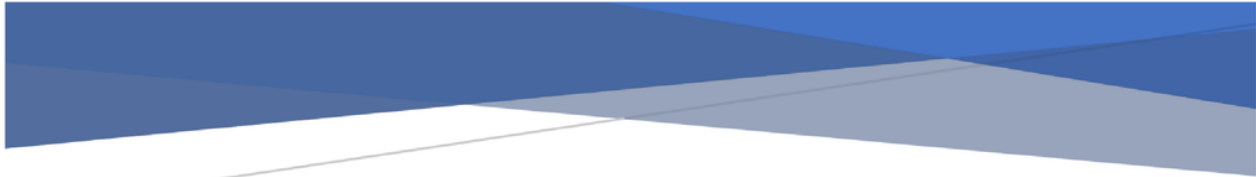


Assignment Ryan Lang

by Ryan Lang

FILE	7936_RYAN_LANG_ASSIGNMENT_RYAN_LANG_178152_223802928.DOCX (943.25K)		
TIME SUBMITTED	05-APR-2019 02:27PM (UTC+0100)	WORD COUNT	3119
SUBMISSION ID	104204096	CHARACTER COUNT	17632



Investigating Emerging Web Technologies

M6X01096 Mike Dacey

Ryan Lang 1603678
1603678@student.uwtsd.ac.uk

Abstract

There are many emerging technologies in the world of web development. These new technologies have the potential of creating a huge impact on the future of web development. Web developers will need to use different technologies for different applications. Different approaches to databases are used depending on how they will be used. This report evaluates a NoSQL database approach and compares it to a traditional SQL approach. A typescript application will showcase different dynamic website elements including using a REST web service to edit a database.

Table of Contents

1. Introduction	1
2. MongoDB	1
3. NoSQL vs Traditional SQL	3
Conclusion.....	6
4. Typescript Application	6
Testing.....	7
5. REST Web Service.....	8
Testing.....	8
6. MongoDB Demonstration Code	9
Testing.....	11
7. References	13

1. Introduction

For a technology to have a huge impact on web development it needs to be innovative and adopted by developers. There have been massive technologies in the history of web development that have become the go to way of development. For example, Adobe Flash Player used to be used for displaying video on websites, but when HTML5 came out it had the capability of playing video without the need for third party technology and most websites adopted it. This changed the way developers approached designing a website for sharing videos [1]. There are many technologies that get created that give developers new approaches to develop their sites, and some have the potential to have a great impact on the future of web development.

2. MongoDB

MongoDB is a DBMS (Database Management System) that is a NoSQL database solution [2]. NoSQL refers to a database technology that does not follow the model of relational databases [3]. The technology is flexible enough to be used in different industries, being adopted by search engine giant Google, video game company SEGA, and even being used by the UK government's digital services [4]. It is this flexibility that makes MongoDB such a compelling choice for developers. MongoDB has also been shown to have good performance in regards to reading and writing to databases [5].

Applications are storing more data than ever, and accessing that data at increasingly higher rates. Running the application on a single server means that you will eventually hit a scaling limit. MongoDB is able to scale by adding more servers, allowing for more capacity when it is required. MongoDB is a document database, meaning that is non-relational [6]. A record in a document database is called a document [7]. Documents are assembled into collections which are similar to tables in SQL databases [8]. It stores and queries data as JSON-like documents. Normalisation is not a thing that is done in MongoDB. Document databases make it easier for developers to work with databases because they use the same document model format as is in their applications code.

MongoDB is a good option for developers because it includes features like full index support, sharding, and replication. Sharding is useful because it allows the database to be partitioned into smaller, easily managed parts called data shards. It is also open source with many developers making unofficial applications for use with MongoDB, such as Robo 3T, a GUI for creating MongoDB databases. MongoDB functions across different platforms, meaning if you are a MAC OS X or Linux user you are able to work with MongoDB.

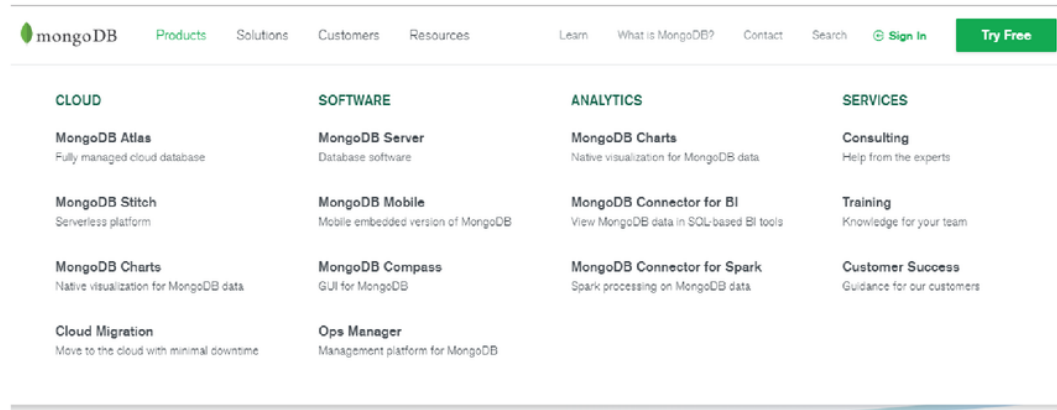


Figure 1: MongoDB Products

Source: Adapted from [9]

Figure 1 shows the products that are officially available from MongoDB. Developers are going to want to be doing different things with this technology and having products available to suite a variety of different needs is a reason why MongoDB is a great choice for developers. The initial release of MongoDB was in 2009, and in 2016 they announced MongoDB Atlas, a way of running MongoDB in the cloud. In 2018 they launched new database cluster sizes and cloud regions on Googles cloud platform. It is clear that MongoDB is continuing to evolve and add more features, improving the developers experience with the technology.

MongoDB also features analytic tools that are not available on competing NoSQL solutions. Real time data reporting is supported, generating a graph to display how much traffic the database is getting. MongoDB does a good job of integrating all of its tools together so you do not to spend huge amounts of time setting things up, compatibility between software works with minimal setup.

MongoDB is one of the most popular NoSQL database solutions available [10]. It is horizontally scalable, which means that more computers can be added to increase the

performance of the system for when there is more data throughput. Sharding allows the database to be distributed across a cluster, further increasing performance and reliability. MapReduce is also supported which is a technology that allows the answer to a query to be organised and reduce the results.

MongoDB allows for executing queries on collections, and is capable of some more complex aggregations, which sets it apart from other NoSQL technologies [11]. Big data and the demands of modern tech companies have shown that NoSQL is very much a required technology, and MongoDB is being widely adopted because of its feature set. In the future as computer technology keeps improving, MongoDB has the potential to offer all of the benefits of NoSQL while also being a good choice for developers who would typically use SQL databases. This is because of MongoDB's strong integration between software and working to try to standardise NoSQL database management.

3. NoSQL vs Traditional SQL

There are reasons that developers sometimes opt to use a NoSQL database over a traditional SQL database. We use NoSQL databases for big data, because relational databases are not able to handle the immense amount of data that some companies are processing [12]. Big data is usually defined by the volume, variety, velocity and the veracity of the data, with the variability sometimes being taken into account.

The volume of the data can be petabytes. The variety of the data can be structured, unstructured and contain many different types of data e.g. images, video, audio etc. The velocity of data refers to the stream of data. Big companies like Amazon and Google are continuously streaming data nonstop. The veracity is the uncertainty of data collected, and the variability refers to different interpretations of the same data [13]. With the sheer amount of data that a company like Google is dealing with, using an SQL database is out of the question. SQL queries would take far too long searching through petabytes of data, and they not as scalable as NoSQL databases.

SQL databases do have their uses though, with some features that are not available on NoSQL databases. With an SQL database you can enable enforcement of referential integrity, and SQL works on lots of different platforms and the code for queries is universal across them.

Creating SQL databases is easier than NoSQL databases. There is no standard for NoSQL databases, meaning switching between them means having to learn how to use each one.

NoSQL databases do not use a schema so any type of data is supported [14]. Big data demands low complexity, so the lack of a schema is fine in this use case.

There are more advantages to NoSQL databases than just their use with big data though. Being flexible they fit the agile methodologies that a lot of developer's teams will be using better. There are lots of options available for tools to manage NoSQL databases, and NoSQL is seeing a lot more innovation than SQL databases. MongoDB is offering some of the features and benefits traditionally associated with SQL, but for NoSQL databases.

A key-value database is an example of a NoSQL database system [15]. Key-value databases provide low latency and high throughput [16]. The concept is to use a hash table and provide each item of data with a unique key and a pointer [17]. Data can then only be accessed by using the key. An example of a key-value database is Amazon's simpleDB. Indexes record the mapping between keys and values which allows for fast queries [18]. The value that is stored is the actual data and because there is no schema this data can be of any type. Key-value store provides two simple operations: KV pair insertion and key lookup [19].

Key-value databases are useful for the duration of a user's session on a website. A user can log in and browse a store and during their time on the store, what products they view may be tracked and personalised themes applied, with recommendations and promotions shown to the user. Each user has a unique identifier and the session data is never queried by anything other than the primary key, which means key-value store will be efficient at this task.

Key-value databases for big data are being stored across multiple computers, splitting the data between machines, while a coordinator machine directs clients to the machine with the resource that they want to access. When a machine goes down, the user wanting to access the data will be unaffected, because of the redundancy built into the database system. An SQL server will not have the same fault tolerance as this, because they cannot be distributed in the same way. Key-value stores are not good for more advanced querying though.

Distributed key-value store consists of four layers: key-value store API layer; data fetch layer; routing layer; and storage layer. The key-value store API layer provides the APIs for

applications, such as get/put operations [20]. The second layer will convert data into chunks and the storage layer stores those chunks. The routing layer controls the look up of storage nodes that has a chunk that is needed. These systems can store billions of key-value pairs and provide speedy connectivity to a huge number of users [21].

One of the most popular SQL database management systems is Microsoft SQL Server. Being a Microsoft product it has good integration with other Microsoft products and services. Visual Studio has integration with SQL server, as does Microsoft Azure services. Microsoft SQL server allows for other software to connect to it and perform operations on the data. Microsoft SQL Server is used when data needs to be structured and the data is not considered big data. Universities will often use Microsoft SQL server, because a university will need data to be organised, and queries to be performed across multiple tables, combining data into an understandable report. A key-value store would not allow for the organisation of data required by many institutions. SQL allows for complex queries, for example, if a university wanted to see all students from Llanelli doing applied computing, who are over the age of 25 and are male, this could easily be achieved with standard SQL code. SQL queries allow for joins between tables, so data can be viewed however the user wants to see the data, assuming they know what SQL code to enter.

SQL databases are what is known as vertically scalable, meaning you can upgrade the existing machine, but is not going to equal the compute power of a resource that is horizontally scaled. SQL emphasises ACID (Atomicity, Consistency, Isolation and Durability) whereas NoSQL follows CAP theorem (Consistency, Availability and Partition tolerance) [22]. Having primary and foreign keys in SQL tables allows for data to be accessible between tables, relationships to be set up to ensure data is accurate, and for unique identifiers to remain unique.

The development of NoSQL databases is continuing with options like MongoDB being updated regularly, with new compelling features added every now and then. NoSQL is essential for big data, which means massive tech companies like Google, Amazon and Facebook are all using various NoSQL technologies that are continuing to evolve.

Conclusion

There are different reasons why a developer would want to choose SQL or NoSQL. SQL is suitable for databases that are smaller in scale and where complex queries are needed to be complete. NoSQL is suited for big data with many options available, like MongoDB, key-value stores, but there is also a lack of standardisation. Managing a NoSQL database can be complicated, and setting up a distributed database for big data is very complicated and pricey challenge, but the performance and reliability when done right is worth the effort.

4. Typescript Application

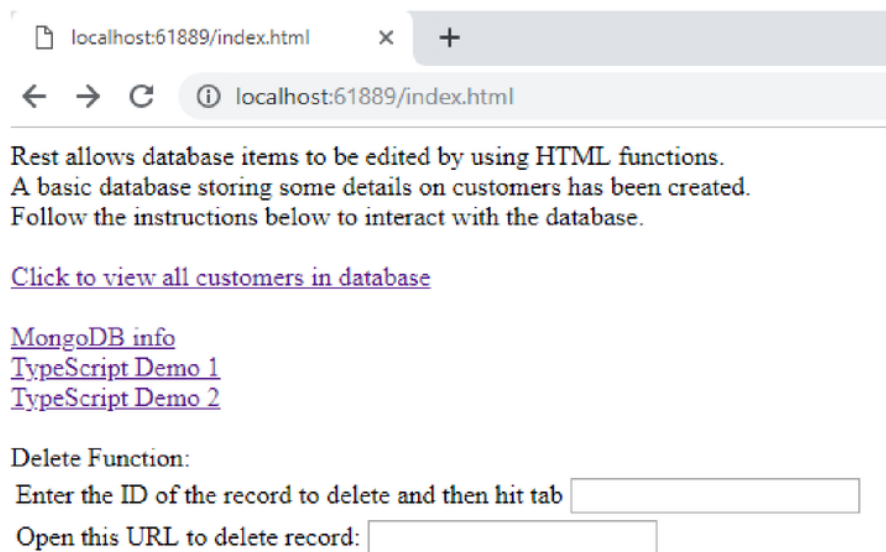


Figure 2: The home page of the web application

There are two links to pages that utilise typescript. The first writes some text and adds two numbers together. The second writes the current time and date.

Testing

Test 1: What happens when first page is loaded?

Expected result: It should write "Hello World", "Goodbye", and the answer to a maths equation.

Actual Result: It displays as expected.

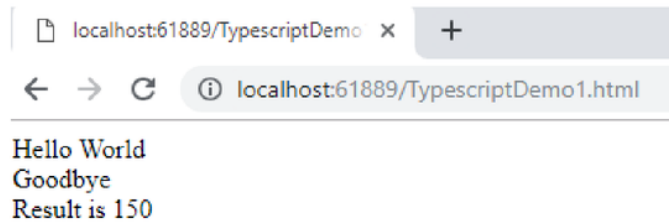


Figure 3: TypescriptDemo1 page

Test 2: What happens when the second page is loaded?

Expected Result: The date and time should be shown.

Actual Result: The data and time is shown.

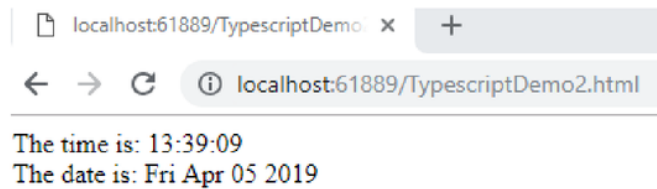


Figure 4: The date and time being shown correctly in TypescriptDemo2

5. REST Web Service

```

namespace WcfService
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]
        [WebInvoke(Method = "GET", ResponseFormat = WebMessageFormat.Json, UriTemplate = "GetCustomer")]
        IEnumerable<Customer> GetCustomers();
        [OperationContract]
        [WebInvoke(Method = "POST", RequestFormat = WebMessageFormat.Json, UriTemplate = "InsertCustomer")]
        void InsertCustomer(Customer cObj);
        [OperationContract]
        [WebInvoke(Method = "POST", RequestFormat = WebMessageFormat.Json, UriTemplate = "UpdateCustomer")]
        void UpdateCustomer(Customer cObj);
        [OperationContract]
        [WebInvoke(Method = "GET", RequestFormat = WebMessageFormat.Json, UriTemplate = "DeleteCustomer/{id}")]
        void DeleteCustomer(string id);
    }
    [DataContract]
    public class Customer
    {
        [DataMember]
        [Key]
        [Required]
        public int id { get; set; }
        [DataMember]
        [Required]
        public string Name { get; set; }
        [DataMember]
        [Required]
        public string Address { get; set; }
    }
}

```

Figure 5: The REST service contracts

A WCF REST service was created and connected to a website.

Testing

Test 1: Can all of the records be viewed?

Expected Result: Clicking the link to view all records should display all of the records.

Actual Result: The records are displayed.



Figure 6: All of the records in the database being displayed

Test 2: Does the delete function remove the record from the database?

Expected Result: The record should be deleted.

Actual Result: The record is deleted.



Figure 7: The record with ID 2 is deleted

6. MongoDB Demonstration Code

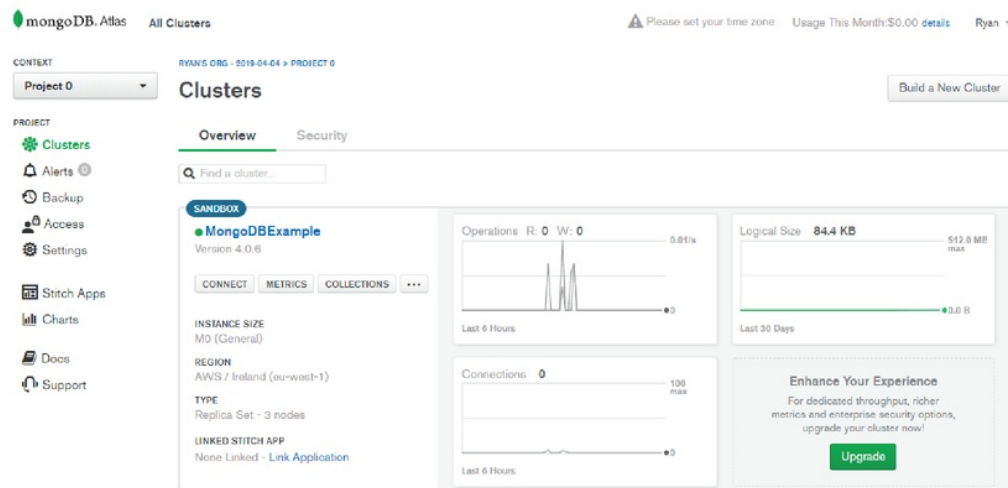


Figure 8: The MongoDB Atlas GUI

The MongoDB database being used for this example has been created on the cloud via MongoDB Atlas.

×

Connect to MongoDBExample

✓ Setup connection security ✓ Choose a connection method Connect

1 Choose your driver version

DRIVER

C# / .NET

VERSION

2.5 or later

2 Add your connection string into your application code

Connection String Only

Full Driver Example

```
var client = new MongoClient("mongodb+srv://mongouser:<password>@mongodt
var database = client.GetDatabase("test");
```

Copy

Replace *mongouser* and *<password>* with your database user's (MongoDB User) credentials.
When entering your password, make sure that any special characters are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close

Figure 9: A look at the driver selection screen on MongoDB Atlas

A lot of different programming languages are supported by MongoDB.

```

1  using MongoDB.Driver;
2  using MongoDB.Bson;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Data.Sql;
9  using System.Data.SqlClient;
10
11 namespace ConnectToMongoDB
12 {
13     class Program
14     {
15         static void Main(string[] args)
16         {
17             string id = "";
18             string name = "";
19             var client = new MongoClient("mongodb+srv://mongouser:mongouser@mongodbexample-57u0z.mongodb.net/test?retryWrites=true");
20             var database = client.GetDatabase("Names");
21             var collect = database.GetCollection<BsonDocument>("Names");
22             Console.WriteLine("Please enter an ID and hit enter");
23             id = Console.ReadLine();
24             Console.WriteLine("Please enter a name");
25             name = Console.ReadLine();
26             var document = new BsonDocument
27             {
28                 {"ID", id },
29                 {"Name", name },
30             };
31             collect.InsertOneAsync(document);
32             Console.WriteLine("Success. Here are all the records in the database");
33             collect.Find(new BsonDocument()).ForEachAsync(X => Console.WriteLine(X));
34             Console.ReadLine();
35         }
36     }
37 }
38

```

Figure 10: The code to read/write to the database

This example is simplistic, but shows that storing data on a database in the cloud with MongoDB is a good option for those looking for a solution that works with different programming languages and are also looking for some analytical tools.

Testing

Test 1: Can the application read the database records?

Expected Result: The records are shown in the console application.

Actual Result: The records are displayed correctly.

```

Success. Here are all the records in the database
{ "_id" : ObjectId("5ca6d784d8d0e0367c8ded81"), "ID" : "2", "Name" : "Rhys" }
{ "_id" : ObjectId("5ca6d869d8d0e039d47c6624"), "ID" : "5", "Name" : "Paul" }
{ "_id" : ObjectId("5ca6d89fd8d0e04188761bed"), "ID" : "8", "Name" : "Dave" }

```

Figure 11: The records from the MongoDB database shown in a console application

Test 2: Can records be added to the database?

Expected Result: Records can be added.

Actual Result: Records are added successfully, however the program needs to be restarted to view the most recent entry into the database.



```
C:\Users\Ryan\Desktop\My Emerging Web Code\WcfService\ConnectToMongoDB\bin\Debug...
Please enter an ID and hit enter
9
Please enter a name
Peter
Success. Here are all the records in the database
{ "_id" : ObjectId("5ca6d784d8d0e0367c8ded81"), "ID" : "2", "Name" : "Rhys" }
{ "_id" : ObjectId("5ca6d869d8d0e039d47c6624"), "ID" : "5", "Name" : "Paul" }
{ "_id" : ObjectId("5ca6d89fd8d0e04188761bed"), "ID" : "8", "Name" : "Dave" }
```

Figure 12: The console application adding records.

7. References

- [1] H. P. Prima Dewi and N. Syifana, "Clickable and interactive video system using HTML5," in *The International Conference on Information Networking 2014 (ICOIN2014)*, Phuket, 2014.
- [2] Y. Widayani, H. Laksmiwati and E. D. Bangun, "Mapping spatio-temporal disaster data into MongoDB," in *2016 International Conference on Data and Software Engineering (ICoDSE)*, Denpasar, 2016.
- [3] Y.-S. Kang, I.-H. Park, J. Rhee and Y.-H. Lee, "MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 485-497, 2015.
- [4] "Our Customers | MongoDB," MongoDB, 2019. [Online]. Available: <https://www.mongodb.com/who-uses-mongodb>. [Accessed 04 April 2019].
- [5] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," in *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, 2013.
- [6] G. Ongo and G. P. Kusuma, "Hybrid Database System of MySQL and MongoDB in Web Application Development," in *2018 International Conference on Information Management and Technology (ICIMTech)*, Jakarta, 2018.
- [7] Y. Widayani, H. Laksmiwati and E. D. Bangun, "Mapping spatio-temporal disaster data into MongoDB," in *2016 International Conference on Data and Software Engineering (ICoDSE)*, Denpasar, 2016.
- [8] J. Kumar and V. Garg, "Security analysis of unstructured data in NOSQL MongoDB database," in *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, Gurgaon, 2017.
- [9] MongoDB, [Online]. Available: <https://www.mongodb.com/>. [Accessed 04 April 2019].
- [10] "MongoDB scheme analysis," in *2017 IEEE 21st International Conference on Intelligent Engineering Systems (INES)*, Larnaca, 2017.
- [11] P. Colombo and E. Ferrari, "Enhancing MongoDB with Purpose-Based Access Control," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 591-604, 2017.
- [12] R. M. Chopade and N. S. Dhavase, "Mongodb, couchbase: Performance comparison for image dataset," in *2017 2nd International Conference for Convergence in Technology (I2CT)*, Mumbai, 2017.
- [13] K. Park, M. C. Nguyen and H. Won, "Web-based collaborative big data analytics on big data as a service platform," in *2015 17th International Conference on Advanced Communication Technology (ICACT)*, Seoul, 2015.

- [14] T. Ciprian-Octavian, F. R'adulescu, A. Boicea and I. Bucur, "Performance evaluation for CRUD operations in asynchronously replicated document oriented database," in *International Conference on Control Systems and Computer Science*, 2015.
- [15] R. Cattell, "Scalable SQL and NoSQL data stores," *SIGMOD Record*, vol. 39, no. 4, pp. 12-27, 2010.
- [16] H. Horie, M. Asahara, H. Yamada and K. Kono, "Pangaea: A Single Key Space, Inter-datacenter Key-Value Store," in *2013 International Conference on Parallel and Distributed Systems*, Seoul, 2013.
- [17] M. M. Patil, A. Hanni, C. H. Tejeshwar and P. Patil, "A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retrieval operations using a web/android application to explore load balancing — Sharding in MongoDB and its advantages," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, 2017.
- [18] J. Li, N. Xiao, Y. Lu, Z. Chen, F. Liu, Y. Xing and S. Li, "Review on HDD-Based, SSD-Based and Hybrid Key-Value Stores," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, Orlando, 2017.
- [19] G. Lu, Y. J. Nam and D. H. C. Du, "BloomStore: Bloom-Filter based memory-efficient key-value store for indexing of data deduplication on flash," in *012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, San Diego, 2012.
- [20] H. Horie, M. Asahara, H. Yamada and K. Kono, "Minimizing WAN Communications in Inter-datacenter Key-Value Stores," in *2014 IEEE 7th International Conference on Cloud Computing*, Anchorage, 2014.
- [21] S. Xiong, Y. Yao, S. Li, Q. Cao, T. He, H. Qi, L. Tolbert and Y. Liu, "kBF: Towards Approximate and Bloom Filter based Key-Value Storage for Cloud Computing Systems," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 85-98, 2014.
- [22] P. A. Bernstein, I. Cseri, N. Dani, N. Ellis, A. Kalhan, G. Kakivaya, D. B. Lomet, R. Manne, L. Novik and T. Talus, "Adapting microsoft SQL server for cloud computing," in *2011 IEEE 27th International Conference on Data Engineering*, Hannover, 2011.

Assignment Ryan Lang

GRADEMARK REPORT

FINAL GRADE

75/100

GENERAL COMMENTS

Instructor

Eval emergent tech (20) 18

Eval of NoSQL (30) 21

Typescript application (20) 14

REST web service (20) 14

Tech demo (10) 8

The emergent technology that you chose was MongoDB. and you provided detailed and a good case for its future importance.

The NoSQL database you chose was MongoDB. Good context and analysis of SQL vs NoSQL. I wanted to see more explanation of how an example NoSQL database could be used e.g. a document in MongoDB vs a table in SQL.

Code provided give two simple demos of TypeScript. A REST service is provided that uses LINQ to access a database but only demonstrates viewing the contents and deleting an entry.

The tech demo illustrates using C# to access a MongoDB database. Simple but quite nice demo.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17
