



REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN CIPTAAN

Dalam rangka perlindungan 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 perlindungan : 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 memberi label isi dokumen secara otomatis. Kemampuan program ini mampu mengelompokkan 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 administrasi 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 *message 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 menunggu proses upload selesai. Setelah proses upload selesai pengguna bisa mengerjakan hal lain. Pengguna akan mendapatkan notifikasi di pojok kanan atas apabila pemrosesan dokumen selesai di proses. Sistem ini Menggunakan algoritma *Kmeans* dalam proses *clustering* dokumen teks. Sistem ini mampu mengelompokkan 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 www.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. pengguna 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 android path
export ANDROID_HOME=/media/myone/Data_mahud/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*, *stemming* hingga proses *clustering*. Proses pembacaan dokumen dilakukan dengan memanggil *class Java* yang ada pada Apache tika kemudian dicompile class bahasa c++ kemudian dijalankan dengan 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 javascript runtime yang dibangun dengan engine V8 chrome. Penggunaan nodejs ialah guna dapat menjalankan aplikasi utama. Pengguna dapat mendownload dan menginstallnya dengan mengunjungi www.nodejs.org. Untuk mendownload NodeJs bisa dilihat pada gambar 2.



Gambar 2. Download NodeJs LTS

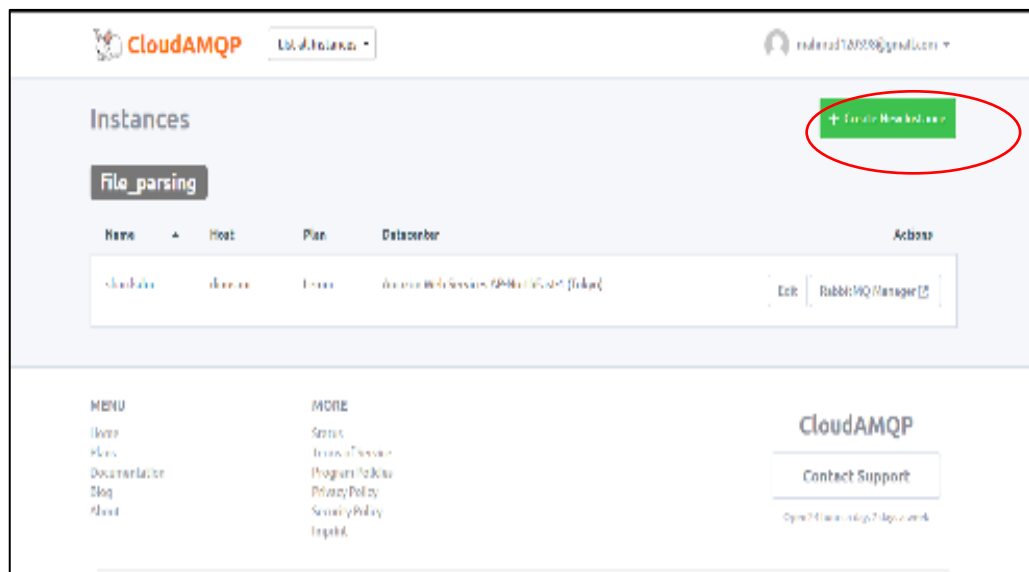
d. Daftar Layanan *Message broker CloudAMQP*

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

dapat mendistribusikan 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 clustering* dokumen dengan program yang bertugas sebagai server, sehingga estimasi waktu bisa ditekan.

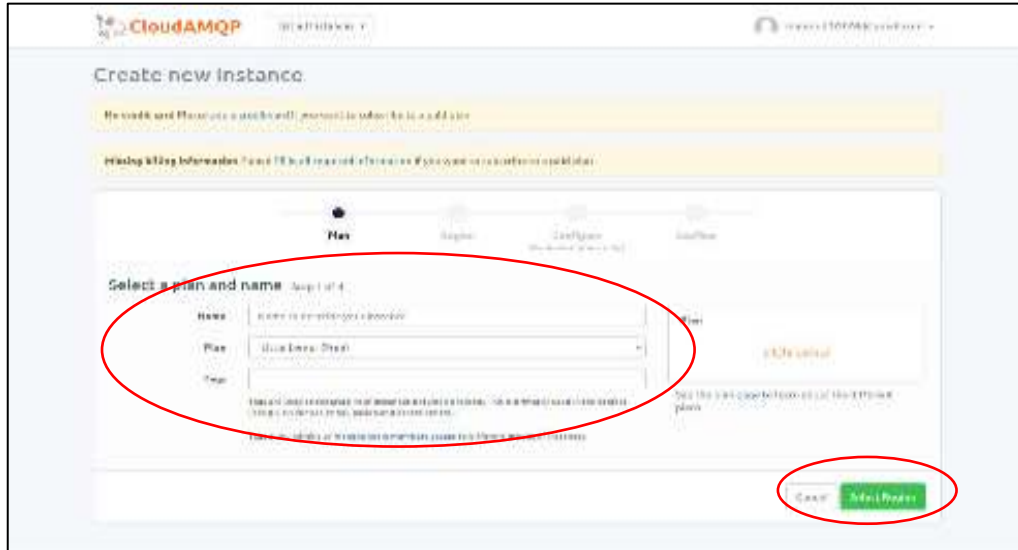
Sebagai contoh pengguna mengupload file PDF dengan size 100Mb, apabila kita tidak menggunakan message broker maka kita akan menunggu sampai proses *upload, read, parsing hingga clustering* dokumen selesai. Proses tersebut tentunya tidak akan memakan waktu lama apabila kita menggunakan layanan *message 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 selesai meng-*upload* dokume, pengguna dapat membuka layanan lain tanpa harus menunggu 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.

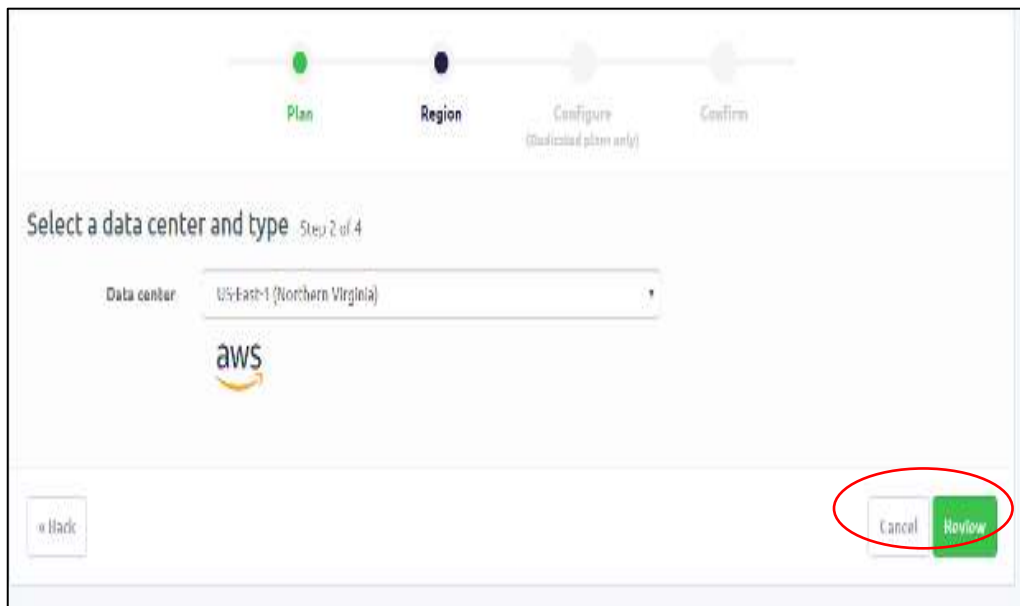


Gambar 3. Membuat Layanan Message Broker Cloudmqp

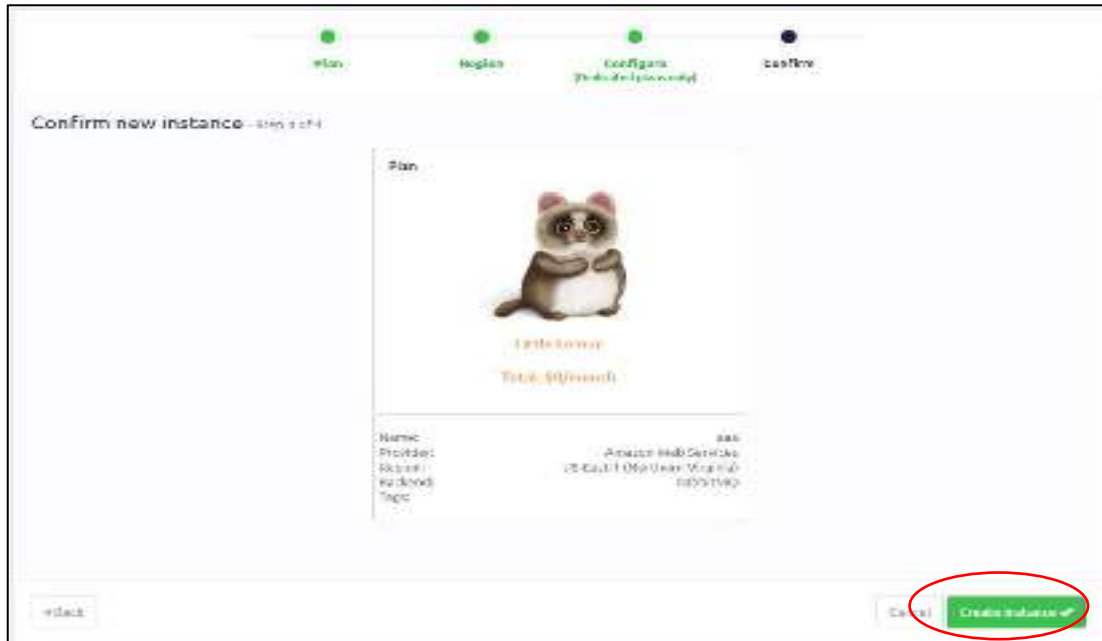
Setelah pengguna mengklik *create instance* maka pengguna akan *redirect* ke halaman selanjutnya yang mana pada halaman ini pengguna mengisi nama dari instance rabbitmq dan tipe *instance*. Pengguna bisa mengisi nama instance sesuai dengan yang diinginkan dan *little lemule lemur (free)* sebagai tipe *instance*. 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.

```
heroku@yone:~/Documents/beraku_app$ cat .env
1 ROOT_APP=/home/yone/Documents/beraku_app/shareds
2 MONGODB_URI=/home/yone/Documents/beraku_app/shareds/db
3 PYTHON_PATH=/home/yone/Documents/beraku_app/shareds/bin/l15
4 UPLOAD_PATH=/home/yone/Documents/beraku_app/shareds/db/public/uploads
5 MONGODB_URI=mongodbs://localhost:27017/
6 MONGODB_URI=mongodbs://localhost:27017/
7 PYTHON_PATH=/home/yone/Documents/beraku_app/shareds/bin/l15
8 PORT=https://localhost:5000/
9 FILE_UPLOAD_SIZE=50000000
10 PORT=5000
```

Gambar 9. Environment Variabel Aplikasi Utama

```
bin > heroku env
1 export BARBT_NO_URI=semp //gllrfsp-7h7kv07NNry1Pl X02epR1bNARHInC5MAdinosaur rmq.cloudapp.com/gllrfsp
2 export PATH_FILE=/home/yone/Documents/beraku_app/shareds/db/public/uploads/file
3 export MONGODB_URI=mongodbs://localhost:27017/
4 export TTKA_PATH=/home/yone/Documents/beraku_app/shareds/bin/l10/tika-app/tika-app.jar
5 export JAVA_HOME=/usr/lib/jvm/jre1.8.0_211/
```

Gambar 10. Environmen Python script untuk Pemrosesan File

g. Instal *Dependencies* 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.

```
myone@yone-1442.RR:~/Documents/beraku_app/mazendb$ npm install
- load up ansi-regex: ■■■ parse range manifest for isovep*2.0.0 fetched in 72ms
```

Gambar 11. Proses *Install Dependencies* Aplikasi

3. Menjalankan Aplikasi

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



```
myone@myone-V442URR:~/Documents/heroku_app/shared$ ./run.sh
? Selamat datang di program shared dashboard manajemen dokument
Start Program
> Add Administrator
```

Gambar 12. Proses Awal menjalankan Aplikasi di Linux

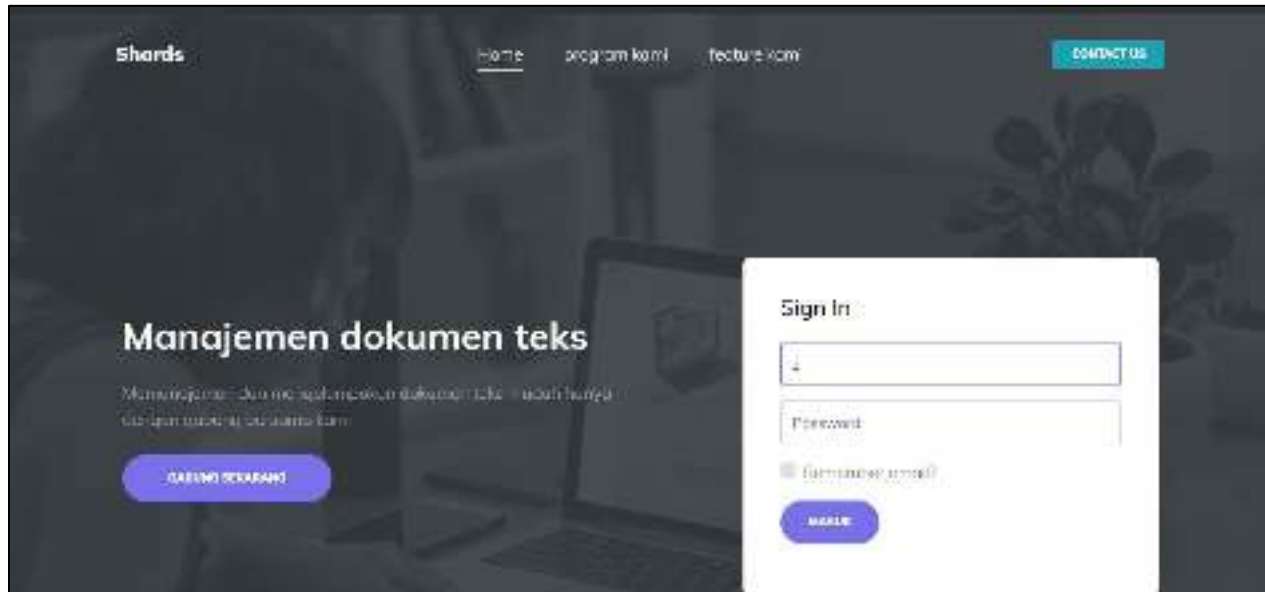
Pada awal pembukaan aplikasi pengguna akan ditampilkan 2 menu menu pertama ialah menu untuk start program. Dan menu kedua ialah add administrator user. Pengguna yang pertama kali membuka aplikasi diharuskan untuk membuat user administrator agar pengguna dapat membuka menu administrator aplikasi. Proses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



```
myone@myone-V442URR:~/Documents/heroku_app/shared$ ./run.sh
? Selamat datang di program shared dashboard manajemen dokument Add Administrator
? Username awal administrator
? Password Administrator
? Password Administrator
```

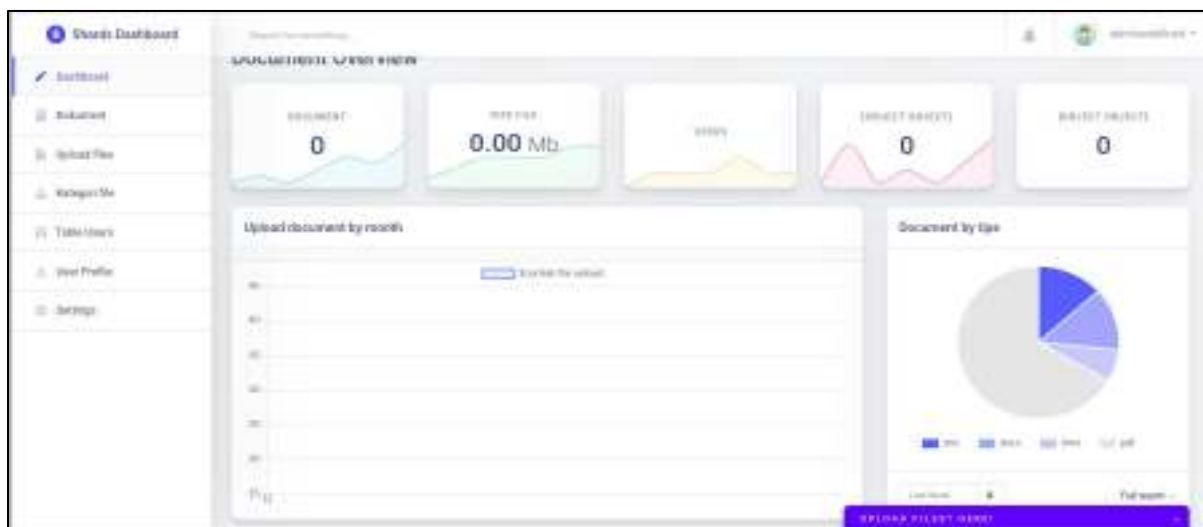
Gambar 13. Proses Pembuatan Administrator User

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.



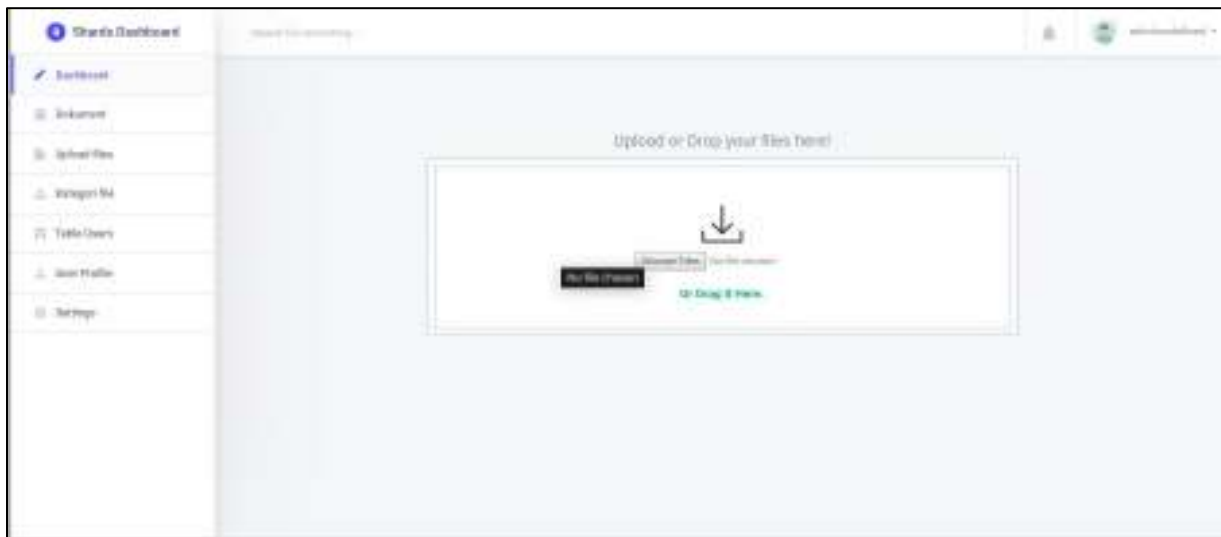
Gambar 14. Halaman Login Sistem

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



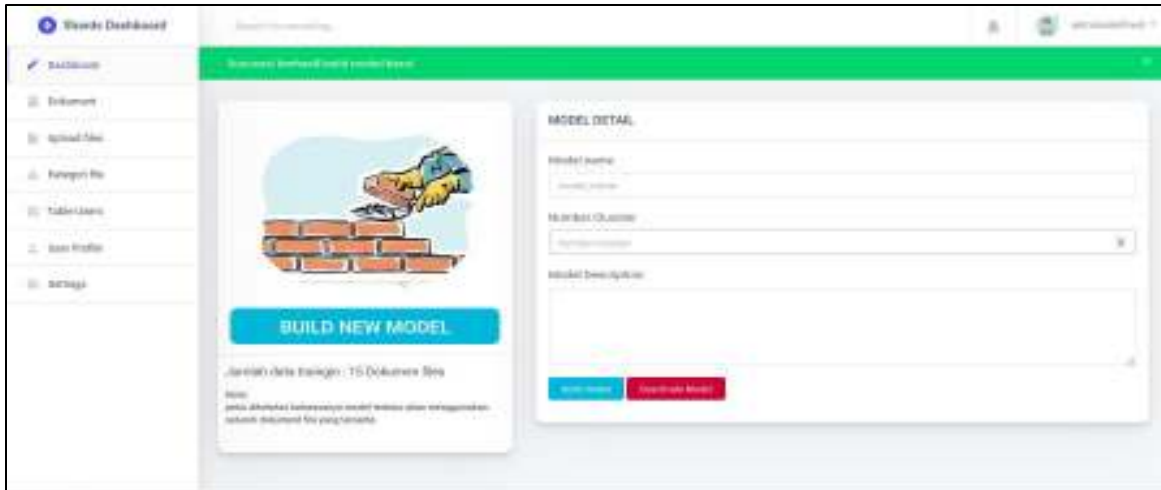
Gambar 15. Dashboard Sistem

Sistem ini belum bisa dijalankan karena pemrosesan *clustering* membutuhkan data sample awal dalam proses *clustering*. Nantinya seluruh dokumen akan di kelompokkan 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-*redirect* 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 file 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 *clustering* 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 fitur yang agak sedikit berbeda dimana admin memiliki menu tambahan dan fitur 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 clustering*.

a. Halaman registrasi akun

Merupakan halaman yang berfungsi untuk proses pendaftaran pengguna baru.

The image shows a registration form with the following fields and elements:

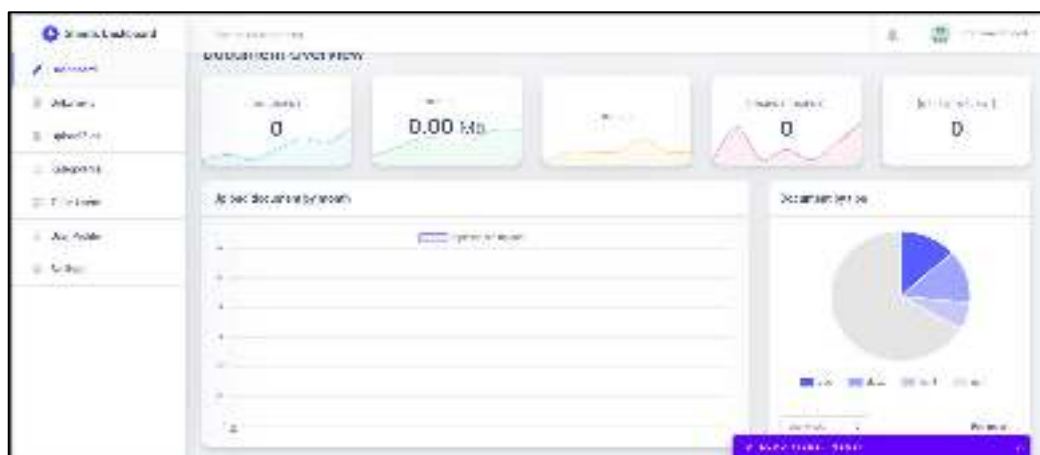
- Registrasi akun** (Title)
- Log In** (Link)
- Nama depan** (First Name) and **Nama belakang** (Last Name) input fields.
- Email address** input field.
- Crea** (Creation Date) input field.
- Gender selection: **Male** and **Female**.
- Password** input field.
- Registrasi akun** (Registration) button.

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 fitur diantaranya ialah sebagai berikut.

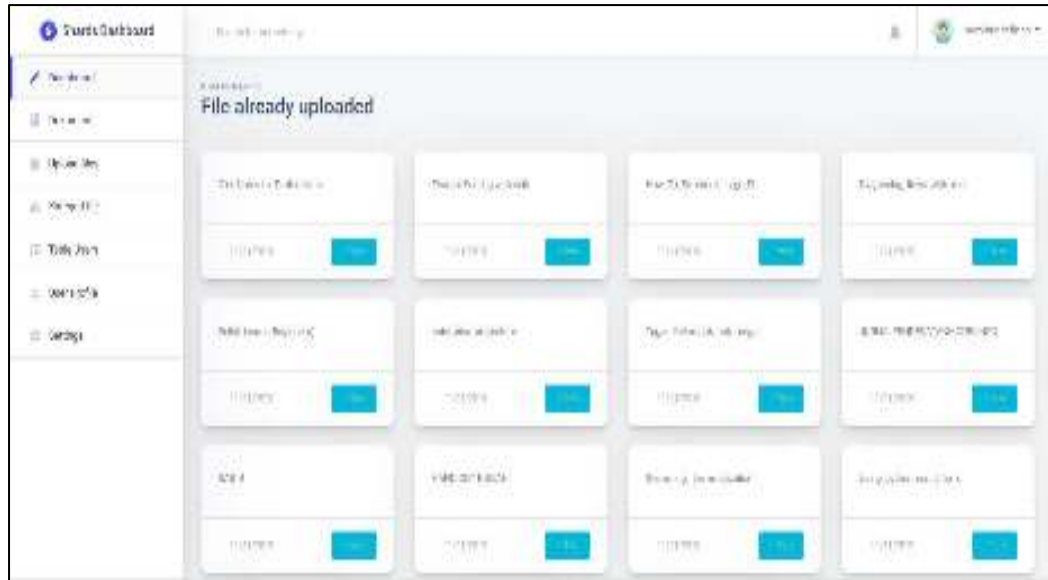
1. Jumlah dokumen yang diupload secara keseluruhan.
2. Jumlah *size* dari seluruh dokumen semua user.
3. Jumlah user yang telah menggunakan sistem.
4. Jumlah dokumen berbahasa inggris.
5. Jumlah dokumen berbahasa indonesia.
6. Grafik line pengupload-an dokumen
7. Grafik *pie chart* mengenai tipe dokumen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



Gambar 19. Dashboard Aplikasi Setelah User Login

c. Menu dokumen

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

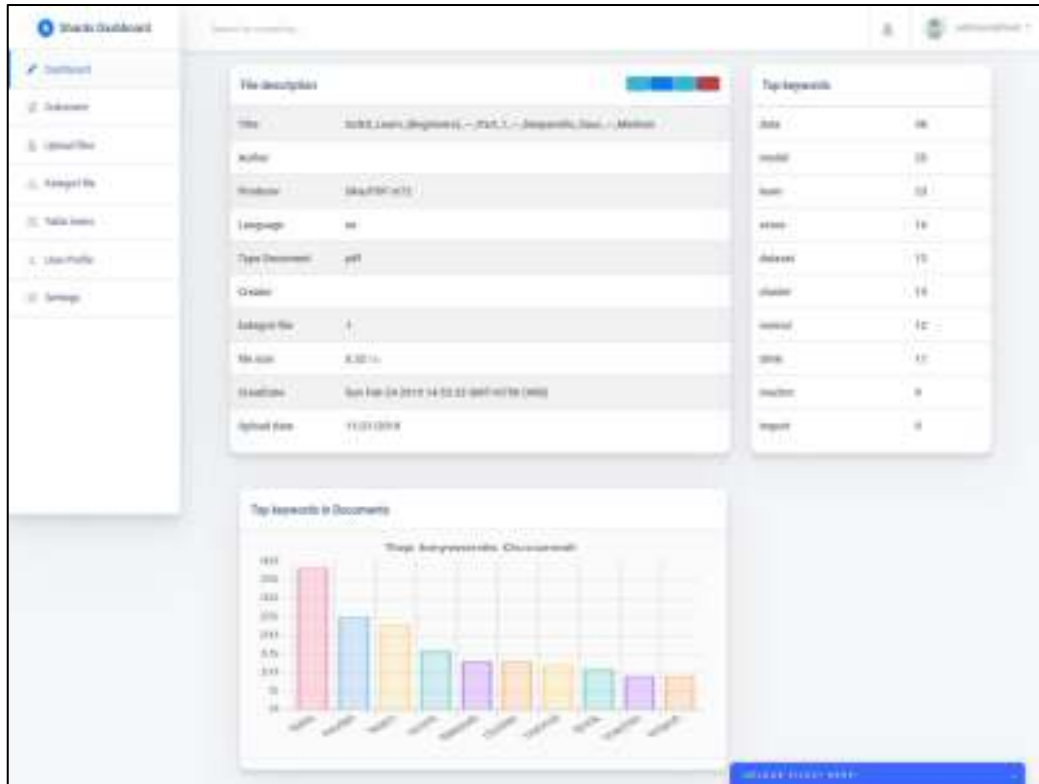


Gambar 20. Menu Dokumen

Dalam menu dokumen terdapat beberapa halaman diantaranya 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 menggunakan *drop and drag* dokumen. Proses ini bisa dilihat pada gambar dibawah ini.



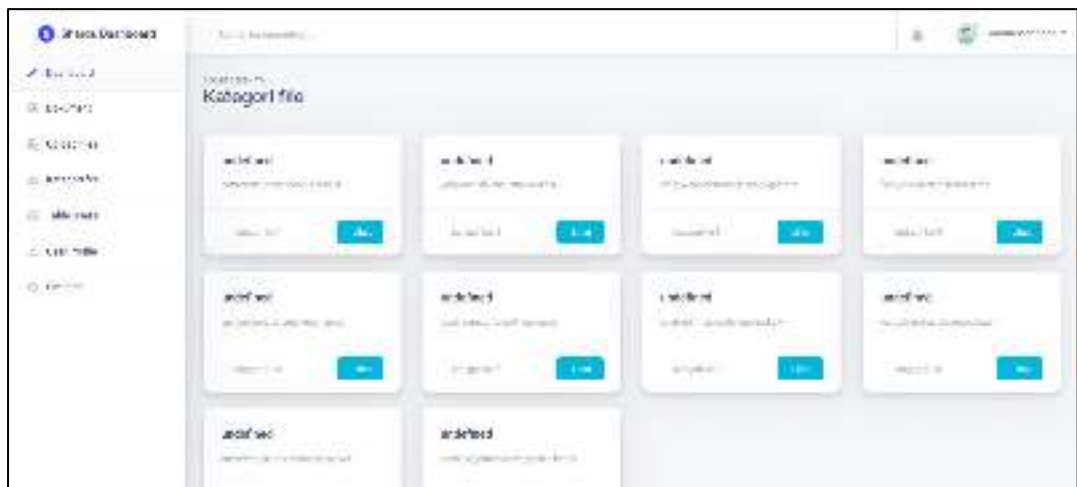
Gambar 22. Menu Upload File



Gambar 23. Notifikasi 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 diantaranya ialah sebagai berikut:

1. Halaman dokumen berdasarkan kategori

Halaman yang menampilkan dokumen berdasarkan kategori tertentu.

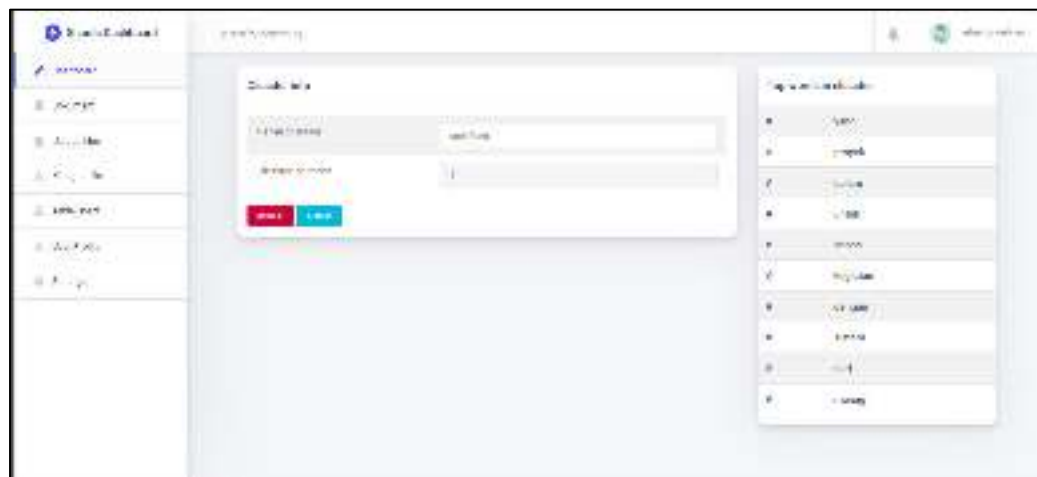
Halaman ini dapat dilihat pada gambar dibawah ini



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.

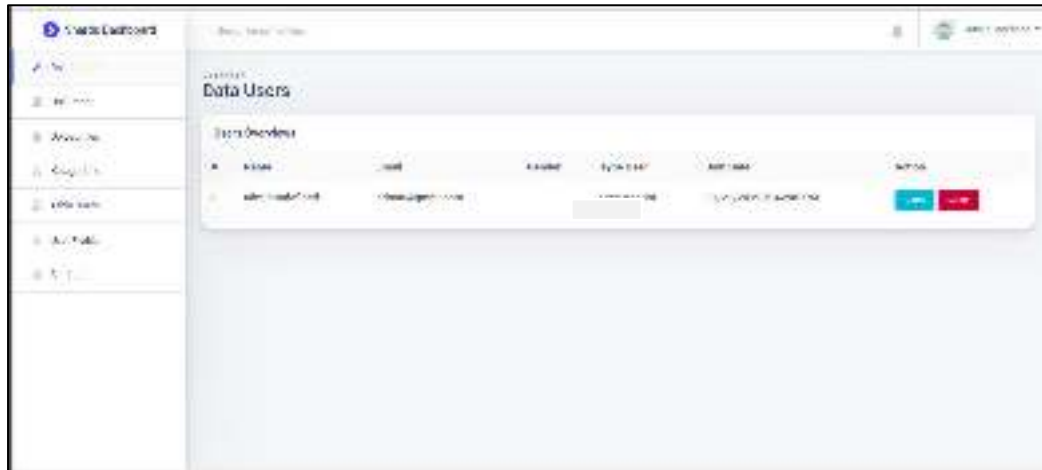


Gambar 26. Memberi atau Mengganti Nama Kategori

f. Menu Tabel *users*

Merupakan menu yang berfungsi untuk menampilkan user yang sudah terdaftar.

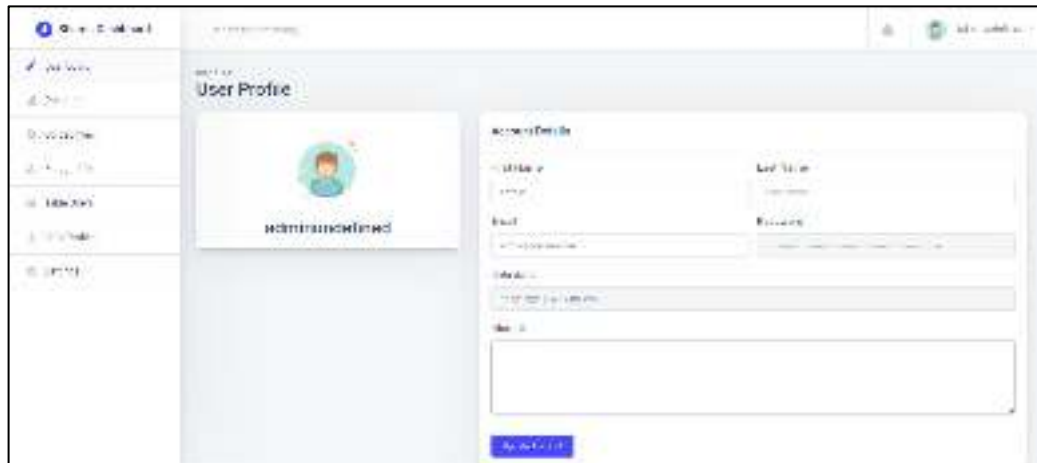
Untuk lebih jelasnya bisa dilihat pada gambar 27.



Gambar 27. Menu user

g. Menu *User Profile*

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

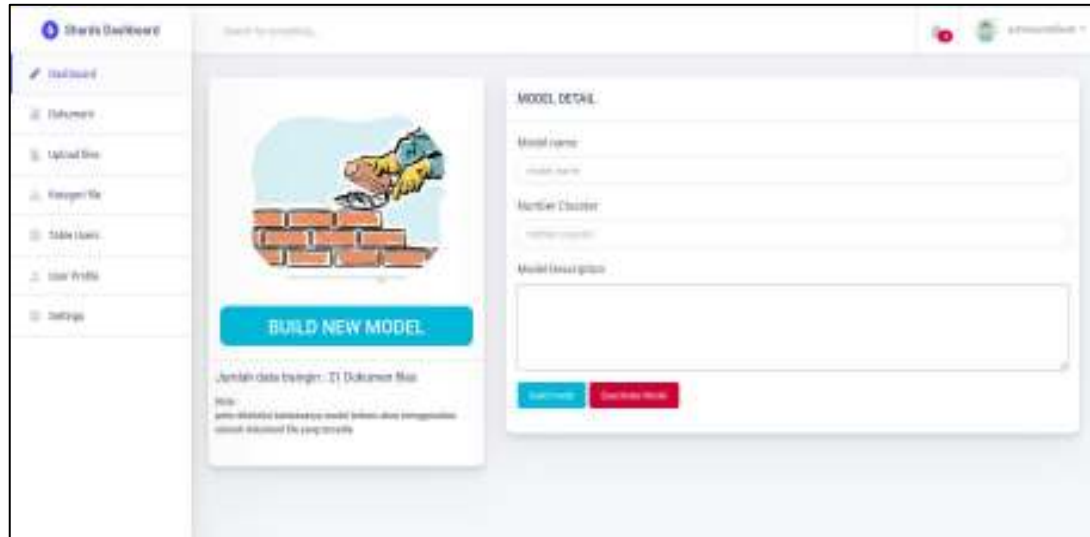


Gambar 28. Menu Profile User

h. Menu Settings

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

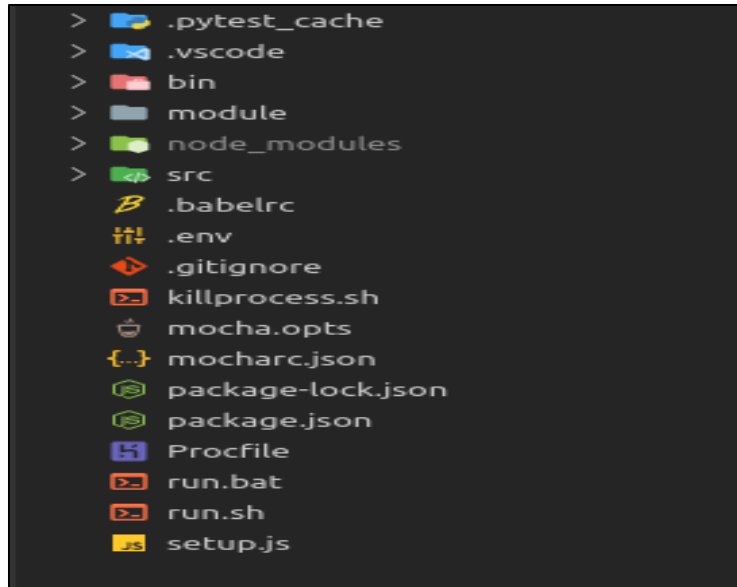
Dalam menu ini juga terdapat fitur aktifasi pengelompokan maupun *deactivate* pengelompokan.



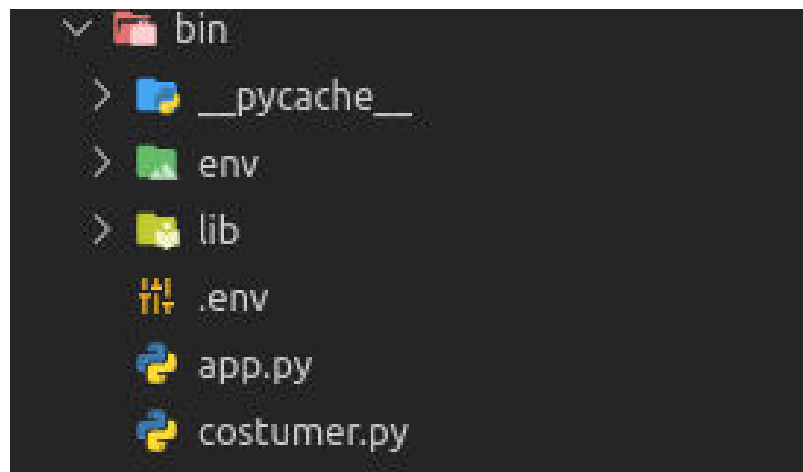
Gambar 29. Menu Settings Build Model cluster berhasil

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 mempermudah 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.

```

11
12
13 # setup connection to the database
14 db_conn = pymysql.connect(host=db_host,
15                             user=db_user,
16                             password=db_password,
17                             database=db_database)
18 cursor = db_conn.cursor()
19
20 # create table for storing data
21 cursor.execute('CREATE TABLE IF NOT EXISTS `data` (
22     `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
23     `file_name` VARCHAR(255) NOT NULL,
24     `data` TEXT NOT NULL)')
25 db_conn.commit()
26
27 # read file and store data
28 def read_file_and_store(file_path):
29     with open(file_path, 'r') as f:
30         data = f.read()
31     cursor.execute('INSERT INTO `data` (`file_name`, `data`) VALUES (%s, %s)',
32                  (file_path, data))
33 db_conn.commit()
34
35 # main function
36 def main():
37     file_path = input('Enter file path: ')
38     read_file_and_store(file_path)
39
40 if __name__ == '__main__':
41     main()

```

Gambar 32. Screenshot File costumer.py

```

1
2 # File Extraction
3
4 # Importing modules
5 import os
6 import shutil
7
8 # Function to extract files
9 def extract_files(source_dir, target_dir):
10     if not os.path.exists(target_dir):
11         os.makedirs(target_dir)
12     for file in os.listdir(source_dir):
13         src_path = os.path.join(source_dir, file)
14         dst_path = os.path.join(target_dir, file)
15         if os.path.isfile(src_path):
16             shutil.copy2(src_path, dst_path)
17         elif os.path.isdir(src_path):
18             extract_files(src_path, dst_path)
19
20 # Main function
21 def main():
22     source_dir = input('Enter source directory: ')
23     target_dir = input('Enter target directory: ')
24     extract_files(source_dir, target_dir)
25
26 if __name__ == '__main__':
27     main()

```

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*.


```

78 | #!/usr/bin/env node
79 |
80 | const mongoose = require('mongoose');
81 | const express = require('express');
82 |
83 | // Create an Express application
84 | const app = express();
85 |
86 | // Create a Mongoose connection
87 | const mongoose = mongoose.createConnection('mongodb://localhost:27020/');
88 |
89 | // Middleware
90 | app.use(express.json());
91 |
92 | // Routes
93 |
94 | // GET /api/users
95 | app.get('/api/users', async (req, res) => {
96 |   const users = await mongoose.model('User').find().exec();
97 |   res.json(users);
98 | });
99 |
100 | // POST /api/users
101 | app.post('/api/users', async (req, res) => {
102 |   const user = new mongoose.model('User')(req.body);
103 |   const savedUser = await user.save();
104 |   res.json(savedUser);
105 | });
106 |
107 | // Error handling
108 | app.use((err, req, res, next) => {
109 |   console.error(err.stack);
110 |   res.status(500).send('Something broke!');
111 | });
112 |
113 | // Start the server
114 | app.listen(3000, () => {
115 |   console.log('Server is running on port 3000');
116 | });

```

Gambar 34. Screenshot dari Script Module Databases

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

```

1 | // Import dependencies
2 | const express = require('express');
3 | const mongoose = require('mongoose');
4 | const bodyParser = require('body-parser');
5 |
6 | // Create an Express application
7 | const app = express();
8 |
9 | // Create a Mongoose connection
10 | const mongoose = mongoose.createConnection('mongodb://localhost:27020/');
11 |
12 | // Middleware
13 | app.use(bodyParser.json());
14 |
15 | // Routes
16 |
17 | // GET /api/users
18 | app.get('/api/users', async (req, res) => {
19 |   const users = await mongoose.model('User').find().exec();
20 |   res.json(users);
21 | });
22 |
23 | // POST /api/users
24 | app.post('/api/users', async (req, res) => {
25 |   const user = new mongoose.model('User')(req.body);
26 |   const savedUser = await user.save();
27 |   res.json(savedUser);
28 | });
29 |
30 | // Error handling
31 | app.use((err, req, res, next) => {
32 |   console.error(err.stack);
33 |   res.status(500).send('Something broke!');
34 | });
35 |
36 | // Start the server
37 | app.listen(3000, () => {
38 |   console.log('Server is running on port 3000');
39 | });

```

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.

```

28
29 Admin.get('/', admin.getViews);
30 Admin.get('/notif', admin.notif);
31 Admin.get('/statement', admin.getStatisticPreview);
32 Admin.get('/users', users.handleContent);
33 Admin.get('/user-profile', userPrefiew.getUserPrefiew);
34 Admin.get('/users/edit/:id', userPrefiew.handleUpdateGet);
35 Admin.get('/users/delete/:id', userPrefiew.handleDeleted);
36
37
38 Admin.get('/document', document.handleContent);
39 Admin.get('/document/delete/:id', document.handleDeleted);
40 Admin.get('/document/edit Loading..', document.handleUpdate);
41 Admin.get('/document/search/', document.SearchDocument);
42 Admin.get('/document/upload-multiple', document.getViewsNewuserPrefiew);
43 Admin.get('/document/kategori', document.kategori);
44 Admin.get('/kategori/:id', document.SpesifikKat);
45 Admin.get('/cluster/setting/:id', document.clusterSet);
46 Admin.get('/document/description/:id', document.description);
47 Admin.get('/document/keywords-bar/:id', document.getStatisticKeyword);
48 Admin.get('/settings/', document.new_model);
49 Admin.get('/cluster/activate_model', document.active_cluster);
50 // Admin.get('/settings/build_new_model', document.new_model);
51

```

Gambar 36. Screenshot dari Router Admin

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

```

59
60 // Index.get('/document', authorization.authenticationUsers, document.handleContent);
61 Index.get('/statement', authorization.authenticationUsers, admin.getStatisticPreview);
62 Index.get('/document', authorization.authenticationUsers, document.handleContent);
63 Index.get('/document/search/', authorization.authenticationUsers, document.SearchDocument);
64 Index.get('/document/kategori', authorization.authenticationUsers, document.kategori);
65 Index.get('/document/edit/:id', authorization.authenticationUsers, document.handleUpdate);
66 Index.get('/document/delete/:id', authorization.authenticationUsers, document.handleDeleted);
67 Index.get('/document/upload-multiple', authorization.authenticationUsers, document.getViewsNewuserPrefiew);
68 Index.get('/document/description/:id', authorization.authenticationUsers, document.description);
69 Index.get('/document/keywords-bar/:id', authorization.authenticationUsers, document.getStatisticKeyword);
70
71 Index.get('/kategori/:id', authorization.authenticationUsers, document.SpesifikKat);
72 Index.get('/cluster/setting/:id', authorization.authenticationUsers, document.clusterSet);
73 Index.get('/user-profile', authorization.authenticationUsers, userPrefiew.getUserPrefiew);
74

```

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.

```
73  /* sending data to rabbitmq publisher */
74  uploads.sendToQueueRabbitmq = function(data, next) {
75      var queue = 'process file';
76      result = 'result';
77
78      amqp.connect(env.RABBITMQ, (err, conn) => {
79          if(err) {
80              return next(err);
81          } else {
82              let channel = conn.createChannel();
83              channel.assertQueue(queue, { durable: false });
84              channel.assertQueue(result, { durable: false });
85              channel.sendToQueue(queue, new Buffer(JSON.stringify(data)), {persistent: false});
86              channel.consume(result, function(msg) {
87                  var data = msg.content.toString();
88                  console.log(data);
89              }, { noAck: true });
90
91              /* close connection when it's done */
92              setImmediate(function () { conn.close(); }, 300);
93          }
94      });
95  }
96  }
```

Gambar 38. Screenshot *Publisher 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 mengganggu tugas tersebut diselesaikan oleh *customer*.