Nigerian Journal of Technology (NIJOTECH)
Vol. 40, No. 4 July, 2021, pp.713 –727
www.nijotech.com

Print ISSN: 0331-8443
Electronic ISSN: 2467-8821
http://dx.doi.org/10.4314/njt.v40i4.18

Database Management System for Mobile Crowdsourcing Applications

F.M. Dahunsi[1*], A. J. Joseph[2], O. A. Sarumi[3], O. O. Obe[4]

[1,2]*Department of Computer Engineering, Federal University of Technology Akure, Ondo State, NIGERIA.*
[3,4]*Department of Computer Science, Federal University of Technology Akure, Ondo State, NIGERIA.*

Abstract

The evaluation of mobile crowdsourcing activities and reports require a viable and large volume of data. These data are gathered in real-time and from a large number of paid or unpaid volunteers over a period. A high volume of quality data from smartphones or mobile devices is pivotal to the accuracy and validity of the results. Therefore, there is a need for a robust and scalable database structure that can effectively manage and store the large volumes of data collected from various volunteers without compromising the integrity of the data. An in-depth review of various database designs to select the most suitable that will meet the needs of a real-time, robust and large volunteer data handling system is presented. A non-relational database was proposed for the mobile-end database: Google Cloud Firestore specifically due to its support for mobile client implementation, this choice also makes the integration of data from the mobile end-users to the cloud-hosted database relatively easier with all proposed services being part of the Google Cloud Platform; although it is not as popular as some other database services. Separate comparative reviews of the Database Management System (DBMS) performance demonstrated that MongoDB (a non-relational database) performed better when reading large datasets and performing full-text queries, while MySQL (relational) and Cassandra (non-relational) performed much better for data insertion. Google BigQuery was proposed as an appropriate data warehouse solution. It will provide continuity and direct integration with Cloud Firestore and its Application Programming Interface (API) for data migration from Cloud Firestore to BigQuery, and the local server. Also Google BigQuery provides machine learning support for data analytics.

.

Keywords: Database management system, mobile applications, crowdsourcing, big data

1.0 INTRODUCTION

A database is a collection of data stored electronically such that information relevant to an enterprise, organization or entity can be retrieved conveniently and efficiently [1]. Although the terms "database" and "database management system (DBMS)" are used interchangeably the former correctly refers to the collection of data which comprises all data significant to an organization while the latter refers to the entire system which comprises of the collection of stored data (database) and a set of programs that handle operations through transactions on the stored data [2]. Appropriate database architecture is critical in ensuring reliable data, minimizing data duplication, effective query execution, high-performance usage and integration. The database architecture becomes more relevant at the point of analysis for the extraction of meaningful information. An important issue in the management of large amounts of data handled by a DBMS is data inconsistency. This occurs when different versions of the same data exist at different places and are usually made worse by data redundancy which occurs when there are multiple entries of the same data in different places [3]. Modern DBMSs sufficiently handle data consistency, redundancy issues, and several others by ensuring that every transaction has the following properties [2], [4]:

1. **Atomicity:** This requires that all concurrent operations or co-dependent operations must be successfully executed, in the case where an error or failure occurs during the execution of any of these operations the transaction should be terminated and all changes reversed.

2. **Consistency:** This ensures that changes made through transactions occur only in predefined ways by always following certain rules and constraints. An example of a constraint is the data type for a column in a relational database which dictates the type of values that can be stored in that specific column any attempt to store data of the wrong type results in a failed transaction.

3. **Isolation:** This defines the rule for cascading transactions and it restricts the use of data to1 only one transaction at a time and that data can only be

*Corresponding author **(Tel:** +234 (0) 8101564160)
Email addresses: fmdahunsi@futa.edu.ng (F. M. Dahunsi), josephayo110@gmail.com (A. J. Joseph), oasarumi@futa.edu.ng (O.A. Sarumi), ooobe@futa.edu.ng (O.O. Obe)

used by another once the current transaction is done. This property determines how changes made by one or more users are committed in the database and become visible to other users, it usually involves a trade-off between perfectly isolated transactions and concurrent transactions through the use of strategies such as serializability which executes concurrent transactions serially depending on the business logic [5].

4. **Durability:** This enforces the changes made by a successful transaction and ensures that those changes cannot be reversed or lost except by the execution of another transaction.

5. **Serializability:** This property ensures that concurrent transactions are properly scheduled serially to ensure consistency in results.

DBMSs have become more important to the efficiency of several tasks ranging from day-to-day activities of large and small organizations to research in various fields. It is especially relevant in the past few decades where the amount of data collected and stored has risen in exponential magnitudes according to several indicators [6]. This high growth rate of created data has highlighted how data can be underutilized due to poor management and the limitations of the present technology. Some of the features considered at various levels in ensuring that a suitable database was selected are normalization, data structure, and prioritized operations.

Database normalization involves the structuring of data into a specified normal form to reduce avoidable duplication of data and improve the integrity of database operations. This gives an advantage of efficiency for relational databases which hold structured data but this is not the case for non-relational databases.

In addition, determining if the data will be in a structured, semi-structured, or unstructured format depends mainly on if the DBMS is relational or non-relational and the intended application of the data. In addition to database normalization and data structure, prioritized operations were examined to determine which operations have precedence. For instance, at the point of data collection, data write operations have priority while at the most crucial stage (retrieving data for analysis) data read and update operations take priority. Hence priority varies, though retrieving data is of greater importance in this application because the accuracy of results is hinged on the retrieved data.

2.0 OVERVIEW OF EXISTING APPROACH

Mobile crowdsourcing applications are software applications developed to leverage the widespread use of mobile devices especially smartphones which have an estimated 3.6 billion users worldwide [7] and 25-30 million users in Nigeria [8] and interact with specific user demographics. Also to gather data that reflects actual user experience and opinions provided that the crowdsourcing project is properly planned and challenges such as crowd engagement with the applications, data and user privacy, data verification and crowdsourcing architecture which are highlighted in [9]–[12] are considered. Some examples of crowdsourcing projects which integrated mobile applications are Waze [13], Open street map [14], OpenSignal [15], Disaster management, and emergency routing with google maps [16].

This research focuses on the comparative analysis of various database management systems (DBMS) and selecting an appropriate one to be used for a mobile crowdsourcing application. The particular mobile crowdsourcing application system considered is one developed to evaluate the performance of mobile communication services through the measurement of certain voice and broadband key performance indicators (KPIs) on volunteer's mobile devices. The KPIs were selected based on requirements from the Nigerian Communication Commission (NCC) for quality-of-service evaluations. For database design, a brief and expected data type for each KPI is presented in this section, though the data type can be changed if a different approach is taken for the computation of the metrics [17].

2.1 Voice Service KPIs

The Voice Service KPIs considered are the Call Setup Success Rate (CSSR), Radio Signal Quality and Strength, Handover Success Rate (HSR), Bit Error Rate (BER), and Traffic Channel Congestion (TCH-CONG).

- **Call Setup Success Rate (CSSR):** The call setup success rate is the percentage of successfully linked calls. The result from a single attempted call is stored as a Boolean value to indicate success or failure.
- **Radio Signal Quality and Strength:** This metric is a measure of the strength and the quality of signal received by a mobile device antenna. It is computed as integer values measured in dBm.
- **Handover Success Rate (HSR):** This is a percentage of successful switches between cell towers which is usually attempted when a mobile device moves to the distance where the signal quality it is currently receiving is weak. The result from a handover attempt is stored as a Boolean value to indicate success or failure.
- **Bit Error Rate (BER):** This is a measurement of the end-to-end bit error that occurs in the data transmitted during a voice call. The result from a single voice call is stored as a floating-point value.
- **Traffic Channel Congestion (TCH-CONG):** This indicates the level of unavailability of resources needed for voice services which results in blocked calls. A single test gives a Boolean value result that indicates whether or not there is congestion.

2.2 Broadband Service KPIs

The broadband Service KPIs considered are Download Speed, Upload Speed, Domain Name Service (DNS) Lookup, Network availability, and Video streaming experience.

- **Download Speed:** This is a measure in megabits per second of the amount of data a user receives from a server. The measured values are stored as floating-point values.
- **Upload Speed:** This is a measure in megabits per second of the amount of data a user sends to a server. The measured values are stored as floating-point values.
- **Domain Name Service (DNS) Lookup:** This metric is the time taken in milliseconds to successfully send a query for the internet protocol (IP) address of a domain name and get a response. Results are stored as integer values.
- **Network availability:** The percentage measure of how often broadband service can be accessed in a given period is the network availability. The measured values are stored as floating-point values.
- **Video streaming experience:** This represents the perceived experience a user gets when using a video streaming service. The measured values are stored as floating-point values.

This study consulted official documentation of standard DBMS; their features, specifications, and previous works where similar systems were implemented. Some important requirements of a mobile crowdsourcing application database are:

a. a flexible schema design that makes it much easier to update the database to handle changing application requirements.
b. an ability to seamlessly and effectively scale the database and local server as data grows.
c. To be readily available and easy to use.

The quality of analysis gleaned from gathered data is highly dependent on the database implementation. Several researchers have used different database architectures in the implementation of mobile crowdsourcing applications. In [18], mobile broadband performance measurement was carried out using MySQL database design for about one hundred Mobile Network Operators (MNOs) subscribers in two cities in Nigeria. This coverage is less than the projected number of users and geographical coverage for this current research.

SQLite database technology was used in [19] is also SQL-based and implements a structured database, but it still has constraints similar to [18], which are the limited amount number of users and geographical coverage. Aside from these research papers mentioning the DBMS used, no further insight was offered as to why these specific systems were used. A comparison of a relational (MySQL) and a non-relational (MongoDB) database was done in [20], the approach of the comparative study carried out here was to first explain the difference in how data was organized with the former implementing a graphical table format while the latter uses a document-collection based structure, this review then moved on to benchmarking four major operations; insert, select, update and delete for large amounts of data and results showed that MongoDB provided better execution times for all four operations and it concludes that although a non-relational database performed better for large amount of data the choice ultimately depends on the particular application that is to be integrated with the database. A more particular comparative study of document-based DBMSs was carried out in [21], this focused on the certain features of the compared DBMS with an eventual conclusion that each DBMS classification addresses specific requirements that suit different application implementation.

Section three of this study explains the types of databases based on usage and application. Section four presents the evaluation of DBMS using four distinct parameters of the system: mobile application, cloud application, web application, and the local server. Section five discusses the results and presents the comparative analysis. Section six presents the critical analysis gleaned from the review and proposed research areas that could be investigated in subsequent works.

3.0 AN OVERVIEW OF DATABASE SYSTEMS

It is important to note that the term “database” covers more than just data but is used to generally refer to the data, database management system (DBMS), and all associated applications [22]. An operational database is used to manage and store data in real-time. It is the source of information for the data warehouse and it is set up to work efficiently with a high volume of transactional processing. It deals mainly with operational information which is the type of information required for day-to-day routine activities [23].

A data warehouse system is used for services that involve data analysis and decision-making [23]. Data warehouses deal with strategic information which must have a uniform and consistent view, conveniently available, accessible, and correct. These systems are optimized mainly for read operations; medium access frequency larger amount of accessed data, and much more complex queries compared to operational databases. Although, it may frequently interact with the operational database for data. In a basic sense, the create, read, update and delete operations are the main activities of an operational database and it includes both relational and non-relational databases.

3.1 Comparative Analysis of Database Types

A relational database uses a structure that helps the user to define and access data in the database concerning some other piece of data. It is a collection of tables representing both data and data relationships [22].

The physical arrangement of each table (known as relations) is similar to that of a spreadsheet, having multiple columns which are labelled with unique names and records (rows) of various types. An instance of a record specifies a set number of fields or attributes and table columns refer to the attributes of the record type, constraints, and data types. The outcome of a relational database is structured data that fits perfectly into defined fields and columns and it is managed using a Structured Query Language (SQL) which has several flavours like SQLite, MySQL.

A non-relational database is a database that does not follow the rows and column tabular structure used in most conventional database systems. Alternatively, non-relational databases utilize a retrieval model that is designed to satisfy the particular needs of the data format being processed [24]. The outcome of this is unstructured and semi-structured data. Unstructured data has loose formatting, with limited structure, it consists mainly of qualitative data (text, audio, video, satellite imagery, etc.) [1]. Semi-structured data is data comprising semantic tags and elements (known as metadata) but not perfectly compatible with the framework associated with traditional relational databases [1]. This data model has some form of structure because similar entities are grouped and organized in a hierarchical format and can consist of both quantitative and qualitative data. Both unstructured and semi-structured data are managed using NoSQL technologies like MongoDB, and Firebase.

A distributed database is a collection of different databases, but at different geographical locations (sites) connected over a network and is managed as a single database [1]. It provides an advantage of improved performances and system reliability. A distributed database can be implemented by data fragmentation, allocation, and replication [1], [25]. Fragmentation involves breaking up data into chunks and storing them in the constituent systems of the distributed database, allocation is the operation by which fragments are placed in the available distributed infrastructures, and replication involves the duplication of data in the various available systems that make up the distributed database. This duplication serves as backups and fallbacks in the event of the failure of any constituent system but comes at a shortfall of cost and data redundancy. Most cloud database services use this model to ensure that data is secure and always available to users quickly and consistently.

A comparison between relational databases (structured data) and non-relational databases (semi-structured and unstructured data) is presented in Table 1 from the standpoint of data structure which is the fundamentals of all databases. The relational database options for data with a predefined schema that is not easily changed are available across various platforms i.e. MySQL. These are quite straightforward to use alongside Object-relational mapping (ORM) libraries which are available in various programming languages but do not easily scale up when data grows quickly [26]. Although a relational database is better suited for quantitative data which favours statistical analysis compared to a non-relational database.

A non-relational database permits a structure that does not necessarily have a rigid schema but can easily accommodate changes to how data is organized or how relationships exist. It is also readily available with variations i.e. MongoDB. It can be integrated using well-versed Object Relational Mapping (ORM) libraries in various programming languages. Non-relational database scales better with data that grows quickly and can hold both unstructured and semi-structured data. The former holds mainly qualitative data (useful for categorization) and the latter includes both qualitative and quantitative data this property makes it suitable for complex data collection and analysis.

It is important to note that gathering and storing data for analysis goes beyond an eventual purpose of displaying data in charts but more importantly extracting useful information from the gathered data [29], the structure of the data inevitably affects this. The measurement of parameters for mobile communication quality of service can be implemented using several algorithms, this implies that during the development of the mobile application different algorithms may be implemented with a consequence of modifying the database schema to accommodate the new attributes of the data like the data type or association constraints [29]. Hence there's a need for a system that can accommodate such changes. Even though a relational database is best suited for quantitative data and a non-relational database for qualitative data both database types can still handle either form of data well. An example is the PostgreSQL relational database having a JSON datatype which can hold qualitative [30]. As mobile telecommunications technology evolves with the introduction of newer services such as 5G and Voice over LTE (VoLTE) recently, the infrastructure for this research must suitably accommodate changes and accommodate continued research. Hence, a mutable database structure will be of an advantage here.

Additional, comparative insights and rankings were retrieved from DB-Engines ranking which calculates ranking scores based on the number of mentions of the system online, frequency of technical discussions, general interests, etc [31] and not the actual performance of the DBMS. Figure 1 shows the ranking scores of some select database systems. It shows that relational databases like Oracle and MySQL are much more popular than their non-relational counterparts like MongoDB, this can be attributed to the fact that they have been around longer than non-relational DBMSs and hence have been and are still used more frequently.

Table 1: Comparison between Relational Databases (structured data) and Non-relational Databases (semi-structured and unstructured data).

Characteristics	Relational (Structured)	Non-relational (Semi-structured and unstructured)
Type of data	Predefined and rigid schema	Dynamic schema configuration
Size of data	Scaling is vertical and expensive because when relational databases become large they have to be scaled to a more powerful server which comes at a higher cost [26].	Scaling is horizontal and cheaper since non-relational databases can be scaled by distributing a single database over multiple servers that are not required to be more powerful than the ones already in use [27]. An example can be seen from the pricing of digital ocean droplets, horizontally scaling two basic droplets cost $10/month each and will provide the same specifications as a single $20/month basic droplet [28] but with an extra 20GB of SSD disk storage.
Applicability	Well suited for quantitative data	Well suited for qualitative data
Availability	Readily available across various platforms (MySQL, SQL Server, etc.)	Readily available across various platforms (MongoDB, Firebase, etc.)
Ease of use	Modifiable, supported, friendly user interface (MySQL Workbench) and availability of object-relational mapping (ORM) libraries	Modifiable and supported friendly user interfaces (MongoDB Atlas, Firebase Console), and availability of object-relational mapping (ORM) libraries

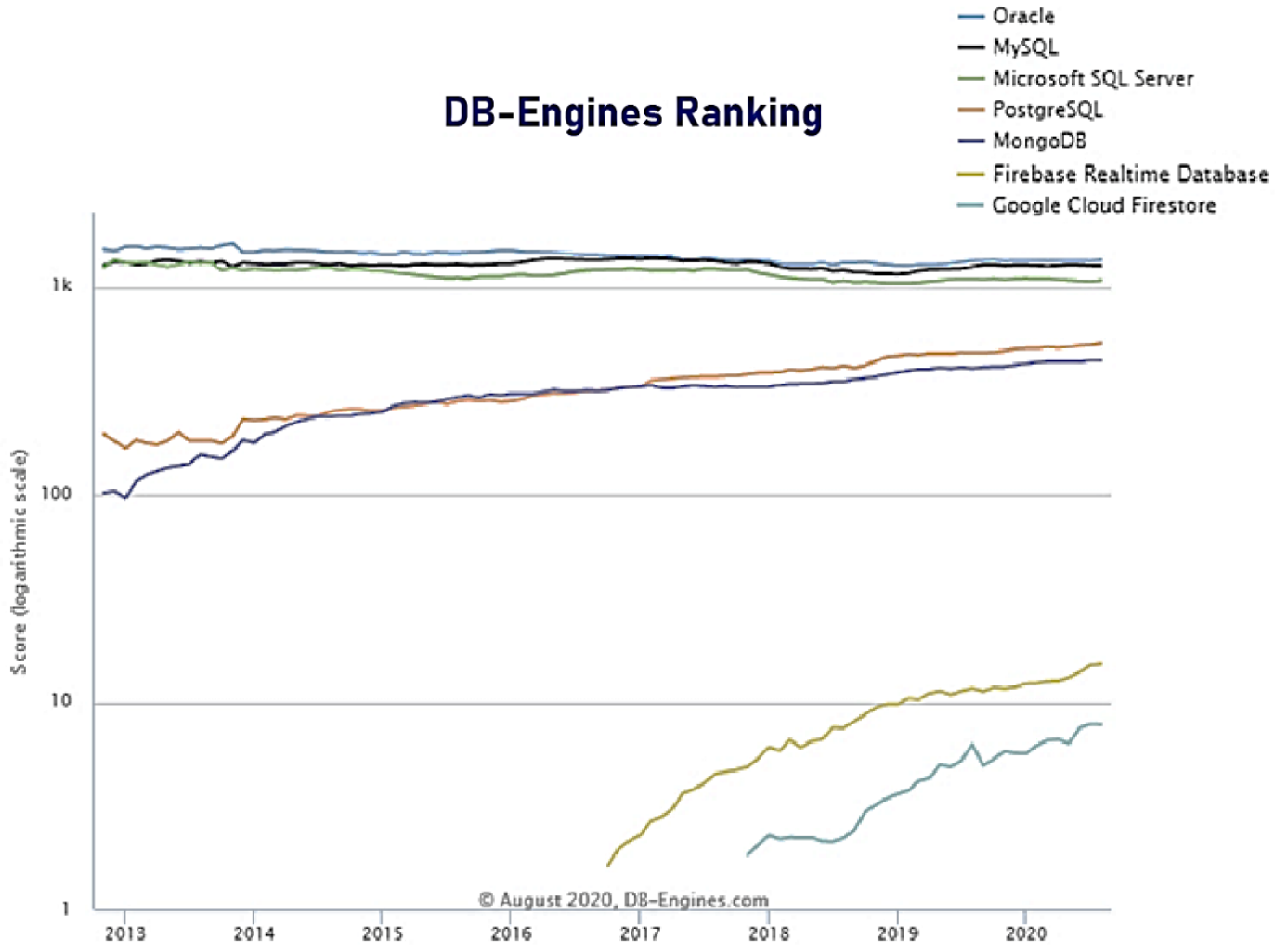

Figure 1: DB-Engines Database Ranking [31]

4.0 DATABASE DESIGN CONSIDERATIONS FOR MOBILE CROWDSOURCING APPLICATION

The system architecture for the mobile crowdsourcing application system considered for this research is a crowdsourcing system to evaluate the performance of mobile communication services as shown in Figure 2. KPIs are measured from volunteer's mobile phones and saved on the mobile application database. The measurements are collated on the phone and sent via the internet to the cloud server database. Thereafter through an API layer, the data and its analysis can be accessed by the web application for presentation and visualization. There are four databases considered for this study, the mobile application database, cloud server database, web application database, and the local server database.

a. Mobile application database: This is the primary database in the database architecture for this system. It is the entry point for data gathered from measurements on the mobile devices of users. Although this database holds measurement data it also doubles as a database for volunteers' information.
b. Cloud server database: This has its functions interleaved with the mobile application database and both can be implemented as a single DBMS depending on the cloud database service being used. For example, Firebase works perfectly for the features required by a mobile application database and also a cloud server database but MongoDB and some others are optimized for a cloud server database rather than a mobile application database except when integrated with complementary mobile application database service [32]. The basic differential as seen in Figure 2 is that data collected from mobile devices is transferred to cloud storage via an internet connection.
c. Web application database: This database serves the purpose of providing a web application with data results extracted from aggregated computations on the data measurements gathered from mobile devices of volunteers. Besides providing data results it also performs the role of an operational database that executes transactional operations that are required by the features of the web applications such as management of authorized and unauthorized users. A web application database is a cloud database that provides database service to a web application [33].
d. Local server database: This is the final layer of the database architecture for this system. This database is the final destination for data gathered from volunteers' mobile devices. Although the cloud database safely holds data and makes it readily available without the restriction of location. Some problems posed are increased cost, large data size, and increased data retrieval time, these issues are solved by having data available on a local machine. The local server pulls data intermittently from the cloud database and stores it in a local machine (see Figure 2) from which it can be accessed any time for analysis without incurring the additional costs.

These databases are reviewed more explicitly in the next sub-sections.

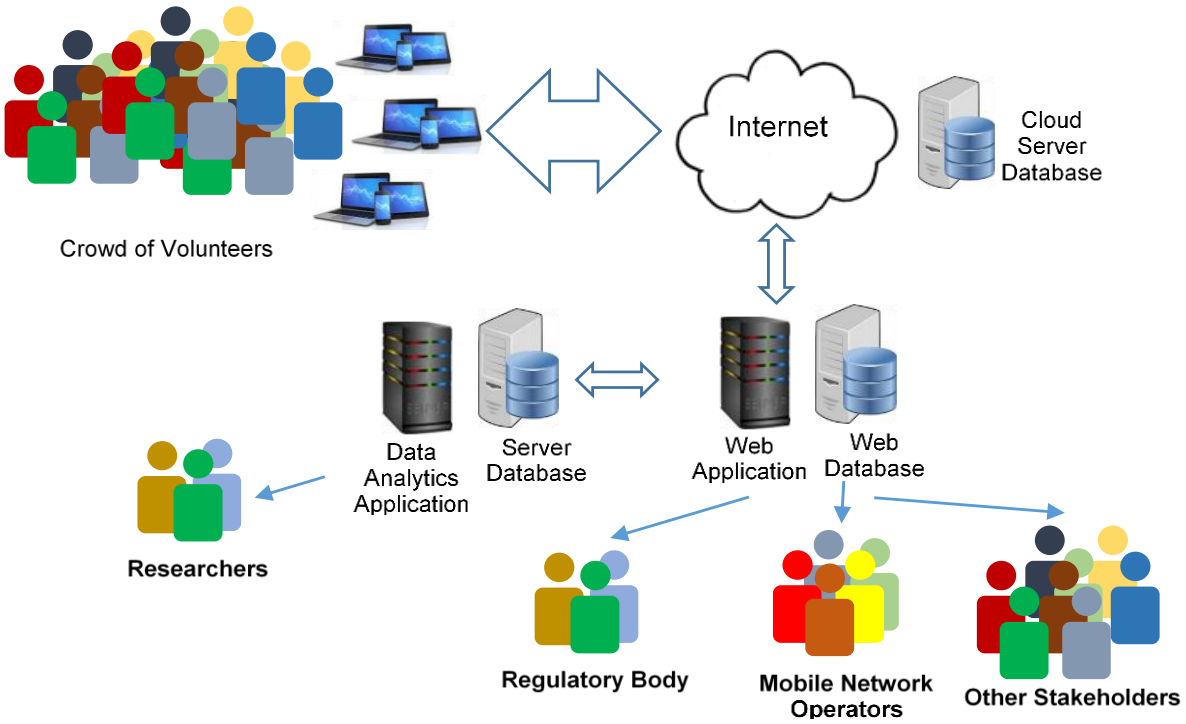

Figure 2: System architecture for the crowdsourced mobile communication quality of service analysis system

4.1 Mobile Application DBMS

Databases are essential to most smartphone applications and selecting the correct one is key to the performance of the application. User details and interactions at specific points during the use of the application are required to be stored for various uses such as displaying the same information at a later time or using the information to ensure consistent and correct updates and read on a more diverse database. Features of databases for mobile applications are:

i. Support for client platform: The level of support of a database service on a mobile client platform is critical to the overall performance of the application. Without basic support for a database service on a mobile client, it will be difficult to integrate such a database with a mobile client application.
ii. Flexible data model: It is common for changes to be made to a mobile application, and in some instances, these changes will involve the data model being used. Hence, the data model must be modifiable without causing crashes or breaks.
iii. Ability to resolve data conflict: A database that will be integrated with any mobile client should adequately compensate for occurrences of data conflict which occurs frequently considering the conditions where user internet connectivity is not always guaranteed and this leads to queued updates or queries from specific users locally on their mobile devices to the same data field or a document in the database cloud.
iv. Synchronization between multiple devices: Since the mobile client application users connect to a database using mobile applications from different geographical locations at different times, data read to the same document or data field must be consistent to separate users from the most recent update at the instant when connectivity is established.
v. Scalability and speed: As the number of users of a mobile client application increases the database must be able to effectively scale to accommodate the increase. Also, delays in response to user queries to the database should be minimal as this is noticeable and directly affects the user perception of the application performance.
vi. Network topology: The database physical network interaction should be in a manner such that data synchronization and availability can be configured in a way that will best suit the mobile client application requirements. Some topologies like mesh for instance allow different sections of data to be available offline without much synchronization issues arising as a result of any modification to the offline data.
vii. Data security and customization: The security of data with authentication protocols should be efficiently handled by the database, to avoid malicious use, data read/write, or misappropriation of user data and even the mobile client application.
viii. Secure data at rest and in motion: Data at the stages where it is stored both locally or on the cloud and when it is being moved between the client and the cloud database should be secured from external and unauthorized access and also should not be lost.
ix. Good access to data: The database should be readily accessible at any instant when there is a request by the mobile client.

4.2 Cloud Application DBMS

Features of databases for Cloud DBMS and the requirement for the DBMSs of mobile crowdsourcing application technique proposed include:

i. An ability to store basic data types such as String, Boolean, and number types. These basic data types are a predefined format for storing data in distinct ranges and forms like alphanumeric characters, numbers, date, Unicode characters, etc., and user custom-defined data format.
ii. The ability to hold as much data that is collected, easy integration, and support with mobile clients.
iii. Cost-effectiveness of the technique
iv. A storage format that supports hierarchical data format and access.
v. The ability for the data to be easily shipped to the local server.

4.3 Web Application DBMS

A web application is a program or software accessible through a web browser with its content delivered over a web server, and like any other native application on a mobile client, it communicates with a database for operational data. The database is important to web applications for the following reasons:

1. A database serves the purpose of storing a vast amount of data and optimizing queries of data, the same data that will be needed in populating the web application with content.
2. A database also provides data-to-program insulation.
3. A database easily allows the distribution of data to multiple users which ultimately makes it an essential part of a multi-user web application as the case is most of the time.

Features of databases for web application include:

1. Scalability: As the number of users of a web application increase, more volume of data will need to be handled to keep up with good performance hence the need for scalability which can be

efficiently handled by the database host service provider.

2. Speed: This is a foreground feature that defines operational capability. Delays attributable to requests in the database would be significant because this is tangible and impacts how the client perceives the application's performance [1]. Improvements can be made through strategies such as distributed database and browser features like caching can be used to improve speed performance for data retrieval on web applications.
3. Structure: This depends mainly on the business solutions for which the application will be used and subsequently the type of data that will be generated and stored, this data may be structured, semi-structured, or unstructured.
4. Availability: Data is always available to connected clients with multiple views for specific users [34].
5. Data consistency and security: Security is essential in preventing data breaches, loss, and availability [35]. Also, data consistency must be guaranteed and modifications must be constrained to set down rules to ensure that any document or data field can only be changed in a specific way [36].

4.4 The Local Server DBMS

The database becomes increasingly important down the workflow. Data on the cloud database will be shipped to the local server, for further analysis and storage. Exporting data from the cloud to the local server can be handled by a scheduled job that performs automated shipping of data from Cloud Firestore to the local server and it is best implemented using Firestore server SDKs [37]. The exported data retains its document-based structure which does not pose many difficulties for analysis and as regards the management of data on the local server is can be done using the Firestore server SDKs or cloud functions [38].

Data archived in the local server database is stored in formats that are suited for how they will be used. This research on mobile communication quality of service, for instance, requires this data to be available locally for analysis, therefore data that is pulled from the cloud database will be stored in a format (possibly .csv) that will serve the purpose of data analysis perfectly, focus on the review here will be on the server application and not particularly on data. Some features of databases for Local Server DBMS.

1. High storage capacity.
2. Cost-effective
3. Low downtimes
4. High-end computers to ensure minimum downtime.

5.0 SELECTED DATABASE FOR MOBILE CROWDSOURCING APPLICATION

5.1 Database Models (Services) for Cloud DBMS

Database Models (Services) considered for the Cloud DBMS are Firebase Real-Time Database, Cloud Firestore, MongoDB, PostgreSQL, Redis, Neo4j, and Cassandra. The non-relational databases were chosen in a way that the four major categories (Document stores, Column stores, Key-value stores, and Graph stores) are represented, except for PostgreSQL which is a relational database.

a. Firebase Real-Time Database

It is a NoSQL cloud-hosted database service offered by Firebase Incorporation. Data is stored as a JSON tree and synchronized in real-time to every connected client and remains available when the application goes offline. As mentioned earlier, data is stored as one simple large JSON tree and this generally gives efficiency and guarantees optimized performance in the execution of high-volume queries without delay or loss (low-latency advantages) [39].

b. Cloud Firestore

It is a NoSQL, document-based cloud-host database service also offered by Firebase Inc. Each document is grouped into collections may further point to other sub-collections. It features queries that are much faster and efficient than Firebase Real-Time Database and it also has better scalability [40], the speed advantage it has over Firebase Real-Time Database is attributed to all queries being indexed by default, this ensures that the query performance is proportional to the size of the result set data unlike Firebase Real-Time Database whose querying performance degrades as data grows [41].

c. MongoDB

It is similar to Cloud Firestore in that each document is grouped into collections and it is a NoSQL, document database. It is an open-source DBMS service offered by MongoDB Inc. and it is the most popular NoSQL database according to DB-Engines rankings [31], [42].

d. PostgreSQL

This is an open-source relational database where data is organized in tables, columns, and rows. It can be deployed on a self-managed cloud server or a fully managed cloud service.

e. Redis

This is a key-value store type non-relational database that natively provides fast response time hence its common use as a caching database and for applications that carry out heavy computation on query results that is to be sent to a client (mobile or web) since it significantly reduces query time.

f. Neo4j

This is a graph store type non-relational database mostly used for systems that are heavily reliant on

relationships that exist between data and uses its graph architecture to optimize complex queries.

g. Cassandra

This is a column store type non-relational database particularly built for storing a large amount of data quickly and easily scales when data becomes large.

Each of the compared database services provides various data types that cover whatever will be required, easy data migration features using REST APIs or Command Line Tools [43], [44], and can be easily integrated with mobile clients using reliable SDKs. Although MongoDB's solution does not provide as wide a range of cloud database management tools compared to the other [45]–[47]. Most especially when the database is not hosted using MongoDB Atlas which is a more expensive alternative to hosting on the personal cloud server also management tools are provided as part of the cloud database service for Firebase Real-Time Database and Cloud Firestore.

Limits to data storage size are based solely on subscription plans in the cases of Cloud Firestore and Firebase Real-Time Database [48] while in the case of MongoDB it is based on the configuration setup on MongoDB Atlas [49] Firebase Real-Time Database has the best presence support for native mobile applications [41] amongst all three although Cloud Firestore can leverage on Firebase Real-Time Database for this functionality [41] and pricing is generally cost-effective on pay-as-you-go plans [42], [48], [50].

Table 2: Comparison between Selected Database Models (Services) for the Cloud DBMS.

Characteristics	Firebase Real-Time Database	Cloud Firestore	MongoDB	PostgreSQL	Redis	Neo4j	Cassandra
Type of data	Supports most data types and generally structured in a JSON tree.	Supports most data types, Cloud Firestore references and generally structured in a JSON tree.	Data structures are represented in JSON, internal data is stored as BSON.	Supports a wide range of data types for table columns which constraints the values to be stored [51].	Supports a wide range of data types with the basic being String which covers various data types [52]	Basic data types are supported [53].	Supports a wide range of data types with the aforementioned basic data types covered by its native data type
Size of data	Dependent on plan The free tier plan (Spark) allows up to 1GB of data to be stored [48].	Dependent on plan The free tier plan (Spark) allows up to 1 GB of stored data to be stored [48].	The maximum BSON document size is 16MB and collection size vary according to shards into which data is split [49]	Data size is primarily capped by the available storage allocated for the database [54] or the database-as-a-service plan.	Depends on the limits set by the cloud management service in use.	Depends on the allocated space for the database or the limit set by the database-as-a-service plan.	Depends on the allocated space for the database or the limit set by the database-as-a-service plan.
Applicability	It is well suited for iOS and Android clients with offline support and can be seamlessly integrated with the aid of mobile-first, real-time SDKs.	It is well suited for iOS, Android, and web clients with offline support and can be seamlessly integrated with the aid of mobile-first, real-time SDKs.	It is more suited for large-scale applications which are mainly web-based.	It works well with web technologies such as websites through server-client web services like web Application Programmable Interfaces (APIs).	It is an excellent option for analyzing data in real-time, caching, and web-based applications.	It is ideal for systems where data is highly connected and complex queries are regularly executed as user-to-user interaction-intensive applications.	It is more suited for large-scale applications that require high uptimes and large amounts of data.
Availability	It is more suited for recording client connection	It is not suited for recording client connection	It does not have active presence	It has no presence	It has no presence	It has no presence support on	It has no presence

	status (it is presence supported) [41]	status (it does not have native presence support) [41]	support on mobile clients	support on mobile clients.	support on mobile clients.	mobile clients.	support on mobile clients.
Ease of use	It can be integrated without much hassle with mobile clients using reliable SDKs [47].	It can be integrated without much hassle with mobile and web clients using reliable SDKs [46].	It can be integrated with mobile clients using the new MongoDB Stitch [45].	Integration with mobile clients cannot be done directly but it requires an extra web API layer.	Integration of mobile clients is not possible without the use of a separate web API layer.	Integration with mobile clients cannot be done directly without an additional API layer.	Integration with mobile clients cannot be done directly without an additional API layer.
Cost	Pricing depends on the Firebase plan. Spark -free tier- (which has a limit caps on features), Blaze -pay as you go- with a rate of $5/GB for stored data [48].	Pricing depends on the Firebase plan. Spark -free tier- (which has a limit caps on features), Blaze -pay as you go- with a rate of $0.18/GB for stored data [48].	Although initial use is on a free tier, pricing is calculated on an hourly basis depending on several factors [42], if MongoDB Atlas is used rather than a self-managed server is.	This DBMS is usually deployed on a self-managed server and so the accrued cost for using this DBMS is just the cost of the cloud server, alternative fully managed cloud services are also offered at different prices.	The cost is determined by the fully-managed cloud service provider [55].	The cost is determined by the database-as-a-service provider or the cost of a self-managed server on which the DBMS is deployed.	While a self-managed cloud deployment cost is that of the server a fully managed Apache Cassandra Pricing varies depending on the service provider.
Storage format	Data is stored as a large JSON tree.	Data is stored as collections of documents.	Data is also stored as collections of documents.	Data is stored in tables; column-row format.	Data is stored in a key-value format.	Data is stored as a property graph model.	Data is stored in wide column stores.
Data acquisition	Data can be exported in a straightforward manner using a REST API service provided by Firebase [44] in JSON or CSV format.	Data can also be exported using the Cloud Firestore API [43] in JSON or CSV format.	Data can be exported using a command-line tool that produces a JSON or CSV export of data stored in a MongoDB instance [56].	Data backup dumps can be executed using the Postgres command line (CLI) tool [57] or a graphical user interface (GUI) like TablePlus [58].	Data exports can be carried out through a file transfer protocol (FTP) server or any of the other options provided by the cloud management service [59].	It supports various methods for whole and partial database exports through CLI commands or Cypher scripts [60].	Cassandra Query Language (CQL) can be easily used to export data through a CLI tool.

The comparison in Table 2 shows that all discussed databases have features that cover each requirement except some specific cases like:

1. PostgreSQL does not easily scale when data grows large.
2. The requirement for a mobile client integration which Cloud Firestore and Firebase Realtime database provide by default, unlike others that require an additional API layer to connect the database to the mobile application.

The proposed procedure for gathering data in this research is not labour-intensive (such as a physical survey) making the exact amount of data to be gathered unpredictable, this window of uncertainty at this point will only reflect in the

pricing incurred for the cloud storage provider and physical storage capacity and limits of local server for analysis.

A cloud-hosted database removes the burden of setting up synchronization configuration procedures with the mobile client because they have an easy-to-configure cloud service. Also, cloud-hosted databases solve issues of data distribution during scale-up operations (which is handled by the cloud host) and availability of data [41]. Cloud Firestore for instance is a Backend-as-a-Service (BaaS) deployment option under the larger Google Cloud Platform, providing server-side services for mobile and web clients.

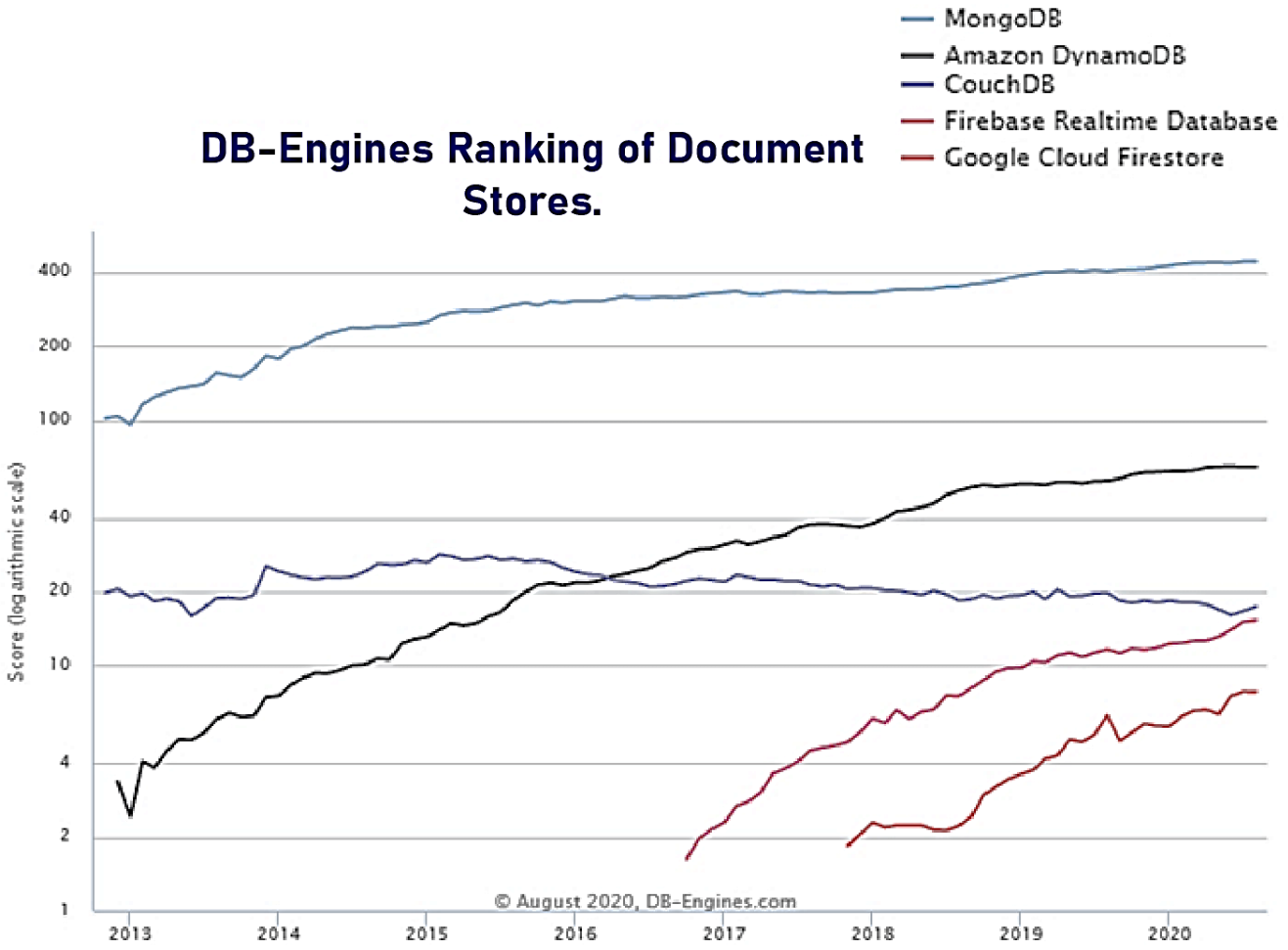

Figure 3: DB-Engines Ranking of Document Stores [31]

The database proposed for a cloud DBMS is a non-relational database. For the cloud storage, the proposed is the Cloud Firestore, the main reasons for choosing Cloud Firestore are as follows:

1. Cloud Firestore does not have native presence support for instant synchronization of data but it can leverage on Firebase Realtime Database's support by syncing Cloud Firestore and Realtime Database using Cloud Functions, the other databases compared do not have this feature hence it would require the software developers to create an additional API service to connect the mobile application to the cloud database.
2. The scalability of Cloud Firestore goes further compared to Firebase Real-time Database, with Cloud Firestore scaling up to 1 million concurrent connections while Firebase Real-time Database scales to about 200,000 concurrent connections [40]. PostgreSQL doesn't scale very well as it is a relational database that has to scale vertically and distributed clusters are not as easy to manage compared to other compared non-relational databases. MongoDB, Redis, Neo4j, and Cassandra all scale easily.
3. MongoDB has been filtered out due to not having active support for mobile clients, compared with both Firebase Real-time Database and Cloud Firestore [45]–[47] which are more commonly used for mobile applications and are more preferred by mobile application developers.
4. The cost of hosting a Cloud Firestore is much cheaper compared to Firebase Realtime Database according to the official pricing lists which prices Cloud Firestore storage at $0.18/GiB and Firebase Real-time database storage at $5/GB [42], [48], [50]. The cost of hosting a cloud instance of the

other compared databases depends on whether it is hosted on a self-managed server or a fully-managed database service of which is the former is usually charged for just the infrastructure while the latter is charged based on storage and query usage.

Firebase Real-Time Databases are limited to zonal availability in a single region and Cloud Firestore ensures data is shared across multiple data centres at once which will provide strong consistency of data at any instant [41].

5.2 *Database Models (Services) for the Web Application DBMS*

The requirements for a Web Application DBMS for operational database operations such as the management, authentication, and authorization of users (stakeholders) as seen in Figure 2 and also evaluation of aggregated results on the web application can be efficiently handled by the existing cloud database which is already managing metrics measurement data. This can be achieved using an API layer software that will interface with the database, execute the required database operations needed by the web application for operational data. Therefore, the review already carried out for Cloud Application Database covers the database service for the web application. The decision of utilizing an API layer instead of a separate database for the web application ensures that a centralized cloud database service is used for the entire system which subsequently saves the cost of running multiple cloud database services.

5.3 *Database Models (Services) for the Local Server DBMS*

The requirements for the local database server are that the server performs an intermittent extraction and storage of data from the cloud database to local storage and also grants data access to other computers that are securely connected locally for analysis. Therefore, decisions to be made concerning the server depends on optimization plans, developers' preference, and existing cloud database service.

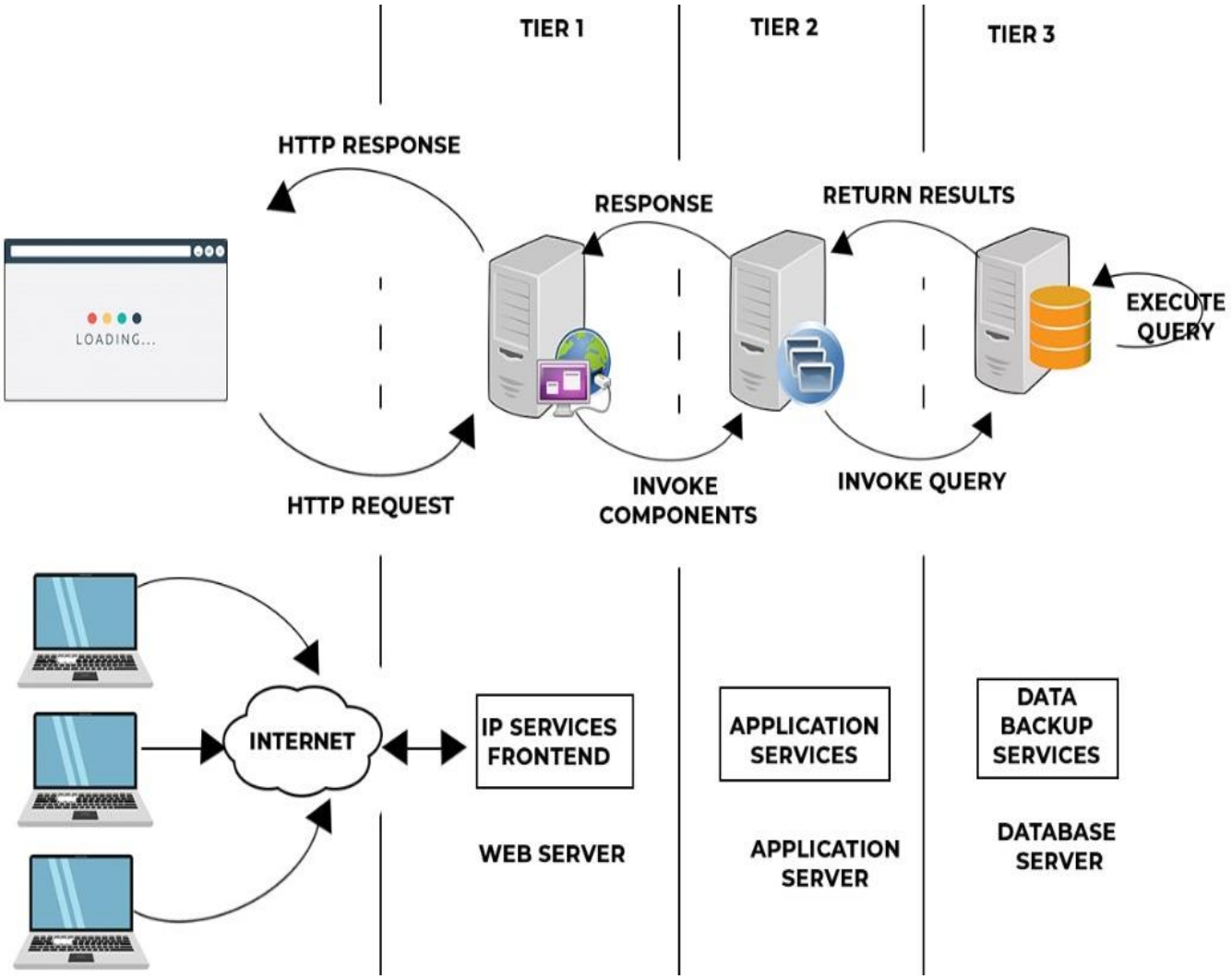

Figure 4: A Typical 3 Tier Server Architecture
(adapted from: https://images.app.goo.gl/icJBVqjMJYHMpvBb9)

a. Optimization: The local server could be run on a regular desktop operating system environment like Windows, but this will only compete with the primary assignments of the server mentioned above for the computing power of the computer [61]. For an optimized server, the computer should run a dedicated server operating system like Ubuntu Server.
b. Developers' Preference: This particularly determines the server's operating system; the choice of the operating system depends on which one the developer who will set up the system can efficiently work with.
c. Existing Cloud Database Service.
The already deployed cloud database service from which the local server will determine what SDK or software to be used to extract data from it. For instance, Firebase has cloud functions that can be implemented with custom scripts that will run on the server to execute data exports [38].

6.0 DISCUSSION

A robust mobile crowdsourcing application data management system is heavily reliant on data for its functionality, the word robust here signifies a system that covers most of the requirements for a DBMS that were identified in this study. In this survey, different features of DBMSs were compared along with the previous implementation of crowdsourcing mobile quality of service measurement systems and benchmark tests and reports carried out by [62], [63]. Although database normalization gives a huge advantage of reducing data redundancy it is most suited for structured data which requires a predefined and rigid schema. The need for normalization was not so desirable as it was pointed out in this review and a flexible data schema will be more suitable. Database query optimization by the DBMS service providers in combination with standard object-relational mapping libraries ensure that the absence efficiency of normalization of structured data is catered for by the aforementioned optimization features. Therefore, a non-relational database was proposed for the mobile-end database with Google Cloud Firestore proposed specifically due to its support for mobile client implementation. This choice also makes the integration of data from the mobile end-users to the cloud-hosted database relatively easier with all proposed services being part of the Google Cloud Platform. Although it is not as popular as some other database services as seen in Figures 1 and 3, separate comparative reviews of the DBMS performance by [62], [63] demonstrated that MongoDB (a non-relational database) performed better when reading large datasets and performing full-text queries, while MySQL (relational) and Cassandra (non-relational) performed much better for data insertion.

Google BigQuery was proposed as an appropriate data warehouse solution since it will provide continuity of direct integration with Cloud Firestore and its APIs for data migration from Cloud Firestore to BigQuery, and the local server. Google BigQuery provides machine learning support for data analytics. Furthermore, from this study, the need for a separate database for the web application was eliminated, considering that the web application mainly serves the purpose of visualization of results gathered from the analysis of collected data from the mobile clients, the required functional data can be provided through the API layer. This serves as an intermediary with specific endpoints and provides the web application with the required data. This also serves the purpose of separation of concerns as the web application only deals with just requesting data via the API endpoints and visualization while the API handles all queries on the database. It also optimizes the already existing database for the mobile client which the API will query for functional data required by the web application.

7.0 CONCLUSION

In this review, a comparison of DBMSs was carried out for a mobile crowdsourcing application and analysis, a broad overview of database types was given, through the description of their features, suitable areas of application and previous implementation of mobile applications along with database performance benchmarking reports helped to narrow down from the vast DBMS options. A mobile crowdsourcing application should be flexible considering the continuous evolution of the technology of mobile communication, this, therefore, makes it necessary that new systems and existing ones should be capable of accommodating changes instead of entirely new systems being developed due to innovations, to this effect the choice of a non-relational database in this review was considered. Also, when choosing a DBMS, it is important to consider how efficient such a system will be at the point of extracting data for analysis and comprehensive result, it will not do much good if data that is gathered cannot serve this end purpose well. Data should exist in a format or structure that will ensure that analysis and its results can be conveyed empirically. The proposed focus for subsequent works is that proper documentation of the process of choosing a DBMS and review of the basis for their preference should be discussed in upcoming works of literature on mobile applications because of its importance in the overall system accuracy and sustainability.

ACKNOWLEDGEMENT

This research was funded by the Nigerian Communication Commission Research Fund Grant 2020

REFERENCES

[1] Silberschatz, A., Korth, H. F. and Sudarshan, S. *Database System Concepts (7th. edition)*, (2019).
[2] Carlos, C., Steven, M. and Rob, P. "Database Systems: Design, Implementation and Management", *13th ed. CENGAGE*, (2010).

[3] Connolly, T., Begg, C. and Begg, C. "System Database A Practical Approach to Design, Implementation, and Management," (2005), 1427. Available: www.booksites.net/connbegg,.

[4] Machado, K., Kank, R., Sonawane, J. and Maitra, S. "A Comparative Study of ACID and BASE in Database Transaction Processing". *International Journal of Scientific & Engineering Research*, 8(5), (2017), 116–119.

[5] Oracle, "Data Concurrency and Consistency," *Oracle9i Database Concepts Release 2 (9.2) Part Number A96524-01*, (2002). https://docs.oracle.com/cd/B10501_01/server.920/a96524/c21cnsis.htm.

[6] Shanhong, L. "Big data - Statistics & Facts Statista," *Statista*, (2020). https://www.statista.com/topics/1464/big-data/.

[7] O'Dea, S. "Smartphone users worldwide 2016-2023, Statista," *Statista*, (2021). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[8] O'Dea, S. "Smartphone users in Nigeria 2014-2025, Statista," *Statista*, (2020). https://www.statista.com/statistics/467187/forecast-of-smartphone-users-in-nigeria/

[9] Phuttharak, J. and Loke, S. W. "A Review of Mobile Crowdsourcing Architectures and Challenges: Toward Crowd-Empowered Internet-of-Things," *IEEE Access*, 7, (2019), 304–324. doi: 10.1109/ACCESS.2018.2885353.

[10] Chatzimilioudis, G., Konstantinidis, A., Laoudias, C. and Zeinalipour-Yazti, D. "Crowdsourcing with smartphones," *IEEE Internet Comput.* 16(5), (2012), 36–44. doi: 10.1109/MIC.2012.70.

[11] Sukhwani, V. and Shaw, R. "Operationalizing crowdsourcing through mobile applications for disaster management in India". Progress in Disaster Science, (5), (2020), 100052. doi: 10.1016/j.pdisas.2019.100052.

[12] Wang, Y., Jia, X., Jin, Q. and Ma, J. "Mobile crowdsourcing: framework, challenges, and solutions," *Concurrency and Computation: Practice and experience*, *29*(3), (2017), e3789. doi: 10.1002/cpe.3789.

[13] Waze, "Harnessing real-time, crowdsourced data to improve crisis response." https://www.waze.com/ccp/casestudies/harnessing_real_time_crowdsourced_data_to_improve_crisis_response.

[14] PennState University, "OpenStreetMap and its use as open data| GEOG 585: Web Mapping," *GEOG 585: Open Web Mapping, Department of Geography*, (2020). https://www.e-education.psu.edu/geog585/node/738.

[15] Gill, B. "Measuring the Consumer Mobile Experience: Let's Get the Facts Straight | OPENSIGNAL," *OPENSIGNAL*, (2017).

[16] Nedkov, S. and Zlatanova, S. "Google Maps for Crowdsourced Emergency Routing," *International Society for Photogrammetry and Remote Sensing* XXXIX-B4, (2012), 477–482, doi: 10.5194/isprsarchives-xxxix-b4-477-2012.

[17] Nigerian Communications Commission, "Quality of Service." https://www.ncc.gov.ng/technology/standards/qos.

[18] Dahunsi, F. M. and Akinlabi, A. A. "Measuring mobile broadband performance in Nigeria: 2G and 3G," *Nigerian Journal of Technology*, *38*(2), (2019), 422-436. doi: 10.4314/njt.v38i2.19.

[19] Dahunsi, F. M. and Kolawole, G. "Participatory Analysis of Cellular Network Quality of Service," *International Journal of Computing & ICT Research* 9,(1),(2015),25-40. http://ijcir.mak.ac.ug/volume9-issue1/article3.pdf.

[20] Győrödi, C., Győrödi, R., Pecherle, G. and Olah, A., "A comparative study: MongoDB vs. MySQL," In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES), (2015),*1-6. doi: 10.1109/EMES.2015.7158433.

[21] Manoj, V. "Comparative study of nosql document, column store databases and evaluation of cassandra." *International Journal of Database Management Systems*, 6(4), (2014), 11–26. doi: 10.5121/ijdms.2014.6402.

[22] Oracle, "What Is a Database | Oracle," *Oracle In-Page Database Topics*, 2020. https://www.oracle.com/database/what-is-database.html.

[23] Ponniah, P. "Data Warehousing Fundamentals for it Professionals", *2nd ed. John Wiley & Sons, Inc*, (2010).

[24] Microsoft, "Non-relational data and NoSQL - Azure Architecture Center | Microsoft Docs," *Microsoft In-Page*, (2020). https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data.

[25] Nashat, D. and Amer, A.A. "A comprehensive taxonomy of fragmentation and allocation techniques in distributed database design". *ACM Computing Surveys (CSUR)*, 51(1), (2008), 1-25. doi: 10.1145/3150223.

[26] Login Radius Inc, "RDBMS vs NoSQL · LoginRadius Engineering," 2020. https://www.loginradius.com/engineering/blog/relational-database-management-system-rdbms-vs-nosql.

[27] Microsoft Corporation, "Relational vs. NoSQL data. Microsoft Docs." https://bit.ly/3oy5fYQ.

[28] DigitalOcean, "Pricing." https://do.co/3wkgdUs.

[29] Hardy, W. C. "QoS: measurement and evaluation of telecommunications quality of service". *1st ed. John Wiley & Sons, Inc*, (2001).

[30] "PostgreSQL: Documentation: 9.4: JSON Types." https://www.postgresql.org/docs/9.4/datatype-json.html (accessed May 07, 2021).

[31] Solid IT gmbh, "Method of calculating the scores of the DB-Engines Ranking," (2020). https://db-engines.com/en/ranking_definition (accessed Oct. 20, 2020).

[32] MongoDB, "Introduction to MongoDB Realm for Mobile Developers — MongoDB Realm," (2020). https://docs.mongodb.com/realm/get-started/introduction-mobile/

[33] Puneet, M. Kaur, J. and Pallavi, M. "DataMining Techniques for Software Defect Prediction," *International Journal of Software and Web Science*, 3(1), (2013), 54–57.

[34] Rex, H. "A Practical Guide to Database Design". *2nd Edition. Routledge*, (2018).

[35] Kedar, S. "Database Management Systems". *First Edition. Technical Publications*, (2009).

[36] Ports, D.R., Clements, A.T., Zhang, I., Madden, S. and Liskov, B. *"Transactional Consistency and Automatic Management in an Application Data Cache". Proceeding 9th USENIX Symposium. Oper. Syst. Des. Implementation, OSDI,* (2010), 279–292.

[37] Firestore, "FirestoreAdminClient - Documentation." https://googleapis.dev/nodejs/firestore/latest/v1.FirestoreAdminClient.html#exportDocuments.

[38] Google, "What is BigQuery Data Transfer Service? Google Cloud." https://cloud.google.com/bigquery-transfer/docs/introduction (accessed Oct. 30, 2020).

[39] Firebase, "Firebase Realtime Database," (2020). https://firebase.google.com/docs/database

[40] Firestore, "Cloud Firestore Data model | Firebase," (2020). https://firebase.google.com/docs/firestore/data-model (accessed Oct. 30, 2020).

[41] Firebase, "Choose a database: Cloud Firestore or Realtime Database | Firebase," 2020. https://firebase.google.com/docs/firestore/rtdb-vs-firestore (accessed Oct. 10, 2020).

[42] MongoDB, "MongoDB Atlas FAQ | MongoDB," 2020. https://www.mongodb.com/cloud/atlas/faq (accessed Oct. 30, 2020).

[43] Firebase, "Export and import data | Firebase." https://firebase.google.com/docs/firestore/manage-data/export-import (accessed Oct. 20, 2020).

[44] Firebase, "Retrieving Data | Firebase," 2020. https://firebase.google.com/docs/database/rest/retrieve-data (accessed Oct. 30, 2020).

[45] mongoDB, "Create a Realm App (Realm UI) — MongoDB Realm," 2020. https://docs.mongodb.com/realm/get-started/create-realm-app/ (accessed Oct. 30, 2020).

[46] Firestore, "Get started with Cloud Firestore | Firebase," 2020. https://firebase.google.com/docs/firestore/quickstart (accessed Oct. 30, 2020).

[47] Firebase, "Installation & Setup on Android | Firebase Realtime Database," 2020. https://firebase.google.com/docs/database/android/start (accessed Oct. 30, 2020).

[48] Firebase, "Firebase Pricing," 2020. https://firebase.google.com/pricing (accessed Oct. 30, 2020).

[49] MongoB, "MongoDB Limits and Thresholds — MongoDB Manual," 2020. https://docs.mongodb.com/manual/reference/limits/

[50] MongoDB, "Pricing | MongoDB," 2020. https://www.mongodb.com/pricing

[51] PostgreSQL, "PostgreSQL: Documentation: 9.5: Data Types." https://www.postgresql.org/docs/9.5/datatype.html

[52] Redis, "Data types." https://redis.io/topics/data-types (accessed May 07, 2021).

[53] Neo4j, "Values and types - Neo4j Cypher Manual." https://neo4j.com/docs/cypher-manual/current/syntax/values/

[54] PostgreSQL, "PostgreSQL: Documentation: 12: Appendix K. PostgreSQL Limits." https://www.postgresql.org/docs/12/limits.html

[55] "Hosted Redis Database on the Kubernetes Cloud Platform - RClusters." https://bit.ly/3v3eFOl

[56] mongoDB, "mongoexport — MongoDB Manual," 2020. https://github.com/mongodb/docs/blob/v4.0/source/reference/program/mongoexport.txt or https://docs.mongodb.com/manual/reference/program/mongoexport/ (accessed Oct. 30, 2020).

[57] "PostgreSQL: Documentation: 9.1: SQL Dump." https://www.postgresql.org/docs/9.1/backup-dump.html.

[58] "PostgreSQL - How to copy a database to another server? | TablePlus." https://tableplus.com/blog/2018/04/postgresql-how-to-copy-database-to-other-server.html (accessed May 07, 2021).

[59] Redis Labs Documentation Center, "Export data from a database." https://docs.redislabs.com/latest/rs/administering/import-export/exporting-data/ (accessed Apr. 27, 2021).

[60] Redis Labs Documentation Center, "Export-APOC Documentation." https://neo4j.com/labs/apoc/4.1/export/ (accessed Apr. 27, 2021).

[61] Ubuntu, "Install Ubuntu Server | Ubuntu," 2020. https://ubuntu.com/tutorials/install-ubuntu-server#1-overview (accessed Dec. 15, 2020).

[62] Boicea, A., Radulescu, F. and Agapin, L.I., "MongoDB vs Oracle - Database comparison," MongoDB vs Oracle--database comparison. In *2012 third international conference on emerging intelligent data and web technologies* 330–335, 2012, doi: 10.1109/EIDWT.2012.32.

[63] S. C. Satapathy, V. K. Prasad, and S. K. Udgata, *Proceedings of the First International Conference on Computational Intelligence and and Informatics*. 2016.