

**PERFORMANCE COMPARISON OF SEARCH FEATURE WITH  
DJANGO QUERYSETS AND ELASTICSEARCH IN A WEB  
APPLICATION**



Composed as one of the requirements to complete Strata I Study Program in  
Informatics Study Program the Faculty of Communication and Information

By:

**ZAHID MUJADDID**  
**L 200 134 010**

**INFORMATICS STUDY PROGRAM  
THE FACULTY OF COMMUNICATION AND INFORMATION  
UNIVERSITAS MUHAMMADIYAH SURAKARTA**

**2020**

**PAGE OF APPROVEMENT**

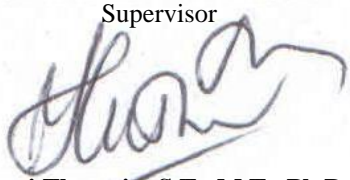
**PERFORMANCE COMPARISON OF SEARCH FEATURE WITH DJANGO  
QUERYSETS AND ELASTICSEARCH IN A WEB APPLICATION**

**SCIENTIFIC PUBLICATION**

By:

**ZAHID MUJADDID**  
**L 200 134 010**

Checked and approved for examination by:

Supervisor  
  
**Husni Thamrin, S.T., M.T., Ph.D.**  
**NIK.706**

## PAGE OF VERIFICATION

### PERFORMANCE COMPARISON OF SEARCH FEATURE WITH DJANGO QUERYSETS AND ELASTICSEARCH IN A WEB APPLICATION

By:

**ZAHID MUJADDID**  
L 200 134 010

Successfully defended in the presence of Board of Examiners  
Faculty of Communication and Information  
Muhammadiyah Surakarta University  
On Wednesday, 20 May 2020  
and acknowledged to be qualified

Board of Examiners:

Husni Thamrin, S.T., M.T., Ph.D.  
(Chief Board of Examiner)

(.....)

Dr., Ir. Bana Handaga, M.T.  
(Member I Board of Examiner)

(.....)

Azizah Fatmawati, S.T., M.Cs.  
(Member II Board of Examiner)

(.....)

Dean  
Faculty of Communication and Information



**Nurgiyatna, S.T., M.Sc., Ph.D.**  
NIK. 881

## STATEMENT OF VOW

I hereby declare this scientific publication is the one and only and to the best of my knowledge no similar work has ever been published to obtain a degree in a college or institution before by other people, except cited in this manuscript as a reference in the bibliography.

In the event of my statement above proven to be otherwise, then I shall be hold fully accountable.

Surakarta, 08 July 2020

Author



ZAHID MUJADDID

L 200 134 010



**UNIVERSITAS MUHAMMADIYAH SURAKARTA  
FAKULTAS KOMUNIKASI DAN INFORMATIKA  
PROGRAM STUDI INFORMATIKA**

Jl. A Yani Tromol Pos 1 Pabelan Kartasura Telp. (0271)717417, 719483 Fax (0271) 714448  
Surakarta 57102 Indonesia. Web: <http://informatika.ums.ac.id>. Email: [informatika@ums.ac.id](mailto:informatika@ums.ac.id)

---

**SURAT KETERANGAN LULUS PLAGIASI**

Assalamu'alaikum Wr. Wb

Biro Skripsi Program Studi Informatika menerangkan bahwa :

Nama : Zahid Mujaddid  
NIM : **L200134010**  
Judul : **PERFORMANCE COMPARISON OF SEARCH FEATURE  
WITH DJANGO QUERYSETS AND ELASTICSEARCH IN A  
WEB APPLICATION**  
Program Studi : Informatika  
Status : **Lulus**

Adalah benar-benar sudah lulus pengecekan plagiasi dari Naskah Publikasi Skripsi, dengan menggunakan aplikasi Turnitin.

Demikian surat keterangan ini dibuat agar dipergunakan sebagaimana mestinya.

Wassalamu'alaikum Wr. Wb

Surakarta, 25 Juni 2020

Biro Skripsi Informatika

**Ihsan Cahyo Utomo, S.Kom., M.Kom.**



The screenshot shows the Turnitin feedback studio interface. The document title is "PERFORMANCE COMPARISON OF SEARCH FEATURE WITH DJANGO QUERYSETS AND ELASTICSEARCH IN A WEB APPLICATION". The abstract is visible, starting with "Search engine is an important tool for user to search relevant information quickly and easily." The match overview on the right shows a 10% match rate with 7 sources, each contributing 1% to the total match. The sources are:

| Rank | Source                    | Match Percentage |
|------|---------------------------|------------------|
| 1    | www.fullstackpython.c...  | 1%               |
| 2    | www.mdpi.com              | 1%               |
| 3    | Submitted to Technisc...  | 1%               |
| 4    | Submitted to Harrisbur... | 1%               |
| 5    | www.dtic.mil              | 1%               |
| 6    | www.ivoryresearch.com     | 1%               |
| 7    | Submitted to Queen M...   | 1%               |

Page: 3 of 18 | Word Count: 3293 | Text-only Report | High Resolution On

# PERFORMANCE COMPARISON OF SEARCH FEATURE WITH DJANGO QUERYSETS AND ELASTICSEARCH IN A WEB APPLICATION

## Abstract

Search engine is an important tool for user to search relevant information quickly and easily. It is especially an essential feature for the application that manage massive influx of data and information in their server. The implementation of search engine is wide and various ranged from famous web crawler such as Google, shopping sites such as Amazon to social media such as Facebook. The purpose of this research is to implement search engine technology to web application Arsip dan Dokumen UMS that handle every archive in University of Muhammadiyah Surakarta and developed with Django web framework. This research focused on the performance comparison of search engine between third-party search engine using Elasticsearch and search engine built with Django Querysets which have become the default implementation in this application. Each search engine must perform 12 search queries against sample of text contained in archive database field. This were repeated ten times for each query to obtain the best possible performance measurement in seconds. The archive database field populated with 1001 text samples extracted randomly from various Indonesian Wikipedia page. This research can prove how useful the implementation of Elasticsearch as search engine and its drawbacks.

**Index Terms:** Django, Elasticsearch, performance comparison, search engine, web application

## Abstrak

Mesin pencari adalah alat penting bagi pengguna untuk mencari informasi yang relevan dengan cepat dan mudah. Ini adalah fitur yang sangat penting terutama untuk aplikasi yang mengelola arus data dan informasi yang besar di server mereka. Implementasi mesin pencari sangat luas dan beragam mulai dari perayap web terkenal seperti Google, situs belanja seperti Amazon hingga media sosial seperti Facebook. Tujuan dari penelitian ini adalah untuk mengimplementasikan teknologi mesin pencari ke aplikasi web Arsip dan Dokumen UMS yang menangani setiap arsip di Universitas Muhammadiyah Surakarta dan dikembangkan dengan web framework Django. Penelitian ini berfokus pada perbandingan kinerja mesin pencari antara mesin pencari pihak ketiga menggunakan Elasticsearch dan mesin pencari yang dibangun dengan Django Querysets yang telah menjadi implementasi standar dalam aplikasi ini. Setiap mesin pencari harus melakukan 12 permintaan pencarian terhadap sampel teks yang terkandung dalam field database arsip. Hal ini diulang sepuluh kali untuk setiap permintaan untuk mendapatkan pengukuran kinerja terbaik dalam hitungan detik. Field database arsip berisi 1001 sampel teks yang diekstraksi secara acak dari berbagai halaman Wikipedia bahasa Indonesia. Penelitian ini dapat membuktikan betapa bermanfaatnya implementasi Elasticsearch sebagai mesin pencari dan kelemahannya.

**Index Terms:** Django, Elasticsearch, perbandingan kinerja, mesin pencari, web aplikasi

## 1. INTRODUCTION

Following the creation of website in 1990 and its availability for everyone with the announcement from CERN (Cailliau, 1995), search engine use began to resurge in popularity. Beginning with the creation of first commercialized search engine Yahoo! Search in 1994 (Oppitz & Tomsu, 2017), the use of search engine gained a traction among the growing community of web users as a tool with the ability to get relevant information quickly and easily. The use of search engine not only limited to web crawler such as Google and Bing, but even in many websites that maintain a large userbase and database such as shopping sites Amazon and Alibaba and social media such as Facebook and Twitter.

Search engine is an information retrieval program and presentation in response to user queries and has become important feature for web application where information changes dynamically. There are many properties for a search engine to get an attribute of a good engine. They include the ability to retrieve as many relevant documents as available in the dataset (Thamrin, Triyono, & Fadlilah, 2015). The crucial aspect of search engines is their quality and scalability without compromising the performance itself. That is not easy to achieve especially when information and user of search engine must handle increased, this in turn can cause the number of search queries also increased (Brin & Page, 1998).

The objective of this research was to compare performance of two different implementation of search feature in web-based Arsip dan Dokumen UMS application developed specifically using Django with a purpose as central archives management in University of Muhammadiyah Surakarta. The first implementation of search feature utilized Django built-in Querysets field lookup which essentially works the same way as SQL **WHERE** clause by specified keyword arguments. The second implementation utilized third-party search backend Elasticsearch, an open source search engine based on the Lucene library developed in Java with capability of near real-time information retrieval (Bendeche et al., 2019). Although there are many documentations and tutorials of how to implement Elasticsearch in Django, we have not yet found published studies that compare their performance.

For this research, the sample of data used for comparison was a collection of contents extracted from various sources of Indonesian Wikipedia which used to populate



fields in Arsip dan Dokumen UMS application database. A series of search query against database were performed to gather search result count and the time taken for each query.

## 2. METHOD

### 2.1. Development Tools

This research required to setup development environment and tools to be installed consist of hardware and software as outlined in Table 1.

Table 1. Development environment and tools

| Hardware  | Software  |
|---|---|
| <ul style="list-style-type: none"> <li>• Laptop Acer Aspire A515-41G-13JX, AMD Quad-Core Processor up to 3.60 GHz,</li> <li>AMD Radeon™ RX 540 with 2 GB VRAM,</li> <li>8GB DDR4 Memory, 1000 GB HDD</li> </ul> | <ul style="list-style-type: none"> <li>• Windows 10 Enterprise LTSC 64-bit</li> <li>• Docker Desktop</li> <li>• Cmdr Console Emulator</li> <li>• VS Code Editor</li> <li>• Sublime Text 3</li> <li>• Chromium Edge Browser</li> </ul> |

### 2.2. Application Development

Arsip dan Dokumen UMS is an application created with Django, a python web framework with the purpose as central management archives in University of Muhammadiyah Surakarta. The features also included uploading the file when creating a new archive and downloading it when user searching for an archive.

#### 2.2.1. Requirements Analysis

The application must meet the following requirements:

- 1) User authentication (login and logout).
- 2) Create, Read, Update and Delete (CRUD) an archive.
- 3) Search from database.
- 4) Download uploaded file online.

#### 2.2.2. Design and Coding

The design of the application follows the principle depicted by use case diagram in Figure 1.

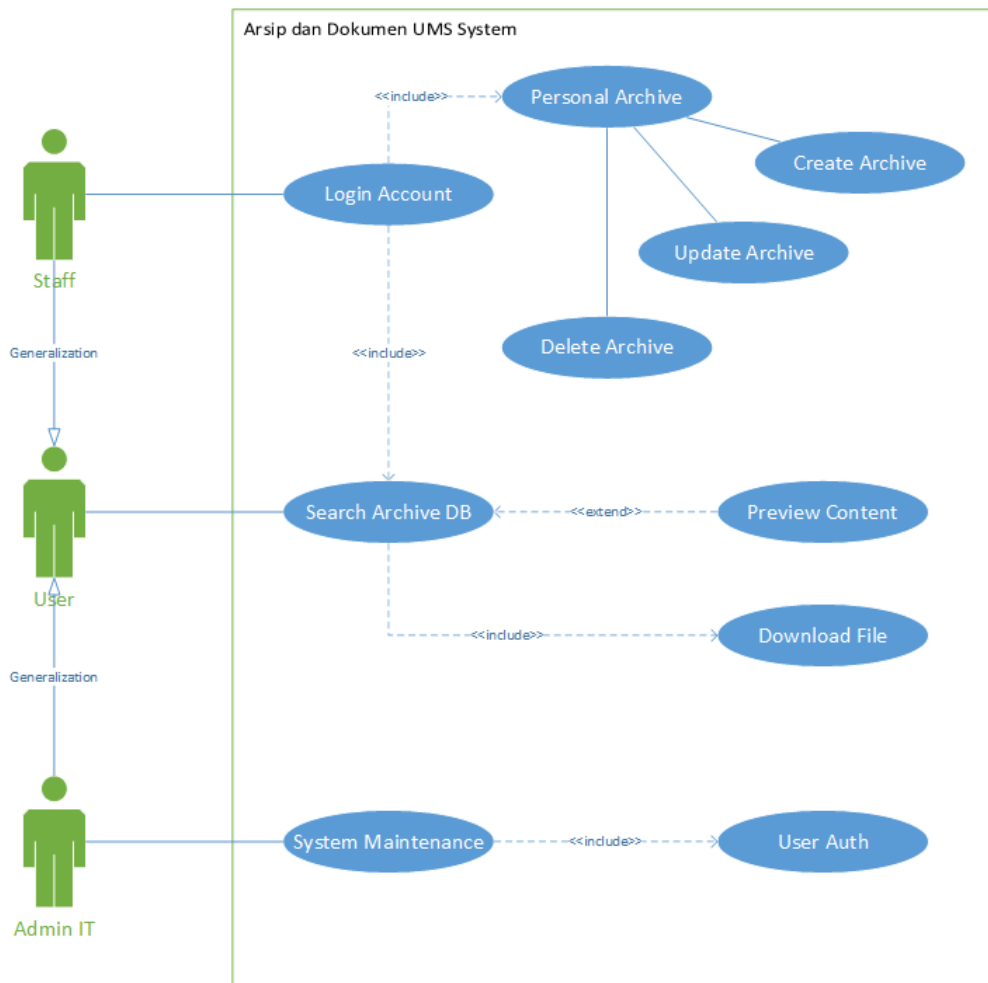


Figure 1. Use case diagram Arsip dan Dokumen UMS

The development done using Docker Desktop, an application for the building and sharing of containerized applications and microservices (Docker, Inc., n.d.). For this application, development setup divided into three services, first the web service that host Alpine Linux installed with Python 3.8.2 and Django 2.2 as the main service to host the web application code and occupied port 8000. The second was the database service installed with Alpine-based PostgreSQL as the application database and occupied default port 5432. The third service is Elasticsearch using centos-based Linux hosted within port 9200. The three services spun up three containers that communicate to each other within one network and together form one virtual development environment.

### 2.2.3. Testing

This research adopted *unit testing* and *black box testing*. The *unit testing* performed to test the code inside the application itself by writing the test code to determine if the archive CRUD operation works as intended.

The *black box testing* were performed after the *unit test* to confirm if the application really working as intended and to discover unexpected bugs or error if any which done directly from web User Interface (UI).

#### 2.2.4. Deployment

The application pushed to Gitlab a Git-repository manager for version control. The deployment of application leveraged Gitlab CI/CD, a built-in tool for automating the deployment. The current application also utilized Heroku, a container-based cloud Platform as a Service (PaaS) where the application deployed and maintained (Salesforce.com, Inc., n.d.). The deployment in Heroku is quite straightforward and easy to manage. Heroku also provides a generous free tier which are extremely useful to test the application within the deployment environment.

### 2.3. Material

The search engine requires to search text against the archive database field. Django web framework includes object-relational mapping layer (ORM) that can be used to interact with application data from various relational databases such as SQLite, PostgreSQL and MySQL. The database generated by creating an archive model first. Model is a Django representation of a table in database. Each model maps to a single database table. Table 2 depict archive model attributes created in this application where each attribute represents each field in the database.

Table 2. Archive model database representation in Django

| Field Name               | Models Type  | Attribute                 |
|--------------------------|--------------|---------------------------|
| id (primary key)         | AutoField    | auto_created=True         |
| pengunggah (foreign key) | ForeignKey   | on_delete=models.SET_NULL |
| Nama                     | CharField    | max_length=200            |
| Deskripsi                | TextField    | null=True                 |
| tanggal_unggah           | DateField    | auto_now_add=True         |
| Jenis                    | IntegerField | default=0                 |
| Sifat                    | IntegerField | default=0                 |
| publik_mulai             | DateField    | null=True                 |
| publik_berakhir          | DateField    | null=True                 |
| file_media               | FileField    | max_length=200            |

### 2.3.1. Sample Acquisition

To search text within the field database, it needs to be populated first. This research use archive database but the search feature implemented in this application does not really care whether the text contained within the database is truly an archive or just some random generated text. The search needs only to do well matching search query from the user. This also allows separating of concern should the application wish to implement a feature where content within the database must be an archive by detecting certain pattern which an archive should have.

For this research, the archive database populated with random text extracted from Wikipedia of Indonesia using the technique called web scrapping with the help from third-party python package *wikipedia* ([pypi.org/project/wikipedia/](https://pypi.org/project/wikipedia/)). As the name implies, the package used to extract content from a Wikipedia page which in this research used to extract both page title and summary of a Wikipedia topic to populate both field nama and field deskripsi respectively in archive database. The current research collected a thousand and one (1001) Wikipedia topics as samples to be searched.

### 2.3.2. Search Engine

The application used two different search implementations which can switched as needed for conducting a performance comparison analysis. The first search implemented with Django Querysets is simply an implementation of object-relational mapping layer (ORM) used to interact with application data from database. The search with Querysets basically performs a field lookup which translates from Django ORM as SQL **WHERE** clause and return a new set of queries based on specified arguments.

The second search implemented with Elasticsearch require an additional package, *django-elasticsearch-dsl* that allows Elasticsearch running in port 9200 to index Django database model. Elasticsearch uses the indexing concept. It is a document oriented tool. Once the document is added, it can be searched within a next second (Kalyani & Mehta, 2017). The application requires at least once for Elasticsearch to run search indexing against the application database. Every time user performs CRUD operation to the archive model, Elasticsearch

automatically update its index to reflect the change within the application database.

The search also applies additional filtering determined by the value of field **sifat** as depicted in Table 2 and whether or not a user is authenticated. The field **sifat** is an **IntegerField** type which store integer number ranged from 0 - 3 where each represents a key of value trait an archive can have. Table 3 outlined each of key-value pair in field **sifat**.

Table 3. Field sifat key-value pain in Archive database

| Key | Value         |
|-----|---------------|
| 0   | Publik        |
| 1   | Publik (temp) |
| 2   | Internal      |
| 3   | Pribadi       |

The archive with value **Publik** and **Publik (temp)** will always visible in search results whether a user is authenticated or not. The difference is that **Publik** remain publicly available as long as the archive still exist in database, while **Publik (temp)** has a time limit which it can remain available. The value Internal means the archives are only available in the search result for authenticated user only. The archive with a trait **Pribadi** is simply a private archive that only the owner of the archive has access to it.

#### 2.4. Comparison Analysis

To obtain the comparison performance, the first step is to determine the keywords for search query. The focus of this research is purely on the performance side and because of that certain keywords must be discarded if the search did not return any result. The other consideration is for search query to return the same result count for both Django Querysets and Elasticsearch which is used to determine the performance both search engine in a balanced situation. To test an extreme case, the author chose two search queries where the search must return result with wide disparity count. The total target is 12 search queries where 10 queries with similar or small difference and 2 others for extreme case with wide disparity search results

The next step is to measure the search performance time using python module *timeit* that provides a way to time small bits of code (Python Software Foundation,

2020). This achieved by specifying a variable start timer before the beginning of search algorithm, and variable end timer after the end of search algorithm each for Django Querysets search and Elasticsearch. The performance result was calculated by subtracting the end timer with the start timer to obtain search time in seconds and then displayed on the search page.

The search was conducted by manually typing search keywords in input box of the search page. This conducted for ten times for each search keywords to obtain the best time possible that can be measured.

### 3. RESULT AND DISCUSSION

#### 3.1. Application Result

The application web-based Arsip dan Dokumen UMS are capable of archival, retrieval and search of file documents. Figure 2 shows the homepage of web application for a guest user.

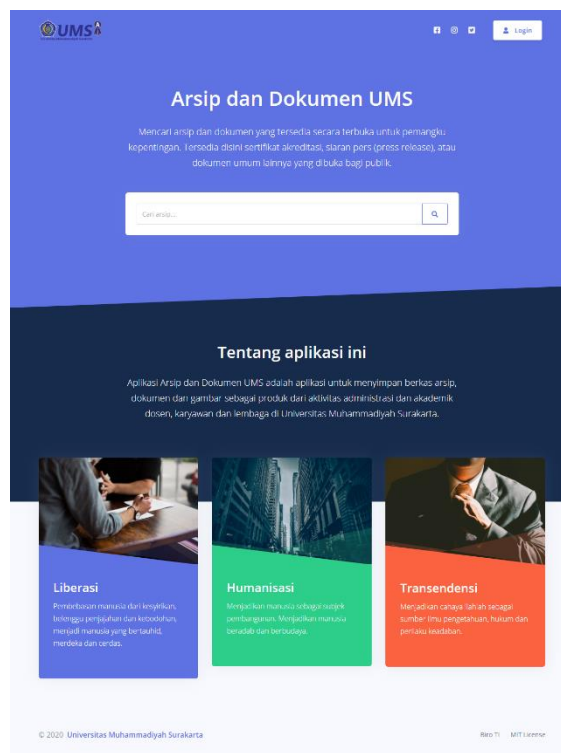


Figure 2. Homepage

There are two important section within the homepage. The top-half section is an introduction section and the bottom-half are about section. The user can login from Login navigation shown at the top-right of the header that will redirect user to login page. The login page consists of one simple login form as depicted in Figure 3.

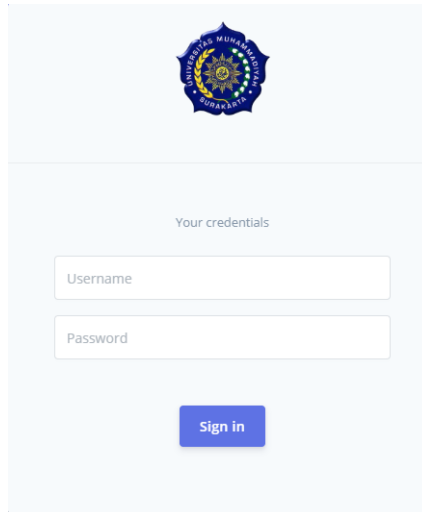


Figure 3. Login form

From the homepage user can also enter a search query within input box that will redirect user to search page depicted in Figure 4.

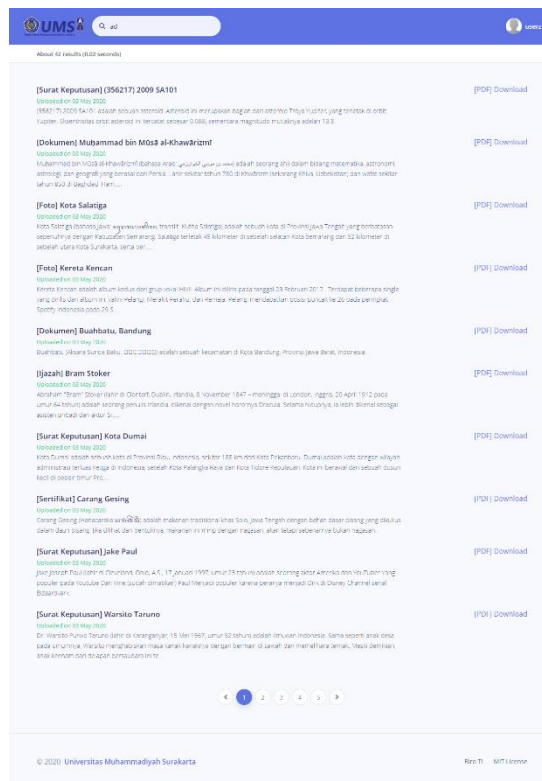


Figure 4. Search page

At the top-right of header after a user login, the previous login navigation was gone and replaced with user icon and username display and when clicked will display a drop-down menu depicted in Figure 5.

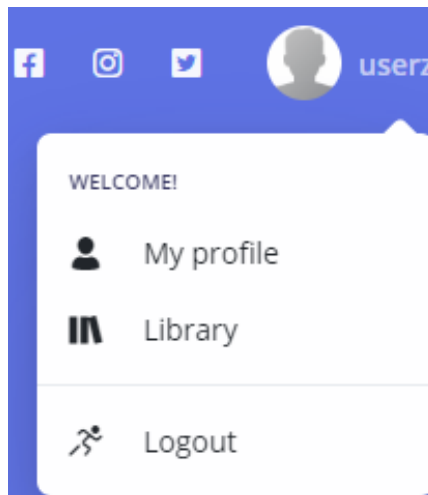


Figure 5. User menu dropdown

There are two navigation bars, My Profile that will redirect to user profile page and Library that will redirect to list of archives uploaded by users as depicted in Figure 6.

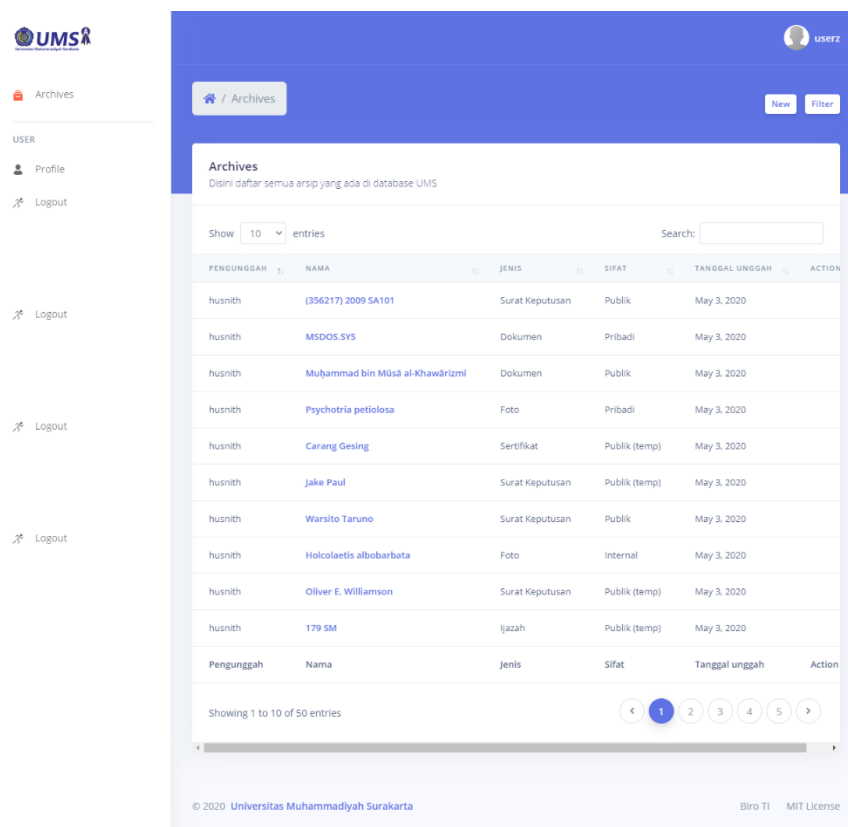


Figure 6. Library page

The library page is also the place which contain links to perform CRUD operation for archive such as new button at the top-right of table that will redirect user to archive create page as depicted in Figure 7.



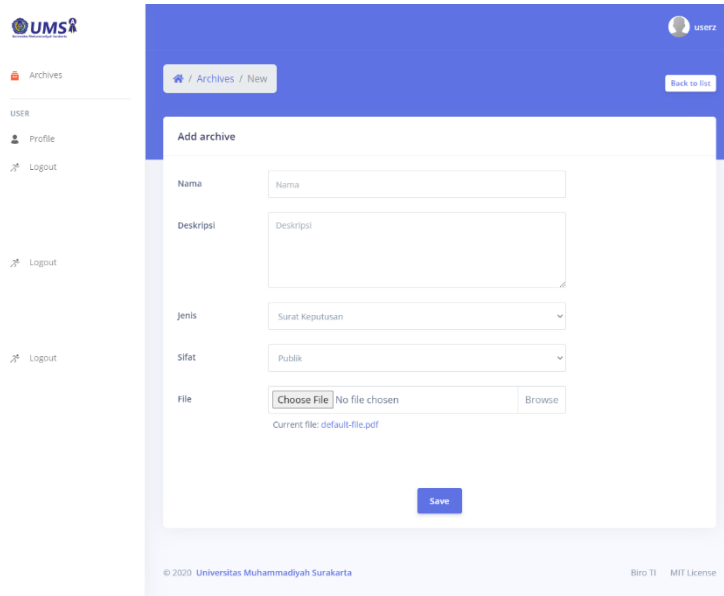


Figure 7. Archive create page

The user can also view the detail of each archive in the list by simply clicking the name of each archive which will redirect user to the page as depicted in Figure 8.

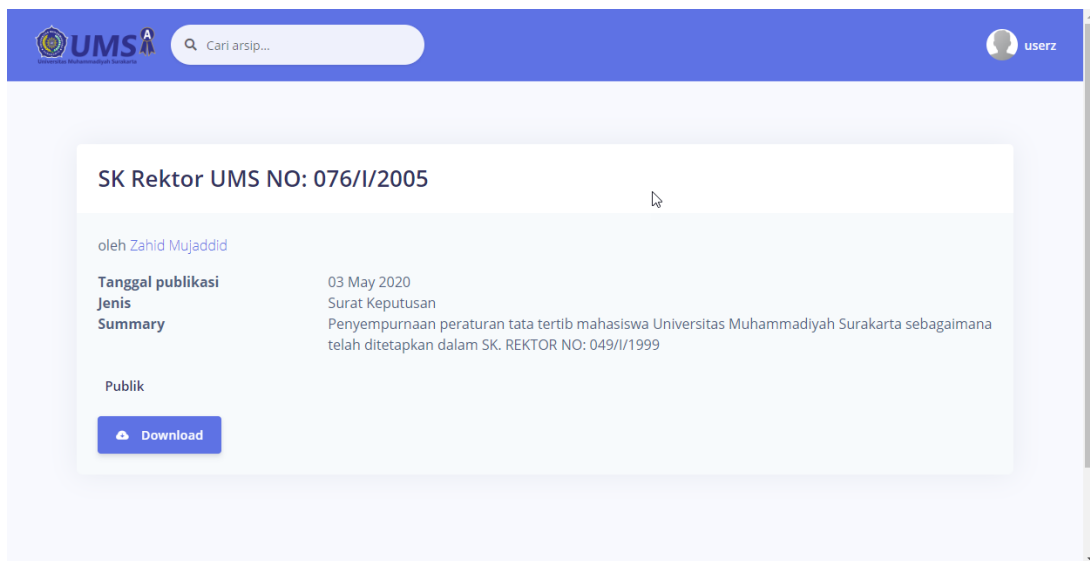


Figure 8. Archive detail page

## 3.2. Test Result

### 3.2.1. Unit Test

Table 4 depict each of the *unit test* performed within the application.

Table 4. Unit test performed

| Test                     | Status |
|--------------------------|--------|
| Test home page displayed | OK     |
| Test archive model works | OK     |

| Test                              | Status |
|-----------------------------------|--------|
| Test add an archive works         | OK     |
| Test update an archive works      | OK     |
| Test remove an archive works      | OK     |
| Test get the list of all archives | OK     |

### 3.2.2. Black Box Test

Table 5 depict each of *black box testing* performed in this application.

Table 5. Black box testing performed

| Test                        | Status |
|-----------------------------|--------|
| User can login              | OK     |
| User can logout             | OK     |
| User can create an archive  | OK     |
| User can update an archive  | OK     |
| User can delete an archive  | OK     |
| User can search the archive | OK     |

### 3.3. Comparison Result

The first stage of comparison is search queries which return the same result count as outlined in Table 6.

Table 6. Performance comparison of both search engine with the same result count

| Search Query | Django Querysets | Elasticsearch | Result |
|--------------|------------------|---------------|--------|
| film         | 0.06             | 0.09          | 46     |
| latin        | 0.06             | 0.1           | 16     |
| pengetahuan  | 0.06             | 0.07          | 4      |
| penggunaan   | 0.06             | 0.1           | 13     |
| permainan    | 0.05             | 0.07          | 4      |

Overall Django Querysets search faster by small margin compared to Elasticsearch for the same search result count with difference by 0.3 seconds in average. Search with Django Querysets also shows rather consistent performance across different search result which best attributed itself for being a built-in function of

Django ORM implementation. The next stage is comparison of search queries with different result count as outlined in Table 7 and Table 8.

Table 7. Performance comparison of both search engine with small difference result count

| Search Query | Django Querysets |          | Elasticsearch |          |
|--------------|------------------|----------|---------------|----------|
|              | Result           | Time/sec | Result        | Time/sec |
| Ekonomi      | 9                | 0.05     | 8             | 0.07     |
| Ibu Kota     | 23               | 0.04     | 16            | 0.07     |
| Industri     | 11               | 0.07     | 8             | 0.08     |
| Kabupaten    | 65               | 0.05     | 64            | 0.13     |
| pusat        | 23               | 0.06     | 17            | 0.1      |

Table 8. Performance comparison for both search engine with massive difference result count

| Search Query | Django Querysets |          | Elasticsearch |          |
|--------------|------------------|----------|---------------|----------|
|              | Result           | Time/sec | Result        | Time/sec |
| Dewa         | 41               | 0.06     | 7             | 0.09     |
| Ad           | 797              | 0.09     | 3             | 0.08     |

In this instance search with Django Querysets still outperform Elasticsearch even when the result count is higher than its counterpart. The only instance Elasticsearch outperform Django Querysets is when there is a massive difference of result count as depicted in search query “Ad”. The performance test in this research only conducted in one query at a time and thus didn’t account for performance of multiple queries at once which requires for this research to be expanded in future studies.

Elasticsearch also shown to always return results with fewer count than search with Django Querysets because of its advanced nature which built upon Lucene library where the relevancy of search result are calculated using practical scoring functions (Bhandarkar & B. N., 2020). This best depicted by search query in Table 8 where Elasticsearch manage to find the meaning of “Dewa” which means God and return seven such relevant result.

Search with Django Querysets is simply an implementation of complex database field lookup using QuerySet, a database-abstraction API. This is very easy to

setup because QuerySet is built-in API provided by Django and is a standard when working with database in Django way.

Search with Elasticsearch on the other hand, require more complicated setup and additional search backend configuration within Django. Elasticsearch also needs to run from a different port first for the application to be able to connect and use it. The algorithm to create search with Elasticsearch also prone to error without proper reading of documentation.

#### **4. CONCLUSIONS**

Quantitative analysis of search requires proper knowledge of the available search engines and their applicability to specific types of application because the choice to build search feature depends entirely on the complexity of data that will be handled by the application itself.

For comparison between two search implementation, Django Querysets generally is faster and easier to work with because its built-in nature as Django object-relational mapping (ORM) implementation and has become standard when interacting with database in Django way. Search with Elasticsearch is slower only by small margin and return fewer result, but more relevant because it utilized Lucene library for score calculation of search result. Elasticsearch is also a lot harder to implement since it requires the help of external package, additional configuration and even more coding to utilize its rich features.

Further studies are still necessary especially in regards search performance and precision where multiple search queries at once involved to fully understand Elasticsearch full capability and its advantage in performance.

#### **REFERENCES**

- Bendeche, M., Svorobj, S., Endo, P. T., Mario, M. N., Ares, M. E., Byrne, J., & Lynn, T. (2019). Modelling and Simulation of ElasticSearch using CloudSim. *Proceedings - 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications, DS-RT 2019*. <https://doi.org/10.1109/DS-RT47707.2019.8958653>.
- Bhandarkar, S., & B. N., N. (2020). A Full-Text-Based Search Algorithm vs Elasticsearch. *Studies in Indian Place Names, UGC Care Journal*, 40(74), 2168–2171.
- Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 107-117.

- Cailliau, R. (1995). *History: W3C*. Retrieved from A Little History of the World Wide Web: <https://www.w3.org/History.html>.
- Docker, Inc. (n.d.). *Docker Desktop: The fastest way to containerize applications on your desktop*. Retrieved from Docker: <https://www.docker.com/products/docker-desktop>.
- Kalyani, D., & Mehta, D. (2017). Paper on Searching and Indexing Using Elasticsearch. *International Journal of Engineering and Computer Science*, 21824-21829. doi:10.18535/ijecs/v6i6.44.
- Oppitz, M., & Tomsu, P. (2017). *Inventing the Cloud Century: How Cloudiness Keeps Changing Our Life, Economy and Technology*. Springer. doi:10.1007/978-3-319-61161-7.
- Python Software Foundation. (2020, April 20). *timeit — Measure execution time of small code snippets*. Retrieved from Python 3.8.2 documentation: <https://docs.python.org/3/library/timeit.html>.
- Salesforce.com, Inc. (n.d.). *What is Heroku?* Retrieved from Heroku: <https://www.heroku.com/about>.
- Thamrin, H., Triyono, A., & Fadlilah, U. (2015). Penggunaan Kamus Sinonim dan Hiponim sebagai Sumber Ekspansi Kueri dalam Sistem Temu Kembali Informasi Berbahasa Indonesia. *University Research Colloquium (URECOL)*, 43-49. Retrieved from <http://hdl.handle.net/11617/5109>.