

Dexur Enterprises Private Limited

A PROJECT REPORT

Long Term Internship

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

by

Palthi Malleswari (N190878)

Under the Guidance of

Ms. Kalavathi, Dept of CSE



Department of Computer Science and Engineering
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
Nuzvid, Eluru - 521 202



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Rajiv Gandhi University of Knowledge Technologies, Nuzvid,

Eluru, Andhra Pradesh - 521 202.

CERTIFICATE OF COMPLETION

This is to certify that the work entitled, “*Longterm Internship at Dexur*” is the bonafied work of **Palthi Malleswari (ID No: N190878)**, carried out under my guidance and supervision for 4th year Major project of Bachelor of Technology in the department of Computer Science and Engineering under RGUKT IIIT, Nuzvid. This work is done during the academic session July 2024 – May 2025, under our guidance.

Ms. Kalavathi

Assistant Professor,
Department of CSE,
RGUKT, Nuzvid.

Ms. S.Bhavani

Assistant Professor,
Head of the Department,
Department of CSE,
RGUKT, Nuzvid.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Rajiv Gandhi University of Knowledge Technologies, Nuzvid,

Eluru, Andhra Pradesh - 521 202.

CERTIFICATE OF EXAMINATION

This is to certify that the work entitled, “*Longterm Internship at Dexur*” is the bonafied work of **Palthi Malleswari (ID No: N190878)**, and here by accord our approval of it as a study carried out and presented in a manner required for its acceptance in the 4th year of Bachelor of Technology for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn, as recorded in this thesis. It only signifies the acceptance of this thesis for the purpose for which it has been submitted.

Ms. Kalavathi

Assistant Professor,
Department of CSE,
RGUKT, Nuzvid.

Project Examiner



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Rajiv Gandhi University of Knowledge Technologies, Nuzvid,

Eluru, Andhra Pradesh - 521 202.

DECLARATION

I “**Palathi Malleswari (ID No: N190878)**” hereby declare that the project report entitled “*Longterm Internship at Dexur*” done by us under the guidance of **Ms. Kalavathi**, Assistant Professor, is submitted for the fulfillment of a Major Intern project during the academic session July 2024 - May 2025 at RGUKT-Nuzvid. I also declare that this project is a result of my own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Date: 29-04-2025

Palathi Malleswari (N190878)

Place: RGUKT-NUZVID

ACKNOWLEDGEMENT

I would like to express my gratitude towards my Internal advisors **Ms. Kalavathi** Mam respectively for guiding me through the extensive research process. Lastly, I would like to thank my family and friends for their encouragement, suggestions and belief in me.

I am extremely grateful for the confidence bestowed in me and entrusting my project entitled "*Longterm Internship at Dexur*".

I express my gratitude to **Ms. S.Bhavani (HOD of CSE)** and other faculty members for being source of inspiration and constant encouragement which helped me in completing the project successfully.

Finally, yet importantly, I would like to express our heartfelt thanks to our beloved God and parents for their blessings, my friends for their help and wishes for the successful completion of this project.

ABSTRACT

At Dexur Enterprises Private Limited, a company focused on AI-driven workflow automation and analytics, I contributed to key initiatives aimed at enhancing product performance and scalability. My work spanned three major projects that significantly improved data handling, reporting capabilities, and access control systems.

In the **Dynamic Query Building project**, I led the backend **transition from PostgreSQL to OpenSearch** to address performance issues associated with filtering large datasets. This involved developing dynamic DSL queries using Python and Django, enabling real-time, efficient querying based on user-selected filters from the frontend.

The **Report Builder project** aimed to replace **Apache Superset**, which had high response latency, with a custom reporting solution. I implemented a streamlined reporting system by integrating OpenSearch with React-based charting components, delivering faster and more interactive data visualizations.

Additionally, I played a key role in **enhancing the permissions system** to offer more granular and flexible access control. I also contributed to various data migration tasks through custom scripting, ensuring seamless transitions during infrastructure upgrades.

Contents

1	Introduction	8
2	Related Work/ Existed Work	9
2.1	Data Listing Served via PostgreSQL Backend	9
2.2	Superset For Reporting	10
2.3	Basic Role Based Permissions	11
2.4	Challenges faced in Existed Work	12
2.4.1	Slow Query Performance with PostgreSQL	12
2.4.2	Limited Scalability of Superset for Reporting	12
2.4.3	High Memory Usage for Visualizations	12
2.4.4	Rigid and Basic Permissions System	12
2.4.5	No Record-Level or Field-Level Access Control	13
2.4.6	Lack of Rule-Based Permission Handling	13
3	Proposed System	14
3.1	Proposed Solutions to Solve the Challenges	14
3.1.1	Migration to OpenSearch	14
3.1.2	Replacing Superset with Custom Report Builder	14
3.1.3	Efficient Visualization via OpenSearch Dashboards and React Components	15
3.1.4	Enhanced Permissions Model	15
3.1.5	Advanced Access Configuration	15

3.1.6	Rule-Based Access Implementation	16
4	Results	17
5	Future Work	21
5.1	Indian CMS-Style Hospital Ratings	21
5.2	AI-Powered Health Predictionsn	21
5.3	Rural Healthcare Connectivity	21
5.4	Smart Patient Feedback Sentiment Analysis	22
5.5	Outcome-Based Payment Models	22
5.6	Unified Digital Health Records	22
6	Conclusion	23
7	References	24

1

Introduction

As a Software Engineer Intern at **Dexur Enterprises**, I had the opportunity to contribute to several high-impact projects focused on enhancing system performance, scalability, and security. My primary responsibilities involved optimizing data handling processes, improving report generation capabilities, and upgrading the permissions system to meet complex client requirements. These projects were instrumental in enhancing the efficiency of our product, particularly in areas where large datasets and complex filtering were previously causing performance bottlenecks.

One of my key contributions was the **migration from PostgreSQL to OpenSearch** to address slow query performance. I also led efforts to replace Apache Superset with a custom-built report builder using **OpenSearch and React**, which significantly reduced report generation times. Additionally, I played a vital role in **revamping the permissions system** by implementing record-level, field-level, and rule-based access controls, ensuring more granular and secure data access. Through these initiatives, I gained valuable experience in **backend optimization, dynamic query generation, and real-world problem-solving in a fast-paced startup environment**.

This internship not only enhanced my technical skills but also strengthened my ability to deliver impactful solutions in high-pressure environments, contributing meaningfully to supply chain innovation.

2

Related Work/ Existed Work

2.1 Data Listing Served via PostgreSQL Backend

The screenshot shows the DEZUR web application interface. At the top, there is a navigation bar with links for Quality, Browse Modules, Analytics, Reporting & Dashboards, Patients & Encounters Reports, Incidents & Events: End User Entry, and Incidents & Events: Manager & Admin Reviews & Update. A search bar and user account dropdown are also present. Below the navigation bar, the 'Encounters' section is active, showing a table of patient encounters. The table has columns for Record ID, Record Added Date, Record Source, Record State, Hospital Name, Patient First Name, Patient Middle Name, Patient Last Name, HAN/Patient Account Number, MIN #, Patient Type, and Patient State. A search dropdown is open, showing options like Discharge Status, Length Of Stay, External Injury 1 POA, External Injury 2 POA, and External Injury 2 POA. The table displays several rows of data, including records for DESHIA, STEPHANIE, KALEY, JAIRO, KIMBERLY, and ROBERT.

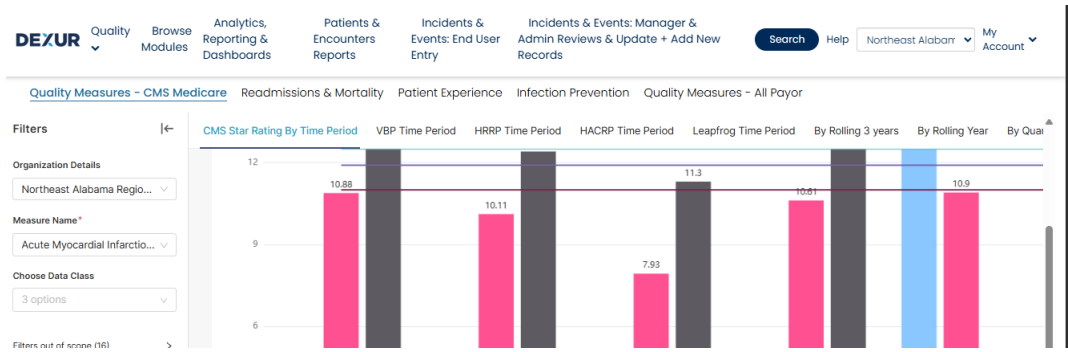
Record ID	Record Added Date	Record Source	Record State	Hospital Name	Patient First Name	Patient Middle Name	Patient Last Name	HAN/Patient Account Number	MIN #	Patient Type	Patient State
676342444	Apr 19 2025, 7:12 AM EST	ADT		REGIONAL MEDICAL CENTER BOARD	DESHIA			394996	67900	Outpatient	
676342443	Apr 19 2025, 7:12 AM EST	ADT		REGIONAL MEDICAL CENTER BOARD	STEPHANIE	A		394995	109120	Outpatient	AL
676342442	Apr 19 2025, 7:12 AM EST	ADT		REGIONAL MEDICAL CENTER BOARD	KALEY			394994	427964	Outpatient	AL
676342441	Apr 19 2025, 7:12 AM EST	ADT		REGIONAL MEDICAL CENTER BOARD	JAIRO			394993	55299	Outpatient	AL
676342440	Apr 19 2025, 7:12 AM EST	ADT		REGIONAL MEDICAL CENTER BOARD	KIMBERLY	L		394992	520989	Outpatient	AL
676342439	Apr 19 2025, 7:12 AM EST	ADT		REGIONAL MEDICAL CENTER BOARD	ROBERT			394991	050321	Outpatient	AL

One of the key performance bottlenecks encountered in our system stemmed from the use of PostgreSQL for executing complex filters on large datasets. As a startup, the initial database design was tailored for a limited client base, with all clients' records stored in a single monolithic table.

Over time, as the volume of data grew, this structure became inefficient—particularly for filtering operations. Queries that required filtering across millions of records experienced significant delays, **with page load times ranging from 9 to 15 seconds**. This was primarily due to **PostgreSQL's limitations in handling complex, large-scale filtering**

operations without optimized indexing or data partitioning, which highlighted the need for a more scalable and high-performance search solution.

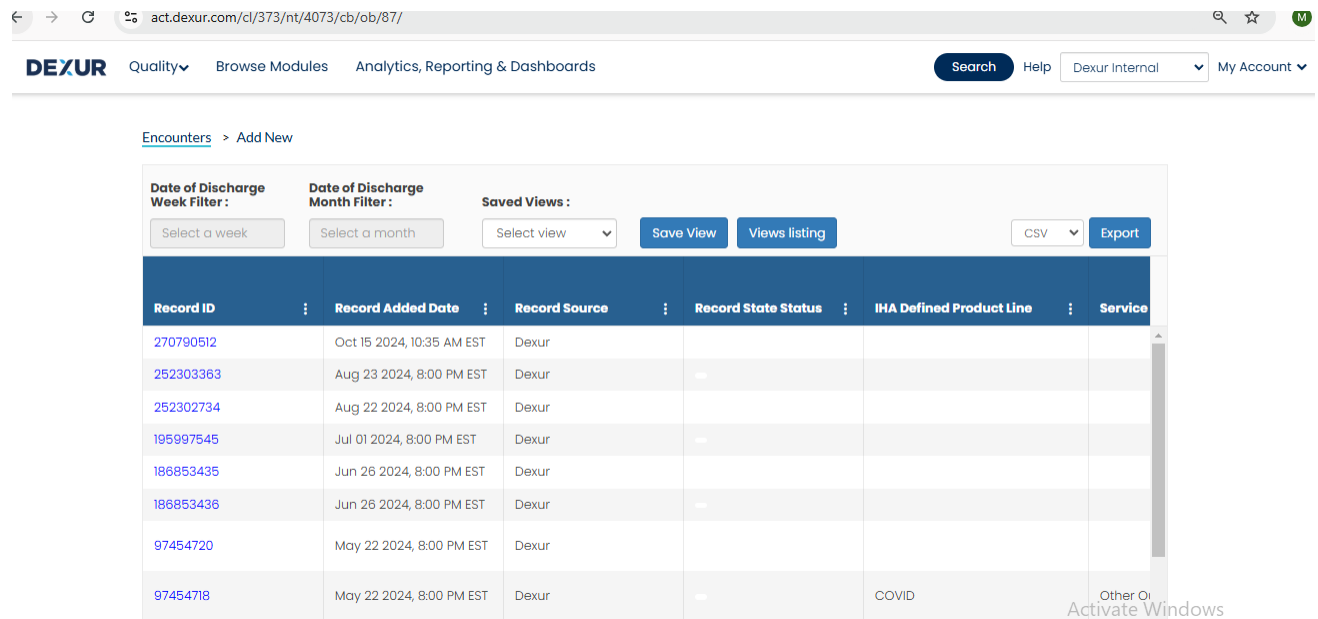
2.2 Superset For Reporting



Apache Superset is an open-source data visualization and dashboarding tool that enables users to create interactive reports, charts, and dashboards using either SQL queries or a no-code interface. It supports a wide range of data sources and provides a user-friendly platform for business intelligence tasks, allowing teams to explore and present data insights quickly. Superset is especially useful for non-developers who want to build dashboards without writing complex backend logic.

However, Superset has significant limitations when working with large-scale data. **As data volume grows, the platform struggles with performance**—especially when executing complex SQL queries involving window functions, subqueries, or multiple joins. **These queries often lead to long wait times (10–15 seconds or more)**, putting heavy load on the database. Additionally, visualizing millions of records consumes a substantial amount of memory, which can slow down or even crash the interface. These limitations made it necessary to adopt a more efficient, customized solution for reporting.

2.3 Basic Role Based Permissions



The screenshot displays the Dexur web application interface. At the top, there is a navigation bar with the Dexur logo, links for Quality, Browse Modules, and Analytics, Reporting & Dashboards, a Search button, a Help link, a dropdown menu for Dexur Internal, and a My Account dropdown. Below the navigation bar, the breadcrumb trail shows 'Encounters > Add New'. The main content area features a table with columns: Record ID, Record Added Date, Record Source, Record State Status, IHA Defined Product Line, and Service. The table contains eight rows of data. Above the table, there are filters for 'Date of Discharge Week Filter' (Select a week), 'Date of Discharge Month Filter' (Select a month), and 'Saved Views' (Select view). There are also buttons for 'Save View', 'Views listing', 'CSV', and 'Export'. A watermark 'Activate Windows' is visible in the bottom right corner of the screenshot.

Record ID	Record Added Date	Record Source	Record State Status	IHA Defined Product Line	Service
270790512	Oct 15 2024, 10:35 AM EST	Dexur			
252303363	Aug 23 2024, 8:00 PM EST	Dexur			
252302734	Aug 22 2024, 8:00 PM EST	Dexur			
195997545	Jul 01 2024, 8:00 PM EST	Dexur			
186853435	Jun 26 2024, 8:00 PM EST	Dexur			
186853436	Jun 26 2024, 8:00 PM EST	Dexur			
97454720	May 22 2024, 8:00 PM EST	Dexur			
97454718	May 22 2024, 8:00 PM EST	Dexur		COVID	Other Or

Initially, the permissions system at Dexur was limited to only **basic operations such as viewing, editing, and adding records**. While this simple access model worked during the early stages of product development, it quickly became insufficient as client needs grew more complex. The system lacked the flexibility to handle scenarios where access needed to be restricted based on specific users, roles, or data types. This limited model meant

that all users with edit rights could access and modify any record, regardless of ownership or sensitivity. **There was no support for finer controls like restricting access to individual records, hiding fields, or applying conditional rules**. As a result, it became difficult to **enforce data privacy and role-based restrictions**, which are essential for enterprise-grade systems handling diverse user roles and sensitive information.

2.4 Challenges faced in Existed Work

During the development and optimization of data handling, reporting, and access control systems, several challenges were encountered that affected performance, scalability, and reliability:

2.4.1 Slow Query Performance with PostgreSQL

PostgreSQL struggled to handle complex filters and large datasets efficiently. As all client records were stored in a single table, filtering required scanning massive volumes of data, resulting in high load times (**9–15 seconds**).

2.4.2 Limited Scalability of Superset for Reporting

Superset, while user-friendly, became a bottleneck for performance when working with large data volumes. Heavy SQL queries with joins and window functions caused delays and sometimes overloaded the system.

2.4.3 High Memory Usage for Visualizations

Generating visualizations with millions of rows in Superset consumed significant memory resources. This led to slow rendering of charts and sometimes caused system instability or timeouts.

2.4.4 Rigid and Basic Permissions System

The earlier permission model was overly simplistic, offering only basic access options like view, edit, and add. It lacked the flexibility needed for modern enterprise use cases with varied user roles and access control requirements.

2.4.5 No Record-Level or Field-Level Access Control

There was no way to restrict data access at the record or field level. This meant users could potentially view or edit data they weren't supposed to, risking data security and client trust.

2.4.6 Lack of Rule-Based Permission Handling

Permissions could not be customized based on conditions or rules. This made it difficult to enforce dynamic access logic—such as granting access based on user roles, ownership, or custom business rules—limiting the system's adaptability to client-specific needs.

To address these challenges, we migrated to OpenSearch for faster filtering, replaced Superset with a custom report builder for better performance, and upgraded the permissions system to support more flexible access controls. We also improved backend logic, automated query generation, and optimized data scripts for better speed and reliability.

3

Proposed System

3.1 Proposed Solutions to Solve the Challenges

3.1.1 Migration to OpenSearch

To overcome the latency issues caused by PostgreSQL when filtering large datasets, we migrated to **OpenSearch**—a distributed, high-performance search and analytics engine. OpenSearch supports full-text search, filtering, and aggregations with much faster **response times (1–1.5 seconds for complex queries)**, **significantly improving page load performance**. Its distributed architecture ensures scalability as data volume grows.

3.1.2 Replacing Superset with Custom Report Builder

Superset was replaced with a custom-built report generation tool using **OpenSearch + React**. This setup provides **low-latency querying and more interactive data visualizations**. OpenSearch allows for real-time analytics and fast querying across indexed datasets, eliminating delays caused by heavy SQL execution in Superset

3.1.3 Efficient Visualization via OpenSearch Dashboards and React Components

Instead of rendering millions of records at once, **we optimized** visualizations by leveraging OpenSearch's aggregation capabilities. Coupled with lightweight **React components**, this allowed for **scalable, memory-efficient, and fast-loading** dashboards that reduced browser strain and resource consumption.

3.1.4 Enhanced Permissions Model

The permissions system was completely revamped to provide granular access control. We implemented:

- **Record-Level Control:** Users see only the records they created, while managers can access a broader scope.
- **Field-Level Control:** Sensitive fields are conditionally shown or hidden based on the user's role.
- **Rule-Based Access:** Permissions adapt to custom business rules.

In total, **56 unique permissions were implemented** to match diverse client needs.

3.1.5 Advanced Access Configuration

By introducing field- and record-level controls, we ensured **sensitive data visibility is limited to authorized users**. This enhances data privacy and aligns with client-specific security requirements, solving a key limitation in the original system.

3.1.6 Rule-Based Access Implementation

We developed a **flexible rule engine** where **permissions adapt dynamically to user roles, conditions, or data contexts**. This allows for more intelligent, context-aware access management across modules.

These improvements significantly enhanced system performance, scalability, and security—resulting in faster data retrieval, more efficient reporting, and robust access control tailored to client needs.

4

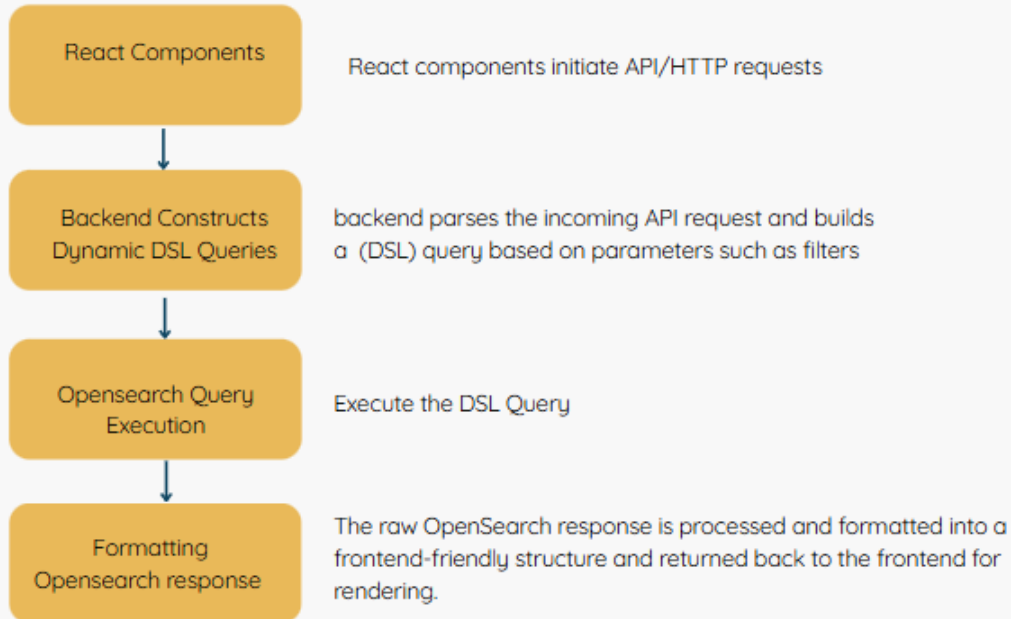
Results

The improvements made across our data handling and reporting systems greatly enhanced performance, scalability, and reliability. Migrating to **OpenSearch reduced response times and improved filtering accuracy, even with large datasets**. The dynamic query builder allowed flexible query generation, reducing backend load and improving responsiveness.

Replacing Superset with a custom report builder using OpenSearch and React resulted in faster, lightweight visualizations, significantly lowering memory usage. The upgraded permissions system, with record-level, field-level, and rule-based controls, ensured secure and role-specific data access. Additionally, optimizing backend scripts and data migration processes contributed to more stable deployments and smoother operations across projects. Overall, these changes made the system faster, more secure, and better aligned with real-world client needs.

Report Builder Platform

Implementation



Filters

Organization Name

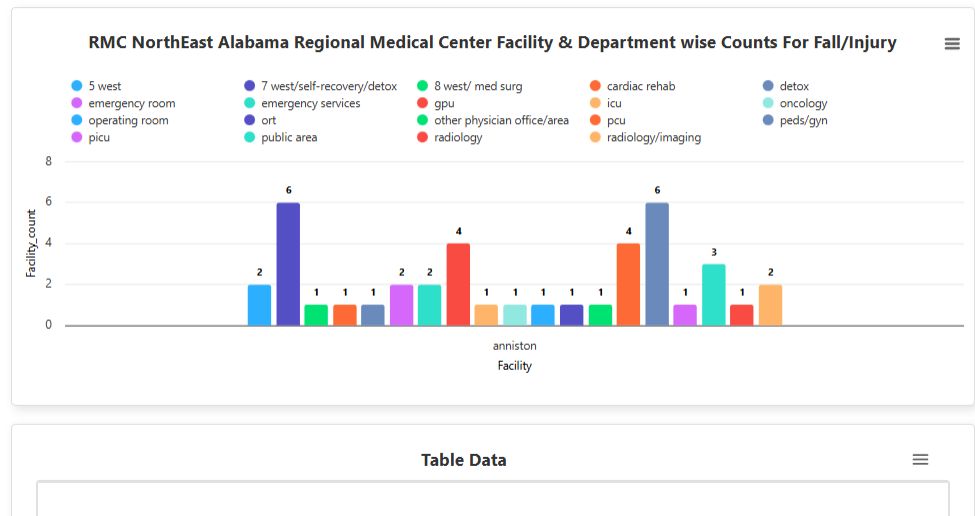
Select Organization Name

Record Source *

Select Record Source *

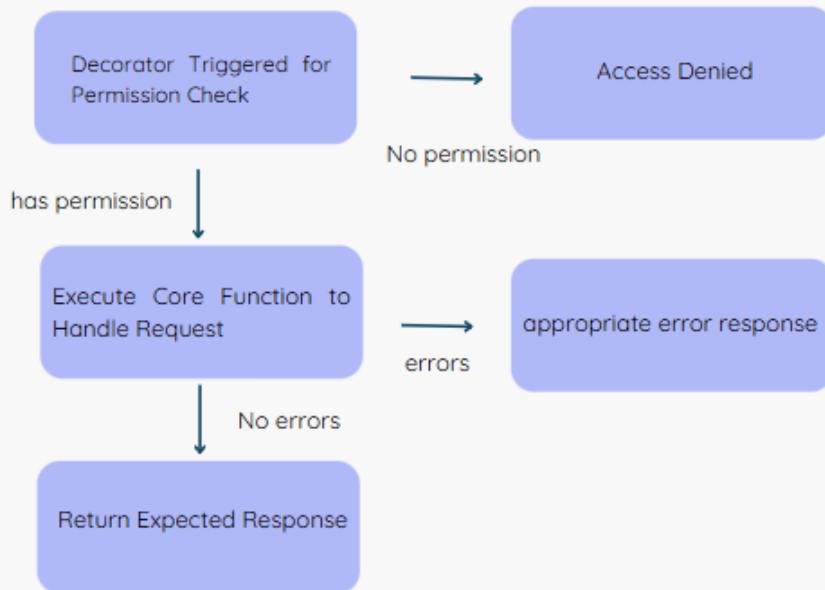
Apply Filters

Facility & Department Cross Tab For Fall/Injury



Enhanced Permissions

Implementation

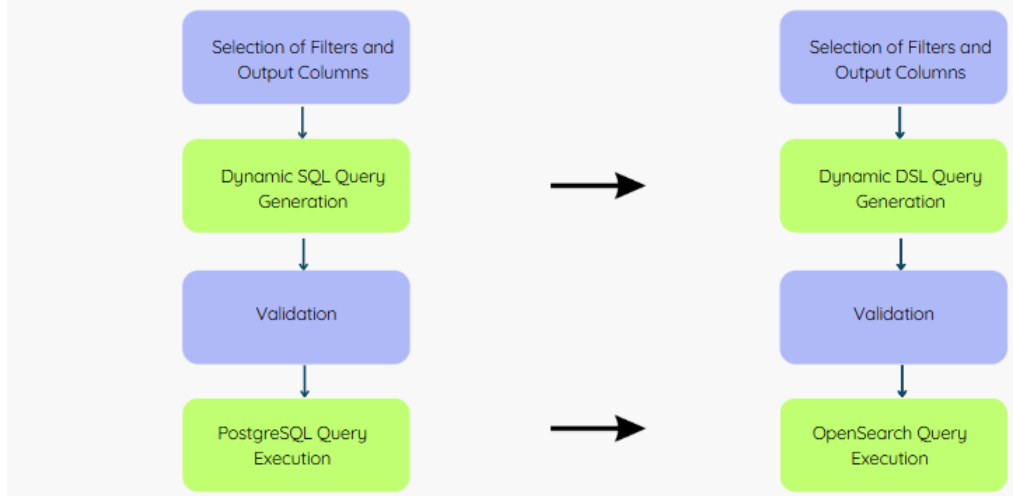


```
permission_internal_id
.....
anyone_can_view_all_records
anyone_can_create_records
anyone_can_edit_all_records
anyone_can_delete_all_records
ability_to_view_records
ability_to_create_records
ability_to_edit_records
ability_to_delete_records
ability_to_only_view_records_created_by_then
ability_to_only_edit_records_created_by_then
ability_to_only_delete_records_created_by_then
ability_to_only_view_records_created_by_specific_usergroup
ability_to_only_edit_records_created_by_specific_usergroup
ability_to_only_delete_records_created_by_specific_usergroup
ability_to_import_records
ability_to_export_records
ability_to_only_view_records_assigned_to_user
ability_to_only_view_records_assigned_to_usergroup
ability_to_only_edit_records_assigned_to_user
ability_to_only_edit_records_assigned_to_usergroup
ability_to_only_delete_records_assigned_to_user
ability_to_only_delete_records_assigned_to_usergroup
ability_to_view_records_based_on_rules
ability_to_delete_records_based_on_rules
ability_to_view_all_object_fields
ability_to_edit_all_object_fields
ability_to_create_all_object_fields
ability_to_create_object_fields
ability_to_view_object_fields
ability_to_edit_object_fields
ability_to_view_object_fields_by_associated_user
ability_to_edit_object_fields_by_associated_user
ability_to_view_object_fields_by_associated_usergroup
ability_to_edit_object_fields_by_associated_usergroup
ability_to_view_object_fields_by_condition
ability_to_edit_object_fields_by_condition
ability_to_view_all_process_states
ability_to_edit_all_process_states
ability_to_create_all_process_states
ability_to_view_specific_process_states
ability_to_edit_specific_process_states
ability_to_create_specific_process_states
ability_to_access_module
ability_to_access_adminview
ability_to_access_crashtab
ability_to_update_model_field_values
ability_to_update_process_state_values
ability_to_export_module_records
ability_to_export_admin_view_records
ability_to_hide_the_fields_for_user_usergroup_based_on_condition
ability_to_hide_the_sections_for_user_based_on_condition
ability_to_hide_the_sections_based_on_condition
fetch_users_based_on_object
fetch_users_based_on_object_field
(56 rows)
```

Migrating to OpenSearch



Implementation



The screenshot shows a web application interface with a REST client. The left pane displays a GET request to `_search` with a query object containing `"match_all": {}`. The right pane shows the JSON response, which includes metadata like `"took": 3`, `"total": 10`, and a list of hits. The first hit is a log entry with details like `"_index": ".ds-s4o_logs_waf-aws_waf-sample-sample-000001"`, `"_type": "waf_logs"`, and `"event": {"result": "ACCEPT", "name": "waf", "domain": "waf"}`.

```
1 GET _search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
```

```
1 {
2   "took": 3,
3   "timed_out": false,
4   "_shards": {
5     "total": 10,
6     "successful": 10,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 10000,
13      "relation": "gte"
14    },
15    "max_score": 1,
16    "hits": [
17      {
18        "_index": ".ds-s4o_logs_waf-aws_waf-sample-sample-000001",
19        "_id": "9V5-cIoB0a8jkDZ9N9N3",
20        "_score": 1,
21        "_source": {
22          "@timestamp": "2023-09-07T16:31:21.462Z",
23          "event": {
24            "result": "ACCEPT",
25            "name": "waf",
26            "domain": "waf"
27          },
28          "attributes": {
29            "data_stream": {
30              "dataset": "waf_log",
31              "namespace": "production",
32              "type": "waf_logs"
33            }
34          },
35          "cloud": {
36            "provider": "aws",
```

5

Future Work

To further improve system performance and scalability, the following future enhancements are planned:

5.1 Indian CMS-Style Hospital Ratings

Develop a hospital rating system tailored to Indian healthcare, based on key metrics like patient safety, readmission, and mortality rates. Helps patients make informed choices.

5.2 AI-Powered Health Predictions

Use AI to predict potential health risks and complications early. Show doctors a “risk score” and deploy chatbots to monitor patients at home and alert doctors if needed.

5.3 Rural Healthcare Connectivity

Build mobile apps for voice/video consultations between rural health workers and urban doctors. Reduces unnecessary travel and improves early diagnosis.

5.4 Smart Patient Feedback Sentiment Analysis

Collect patient feedback digitally and analyze it using NLP to detect tone, urgency, and satisfaction. Helps hospitals identify and resolve issues faster.

5.5 Outcome-Based Payment Models

Reward hospitals for successful patient outcomes rather than the number of procedures or tests. Encourages quality care over quantity.

5.6 Unified Digital Health Records

Create centralized, secure digital records accessible across hospitals and clinics. Enhances coordination and reduces redundant testing.

6

Conclusion

The transition to OpenSearch and the **replacement of legacy tools like Superset with custom-built solutions** have significantly improved system performance, scalability, and user experience. **Dynamic query generation and optimized data pipelines** now allow real-time filtering and analytics, reducing response times and enhancing overall efficiency. The redesigned Report Builder enables faster and more responsive visualizations, while robust backend optimizations have improved reliability and reduced load on the system.

The introduction of a flexible and **secure permissions model—covering record-level, field-level, and rule-based access—has strengthened data security and better aligned with client-specific access needs.** Together with streamlined data migration scripts and system refinements, these improvements have led to a more maintainable, high-performance environment. The result is a scalable and intelligent platform that supports faster feature development, reduces operational overhead, and is well-positioned for future growth and innovation in healthcare analytics.

7

References

- "Opensearch Documentation for Scalable storage": [*https://docs.opensearch.org/docs/latest/*](https://docs.opensearch.org/docs/latest/)
- "Superset for Analytics and DashBoarding": [*https://superset.apache.org/docs/intro*](https://superset.apache.org/docs/intro)
- "React For Front-end" :: [*https://react.dev/learn*](https://react.dev/learn)
- "Django Backend Docs": [*https://docs.djangoproject.com/en/5.2/*](https://docs.djangoproject.com/en/5.2/)
- "PSQL For Database": [*https://www.postgresql.org/docs/*](https://www.postgresql.org/docs/)