# Backend Development in Modern Care Platform Sprinklr

Student Name: Vin Patel
Enrollment ID: **201901288**
B. Tech. Project (BTP) Report
BTP Mode: ***Off Campus***
*Dhirubhai Ambani Institute of ICT (DA-IICT)*
Gandhinagar, India
201901288@daiict.ac.in

Mentor's Name: Kavish Manubolu
Company's Name: *Sprinklr*
Company's Address: 6th Floor, Building 2, DLF Downtown
DLF City Phase 3, Gurugram, Haryana 122010, India
manubolu.kavish@sprinklr.com
On-Campus Mentor: BTP Coordinator

***Abstract*—This report details my educational journey and the work I completed while working as a product engineer intern at Sprinklr. Not only do customers provide feedback on items in the modern social media environment, but businesses also keep an eye on their customers on different types of social media. Additionally, businesses develop marketing campaigns for their goods, keep tabs on market trends, and pay attention to corporate rivalry. All of the aforementioned objectives can be accomplished more easily using Sprinklr. Managing a sizable amount of data is a challenge for any business and organisation in the sector. The backend is in charge of managing, maintaining, and storing data. In this report, I'll discuss my job at Sprinklr, which entails becoming familiar with the company's many technological resources, lowering database latency across numerous production settings, and creating new features for the care bot product.**

***Index Terms*—Java, Spring Framework, MongoDB, Redis, Git, Kafka, Kubernetes.**

## I. Introduction

The client nowadays is the individual who determines the future of the businesses. For businesses, customer experience is important. Therefore, every business must address the concerns and complaints of its clients. Customer experience supervision is therefore crucial for large corporations. It is challenging to monitor consumer experiences and respond to their comments. However, any successful organisation must consider customer feedback and take appropriate action.As a solution to this issue, Sprinklr offers a platform for managing the customer experience. The company Sprinklr offers tools for engagement, sales, marketing, and research. With the use of care goods, businesses can keep an eye on their clients, pay attention to their concerns, and look after them. I am a member of Sprinklr's Care Team, which specialises in Modern Care. Customer service is crucial in today's world. Companies may manage their client contacts across several channels and deliver pro-active customer care with the help of the modern care solution.

## II. Learning Outcomes

We are given seven weeks to finish the Backend course at the beginning of our internship. During this learning period, we acquired knowledge of various tools and technologies from that course.

### *A. Java*

Java is the most globally used programming language for back-end software design in the IT sector. It is used to write the back-end code for the application. Because of its Object-Oriented features, system code is simpler to create and maintain. It's critical to understand the foundations of Java in order to write clean code. Some of the important Java guidelines that were helpful in the project are the ones listed below:

- **Generics** : Generics in Java is a feature that allows classes, interfaces, and methods to be parameterized with one or more types. It enables developers to create reusable code that can be used with different data types, without the need for casting or converting between different data types.
- **Collection Framework** : Collection Framework is a set of interfaces and classes that provide a unified way of managing and manipulating groups of objects. It is a powerful tool that allows developers to store, retrieve, manipulate, and organize collections of objects in a simple and efficient manner.
- **Concurrency** : Concurrency in Java refers to the ability of a program to perform multiple tasks simultaneously. In other words, concurrency allows multiple threads of execution to run concurrently within a single program.

In Java, threads are offered by default. To utilise a multi-core processor and to make the most of CPU cycles, multi-threading is necessary. Understanding the java thread development process makes managing threads simpler. Java Concurrency in Practise Book taught me about concurrent programming. I learnt about parallelization in this book and how can we can use multi-core CPU effectively utilising the CPU and thread cycle. I then used the Effective Java book to learn about Java design patterns and best practises. The industry's recommended practises for writing manageable, maintainable code are discussed in this book.

## B. MongoDB

MongoDB is a document-based NoSQL database. The industry uses it extensively to store data. We discovered some fundamentals and deeper ideas about MongoDB. A document-based NoSQL database is MongoDB. Data is kept in the form of binary JSON (BSON) objects. Every database in MongoDB has collections, and each of these collections has documents in it. A gathering of documents is referred to as a collection. Every document contains a few fields. Documents are the main type of data in MongoDB. The database's fields (key-value pairs) are used to build the document. Additionally, MongoDB has indexes that speed up queries. We gained knowledge of various index types, the inner workings of mongo indexing, and index creation principles. For internal usage, we have also developed documentation on MongoDB.

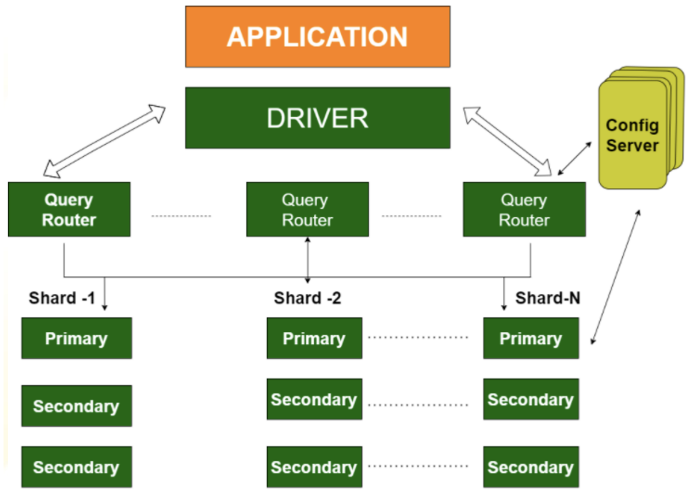

Fig. 1. MongoDB Architecture

In MongoDB, data is organized into collections, with each collection consisting of multiple documents. Each document contains several fields, and the schema of the document is flexible. The fields in a collection can hold different data types, and the documents in the same collection can have varying structures and fields. To optimize query performance, MongoDB uses indexes in various forms. MongoDB is known for its high performance, high availability, and scalability, making it a popular choice for many applications.

## C. Redis

Redis is a database and caching server that uses in-memory data structures. In-memory data storage reduce latency because a trip to the disc is not necessary. Data structures including text, hashmap, sorted set, array, set, geospatial, bitmaps etc. are available in Redis. It operates by using a preset data model to map keys to values. Redis values are mostly accessed through keys. Each key in this system is distinct, binary safe, and has a maximum size of 512 MB. We utilise any binary sequence as a key to store value because of the binary safe key. Key names automatically break down into designated categories like buckets and collections. We create a key-value pair and set a time-out for that key at the same time. It may be used as support for other databases to save load and boost performance. It is an in-memory database, but by making the database durable, it may also be used as a main database.

**Use Cases:**

- String : Store user sessions, shopping cart items, and other short-lived data.
- Hash : Store objects with many properties.
- List : Store timelines, user activity streams, and logs.
- Set : Store unique values for processing and filtering.
- Sorted sort : Store sorted and unique data that can be searched by score or rank.
- BITField : Implement bloom filters, where elements are checked for membership in a set with false positives.
- GeoHash : Implement location-based services, such as finding the nearest store, restaurant, or gas station.

## D. Elastic Search

Elasticsearch is an Apache Lucene-based, distributed, freely available search and analytics engine. Large volumes of data may be stored, easily searched for, and analysed in almost real-time with Elasticsearch. Because it searches an index rather than the text itself, it can produce fast search results. We studied fundamental ideas and elasticsearch query optimisation methods. For quick database queries, businesses utilise elasticsearch as a backup database. For displaying data in the database, Elasticsearch also supports a number of data visualisation tools, including kibana, logstash, and beats.

## E. graphqlh

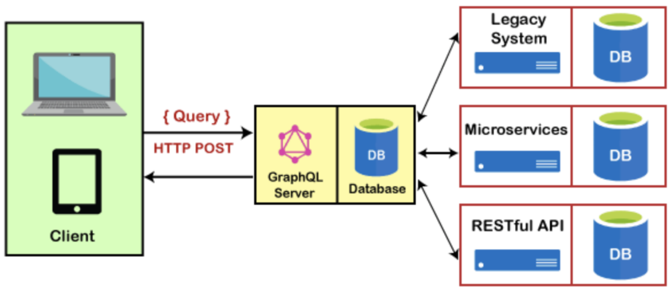

Fig. 2. GraphQL Architecture

When using GraphQL, the API server only sends the one thing that the client has specifically requested. In contrast to the REST API, which always sends the entire object from one endpoint, graphQL allows the client to specify the required fields in a query, and the server only sends the values of those fields. As a result, we can significantly minimise network IO by utilising graphQL. Using the graphQL persistent query cache, which caches the graphQL query, we can further reduce the network IO. GraphQL keeps track of its own schema, which is loaded when the graphQL servlet is created. This schema was utilised by GraphQL to resolve the query fields.

### *F. Kafka*

A distributed messaging system called Kafka is used to publish and subscribe to record(message) streams. Publishers may publish records using Kafka's Api, while subscribers can receive messages. Kafka is a distributed, scalable, and high throughput technology. Topic identifies a certain message type. Subscribers receive the record from the topic once the producer publishes or sends it to it. A broker is a location where published Messages are stored. A distributed system cluster can have several servers, numerous brokers, and one or more partitions thanks to Kafka. A logical log and each division of a subject are equivalent. A partition offset serves as the representation of a record. Messages from a single partition are always consumed sequentially by consumers.

### *G. Git and Gitlab*

A centralised version control system is called Git. Each branch is downloaded and then sent to the server. Git stores the files in a commit-based structure. Only the changes made in relation to the parent commit are stored in each commit. Git compresses the data in this manner. Because only one commit needs to be changed before two branches share the same parent, switching between branches happens instantly.

Source code management, continuous integration, and deployment, as well as other capabilities like bug tracking and wiki documentation, are all offered by the web-based Git repository manager GitLab. It is comparable to GitHub but offers the option of self-hosting, giving users greater control over their data and security.

## III. Work Involved

### *A. Database latency Reduction*

MongoDB is used by Sprinklr for storing data because it offers an effective indexing method for lowering query latency. My task is to develop the proper indexes for that specific database by analysing the various queries and reducing the overall database latency. I must first have a thorough understanding of mongo indexing in order to lessen this database latency.

The Graylog platform in Sprinklr displays all mongo queries executed over the previous five days. You can perform several operations like a filter, sort, groupBy, etc. on that platform. On that platform, you can also visualise data and display the top values according to a particular field in the query. I utilise this platform to identify queries that require optimisation.

The following are some metrics related to query performance in a MongoDB database:

- **DocExamined** : The number of documents to be examined in order to find a result for the query.
- **Frequency** : How often the query is executed in production.
- **Duration** : The amount of time it takes to execute a query.
- **nReturned** : The number of documents that are returned by the query.
- **KeyExamined** : The number of keys to be examined to filter out the necessary data.

Filtering sluggish Mongo queries that either utilise an inefficient index or no index at all is the first thing I do. In order to visualise them, I performed groupBy operations and certain filters to those searches based on statistics. I filtered the sluggish mongo queries and then looked at their statistics.

Given the large number of requests that are made every day in Production, it is not feasible to optimize every query. Instead, queries with higher frequency and longer duration, DocExamined or KeyExamined are optimized as they have a greater impact on server performance. Even if a query takes a longer time but has low frequency, it is not optimized since it does not have a significant impact on server performance. By focusing on one query at a time, latency for other inquiries that are being held up by collateral will be minimised in addition to that query's delay. We have examined the inquiries after filtering them. Since multiple queries may use the same fields but be executed in different orders, I took into consideration different criteria for index creation in the analysis section. To generate indexes, I used the Mongo index generation rules.

1) **ESR rule**: To optimize the performance of a MongoDB index, it is recommended to order the fields in the index by Equality, Sort, and Range.
2) **Prefix Rule**: Rather than creating two separate indexes that differ by only one field, it is more efficient to create a single index with the proper ordering such that the second index becomes a prefix of the new index.

### *B. Automatic Case Distribution Engine*

In sprinklr, I am working on Automatic Case Distribution Engine which manage the all kind of assignments, capacity, agent and routing configuration. Assignment Engine is a single screen from where you can configure and manage assignments of messages/cases. Based on capacity, availability, priority and past interactions, it routes right message to the right agent. It gives a single screen for configuring and managing assignments across the partner, allows you to configure backup agents to balance uneven workload in some groups and supervisors to see all assignment details and assignment logs in a single screen.

I contributed to some tasks. I have been assigned some mongo queries which I have to provide support of elastic search through dynamic properties. Elastic search gives faster searching compared to mongoDB. Some partners are requesting to optimize the query so it will give faster response. I have added a dynamic property which checks that elastic search is

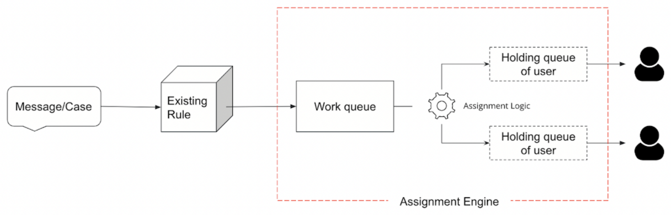

Fig. 3. Automatic Case Distribution Engine

enabled or not. If it is enabled for that partner then we will search through elastic search query otherwise it will search through mongo query.

Next task was to add a sanity job which checks for the assigneeType in current assignment and Universal case. Current Assignment stores the current status of a message or case then it will be populated to the Universal Case. This is a needed feature because sometimes it is taking time to populate the field and then consistency problems arised. So, if this happens, an alert will be sent as per task. So I have added code for it and tested it through different scenarios.

Next task was to add enhancement for the quality evaluation task. I have to distribute quality evaluation tasks equally(in round robin manner) to the quality manager(which is one kind of agent) in such a way that these tasks do not consume the capacity of the agent. I have changed the channel and capacity configuration for that and added a comparator for the agents so that they will be assigned tasks on the basis of activity time in a round robin manner.

## Acknowledgment

I appreciate Prof. Abhishek Jindal keeping an eye on our BTP project. Courses like CNS, OS, and DBMS were really beneficial throughout the internships.I would like to express my heartfelt gratitude to my manager and mentor for their unwavering support and guidance throughout my journey here.

First and foremost, I would like to thank my manager Kavish Manubolu (Vice President - Product Engineering) for providing me with the opportunity to work in this organization and for trusting me with important projects. Your leadership and mentorship have been invaluable to me and have helped me grow both professionally and personally. Your willingness to provide constructive feedback and support has allowed me to become a better employee, and I am grateful for your guidance. I would also like to express my gratitude to my mentor Navaneshwar Reddy (Lead - Product Engineering) for their constant support and encouragement. Your insights and advice have been instrumental in helping me navigate through challenges and improve my skills. Your willingness to share your knowledge and experience has inspired me to strive for excellence in my work.

I am truly grateful for the time, effort, and dedication that my manager and mentor have put into guiding me. Your support has been invaluable to me, and I will always be grateful for your mentorship. Thank you for being an important part of my professional journey.

## References

[1] Sierra, Kathy and Bates, Bert, “Head First Java, 2nd Edition,” PO’Reilly Media, Inc, 2005.
[2] https://spring.io/projects/spring-data
[3] https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
[4] https://redis.io/docs/
[5] https://www.mongodb.com/what-is-mongodb
[6] https://university.redis.com/courses/ru101/
[7] https://about.gitlab.com/stages-devops-lifecycle/