REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN

CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202003141, 23 Januari 2020
Pencipta		
Nama	:	**Mahmud, Yesi Novaria Kunang, , dkk**
Alamat	:	Universitas Bina Darma, Jl. A. Yani No. 3 Plaju, Palembang, Sumatera Selatan, 30264
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Mahmud, Yesi Novaria Kunang, , dkk**
Alamat	:	Universitas Bina Darma, Jl. A. Yani No. 3 Plaju, Palembang, 5, 30264
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**Aplikasi Pengelompokan Dokumen Teks Secara Otomatis**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	23 Januari 2020, di Palembang
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000176648

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM DAN HAK ASASI MANUSIA
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL

Dr. Freddy Harris, S.H., LL.M., ACCS.
NIP. 196611181994031001

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Mahmud	Universitas Bina Darma, Jl. A. Yani No. 3 Plaju
2	Yesi Novaria Kunang	Jl Kijang Mas Bliok E 12 No. 4431 Rt 41/ Rw 11 Demang Lebar Daun
3	Ilman Zuhri Yadi	Jl Kijang Mas Bliok E 12 No. 4431 Rt 41/ Rw 11 Demang Lebar Daun

LAMPIRAN PEMEGANG

No	Nama	Alamat
1	Mahmud	Universitas Bina Darma, Jl. A. Yani No. 3 Plaju
2	Yesi Novaria Kunang	Jl Kijang Mas Bliok E 12 No. 4431 Rt 41/ Rw 11 Demang Lebar Daun
3	Ilman Zuhri Yadi	Jl Kijang Mas Bliok E 12 No. 4431 Rt 41/ Rw 11 Demang Lebar Daun

Aplikasi Pengelompokan Dokumen Teks Secara Otomatis

Pengembang:

1. Mahmud
2. Yesi Novaria Kunang
3. Ilman Zuhri Yadi

1. Pengenalan sistem

Aplikasi pengelompokan dokumen secara otomatis merupakan program komputer yang bisa mengolah dan mengelompokkan dokumen teks serta meberi label isi dokumen secara otomatis. Kemampuan program ini mampu mengelompokan seluruh dokumen berbasis text dalam format pdf, txt, doc, rtf, serta didukung kemampuan mengelompokkan dokumen dua bahasa, yakni bahasa Inggris dan bahasa Indonesia. Aplikasi atau program ini bisa dimanfaatkan oleh perpustakaan dan bagian adminitrasi yang bertugas melakukan sorting dokumen. Untuk mempermudah pengguna, aplikasi ini disiapkan untuk bisa diinstal dan langsung bisa diakses melalui *browser*, sehingga bisa digunakan oleh banyak user. Dengan aplikasi/program ini operator atau petugas tinggal mengupload file/dokumen di *browser*, kemudian aplikasi akan secara otomatis mengelompokkan dokumen yang sejenis berdasarkan kemiripan isi dokumen. Sehingga dengan aplikasi ini bisa membantu operator mengelompokkan dokumen yang jumlahnya ribuan tanpa harus membaca isi dokumen dalam waktu singkat.

Sistem pengelompokan dokumen secara otomatis merupakan suatu sistem yang dibuat dengan menggunakan NodeJs sebagai pemrograman server aplikasi dan Python dalam proses pengolahan dokumen berbasis teks. Ditenagai dengan NodeJs yang bersifat *asynchronous* dan menggunakan layanan *massage broker* dari *RabbitMq* yang bertugas dalam pertukaran data antar program utama dan program yang bertugas dalam proses pemrosesan data sehingga pemrosesan akan lebih cepat. Pengguna tidak harus menunggu hingga pemrosesan suatu dokumen selesai pengguna hanya menungu proses upload selesai. Setelah proses upload selesai penguna bisa mengerjakan hal lain. Pengguna akan mendapatkan notifikasi di pojok kanan atas apabila pemrosesan dokumen seleasi di proses. Sistem ini Menggunakan algoritma *Kmeans* dalam proses *clusstering* dokumen teks. Sistem ini mampu mengelompokan seluruh dokumen berbasis text yang didukung dengan dua bahasa, yakni bahasa Inggris dan bahasa Indonesia. Sistem ini dapat di instal baik di

Sistem Operasi Windows maupun di Linux. Sistem ini memiliki dua user yaitu admin dan operator.

2. Penginstalan sistem

Dalam menggunakan program Pengelompokkan Dokumen Otomatis, pengguna haruslah terlebih dahulu memiliki beberapa komponent yang harus dipersiapkan terlebih dahulu diantaranya ialah sebagai berikut:

a. Java Sdk

Pengguna haruslah terlebih dahulu menginstall Java Sdk di komputer agar bisa menjalankan aplikasi. Java Sdk digunakan untuk menjalankan framework Apache Tika. Apache tika merupakan suatu framework yang terbuat dari bahasa Java bertugas dalam proses pembacaan dokumen. Dipilihnya Apache tika dibanding dengan module python ialah karena apache tika dapat membaca seluruh tulisan yang ada di dokumen berbasis text. Pengguna dapat menginstall Java Sdk di official resmi penyedia Java Sdk yakni di ww.oracle.com/technetwork/java/javase. Disarankan untuk menginstall java sdk versi 8 keatas. Setelah pengguna menginstall Java Sdk maka step selanjutnya pengguna diharuskan membuat path System Variabel. penguna bisa menambahkan environment variabel dnegan mengedit file *~/.bashrc* yang ada di folder home. Proses penambahan system variabel bisa di gambarkan pada gambar 1.

```
# export java path

export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_221/
#export JAVA_HOME='/usr/lib/jvm/jre1.8.0_211/'
export PATH=$PATH:$JAVA_HOME/bin
export LD_LIBRARY_PATH==/usr/lib/jvm/jre1.8.0_211/lib/amd64:/usr/lib/jvm/jre1.8.0_211/lib/amd64/server
#export LD_LIBRARY_PATH=/usr/lib/jvm/jre1.8.0_211/lib/amd64/server/

# exports andoid path
export ANDROID_HOME=/media/myone/Data_mahmud/AndroidSdk/
export PATH=$PATH:$ANDROID_HOME/tools/bin
export PATH=$PATH:$ANDROID_HOME/platform-tools
```

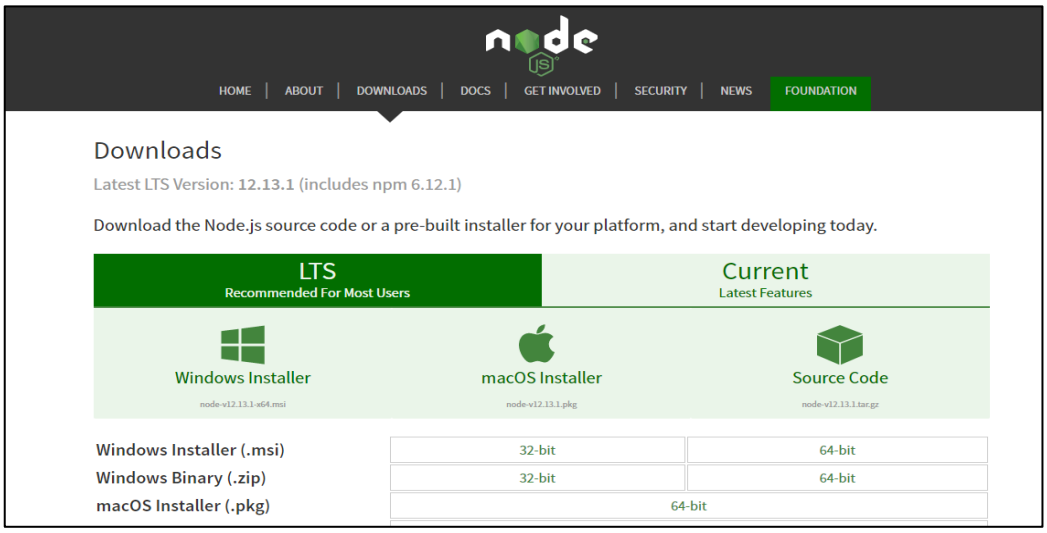
Gambar 1. Penambahan *Environment* Variabel di Linux

b. Python3

Python digunakan untuk menjalankan aplikasi script python yang bertugas dalam pemrosesan dokumen mulai dari membaca dokumen, parsing dokumen, *remove stopword*, *steaming* hingga proses *clusstering*. Proses pembacaan dokumen dilakukan dengan memanggil *class Java* yang ada pada Apache tika kemudian dicompile class bahasa c++ kemudian dijalankan dengna bantuan module *Cpython.* Pengguna bisa mendownload python3 di terminal dengan mengetik sudo apt-get install python3 atau mengunjungi situs resmi di www.python.org

c. Node Js

NodeJs Merupakan javacript runtime yang dibangun dengan enggine V8 chrome. Penggunaan nodejs ialah guna dapat menjalankan aplikasi utama. Pengguna dapat mendownload dan mengistallnya dengan mengunjungi www.nodejs.org. Untuk mendownload NodeJs bisa dilihat pada gambar 2.



Gambar 2. Download NodeJs *LTS*

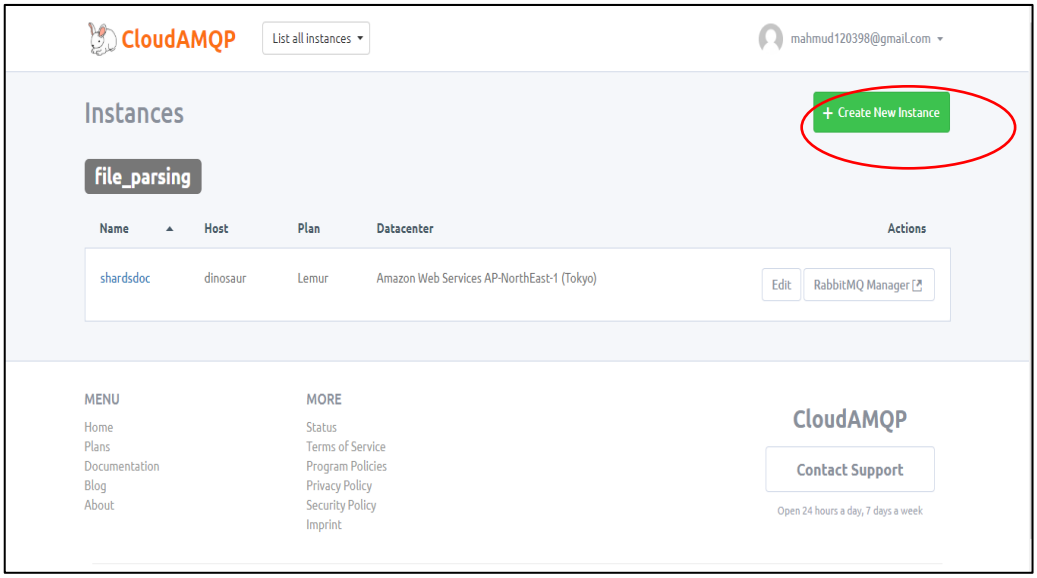
d. Daftar Layanan *Message broker CloudAMQP*

Pengguna selanjutnya diharuskan untuk mendafatar layanan *online* dari **rabbitMq cloud**. Layanan RabbitMq ini berfugsi sebagai *message broker* yang

dapat mendistribusian data dari program satu ke program lain. Dengan menggunakan layanan message broker yang ada pada layanan *rabbitmq cloud* sistem aplikasi dapat berjalan dengan lebih cepat. Program dipisah antara program yang bertugas *read, parsing hingga clusstering* dokumen dengan program yang bertugas sebagai server, sehingga estimasi waktu bisa ditekan.

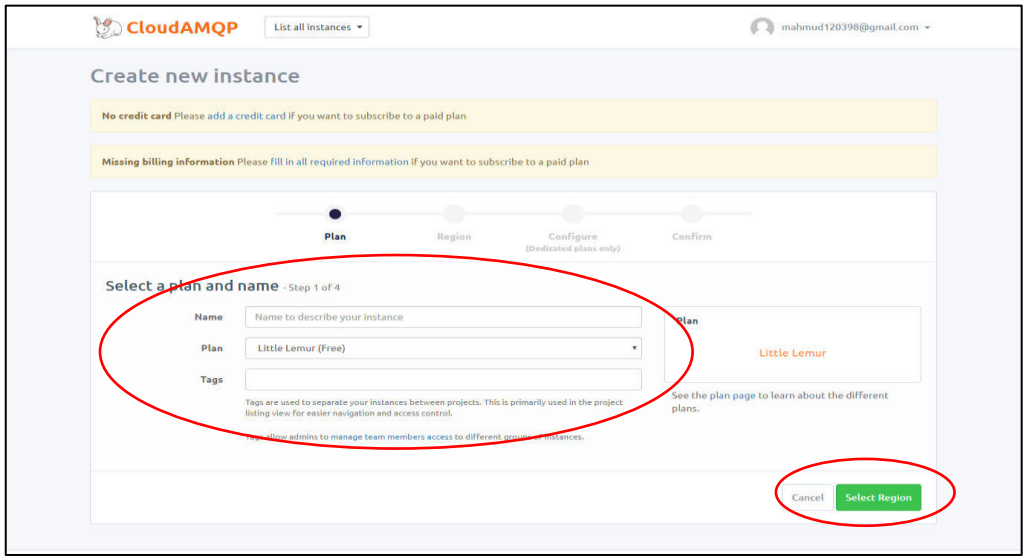
Sebagai contoh pengguna mengupload file PDF dengan size 100Mb, apabila kita tidak menggunakan message broker maka kita akan menunggu sampai proses *upload*, *read*, *parsing* hingga *clusstering* dokumen selesai. Proses tersebut tentunya tidak akan memakan waktu lama apabila kita menggunakan layanan *message broker* karena pemrosesan suatu dokumen bisa di-*handle* oleh script atau program lain yang berjalan di latar belakang (*process background*). Pengguna hanya menunggu sampai proses upload selesai. Setelah pengguna seleai meng-*upload* dokume, pengguna dapat membuka layanan lain tanpa harus mengunggu proses pengkategorian dokumen selesai. Pengguna akan mendapatkan notifikasi apabila proses pengkategorian yang berjalan di *background* selesai.

Pengguna bisa layaaan dari www.cloudamqp.com untuk mendapatkan layanan gratis dari *massage broker*. proses dibawah ini menganggap pengguna sudah mendfatar dan login di layanan cloudamqp.com.

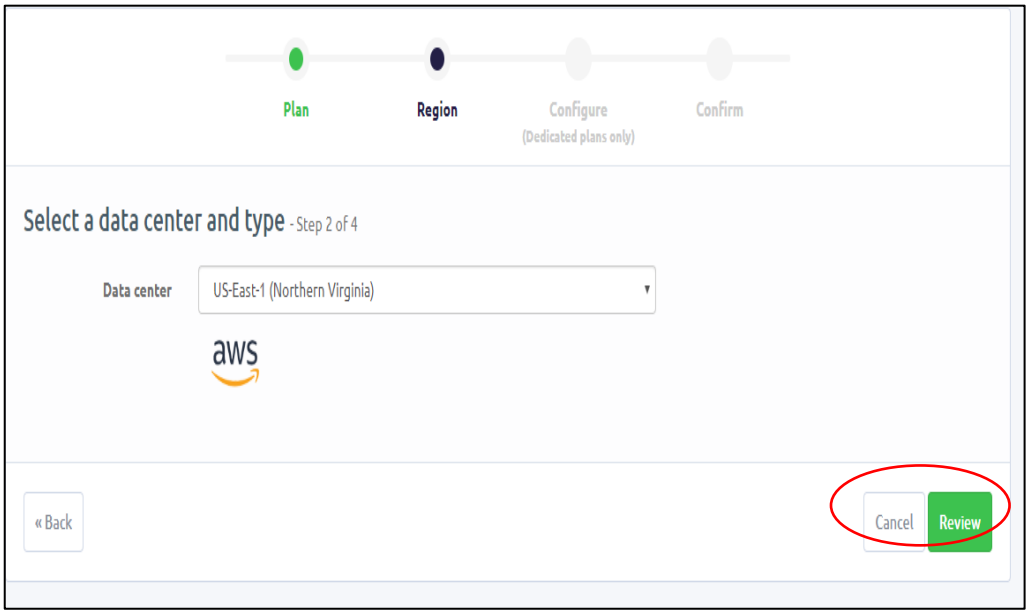


Gambar 3. Membuat Layanan *Message Broker Cloudmqp*

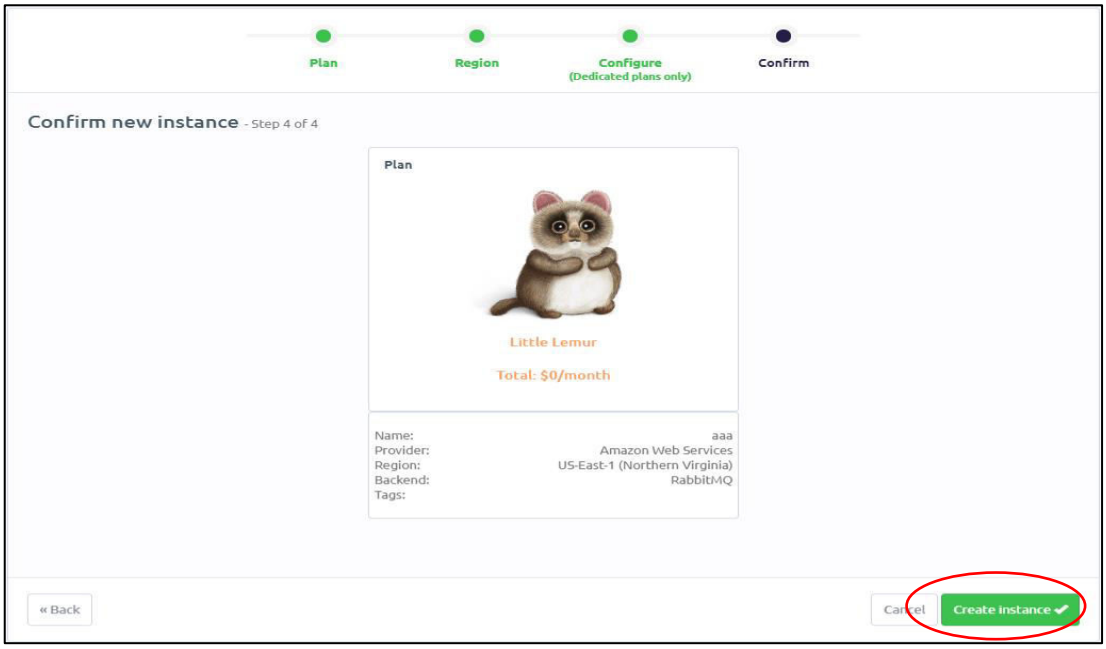
Setelah pengguna mengklik *create instance* maka pengguna akan *diredirect* ke halaman selanjutnya yang mana pada halaman ini pengguna mengisi nama dari instance rabbitmq dan tipe *intance*. Pengguna bisa mengisi nama instance seseuai dengan yang diinginkan dan *litle lemule lemur (free*) sebagai tipe *intance*. Untuk tags pengguna bisa mengosongkan kemudian klik region. Untuk proses ini bisa dilihat pada gambar 3 dibawah ini .



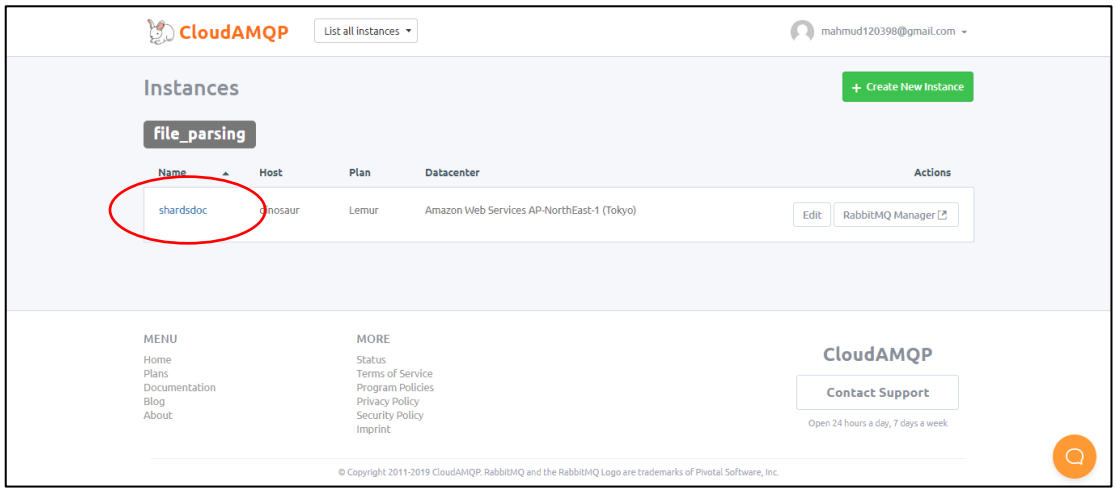
Gambar 4. Isi Nama dan Pilih Tipe *Cloud Instance*



Gambar 5. *Select Data Center Region*

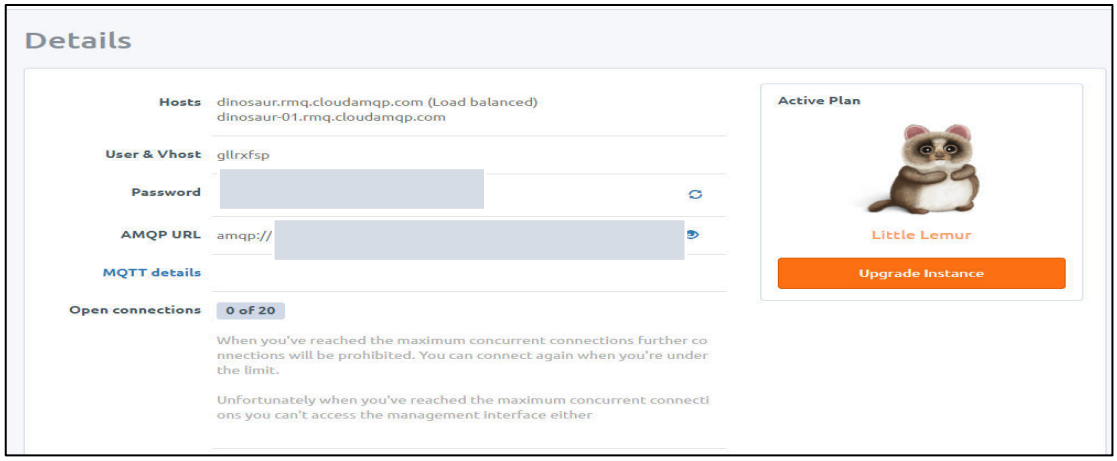


Gambar 6. *Confirm New Instance*



Gambar 7. Contoh Nama *Instance* yang Telah Dibuat

Setelah pengguna mengikuti langkah langkah diatas maka pengguna akan ditampilkan ke halaman deskripsi dari *instance* RabbitMQ yang telah dibuat. Terdapat nama, *host, password, Amqp host* dan beberapa setingan lainnya. *Amqp* ini berguna sebagai *url* layanan *message broker.* proses ini bisa dilihat pada gambar 8.



Gambar 8. Deskripsi dari *Instance* yang Telah Dibuat

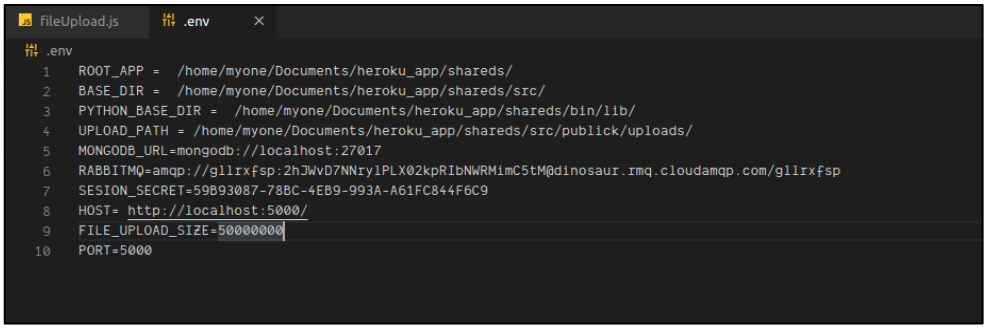
e. Install Mongodb Database

Database *Mongodb* digunakan sebagai proses penyimpanan data. Pengguna bisa mendownload dan menginstall aplikasi ini dengan mengikuti tutorial di link https://docs.mongodb.com/manual/installation/.

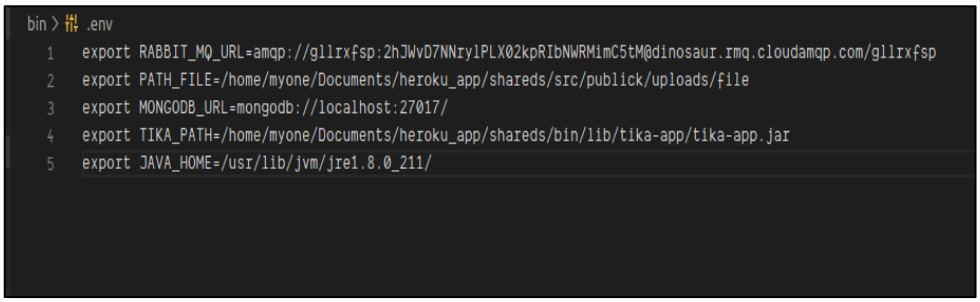
f. Seting *Environment Variabel*

Environmen variabel merupakan sebuah variabel dinamis yang digunakan oleh suatu program. Biasanya *environment variabel* ini terletak pada file **.env** pada *root project.* Dalam project ini terdapat dua file *environment variabel* dengan fungsi yang berbeda-beda. *Environment* yang terletak di folder root berguna untuk variabel global aplikasi. Sedangkan *environment* variabel yang terletak di folder bin berguna sebagai *environment* pemrosesan file yang dikerjakan oleh *script* Python. Berikut ini gambar dari *environment variabel* yang digunakan dalam menunjang aplikasi ini dapat berjalan.

Sesuaikan dengan *environment variabel* disini dengan settingan kita. Seperti *rabbitmq* yang merupakan host dari layanan *message broker* yang kita daftarkan sebelumnya. Untuk lebih jelasnya bisa dilihat pada gambar 9 dan 10.



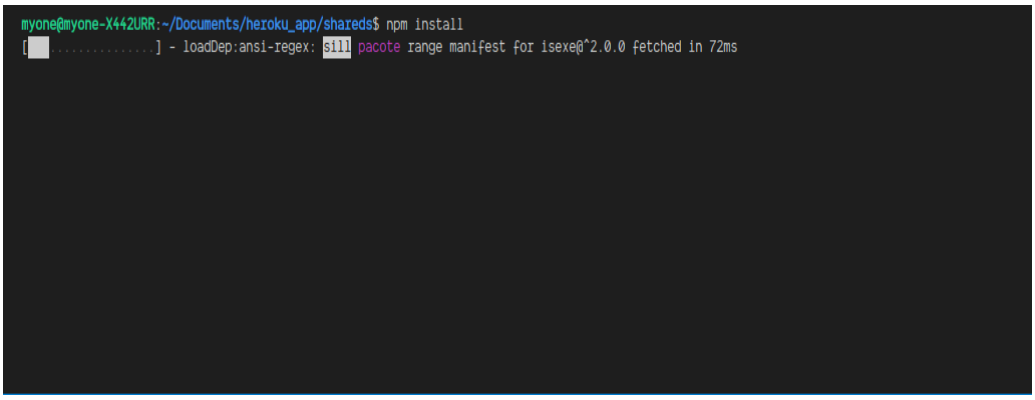
Gambar 9. *Environment Variabel* Aplikasi Utama



Gambar 10. *Environmen* Python *script* untuk Pemrosesan File

g. Instal *Dependecies* Aplikasi

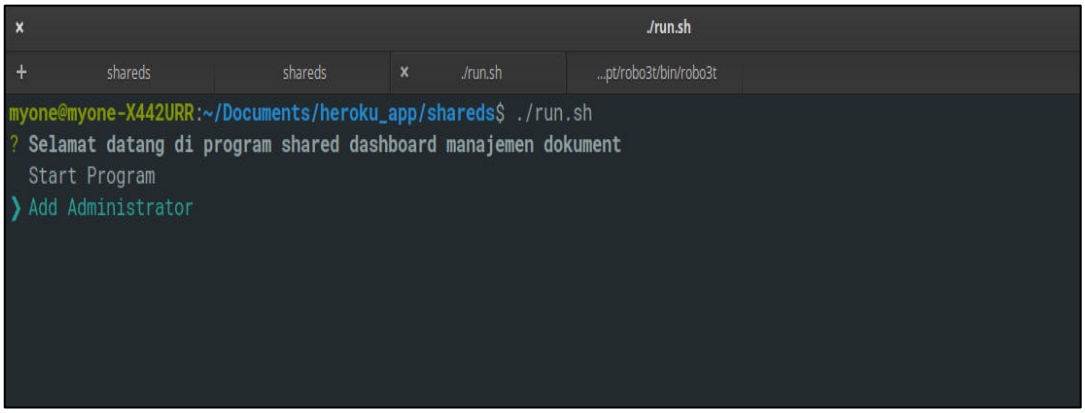
Sistem pengelompokan aplikasi ini dibuat dengan menggunakan NodeJs sebagai bahasa server dan menggunakan *framework express* sehingga kita perlu menginstallnya agar aplikasi dapat berjalan. Pengguna cukup membuka terminal *foder root* aplikasi kemudian ketik **npm install.** Proses bisa dilihat pada gambar 11.



Gambar 11. Proses *Install Dependencis* Aplikasi

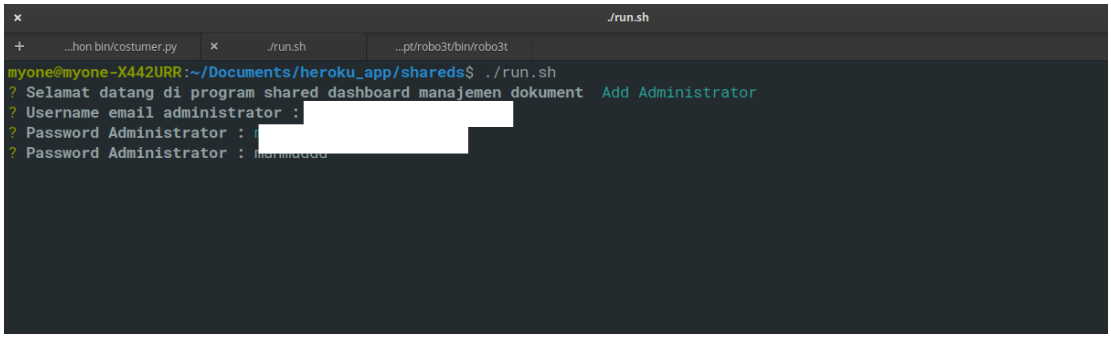
3. Menjalankan Aplikasi

Setelah semua konfigurasi selesai maka pengguna bisa manggunakan aplikasi yakni dengan mengklik run.bat untuk pengguna windows dan run.sh untuk linux di *folder root* project. proses berikuti ini bisa dilihat pada gambar 12 dibawah ini.



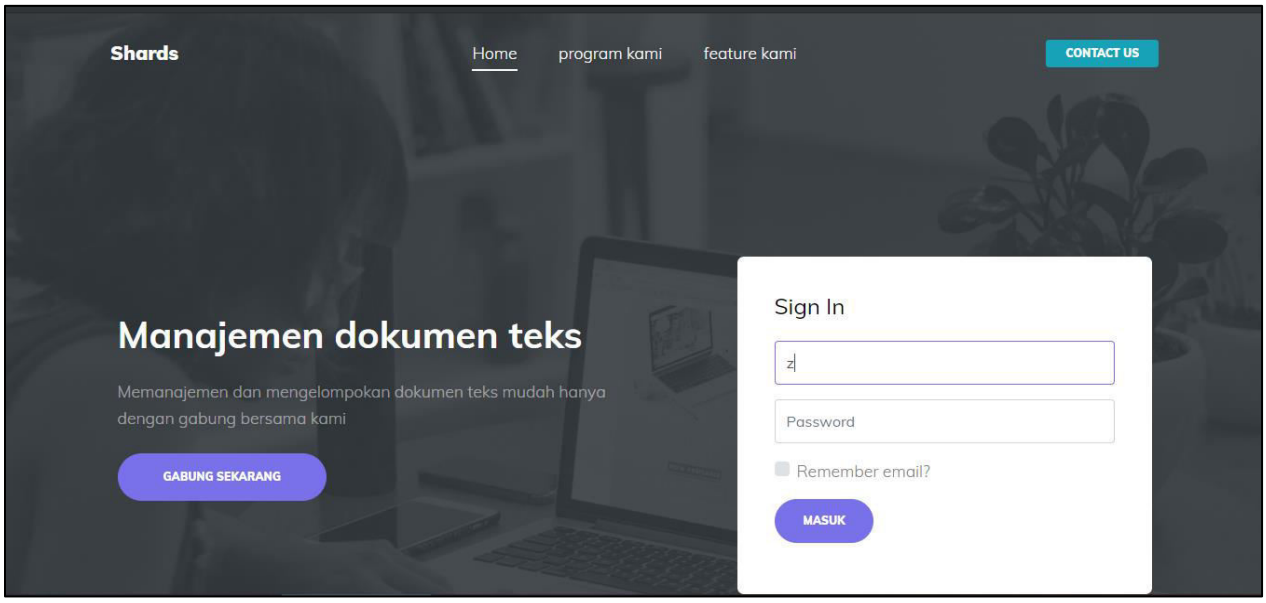
Gambar 12. Proses Awal menjalankan Aplikasi di Linux

Pada awal pembukaan aplikasi pengguna akan ditampilkan 2 menu menu pertama ialah menu untuk start program. Dan menu kedua ialah add administrator user. Pengguna yang pertama kali membuka aplikasi diharuskan untuk membuat user administrator agar pengguna dapat membuka menu administrator aplikasi. Proses ini bisa di lihat pada gambar berikut ini.



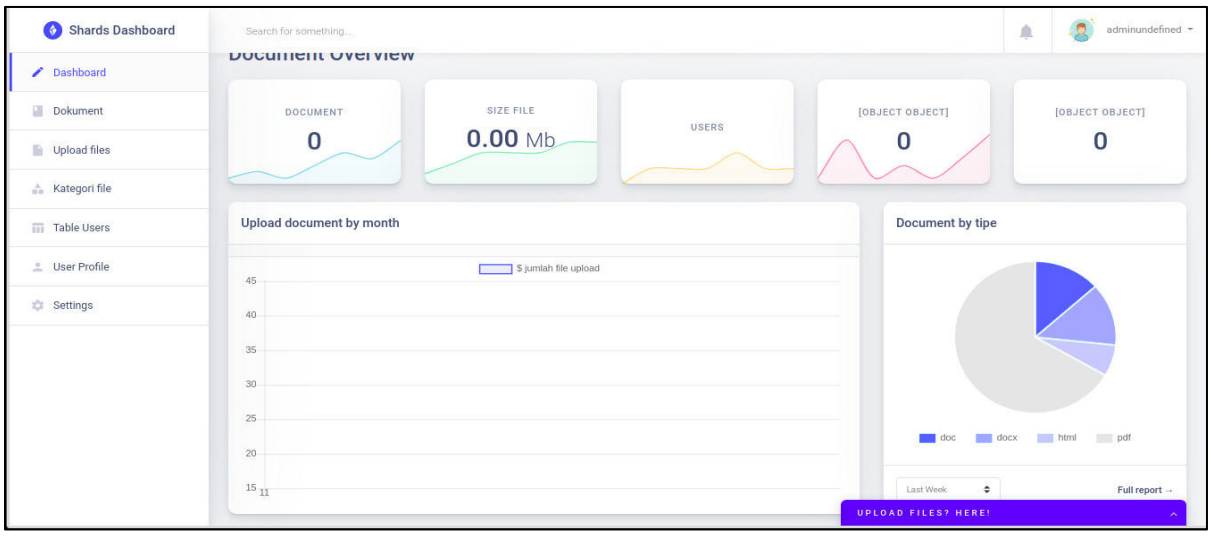
Gambar 13. Proses Pembuatan Administrator User

Setelah pengguna membuat administrator user maka pengguna akan di-*redirect* kehalaman web aplikasi. Pengguna dapat langsung login dengan user yang telah dibuat pada proses sebelumnya. Proses ini bisa dilihat pada gambar 14.



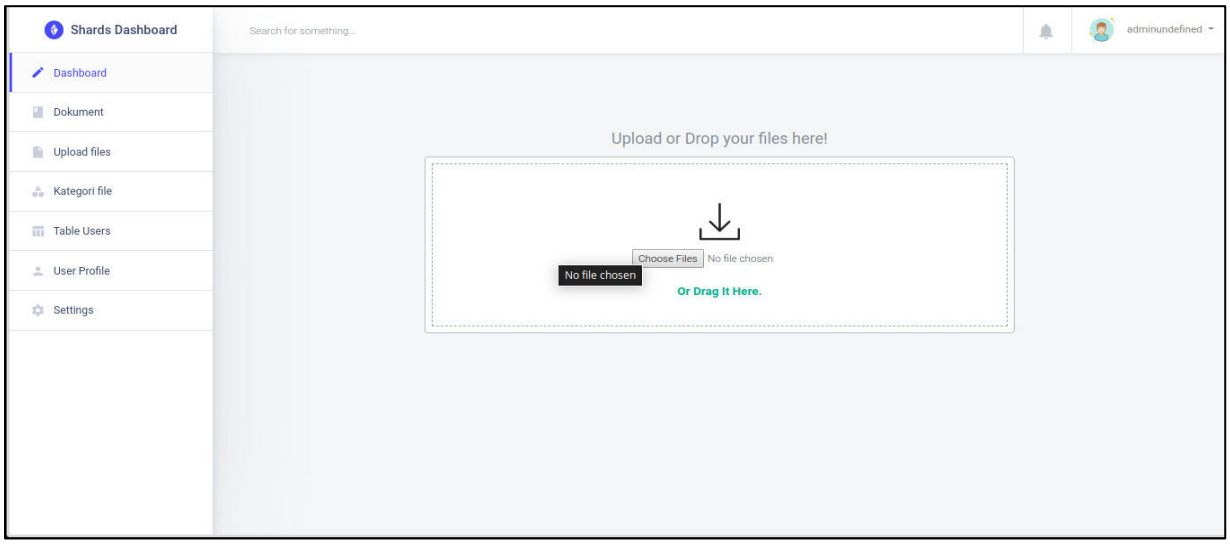
Gambar 14. Halaman Login Sistem

Setelah pengguna login maka pengguna akan di-*redirect* di halaman *dashboard.* Halaman awal ketika user telah melakkukan proses login. Halaman *dashboard* dapat dilihat pada gambar 15.



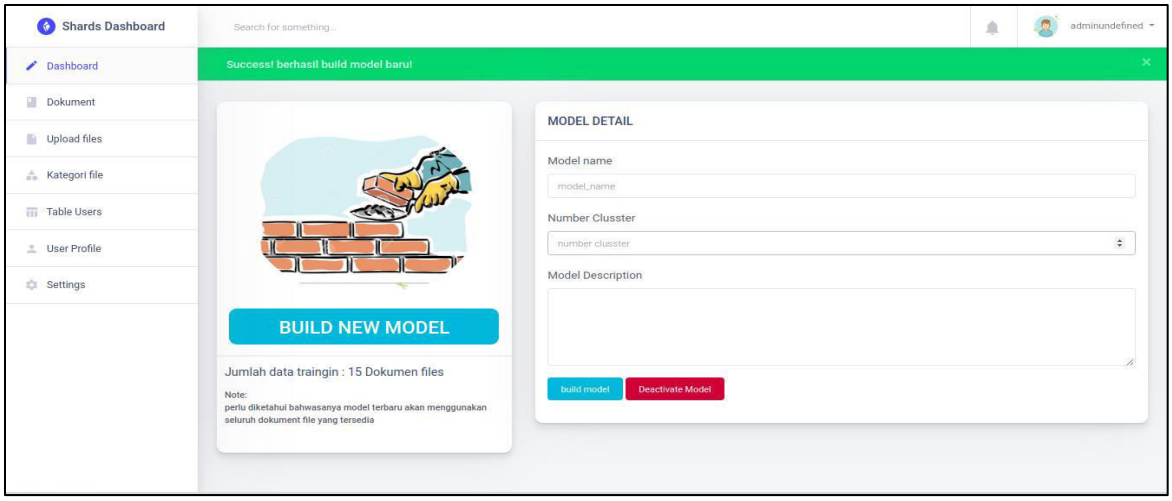
Gambar 15. *Dashboard* Sistem

Sistem ini belum dibasa dijalakan karena pemrosesan *clusstering* membutuhkan data sample awal dalam proses *clustering*. Nantinya seluruh dokumen akan di kelompokan dari sample *clustering* yang telah di daftarkan dalam sebuah model KMeans. Administrator user dapat menginputkan minimal 11 dokumen kedalam suatu sistem sebelum sistem ini bisa dijalankan. Administrator bisa melakukan proses ini dengan mengklik menu upload file disebelah kiri kemudian user akan di-r*edirect* dalam halaman proses upload file. Proses ini bisa dilihat pada gambar 16.



Gambar 16. Proses *Upload* Dokumen Teks.

Setelah administrator selesai menginputkan minimal 11 sample dokumen maka administrator bisa mengklik untuk membuat suatu model. Klik menu seting dan admin akan mendapatkan halaman untuk proses *generate* model dan pelabelan dokumen. Proses ini bisa dilihat pada gambar 17 dibawah ini.



Gambar 17. Proses Buat Model *Clustering*

Nama filed ialah nama dari model *clustering* yang ingin kita buat. Sedangkan *number cluster* berfungsi sebagai jumlah kategori *(cluster).* Sedangkan *model description* berfungsi sebagai deskripsi lengkap atas model yang kita buat. Apabila proses selesai maka pengguna akan mendapatkan notifikasi build model clusstering selesai. Setelah ini sistem ini siap untuk digunakan.

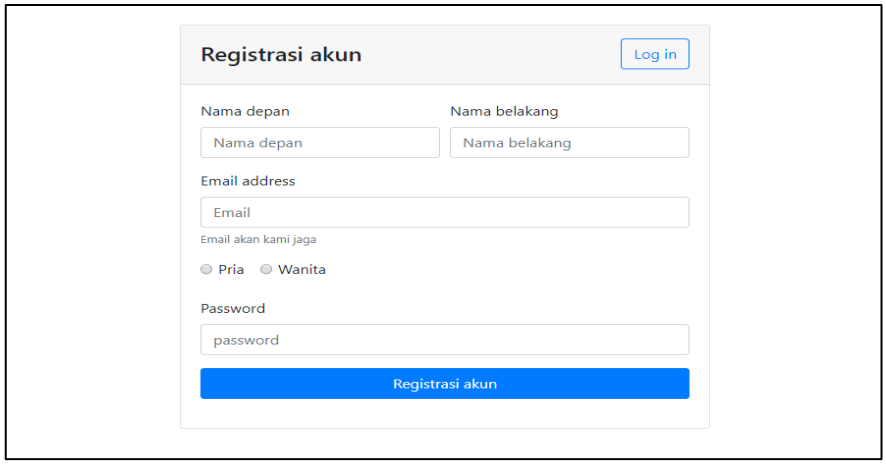
4. Halaman-Halaman Sistem

Dalam sistem ini memiliki dua type user, user pertama sebagai admin dan user kedua sebagai user biasa. Keduanya memiliki menu maupun fiture yang agak sedikit berbeda dimana admin memiliki menu tambahan dan fiture tambahan dibanding user biasa. Yang membedakan antara admin dan user biasa ialah dimana user biasa hanya memiliki beberapa menu diantaranya menu dashboard, menu upload files, menu kategori file, menu user profile. Sedangkan admin memiliki tambahan menu yakni menu table users, dan menu setting. User biasa hanya bisa melihat atas file-file yang diupload nya tetapi admin dapat melihat semua file yang telah di input oleh user biasa.

Berikut ini beberapa menu dan page halaman dalam *an automatic document clusstering.*

a. Halaman registrasi akun

Merupakan halaman yang berfungsi untuk proses pendaftaran pengguna baru.

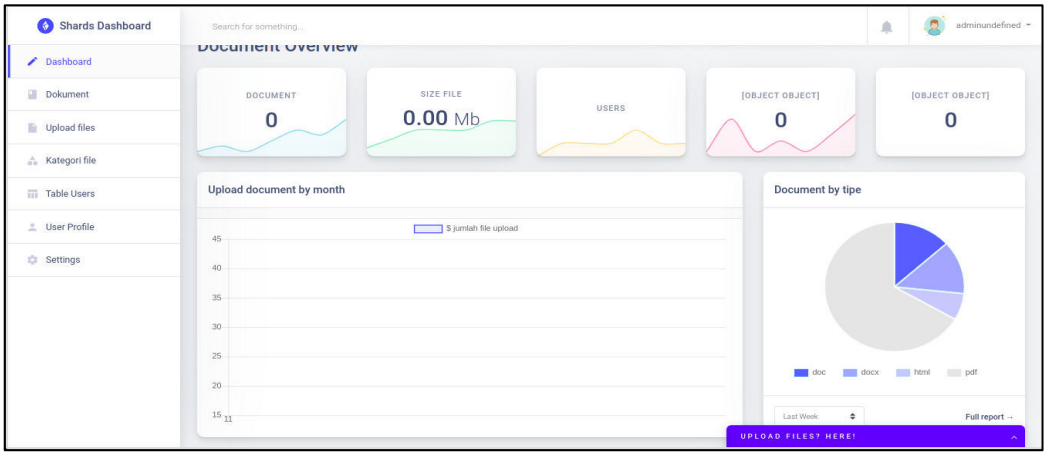


Gambar 18. Halaman Register

b. Menu utama sistem

Merupakan menu pertama kali yang ditampilkan ketika user admin login ke dalam suatu sistem. Dalam menu dashboard ini terdapat beberapa fiture diantaranya ialah sebagai berikut.

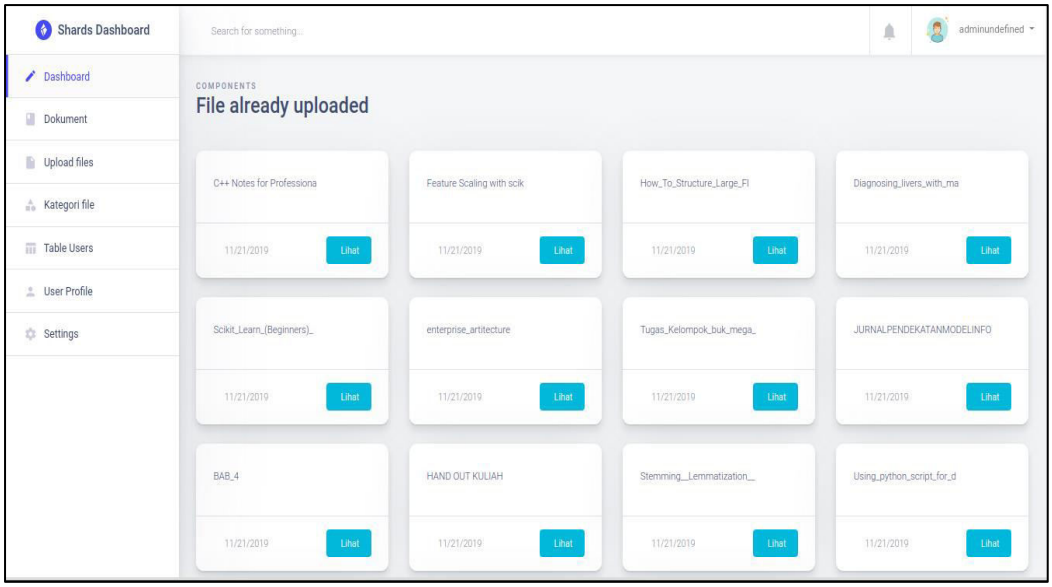
1. Jumlah dokumen yang diupload secara keseluruhan.
2. Jumlah *size* dari seluruh dokumen semua user.
3. Jumlah user yang telah menggunakan sistem.
4. Jumlah dokumen berbahasa inggris.
5. Jumlah dokumen berbahasa indonesia.
6. Grafik line peng*upload*-an dokumen
7. Grafik *pie chart* mengenai tipe dokumen yang di upload.



Gambar 19. *Dashboard* Aplikasi Setelah User *Login*

c. Menu dokumen

Meampilkan seluruh dokumen berberntuk card yang telah di upload dari seluruh user yang menggunakan sistem.

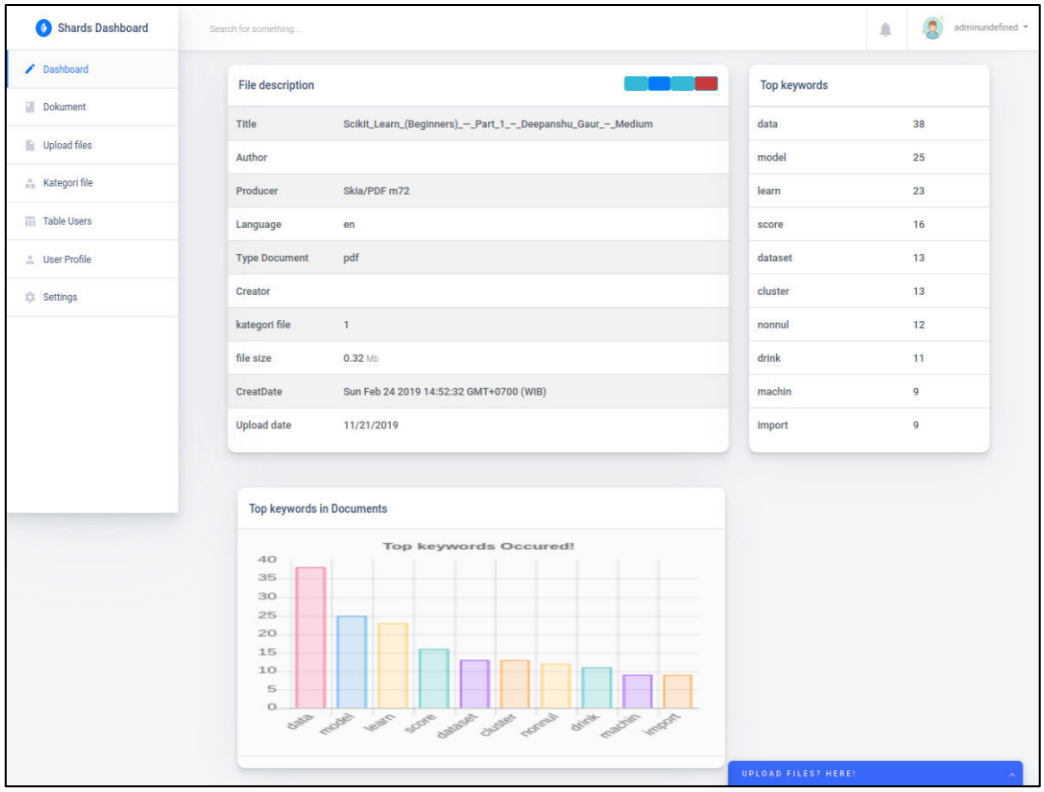


Gambar 20. Menu Dokumen

Dalam menu dokumen terdapat beberapa halaman diantarnya ialah sebagai berikut:

1. Halaman Deskripsi dokumen.

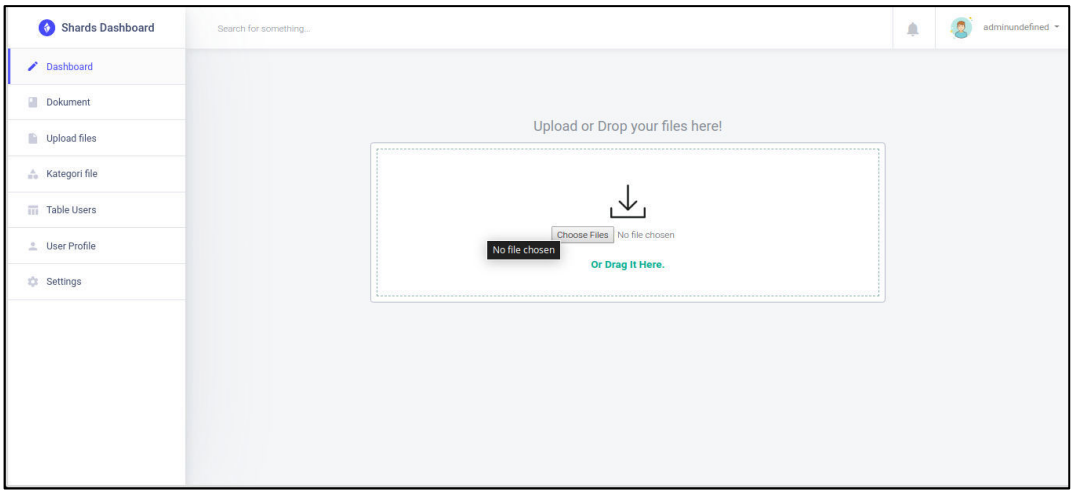
Merupakan halaman ini berisi mengenai deskripsi lengkap dari suatu dokumen. Mulai dari author, publisher, tanggal pembuatan, tanggal diedit, size dokumen, tipe dokumen dan lain lain. Dalam halaman ini juga terdapat *top keywords* yang dapat dilihat lewat *chart* maupun list *keywords*. Untuk lebih jelasnya bisa dilihat pada gambar dibawah ini.



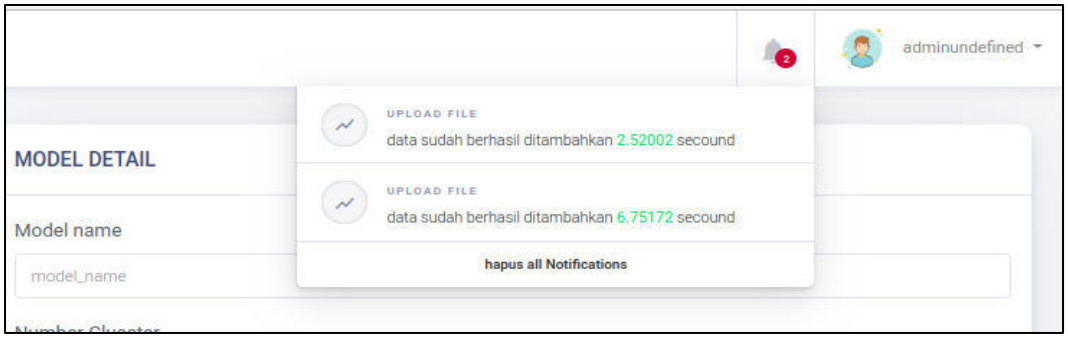
Gambar 21. Deskripsi Dokumen

d. Menu upload Dokumen

Merupakan menu yang berfungsi untuk proses *upload* dokumen berbasis teks. Pengguna bisa memilih dokumen atau dengan menggunkan *drop* and *drag* dokumen. Proses ini bisa dilihat pada gambar dibawah ini.



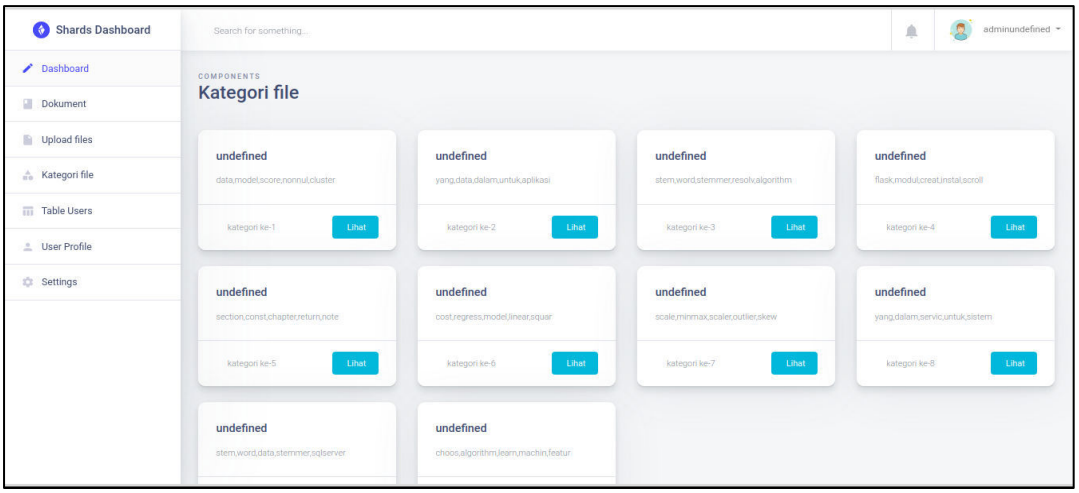
Gambar 22. Menu Upload File



Gambar 23. Notififikasi Proses Pengelompokan Dokumen Selesai

e. Menu Kategori Dokumen

Menu ini merupakan menu yang berisi macam-macam kategori dokumen yang mana kategori (*cluster*) ini ialah jumlah *cluster* saat membuat suatu model k-means.

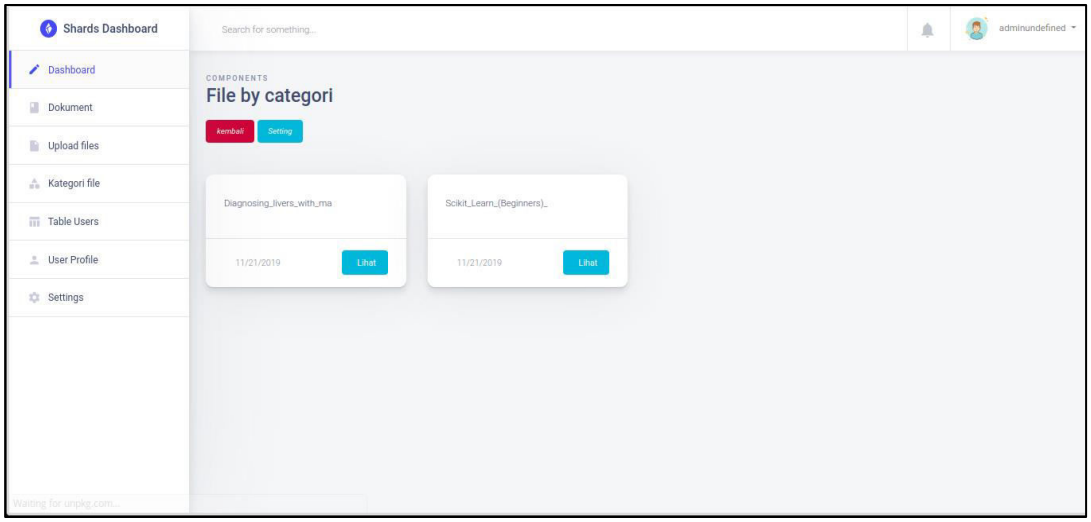


Gambar 24. Menu Kategori *(Cluster)*

Menu kategori dokumen berisi beberapa halaman diantarnya ialah sebagai berikut:

1. Halaman dokumen berdasarkan kategori

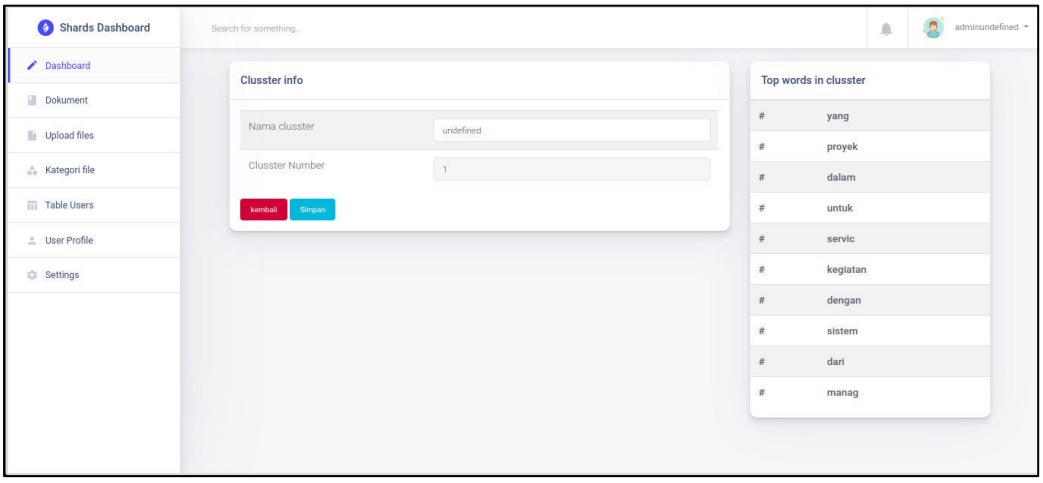
Halaman yang menampilkan dokumen berdasarkan kategori tertentu. Halaman ini dapat dilihat pada gambar dibawah ini



Gambar 25. Dokumen by Kategori (*cluster*)

2. Halaman settings kategori dokumen

Merupakan halaman yang berfungsi untuk keperluan penamaan maupun pengeditan nama dari suatu kategori dokumen. Dalam halaman ini terdapat list berupa *top words* dalam suatu kategori *(cluster).* Berikut ini gambaran yang dapat dilihat pada gambar 26.

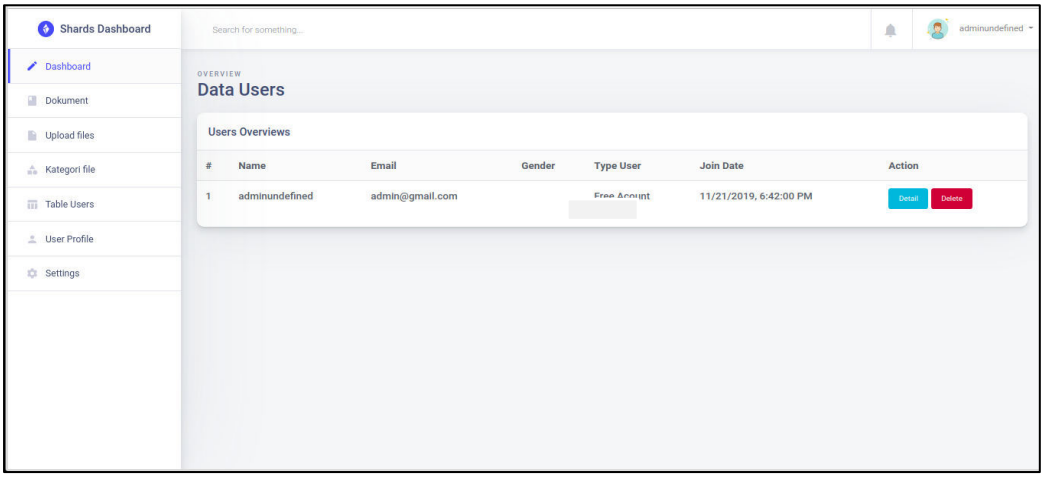


Gambar 26. Memberi atau Mengganti Nama Kategori

f. Menu Tabel *users*

Merupakan menu yang berfungsi untuk menampilkan user yang sudah terdaftar.

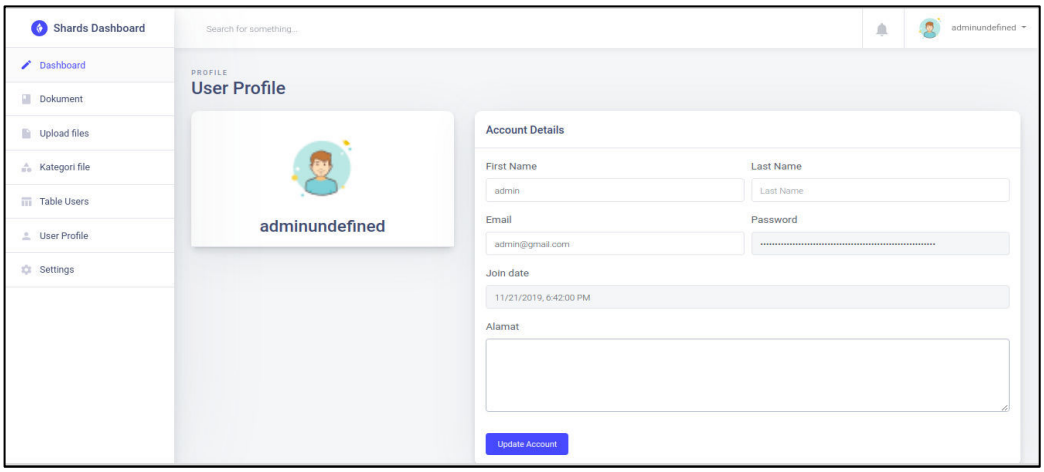
Untuk lebih jelasnya bisa dilihat pada gambar 27.



Gambar 27. Menu user

g. Menu *User Profile*

Merupkaan menu yang berfungsi untuk menampilkan deskripsi lengkap dan mengedit mengenai user yang login. Gambar dibwah ini merupakan *screenshot* dari menu *user profile*

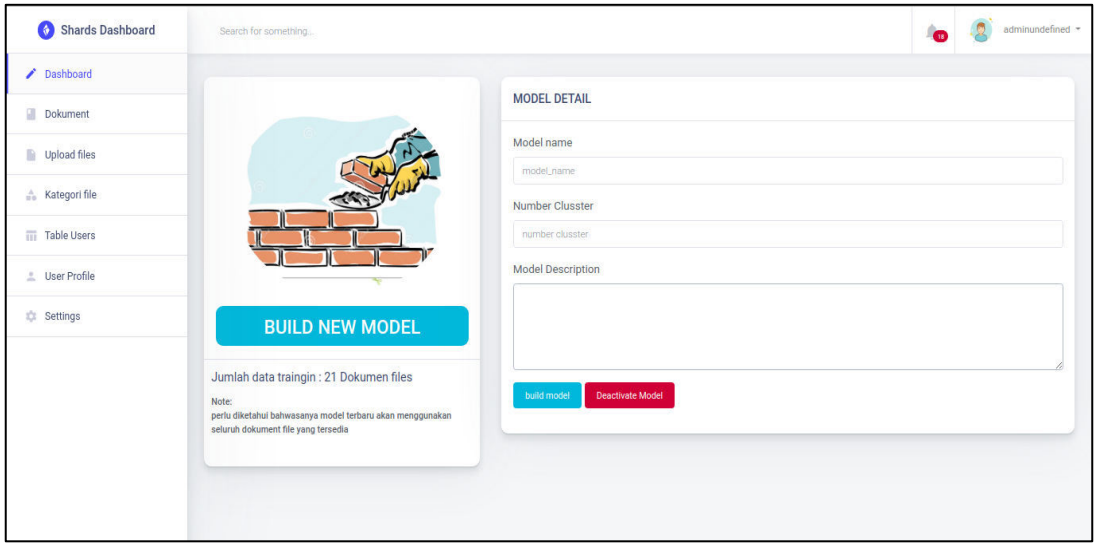


Gambar 28. Menu Profile User

h. Menu Settings

Merupakan menu yang berfungsi untuk untuk membuat suatu model baru yang mana model tersebut digunakan dalam proses pengelompokan suatu dokumen.

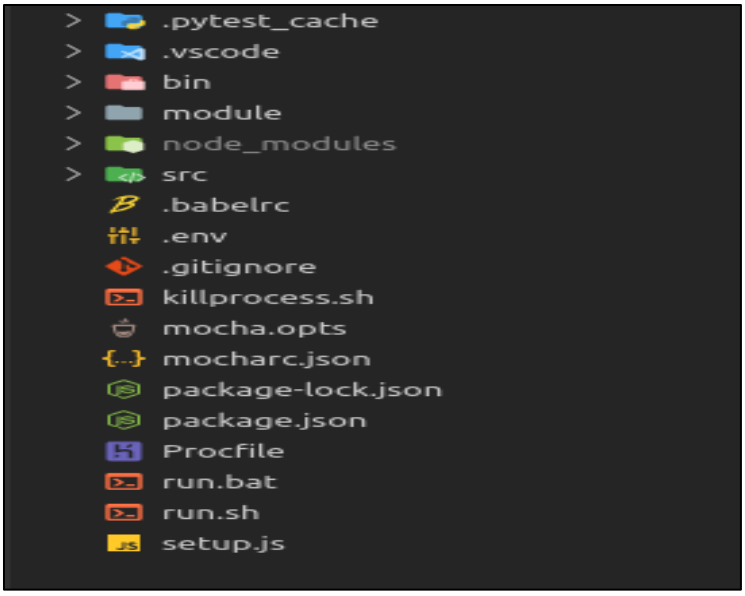
Dalam menu ini juga terdapat fitur aktifasi pengelompokan maupun d*eactivate* pengelompokan.



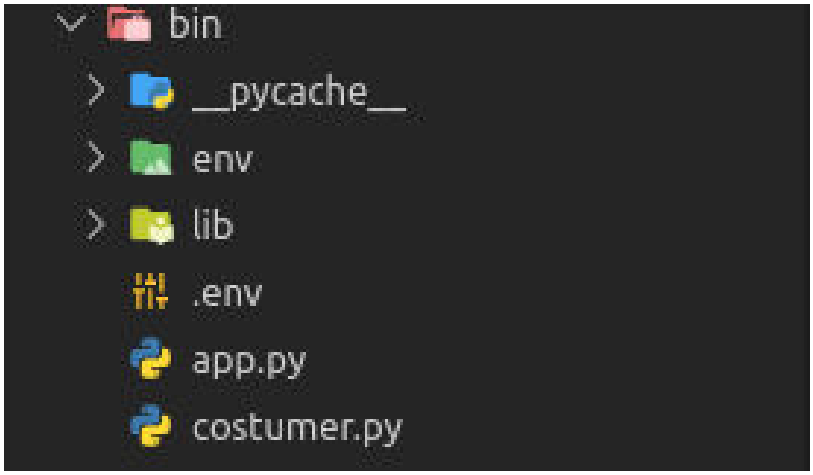
Gambar 29. *Menu Settings Build Model clusster* berhasil

5. *Source Code* Sistem

Dalam pembuatan project sistem *sistem pengelompokan dokumen teks secara otomatis,* kode program dipecah menjadi beberapa folder dan beberapa file. Hal itu dilakukan guna memermudah dalam proses pengembangan. Sistem ini memiliki struktur yakni terdapat folder diantaranya bisa dilihat pada gambar 30.

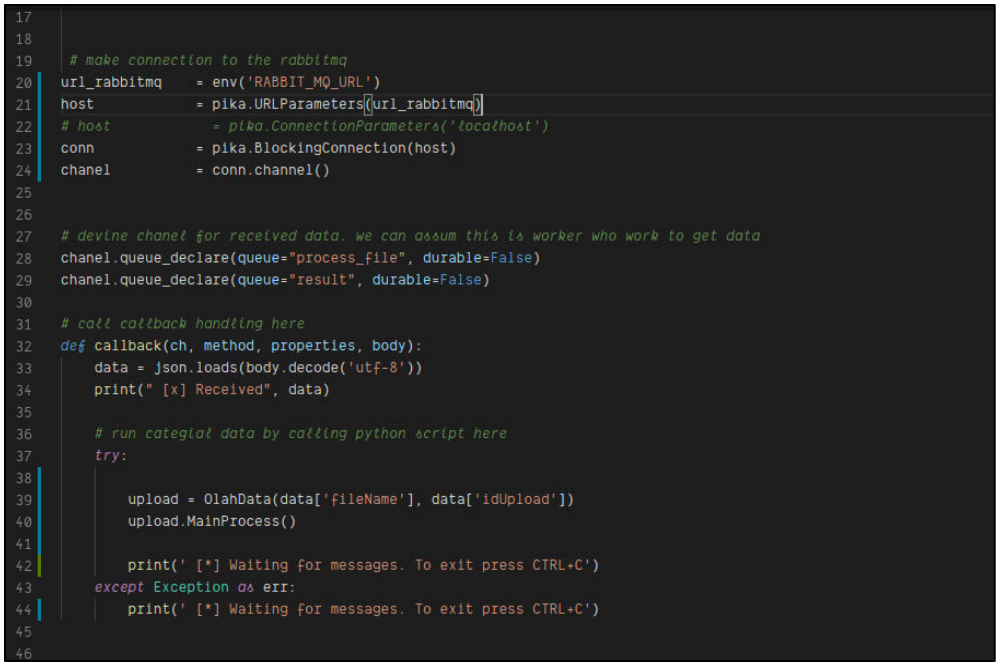


Gambar 30. Project struktur utama sistem



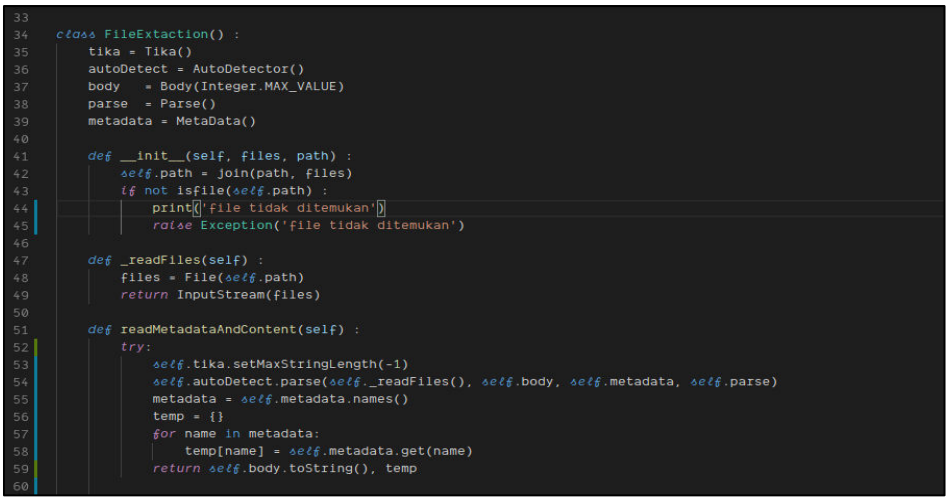
Gambar 31. Struktur *Projcet* Python *Script*

Script file Python yang bernama costumer.py berperan sebagai costumer yang bertugas dalam menangkap data yang dikirim dari *message broker producer.* Untuk *screenshot* dari *source code* costumer.py bisa dilihat pada gambar 32.



```
17
18
19  # make connection to the rabbitmq
20  url_rabbitmq    = env('RABBIT_MQ_URL')
21  host            = pika.URLParameters(url_rabbitmq)
22  # host            = pika.ConnectionParameters('localhost')
23  conn            = pika.BlockingConnection(host)
24  chanel          = conn.channel()
25
26
27  # devine chanel for received data. we can assum this is worker who work to get data
28  chanel.queue_declare(queue="process_file", durable=False)
29  chanel.queue_declare(queue="result", durable=False)
30
31  # call callback handling here
32  def callback(ch, method, properties, body):
33      data = json.loads(body.decode('utf-8'))
34      print(" [x] Received", data)
35
36      # run categiat data by calling python script here
37      try:
38
39          upload = OlahData(data['fileName'], data['idUpload'])
40          upload.MainProcess()
41
42          print(' [*] Waiting for messages. To exit press CTRL+C')
43      except Exception as err:
44          print(' [*] Waiting for messages. To exit press CTRL+C')
45
46
```

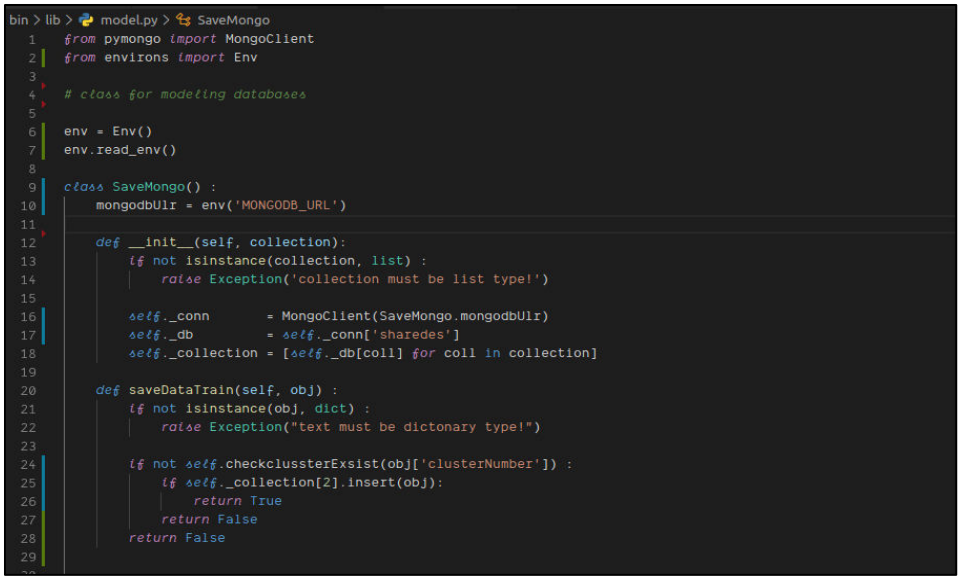
Gambar 32. *Screenshot* File costumer.py



```
33
34  class FileExtaction() :
35      tika = Tika()
36      autoDetect = AutoDetector()
37      body   = Body(Integer.MAX_VALUE)
38      parse  = Parse()
39      metadata = MetaData()
40
41      def __init__(self, files, path) :
42          self.path = join(path, files)
43          if not isfile(self.path) :
44              print('file tidak ditemukan')
45              raise Exception('file tidak ditemukan')
46
47      def _readFiles(self) :
48          files = File(self.path)
49          return InputStream(files)
50
51      def readMetadataAndContent(self) :
52          try:
53              self.tika.setMaxStringLength(-1)
54              self.autoDetect.parse(self._readFiles(), self.body, self.metadata, self.parse)
55              metadata = self.metadata.names()
56              temp = {}
57              for name in metadata:
58                  temp[name] = self.metadata.get(name)
59              return self.body.toString(), temp
60
```

Gambar 33. *File Extraction*

Gambar diatas merupakan *schreenshot* dari *script* yang berguna untuk proses pembacaan teks dari suatu dokumen. Proses pembacaan dilakukan dengan menggunakan *framework* Apache tika yang dikompile dengan bantuan module *Cpython*.



Gambar 34. Screenshot dari Script Module Databases

Proses pembacaan maupun *insert* dokumen dibuat dalam satu class yang diberi nama *class savemongo*.

Gambar 35. *Shreenshot* dari app.js

File app.js merupakan file utama dari sistem *sistem pengelompokan dokumen teks secara otomatis. Script* diatas merupakan bentuk konfigurasi agar sistem dapat berjalan dengan baik.

```
Admin.get('/', admin.getviews);
Admin.get('/notif', admin.notif);
Admin.get('/statetment',admin.getStatisticPreview);
Admin.get('/users', users.handleContent);
Admin.get('/user-profile', userPrefiew.getUserPrefiew);
Admin.get('/users/edit/:id', userPrefiew.handleUpdateGet);
Admin.get('/users/delete/:id', userPrefiew.handleDeleted);


Admin.get('/document', document.handleContent);
Admin.get('/document/delete/:id', document.handleDeleted);
Admin.get('/document/ed Loading... ocument.handleUpdate);
Admin.get('/document/search/', document.SearchDocument);
Admin.get('/document/upload-muliples', document.getViewsNewuserPrefiews);
Admin.get('/document/kategori', document.kategori);
Admin.get('/kategori/:id', document.SpesificKat);
Admin.get('/clusster/setting/:id', document.clussterSet);
Admin.get('/document/description/:id', document.description);
Admin.get('/document/keywords-bar/:id', document.getStatistiKeyword);
Admin.get('/settings/',document.new_model);
Admin.get('/clusster/activate_model', document.active_clusster);
// Admin.get('/settings/build_new_model', document.new_model)

```

Gambar 36. Screenshot dari Router Admin

Script diatas merupakan registrasi router yang dapat diakses oleh seorang admin. Didalam file ini terdapat 2 bentuk regitrasi router yakni get dan post.

```
// Index.get('/document',  authorization.authenticationUsers, document.handleContent);
Index.get('/statetment', authorization.authenticationUsers, admin.getStatisticPreview);
Index.get('/document', authorization.authenticationUsers, document.handleContent);
Index.get('/document/search/', authorization.authenticationUsers, document.SearchDocument);
Index.get('/document/kategori', authorization.authenticationUsers, document.kategori);
Index.get('/document/edit/:id', authorization.authenticationUsers, document.handleUpdate);
Index.get('/document/delete/:id', authorization.authenticationUsers, document.handleDeleted);
Index.get('/document/upload-muliples', authorization.authenticationUsers, document.getViewsNewuserPrefiews);
Index.get('/document/description/:id', authorization.authenticationUsers, document.description);
Index.get('/document/keywords-bar/:id', authorization.authenticationUsers, document.getStatistiKeyword);

Index.get('/kategori/:id', authorization.authenticationUsers, document.SpesificKat);
Index.get('/clusster/setting/:id', authorization.authenticationUsers, document.clussterSet);
Index.get('/user-profile', authorization.authenticationUsers, userPrefiew.getUserPrefiew);

```

Gambar 37. Screenshot Router User Biasa

Didalam File ini berisi seluruh registrasi router user. User hanya dapat melihat dan mengakses router yang ada dalam file ini.

```
/* sending data to rabbitmq costumer  */
uploads.sendTocostumerRabbitmq =  function(data, next) {
    var queue  = 'process_file',
        result = 'result';

    amqp.connect(env.RABBITMQ, (err, conn) =>  {
        if(err) {
            return next(err);
        } else {
            let channel = conn.createChannel();
            channel.assertQueue(queue, { durable: false });
            channel.assertQueue(result, { durable: false });
            channel.sendToQueue(queue, new Buffer(JSON.stringify(data), {persistent: false}));
            channel.consume(result, function (msg) {
                var data =  msg.content.toString();
                console.log(data)
            },{ noAck: true });

            /* close connection when it's done! */
            setTimeout(function () { conn.close(); }, 500);
        }
    })
};

```

Gambar 38. Screenshot *Publiser* Rabbitmq

Script di atas berfungsi sebagai publisher data. Maksud dari *publisher* ialah script yang bertugas mengirimkan data ke server rabbitmq / *message broker* yang kemudian server message broker akan mengirimkan data dari producer ke *customer*. Data yang sudah dikirim maka akan di proses oleh kostumer dan producer bisa melanjutkan untuk melakukan tugas lainnya tanpa harus mengunggu tugas tersebut diselesaikan oleh *customer.*