A qualitative analysis of the performance of MongoDB vs MySQL Database based on insertion and retriewal operations using a web/android application to explore Load Balancing – Sharding in MongoDB and its advantages

Mayur M Patil
Team 31, Store Colleague Mobility
Tesco Bengaluru
Bangalore, India
mayurpatil65@gmail.com
Mayur.Patil@in.tesco.com

Akkamahadevi Hanni#, CH Tejeshwar#,
Priyadarshini Patil*
#Student, *Ass. Professor
B V Bhoomaraddi College of Enginnering and Technology
Hubli, Karnataka, India
arhanni18@gmail.com
chtejeshwar2@gmail.com
priyadarshini.patil@bvb.edu

***Abstract*— Our world has evolved to an optimal point of advancement. The extravagant growth has helped in the invention of technologies, industry standards, gadgets, and devices that produce enormous amount of data that all require an essential data management and manipulation system. The data acquired from the various input and output sources are indulged in providing a certain infrastructure are also susceptible to damages if not treated well which may result in loss of data. To overcome this loss, various strategies that run parallel to prevent such loss are being used, one such example is the NoSQL MongoDB. MongoDb is a cross-platform, document oriented database that provides, high performance and easy scalability ensuring effective data management with its prominent feature of auto sharding. Sharding splits the database across multiple servers, increasing the capacity and scalability as required. This feature handles distribution of data in different nodes to maximize disk space and dynamically load balance queries. Partitioning the databases appropriately is a major step that determines the efficiency of sharding. This involves choosing an index of the MongoDB, competently as a shared key for further horizontal scaling of the database. Our current research involves the study of this load balancer. This paper intends to ascertain the need for NoSQL databases in the present situation and emphasize advancement of document-oriented database - MongoDB in particular by describing with a quantitative example that SQL databases are prone to deterioration when data is over loaded and MongoDB comes with inbuilt load balancer which makes it a better solution in applications with high data load. We describe the technology of sharding – auto load balancing feature of MongoDB and hope to provide a comprehensive insight of the process.**

Keywords*— *Relational databases, Non-relational databases, MySQL, NoSQL, MongoDB, Sharding

I. INTRODUCTION

Today data is being created, distributed, analyzed and are also used to make business decisions at a rate more than ever before. To give an idea of how much data is being generated, in 2013, IBM in their reports has said that more than 90% of the world's data had been created in the previous two years alone[1]. Conventionally, we have been using Relational Database Management Systems: RDBMS for managing the enormous amount of data being created by business applications and handling storage requirements of IT. They also provide efficient data manipulation of structured data. But in the current situation where the velocity and nature of data used/generated is increasing exponentially, RDBMS becomes incompetent in accomplishing management of such huge amount of unstructured data. This led to the invention of an easier and efficient way to handle big data, where the concept of NoSQL was introduced. NoSQL stands for "Not Only SQL" and provides more possibilities beyond the classic approach of data storage and retrieval. Application Development teams are also greatly influenced by this evolution. Since almost all modern applications require developers to create new, rapidly changing data types as they work in quick iterations as agile sprints, NoSQL provides a more flexible data model, higher scalability and also superior performance while still incorporating several key features of relational databases. Use of monolithic servers and storage infrastructure is ended as business organizations are turning to scale out their architecture by using commodity servers, open source software and cloud computing. NoSQL databases take all these dynamic and growing requirements of IT world into consideration and provide efficient solutions.

Non-relational databases come mainly in four different types:

I. Key-Value: The key-value model is the easiest and simplest model of non-relational database to implement. The concept here is to use a hash table and providing each item of data with a unique key and a pointer; data can only be queried using key. This model is useful in representing polymorphic and unstructured data but is inefficient when the application is only interested in querying or updating part of a value. Examples of key-value databases are Amazon simpleDB and OracleBDB.

II. Column-oriented: This database model was developed particularly to process huge amounts of data that is spread across multiple servers. They store data using a sparse, distributed, multidimensional sorted map. Variable number of columns can be stored in each records; there are still keys but they point to multiple columns. Column families can be formed by grouping multiple columns together which can be accessed as single column or columns can be spread across various column families. Examples of column-oriented databases are Cassandra and HBase.

III. Graph-stored: This database model is modeled as a network of relationships between specific elements of graph structures like nodes and edges to represent data. Unlike SQL which provides high level declarative query languages, in NoSQL querying is data model specific. The advantage of graph stored model is that the process of data modelling and directing relationships between entities is greatly simplified here. Example of graph-stored databases are Neo4j and Giraph.

IV. Document-oriented: Unlike relational databases where data is stored in rows and columns, document oriented databases store data in documents. These documents typically use a structure similar to JSON(JavaScript Object Notation); they provide a natural way to model data that is closely aligned with object oriented programming. Each document is considered as an object in object oriented programming, similarly each document is a JSON in document-oriented database. The concept of schema in document databases is dynamic: every document might contain different fields. This adaptability is useful in modeling unstructured and polymorphic data. Also, document databases allow robust queries, any combination of fields in the document can be combined for querying data. Examples of document-oriented databases are MongoDB and CouchDB.

II. Related Work

The advent of different databases in the industry has made way for pros and cons to be listed with respect to certain parameters and make a comparative study of the booming databases. Our research started by referring various papers about similar topics and briefing papers closely related to study of SQL and NoSQL(MongoDB in particular) databases and how MongoDB has an upper hand in efficiently handling increasing data loads. The first paper we considered was on analysis and comparison of MongoDB with MySQL database authored by Lokesh Kumar et al[2]. This paper mainly discusses the need for efficient handling of increasing unstructured data, replaces traditional MySQL databases with MongoDB to compare the two databases in order to justify the pertinence of MongoDB over MySQL. The paper walks us through the overview of MongoDB database that includes its architecture, query model, query types and the auto sharding feature. Kumar et al further compare the two databases on the basis of terminologies used in both databases, query format/syntax used in the two technologies and finally based on the query execution speed/performance of the two databases. The paper focuses on defining a method for integrating the two databases by adding a middleware or interface layer between the web applications and database layer.

Cornelia GYŐRÖDI, Robert GYŐRÖDI, George PECHERLE and Andrada OLAH[3] present their work on comparative study of MongoDB and MySQL which provides an insight about the performance of two databases under various workload. The authors emphasize how MongoDB provides flexibility and customization according to user needs which makes it a better option than MySQL where application flexibility is limited. The comparative study is based on a dynamically structured application forum and justifies how the use of MongoDB provides custom settings than the use of relational database. Further comparison includes comparison on the basis of terms/concepts used and also in terms of performance, after executing different operations. Four basic operations – Insert, Select, Delete and Update are performed on the two databases and the results are recorded, which depicts that MongoDB is faster in operating at higher data loads.

Another paper we studied is authored by Rajat Aghi et al[4] and it is a detailed comparison of SQL and MongoDB databases[4]. The paper discusses about the prevailing database models – relational and non-relational database models, followed by a comparative analysis between the two databases for various data sets. The analysis interprets the time taken by the two databases for insertions, joins and retrievals and showed how MySQL is efficient for simple queries with small datasets and MongoDB is more efficient for complex queries with large data sets.

The next topic of study was in relation to the efficiency of MongoDB in particular compared to other NoSQL databases. Veronika Abramova, Jorge Bernardino and Pedro Furtado presented a work on performance overview of various NoSQL databases.[5] Their work includes a comparative study of five popular NoSQL databases – Cassandra, HBase, MongoDB, OrientDB and Redis. Abramova et al use Yahoo! Cloud Serving Benchmark[6] to compare the execution time of those five databases over various workloads. The optimization mechanisms and data store approaches used by these databases are also considered in the performance evaluation. Their evaluation revealed that databases like Redis, where data is loaded into volatile memory, have faster response times regardless of workloads because access to volatile memory is faster than access to files stored in hard drive. But this database

depends on the amount of volatile memory, which is a very expensive storage. The overall analysis declared that Redis, MongoDB are databases optimized for read operations and Cassandra, HBase are databases optimized for update operations.

III. Proposed Work

SQL a crude database query language has been a prevalent query language to create, manage, store and process the data in the relational databases. The structured, tabular arrangement of the data with the effortless means to query made the use of SQL pertinent then. Now, the idea of technology and interest has changed. The data pool is expanding with time and the management of the same has become a major concern. Data is no longer limited to particular sites nor is a simple load to oversee with ease. Along with other evaluators, scalability and availability outlines a significant measure of performance. Whereas, SQL excels in certain parameters NOSQL has a larger impact on today's industrial requirement due to its distinctive features making it faster and reliable.

We made a detailed study about the two eminent databases and discovered the prominence of NOSQL databases when the work load is exceeding because of the presence of large data sets that need an efficient mode of management. The qualitative approach that we followed led us to various resources that provided us the benefits and limitations of the same. The major drawback in SQL databases that should be mentioned is the auto load balancing property called Sharding. Shard in its literal sense means "slice/piece", database sharding refers to partitioning the data bases into chunks and distributing it into different servers or shards. Sharding can be performed horizontally and vertically depending on the need. This property of NOSQL databases like Mongodb has made it more efficient for the current day data management necessity. One of the reasons for this is the load balanced on a single server; due to sharding, the work load is distributed amongst the different shards which predictably reduces the burden on one particular server. This ensures the work load to be sustainable for large data sets. Also, distributing data in different data sites reduces the amount of data to be stored in each shard thereby easing the processing load on the same. With the sharded data, the querying and processing becomes faster, thereby reflecting the result on performance. SQL databases takes a back seat when scalability is the topic of discussion. In earlier times, the data set didn't pose an issue of data management as it was small and manageable in a single server without the necessity of load balancing. To be suitable and effective for the modern day business technology SQL organizations should update their resources to costlier, complex and bigger ones which can possibly scale as per the need. This update has become an essential means to cope with the increase in the customer demand with time. But this can be rather hectic and expensive due to which customers are switching to NOSQL databases.

SQL databases store data in structurally organized inter-related tables according to the pre-defined schema. This mechanism turns out to become a bottleneck while storing and retrieving large dynamic datasets as per application requirements. Since business applications need more efficient and scalable solutions, MongoDB serves all of their requirements. Also, SQL database is transactional database, where transactions are atomic, consistent, isolated and durable. Although this feature of MySQL is greatly appreciated in mission-critical applications it limits horizontal scalability and is intolerant for data loss or inconsistency. MongoDB on the other hand works at single document level atomically. Multiple documents can be updated using single update operation and modification of each document is atomic but the operation as whole is not, since other operations can be interleaved along with it. We propose to readers that applications with MongoDB in the backend can be faster and more reliable than applications with MySQL backend through an experiment we have conducted. The main aim of the experiment is to prove that use of MongoDB is well suited for business applications with changing dynamic requirements.

IV. Implementation Details

The experiment consists of two datasets – MongoDB dataset and MySQL dataset - of registration and login pages of a web application. The work has been carried out after the entry of one million records into the tables because we are trying to figure out the performance after these many records already exist in the databases. The functionality offered by the registration page of application is to register any user with valid details to the website by storing user's information into the database.

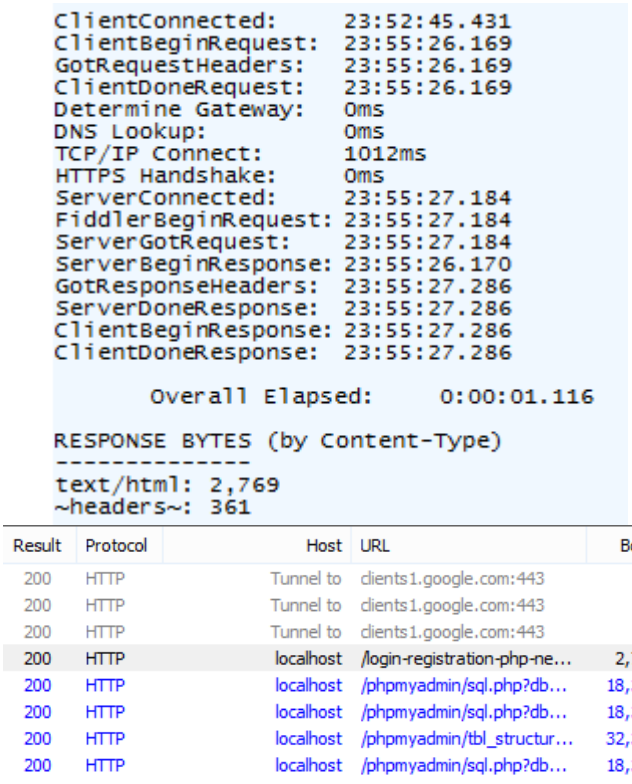

Fig. 1. Registration using MySQL

The login page validates any user trying to log in to the website by checking user credentials against the stored database of registered users. We have created two instances of

registration and login pages and connected one of them to MongoDB database and another to MySQL database. We inserted single user's details into both instances of registration page which stored identical data in MongoDB and MySQL databases. We then performed login operation on both instances of login page using the same user credentials. The performance of application is measured by the total time taken for the webpages to complete registration/login operations, and we have measured this time using Fiddler [7].

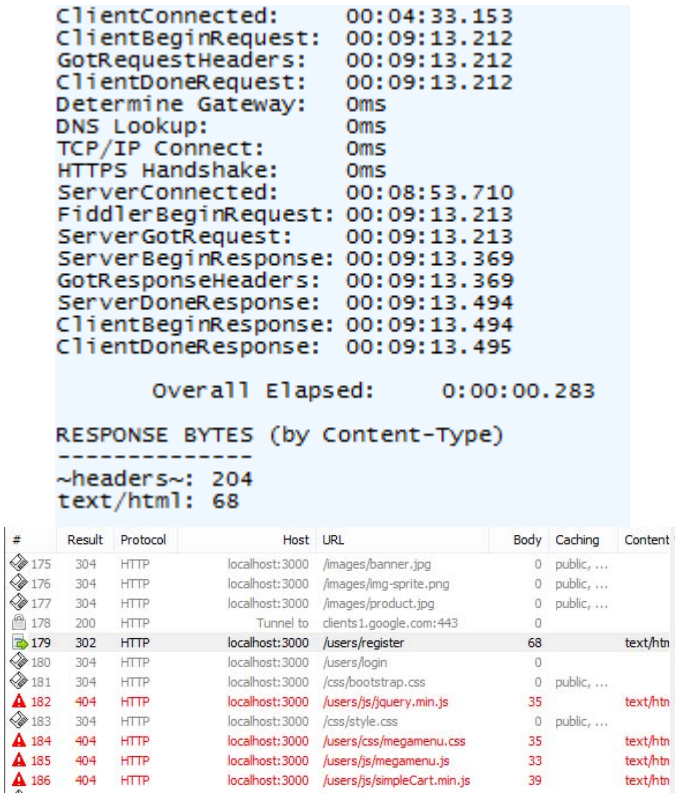

Fig. 2. Registration using MongoDB

Similarly we inserted 10,20,30, 40 and 50 records at a time into each of the databases and noted down the load time of respective databases. We have used a tool called Robomongo[8] to load multiple documents into MongoDB database and a tool called PhpMyAdmin[9] to load multiple rows into MySQL database. The load time of MongoDB is measured using Robomongo and of MySQL by PhpMyAdmin. We have generated a graph of load times of both MongoDB and MySQL against number of records to give better clarity about the behavior of the two databases. These checks were made on a Windows server 2008 R2 which is a 64 bit version with 4GB RAM having a disk space of 45GB running on a x64 processor with clock speed of minimum of 1.4GHz.

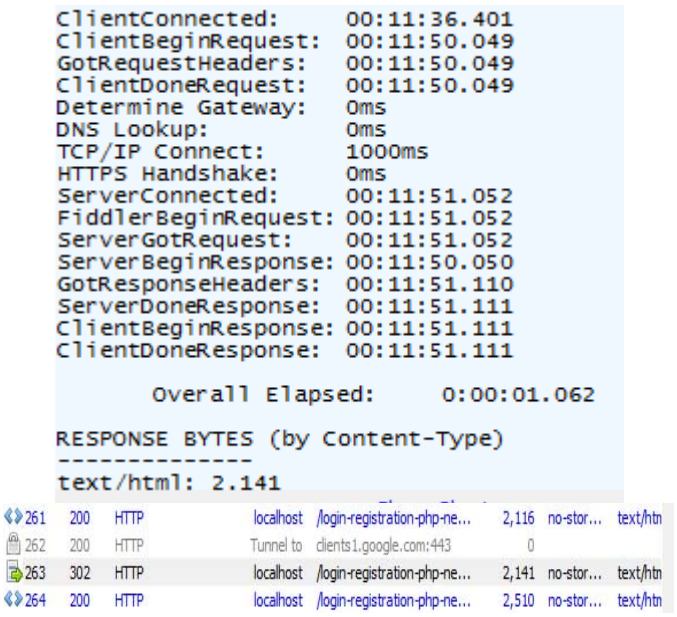

Fig. 3. Login using MySQL

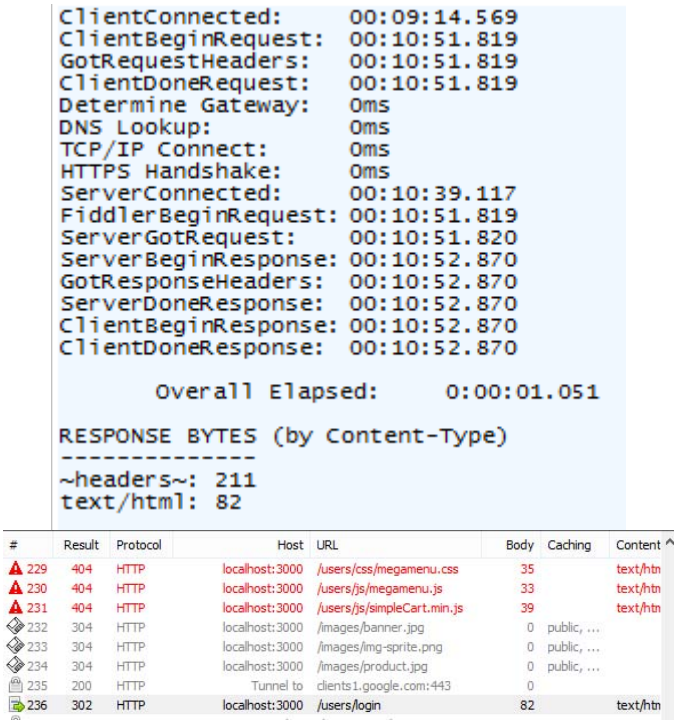

Fig. 4. Login using MongoDB

V. Results Discussion And Conclusion

We can see from Fig(1)which depicts the time taken by the system to load/ insert the data into a MySQL database and from Fig(2) which depicts the time taken by the system to load/ insert the data into a Mongo database, the load time or total time taken to insert a single user details into SQL database is greater than load time of MongoDB database. Also, the retrieval time or total time taken by MySQL database to validate user by fetching his details is more than total time taken by MongoDB to do the same and this can be seen in Fig(3) which shows the time taken by System with a MySQL Database and Fig(4) which shows the time taken by the System with a Mongo Database. With Fig(5) which depicts a graph plotted against number of records inserted and time taken by respective databases to load those records, we can clearly see the time taken to perfrom the basic operations of insertion and retriewal are substantially lesser in a system with MongoDB as the database which falls in line with our objective to prove that MongoDB is faster than MySQL in loading data dynamically.

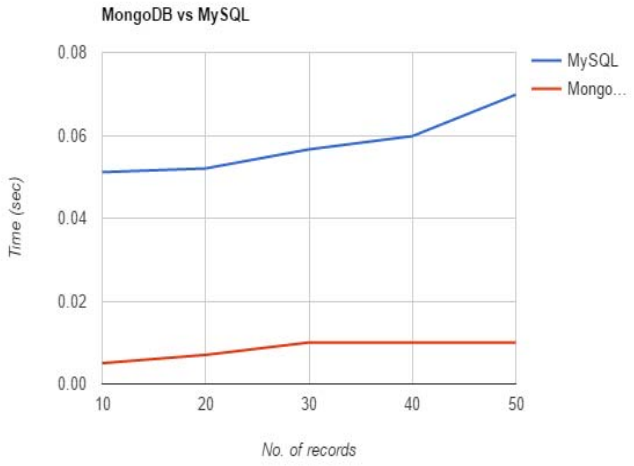

Fig. 5. Image of the graph plotted for the Results comparison

It is evident from the table of Results Table, Table(1) that the times taken in an application driven by MongoDB at the backend take lesser time to perfrom the same action that is being performed by using an application with MySQL running at the backend.

TABLE I. Results Table For Time Taken For Insertion and Retriewal In MySQL And MongoDB

Test #	No Of Records VS Time Taken		
	Number of Records	***MySQL (time in second)***	***MongoDB (time in second)***
1	10	0.0511	0.005
2	20	0.0520	0.007
3	30	0.0566	0.01
4	40	0.0598	0.01
5	50	0.0698	0.01

References

[1] A short history of databases: From RDBMS to NoSQL and beyond. [Online] Available: https://www.3pillarglobal.com/insights/short-history-databases-rdbms-nosql-beyond

[2] Lokesh Kumar, Computer Science & Engineering, Vivekananda Global University Jaipur, Rajasthan, India; Dr.Shalini Rajawat, Computer Science & Engineering, Vivekananda Global University Jaipur, Rajasthan, India; Krati Joshi, Computer Science & Engineering, Vivekananda Global University Jaipur, Rajasthan, India – "Comparative analysis of NoSQL(MongoDB) with MySQL database"- IJMTER, ISSN-2393-8161. http://www.ijmter.com/papers/volume-2/issue-5/comparative-analysis-of-nosql-mongodb-with-mysql-database.pdf

[3] Cornelia GYŐRÖDI, Robert GYŐRÖDI, George PECHERLE, Andrada OLAH, "A comparative study: MongoDB vs MySQL", ResearchGate Conference Paper • June 2015, DOI: 10.13140/RG.2.1.1226.7685

[4] Rajat Aghi, Sumeet Mehta, Rahul Chauhan, Siddhant Chaudhary and Navdeep Bohra Department of Computer Science, Maharaja Surajmal Institute of Technology, Janakpuri, N.delhi 110058, India – "A comprehensive comparison of SQL and MongoDB databases", International Journal of Scientific and Research Publications, Volume 5, Issue 2, February 2015 1 ISSN 2250-3153 http://www.ijsrp.org/research-paper-0215/ijsrp-p3841.pdf

[5] Veronika Abramova, Polytechnic Institute of Coimbra - ISEC / CISUC, Coimbra, Portugal, Jorge Bernardino, Polytechnic Institute of Coimbra - ISEC / CISUC, Coimbra, Portugal, Pedro Furtado, University of Coimbra – DEI / CISUC, Coimbra, Portugal – "Which NoSQL database? A performance overview", Open Journal of Databases(OJDB) Volume 1, Issue 2, 2014. ISSN 2199-3459 www.ronpub.com/journals/ojbd

[6] United Software Associates – "High Performance Benchmarking: MongoDB and NoSQL systems" [Online] Avialable: http://info-mongodb-com.s3.amazonaws.com/High%2BPerformance%2BBenchmark%2BWhite%2BPaper_final.pdf

[7] Fiddler – The free web debugging proxy for any browser, system or platform. [Online] Avialable: http://www.telerik.com/fiddler

[8] Robomongo – Native and cross platform MongoDB manager. [Online] Avialable: https://robomongo.org/

[9] phpMyAdmin – A tool to bring MySQL to web. [Online] Avialable: https://www.phpmyadmin.net/

[10] "MongoDB set to become the new 'default database'- eWEEK" [Online] Avialable: http://www.eweek.com/database/mongodb-set-to-become-the-new-default-database.html

[11] Xiaolin Wang, Sch. of Software, Shanghai Jiao Tong Univ., Shanghai, China, Haopeng Chen, Sch. of Software, Shanghai Jiao Tong Univ., Shanghai, China, Zhenhua Wang, Sch. of Software, Shanghai Jiao Tong Univ., Shanghai, China - "Research paper on Dynamic load balancing in MongoDB". Published in: Dependable, Autonomic and Secure

Computing(DASC), 2013 IEEE 11th conference, INSPEC Accession Number-14400155, DOI: 10.1109/DASC.2013.49

[12] Zachary Parker, The University of Alabama, Tuscaloosa, AL, Scott Poe, The University of Alabama, Tuscaloosa, AL, Susan V Vrbsky, The University of Alabama, Tuscaloosa, AL – "Comparing NoSQL MongoDB to an SQL DB". Published in: Proceeding ACME'13, Proceedings of the 51st ACM Southeast Conference, Article No. 5, ISBN: 978-1-4503-1901-0

[13] Michael Stonebraker, Massachussets Institute of Technology – "SQL databases vs NoSQL databases". Published in: Magazine, Communications of the ACM, Volume 53, Issue 4, April 2010, ACM New York, USA. doi>10.1145/1721654.1721659

[14] A. R. Hanni, M. M. Patil and P. M. Patil, "Summarization of customer reviews for a product on a website using natural language processing," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 2280-2285. doi: 10.1109/ICACCI.2016.7732392 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7732392&isnumber=7732013

[15] The Little MongoDB Book – Karl Seguin . [Online] Avialable: http://openmymind.net/mongodb.pdf