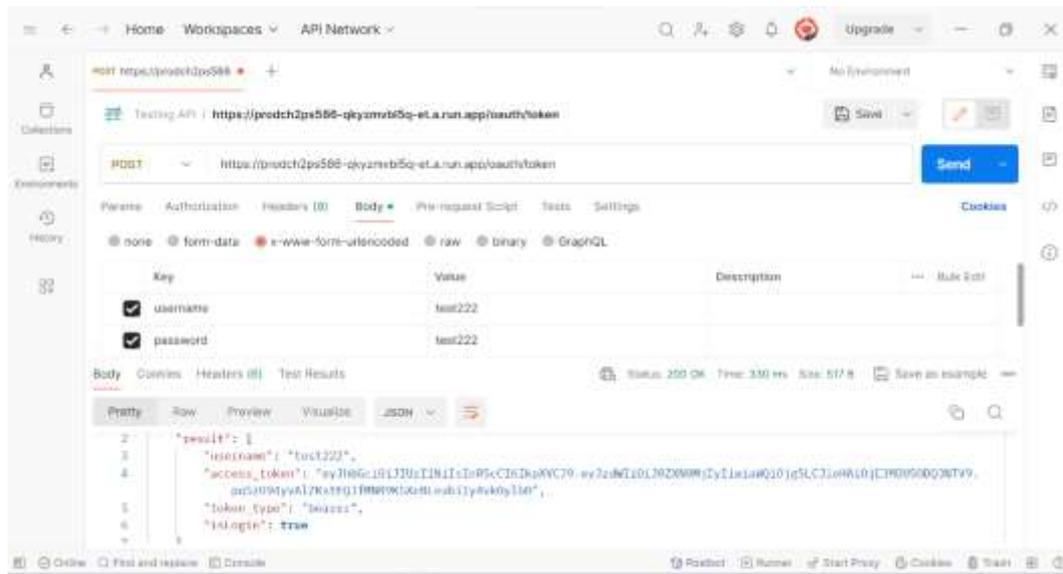
*Route* ini digunakan untuk melakukan registrasi. Pada *route* ini client akan mengirimkan *http request* dengan *method POST* dan mengirimkan *body* dengan format *JSON*.



**Gambar 5. 12 Postman Testing route /oauth**

#### 5.2.2.2 Testing route /oauth/token

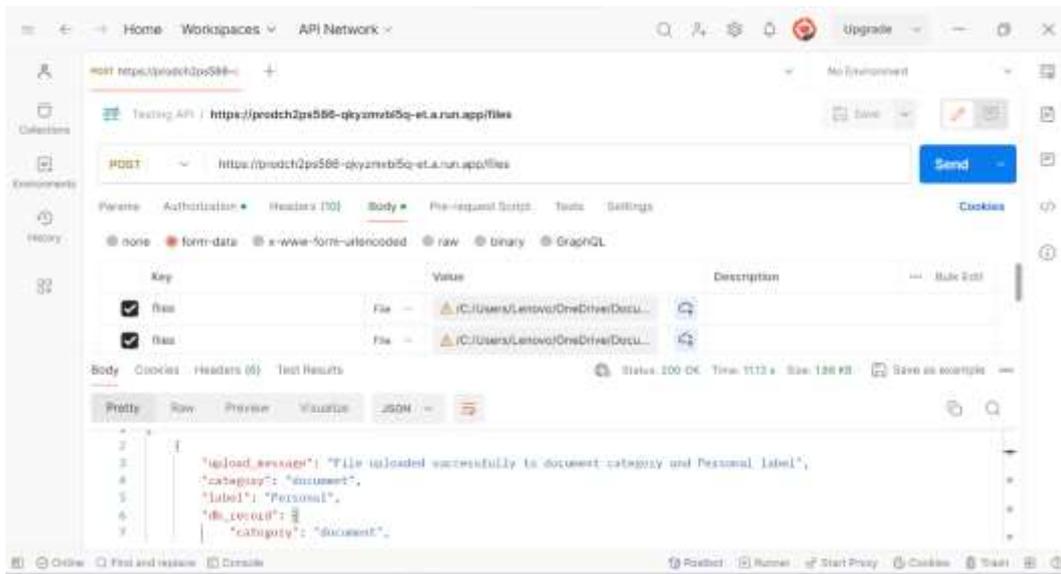
*Route* ini digunakan untuk melakukan *login*. Pada *route* ini *client* akan mengirimkan *http request* dengan *method POST* dan mengirimkan *body* dengan format *x-www-form-urlencoded* dan mengirimkan *username* dan *password*. Setelah *request* terkirim *username* dan *password* adalah tepat, *route* ini akan mengembalikan *response access\_token* yang dapat digunakan oleh untuk *authentication*.



**Gambar 5. 13 Postman Testing route /oauth/token**

### 5.2.2.3 Testing route /files

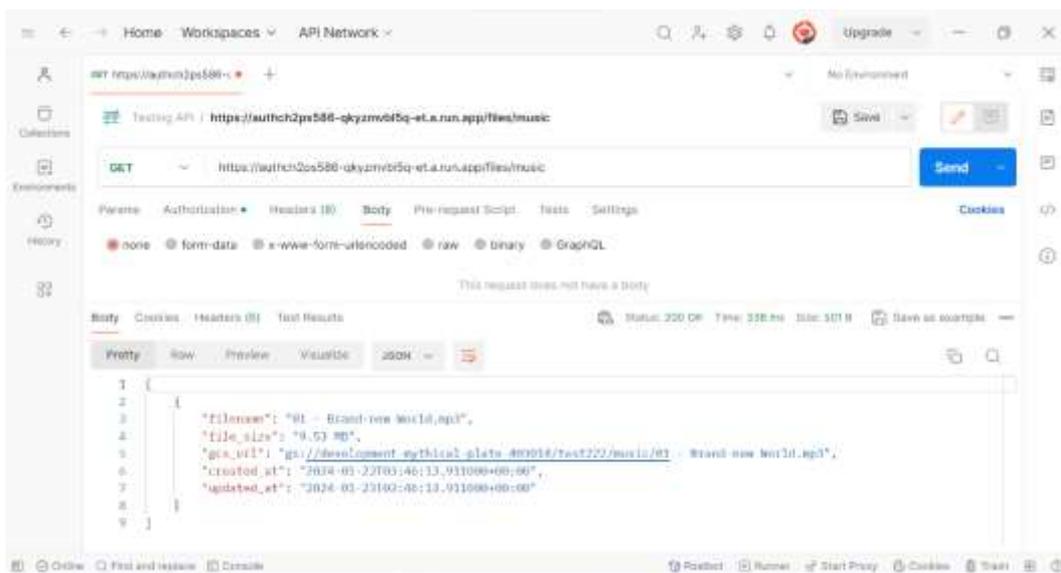
*Route* ini digunakan untuk melakukan *upload file*. Pada *route* ini *client* akan mengirimkan *http request* dengan *method POST* dan diharuskan mengirimkan *access token* yang di dapat pada saat *login*. *Route* ini menerima *request* dengan format *body form-data* dan *list* dari *file* yang ingin dikirimkan. Setelah *request* berhasil dikirim *route* ini akan mengembalikan beberapa data termasuk label kategori dan label *file* yang di tentukan dari *backend*.



**Gambar 5. 14 Postman Testing route /files**

#### 5.2.2.4 Testing route /files/{category}

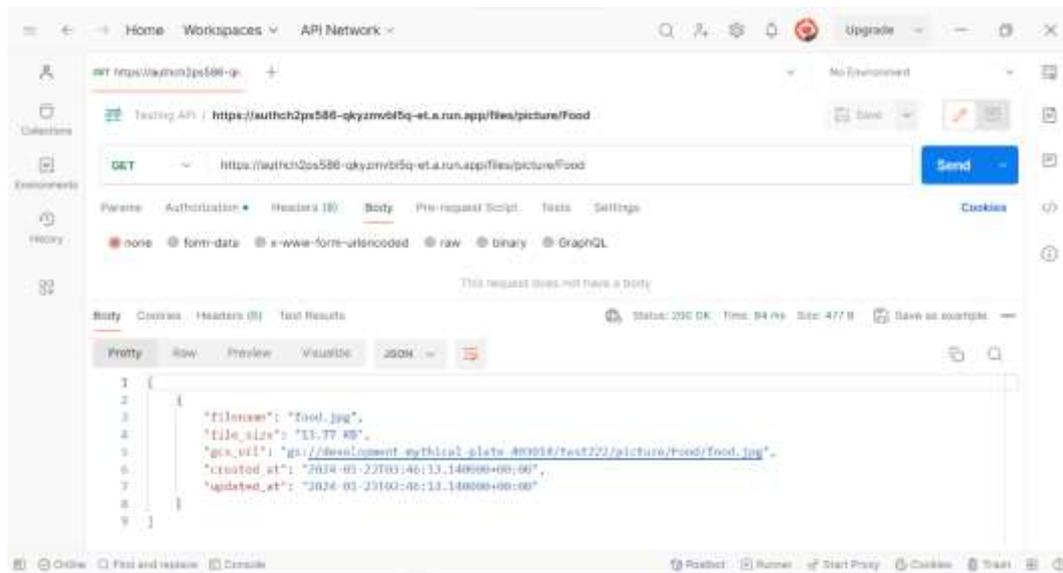
Route ini digunakan untuk memberikan *list* dari *file* dengan folder yang berkategori “*music*”, “*video*” dan “*others*”. Pada *route* ini *client* akan mengirimkan *http request* dengan *method GET* dan diharuskan mengirimkan *access token* yang di dapat pada saat *login*.



**Gambar 5. 15 Postman Testing route /files/{category}**

### 5.2.2.5 Testing route /files/{category}/{label}

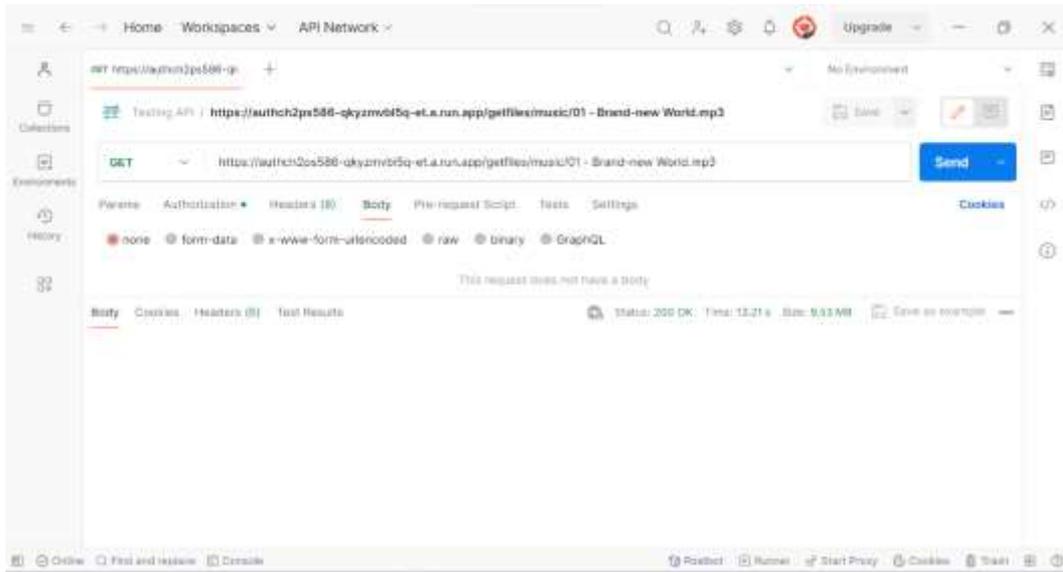
Route ini digunakan untuk memberikan *list* dari *file* dengan folder yang berkategori “*Picture*” dan “*Document*”. Pada *route* ini *client* harus mengirimkan *category* dan *label*. Pada *route* ini *client* akan mengirimkan *http request* dengan *method GET* dan diharuskan mengirimkan *access token* yang di dapat pada saat *login*.



**Gambar 5. 16 Postman Testing route /files/{category}/{label}**

### 5.2.2.6 Testing route /getfiles/{category}/{filename}

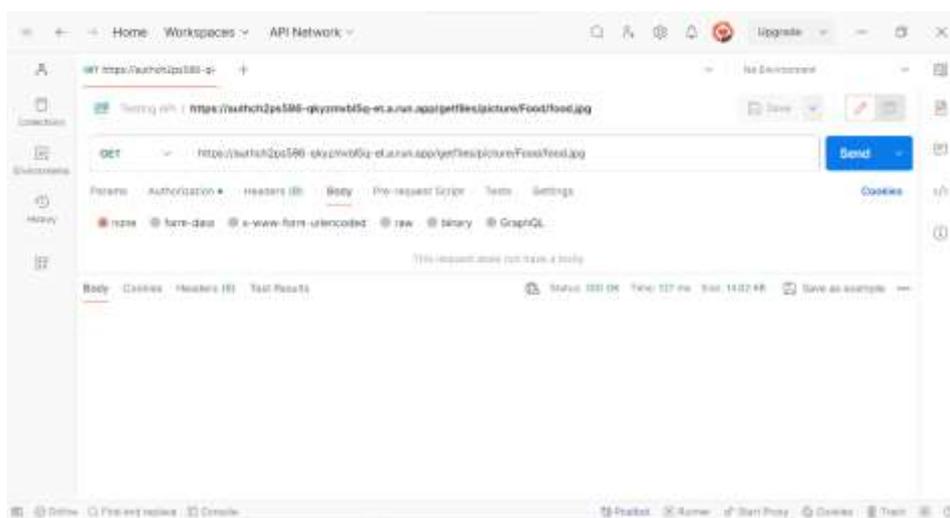
Route ini digunakan untuk mengunduh *file* dari *backend* dengan *file* yang berkategori “*music*”, “*video*” dan “*others*”. Pada *route* ini *client* harus mengirimkan *category* dan *filename*. Pada *route* ini *client* akan mengirimkan *http request* dengan *method GET* dan diharuskan mengirimkan *access token* yang di dapat pada saat *login*.



**Gambar 5. 17 Postman Testing route /getfiles/{category}/{filename}**

### 5.2.2.7 Testing route /getfiles/{category}/{label}/{filename}

Route ini digunakan untuk mengunduh *file* dari *backend* dengan *file* yang berkategori “*Picture*” dan “*Document*”. Pada *route* ini *client* harus mengirimkan *category*, *label* dan *filename*. Pada *route* ini *client* akan mengirimkan *http request* dengan *method GET* dan diharuskan mengirimkan *access token* yang di dapat pada saat *login*.



**Gambar 5. 18 Postman Testing route /getfiles/{category}/{label}/{filename}**

### 5.3 ANALISIS HASIL YANG DICAPAI OLEH SISTEM DAN PERANGKAT LUNAK

Analisis hasil yang dicapai oleh sistem dan perangkat lunak FileManager.AI yang telah dikerjakan oleh tim CH2-PS586 selama satu bulan adalah langkah penting untuk mengevaluasi kualitas, kinerja, dan hasil dari proyek tersebut. Berikut adalah beberapa aspek analisis:

1. Kelebihan aplikasi FileManager.AI
  - a. Aplikasi FileManager.AI memberikan fitur utama yang menjadi kebutuhan khalayak umum. Dengan adanya FileManager.AI pengguna mendapatkan efisiensi *file* yang di atur secara otomatis.
  - b. Fitur utama FileManager.AI yaitu *file* yang di atur secara otomatis, saat ini masih belum tersedia pada *Cloud Storage* populer seperti *Google Drive* dan *One Drive*. Hal ini memberikan nilai tambah pada aplikasi FileManger.AI dan juga potensi aplikasi FileManger.AI di akuisisi oleh *Cloud Storage* populer.
2. Kekurangan Aplikasi FileManager.AI
  - a. Untuk aplikasi FileManager.AI digunakan secara efektif, FileManager.AI perlu mengembangkan aplikasi menjadi *multi-platform* yang dapat di gunakan pada *Android*, *IOS*, *Windows*, *Linux*, dan *Web*. Namun saat ini karena keterbatasan waktu dan biaya *client* yang digunakan hanyalah *Android*.