A TOOL FOR MANAGING KEY VALUE PAIRS ON ORACLE NOSQL DATABASE

A

Project Report submitted to University of Pune,Ganeshkhind

In Fulfillment for the awards of Degree of Bachelor in
Computer Engineering

Submitted by

Anjali D. Kapadni **B80414274**
Sushank Dahiwadkar **B804142**

Under the Guidance of

Prof. M. R. Sanghavi

DEPARTMENT OF COMPUTER ENGINEERING

SNJB'S LATE SAU. KANTABAI BHAVARLALJI JAIN,
COLLEGE OF ENGINEERING, CHANDWAD, DIST:NASHIK

May 2014

SNJB'S LATE SAU. KANTABAI BHAVARLALJI JAIN, COLLEGE OF ENGINEERING, CHANDWAD, DIST:NASHIK
2013 - 2014

CERTIFICATE

This is to certify that,

Anjali D. Kapadni **B80414274**
Sushank Dahiwadkar **B804142**

have successfully completed this project report entitled *A Tool for Managing Key Value pairs on Oracle NoSQL Database*under my guidance in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer under the University of Pune during the academic *2013 - 2014*

Date: April 27, 2014
Place: Chandwad

Prof. M. R. Sanghavi
Guide **External**

Prof. M. R. Sanghavi
Head

Dr. V.J.Gond
Principal

Acknowledgement

We would like to acknowledge all the people who have been of the help and assisted us throughout our project work.

First of all we would like to thank our respected guide Prof. M.R. Sanghavi, Head of Department of Computer Engineering for introducing us throughout features needed. The time-to-time guidance, encouragement, and valuable suggestions received from her are unforgettable in our life. This work would not have been possible without the enthusiastic response, insight, and new ideas from her. We like to thank our mentor Mr. Shirish Joshi working at Persistent Systems Private Limited,Pune.

We gladly take this opportunity to thank Dr.V.J.Gond, Principal, SNJBs College of Engineering, Chandwad for providing all the facilities during the progress of dissertation. We are also grateful to all the faculty members of SNJB's College of Engineering for their support and cooperation.

We would like to thank our lovely parents for time-to-time support and encouragement and valuable suggestions, for the emotional as well as strong support each and every time also for continued loving care and emotional support at every stage of this project.

We would like to thank our friends for their valuable support and encouragement. In a countless way we have received support and love from all our family members who have constantly encouraged us to achieve new heights. The acknowledgement would be incomplete without mention of the blessing of the Almighty, which helped us in keeping high moral during most difficult period.

Anjali D. Kapadni
Sushak dahiwadkar

Abstract

Keyword :

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

In today's Information Technology world Due to huge usage of internet, online data exchange is exponentially increased. And it is continuously increasing. All this data is generated from social networking websites, E-Commerce portals, and many more Networks. This data includes uploaded images, posted articles, web crawling records, online shopping records, Sensor data etc. As the data increases the need to store that data also increases. And this need of more storage leads to increase in storage cost. This huge data is referred as **Big Data**.

Big data is very large in size which exceeds some petabytes. Managing big data is very tedious job. Many technologies are introduced to manage this big data, but these tools were able to manage the data up to some extent. The data is been stored onto Relational database management systems. But RDBMS are also having their own limitations. As the data increases the operations to manage them also gets more complex. Many operations such as complex queries and joins are tedious to workout. And many more technical issues come up which demands something new for handling this Problem.

Many promising technologies were used to handle the data storage needs in past decades but none of them proven to be sufficient. As a result every industry facing the problem of data storage came up with in-house solution, and these lead to foundation of standard of NoSQL, means Not Only SQL (Structured Query Language). NoSQL is the technology which defines many standard and techniques to handle the big data. It does not include structured query language but other techniques to handle the data. NoSQL defines various methods which ensure the data storage over distributed environment. Some NoSQL technologies include MongoDB, CoughDB, and Cassandra etc.

Oracle is a leading player in the Database technologies. So oracle also came up with its product i.e. Oracle NoSQL. Oracle NoSQL is a Key Value Paired database. Oracle

NoSQL has a strong architecture which ensures all the functions required for storage of Big Data. It is in developing phase. Oracle NoSQL does not have any shell or GUI to simplify the user's need to use the product.

This encouraged us for choosing the task to develop a GUI which will be able to manage the Key value pairs on Oracle NoSQL database. Project does not only include the GUI development, but the project is the complete package to manage the Oracle NoSQL Database. The first part of project includes the study of NoSQL and Oracle NoSQL, development of a tool to manage Key Value pairs. The Application is platform independent. The second part of project development includes developing RESTFul web services to manage the Oracle NoSQL database, and developing an Android application to demonstrate the consumption of this web services on Android Platform.

This project is a complete package to work with Oracle NoSQL Database.

1.2 Brief Description

As the project is developed for managing the database, it covers all the functionalities to manage the Oracle NoSQL Database. Various CRUD functions i.e. create, read, update, delete are included in this project. All the functions are based upon key values. It also includes importing data form CSV file to database and exporting the data to CSV file. RESTFul web services are developed for particular purpose. Web services are been developed for attendance management system. This web services work upon the sensor data which is been stored into the database. The sensor data is generated from bio metrics thumb machine which is used into many organization for employee's attendance purpose. The web services evaluate the information needed to the user and sends to android based device. This android application is specially developed to demonstrate the consumption of this web services and use of Oracle NoSQL for managing Big Data. The Android Application has two options i.e. Daily attendance and Monthly Attendance.

1.3 Problem Definition

"Develop a tool for managing Key Value pairs on Oracle NoSQL Database and Web Services to be used on any platform."

The attempt is develop and GUI based application to handle the Oracle NoSQL database which will help all class of users to use Oracle NoSQL. Basically the application is for users who only need to manage the Oracle NoSQL database and not the developers. The application includes all the basic functionalities.

The second part of the project is web services which enables any class of user to

use these project functionalities into their application. The project also demonstrates the efficient use to Oracle NoSQL for managing the big data.

1.4 Organization of Report

- **Chapter 1** : Introduction It contains social and technical scenario, introduction about topic(A Tool for Managing Key-Value Pairs on Oracle NoSQL Database.), basic concept(Provide GUI to perform CRUD as well as Import Export Operation on NoSQL KVStore). Project work, Objective etc. are discussed in this chapter.

- **Chapter 2** : Literature Survey A review of the related work in the area of Database Management System is presented in this chapter. The existing approaches to database management on BigData is discussed first, different types of databses are also discussed and finally literature survey is concluded. Also proposed system is discussed.

- **Chapter 3** : Software Requirement Specification. It includes introduction, System features, External and internal interface, and Non-functional, other requirements, Goals, objective, Need ,System Implementation Plan, Analysis Model, hardware and software interfaces.

- **Chapter 4** : System Design It contain System Architecture, Mathematical Model, UML Diagrams are included in this chapter.

- **Chapter 5** : Technical Specification it contains details of technology that are used in project i.e details about java7, java servlet 3.0, glassfish server 4.0, android sdk, html 5, restful webservices etc.

- **Chapter 6** : Project Estimate , schedule, Team Structure. It contains details about project estimation cost, scheduling i.e which task is done by which member. It also include Team structure and details about duration required to complete Project.

- **Chapter 7** : Software Implementation It gives detail about Algorithm which has been followed for the implementation of system. It also give detail about Mathematical Model.

- **Chapter 8** : Software Testing This chapter gives introduction to Testing, introduction to Types of Testing. It also gives test cases that are used in Project for Testing and snapshot of test cases and Test Plan

- **Chapter 9** : Result This chapter contain snapshot of system which gives details about how system flow going on and final output is displayed

- **Chapter 10** : Deployment And Maintenance This chapter gives detail about how to Install or Un-install our system. It includes snapshot of process of Installation or Un-installation of our system

- **Chapter 11** : Conclusion It contains conclusion and Future scope

Chapter 2

Literature Survey

2.1 Database

A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies. Database management systems (DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker Pro. A database is not generally portable across different DBMSs, but different DBMSs can interoperate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one database. [1]

Choosing between databases used to boil down to examining the differences between the available commercial and open source relational databases. The term "database" had become synonymous with SQL, and for a while not much else came close to being a viable solution for data storage. But recently there has been a shift in the database landscape. When considering options for data storage, there is a new game in town: NoSQL databases [2].

2.2 NoSQL

NoSQL databases represent a recent evolution in enterprise application architecture, continuing the evolution of the past twenty years. In the 1990's, vertically integrated

applications gave way to client-server architectures, and more recently, client-server architectures gave way to three-tier web application architectures. In parallel, the demands of web-scale data analysis added map-reduce processing into the mix and data architects started avoiding transactional consistency in exchange for incremental scalability and large-scale distribution. The NoSQL movement emerged out of this second ecosystem. NoSQL is often characterized by what it's not depending on whom you ask, it's either not only a SQL-based relational database management system or it's simply not a SQL-based RDBMS. While those definitions explain what NoSQL is not, they do little to explain what NoSQL is. Consider the fundamentals that have guided data management for the past forty years [3].

In recent years there was a high demand for massively distributed databases with high partition tolerance but according to the CAP theorem it is impossible for a distributed system to simultaneously provide consistency, availability and partition tolerance guarantees. A distributed system can satisfy any two of these guarantees at the same time, but not all three. For that reason many NoSQL databases are using what is called eventual consistency to provide both availability and partition tolerance guarantees with a maximum level of data consistency. NewSQL is a class of modern relational databases that aims to provide the same scalable performance of NoSQL systems for online transaction processing (read-write) workloads while still using SQL and maintaining the ACID guarantees of a traditional database system. Such databases include Clustrix, EnterpriseDB, NuoDB and VoltDB [1].

The only thing that all NoSQL solutions providers generally agree on is that the term "NoSQL" isn't perfect, but it is catchy. Most agree that the "no" stands for "not only", an admission that the goal is not to reject SQL but, rather, to compensate for the technical limitations shared by the majority of relational database implementations. In fact, NoSQL is more a rejection of a particular software and hardware architecture for databases than of any single technology, language, or product. Relational databases evolved in a different era with different technological constraints, leading to a design that was optimal for the typical deployment prevalent at that time. But times have changed, and that once successful design is now a limitation. You might hear conversations suggesting that a better term for this category is NoRDBMS or half a dozen other labels, but the critical thing to remember is that NoSQL solutions started off with a different set of goals and evolved in a different environment, and so they are operationally different and, arguably, provide better suited solutions for many of today's data storage problems[2].

NoSQL emerged as companies, such as Amazon, Google, LinkedIn and Twitter struggled to deal with unprecedented data and operation volumes under tight latency constraints. Analyzing high-volume, real time data, such as web-site click streams, provides significant business advantage by harnessing unstructured and semi-structured data sources

to create more business value. Traditional relational databases were not up to the task, so enterprises built upon a decade of research on distributed hash tables (DHTs) and either conventional relational database systems or embedded key/value stores, such as Oracle's Berkeley DB, to develop highly available, distributed key-value stores. Although some of the early NoSQL solutions built their systems atop existing relational database engines, they quickly realized that such systems were designed for SQL-based access patterns and latency demands that are quite different from those of NoSQL systems, so these same organizations began to develop brand new storage layers. In contrast, Oracle's Berkeley DB product line was the original key/value store; Oracle Berkeley DB Java Edition has been in commercial use for over eight years. By using Oracle Berkeley DB Java Edition as the underlying storage engine beneath a NoSQL system, Oracle brings enterprise robustness and stability to the NoSQL landscape.[3]

The next generation of post-relational databases in the 2000s became known as NoSQL databases, including fast key-value stores and document-oriented databases. NoSQL databases are often very fast, do not require fixed table schemas, avoid join operations by storing denormalized data, and are designed to scale horizontally. The most popular NoSQL systems include MongoDB, Couchbase, Riak, memcached, Redis, CouchDB, Hazelcast, Apache Cassandra and HBase, which are all open-source software products.

2.2.1 Why NoSQL

NoSQL databases first started out as in-house solutions to real problems in companies such as Amazon Dynamo, Google BigTable, LinkedIn Voldemort, Twitter FlockDB, Facebook Cassandra, Yahoo! PNUTS, and others. These companies didn't start off by rejecting SQL and relational technologies; they tried them and found that they didn't meet their requirements. In particular, these companies faced three primary issues: unprecedented transaction volumes, expectations of low-latency access to massive datasets, and nearly perfect service availability while operating in an unreliable environment. Initially, companies tried the traditional approach: they added more hardware or upgraded to faster hardware as it became available. When that didn't work, they tried to scale existing relational solutions by simplifying their database schema, de-normalizing the schema, relaxing durability and referential integrity, introducing various query caching layers, separating read-only from write-dedicated replicas, and, finally, data partitioning in an attempt to address these new requirements. Although each of these techniques extended the functionality of existing relational technologies, none fundamentally addressed the core limitations, and they all introduced additional overhead and technical tradeoffs. In other words, these were good band aids but not cures.[2]

A major influence on the eventual design of NoSQL databases came from a dramatic shift in IT operations. When the majority of relational database technology was designed, the predominant model for hardware deployments involved buying large servers attached to dedicated storage area networks (SANs). Databases were designed with this model in mind: They expected there to be a single machine with the responsibility of managing the consistent state of the database on that system's connected storage. In other words, databases managed local data in files and provided as much concurrent access as possible given the machine's hardware limitations. Replication of data to scale concurrent access across multiple systems was generally unnecessary, as most systems met design goals with a single server and reliability goals with a hot stand-by ready to take over query processing in the event of master failure. Beyond simple failover replication, there were only a few options, and they were all predicated on this same notion of completely consistent centralized data management. Technologies such as two-phase commit and products such as Oracle's RAC were available, but they were hard to manage, very expensive, and scaled to only a handful of machines. Other solutions available included logical SQL statement-level replication, single-master multi-replica log-based replication, and other home-grown approaches, all of which have serious limitations and generally introduce a lot of administrative and technical overhead. In the end, it was the common architecture and design assumptions underlying most relational databases that failed to address the scalability, latency, and availability requirements of many of the largest sites during the massive growth of the Internet [2].

Given that databases were centralized and generally running on an organization's most expensive hardware containing its most precious information, it made sense to create an organizational structure that required at least a 1:1 ratio of database administrators to database systems to protect and nurture that investment. This, too, was not easy to scale, was costly, and could slow innovation. A growing number of companies were still hitting the scalability and performance wall even when using the best practices and the most advanced technologies of the time. Database architects had sacrificed many of the most central aspects of a relational database, such as joins and fully consistent data, while introducing many complex and fragile pieces into the operations puzzle. Schema devolved from many interrelated fully expressed tables to something much more like a simple key/value look-up. Deployments of expensive servers were not able to keep up with demand. At this point these companies had taken relational databases so far outside their intended use cases that it was no wonder that they were unable to meet performance requirements. It quickly became clear to them that they could do much better by building something in-house that was tailored to their particular workloads. These in-house custom solutions are the inspiration behind the many NoSQL products we now see on the market [2].

Furthermore, until recently, integrating NoSQL solutions with an enterprise application architecture required manual integration and custom development; Oracle's NoSQL Database provides all the desirable features of NoSQL solutions necessary for seamless integration into an enterprise application architecture [3].

2.3 Types of NoSQL Databases

In isolation, NoSQL databases are enhanced for retrieve and append operations and also offer functionality beyond record storage. NoSQL databases shows their strong suit with regard to the elastic handling of variable data by document-oriented databases, the graph databases for representing relationships and in the reduction of a database to a container with KVpairs provided by key-value databases.
The four types of NoSQL Databases are as follows :

- Graph Databases
- Document Databases
- Columnar Databses
- key-Values Stores

2.3.1 Graph Databases

used which, again, can scale across multiple machines. NoSQL databases do not provide a high-level declarative query language like SQL to avoid overtime in processing. Rather, querying these databases is data-model specific. Many of the NoSQL platforms allow for RESTful interfaces to the data, while other offer query APIs. Examples are Neo4J, InfoGrid, Infinite Graph [4]. Looking at the projection of domain models onto a data structure, there are two dominating schools - the relational way as used for RDBMS and graph and network structures, used for e.g. the Semantic Web. While graph structures in theory are normalizable even in RDBMS, this has serious query performance implications for recursive structures like for instance file trees and network structures like e.g. social graphs, due to the implementation characteristics of relational databases. Every operation over a relationship of a network results in a "join" operation in the RDBMS, implemented as a set-operation between the sets of primary keys for two tables - a slow operation and not scalable over growing numbers of tuples in these tables [5].

There is no existing general consensus on terminology regarding the area of graphs. There exist many different types of graph models. However, there is some effort to create the

Property Graph Model, unifying most of the different graph implementations. According to it, information in a Property Graph is modeled using three basic building blocks:

- node (a.k.a. vertex)
- relationship (a.k.a. edge) - with direction and Type (labeled and directed)
- property(a.k.a attribute) on nodes and relationships

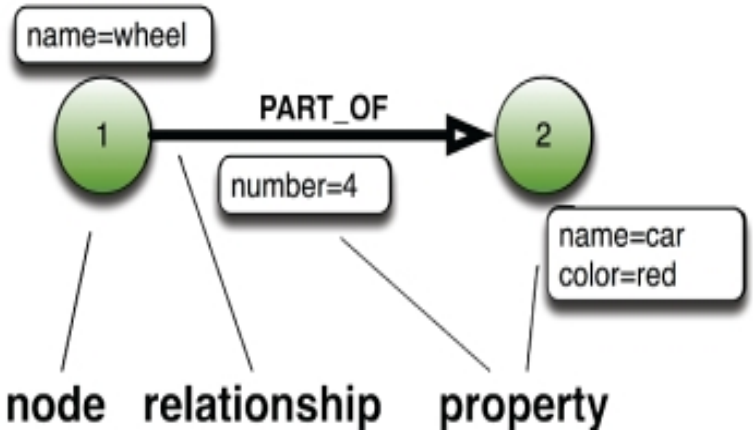

Figure 2.1: Basic terminology for labeled property graphs

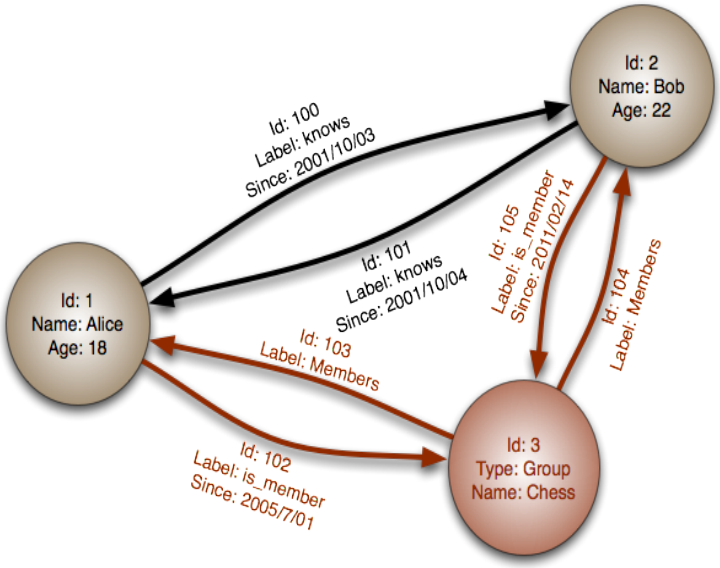

Figure 2.2: labeled property graph.

- **For Example**

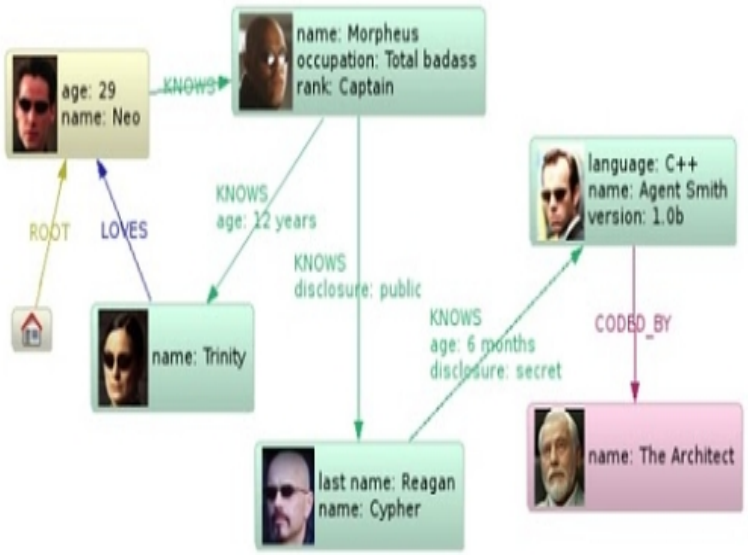

Figure 2.3: Graph Database exapmle.

As mentioned before, Social Networks represent just a tiny fraction of the applications of graph databases, as they are easy to understand for this example.

2.3.2 Document Databases

These were inspired by Lotus Notes and are similar to key-value stores. The model is basically versioned documents that are collections of other key-value collections. The semi-structured documents are stored in formats like JSON. Document databases are essentially the next level of Key/value, allowing nested values associated with each key. Document databases support querying more efficiently. Examples are CouchDB, MongoDb [4].

The central concept of a document-oriented database is the notion of a Document. While each document-oriented database implementation differs on the details of this definition, in general, they all assume documents encapsulate and encode data (or information) in some standard formats or encodings. Encodings in use include XML, YAML, JSON, and BSON, as well as binary forms like PDF and Microsoft Office documents (MS Word, Excel, and so on). Documents inside a document-oriented database are similar, in some ways, to records or rows in relational databases, but they are less rigid. They are not required to adhere to a standard schema, nor will they have all the same sections, slots, parts, or keys. For example, the following is a document: {

FirstName: "Bob",

Address: "5 Oak St.",

```
    Hobby: "sailing"
}
```

A second document might be:

```
{
    FirstName: "Jonathan",
    Address: "15 Wanamassa Point Road",
    Children: [
    {Name: "Michael", Age: 10},
    {Name: "Jennifer", Age: 8},
    {Name: "Samantha", Age: 5},
    {Name: "Elena", Age: 2}
    ]
}
```

These two documents share some structural elements with one another, but each also has unique elements. Unlike a relational database where every record contains the same fields, leaving unused fields empty; there are no empty 'fields' in either document (record) in the above example. This approach allows new information to be added to some records without requiring that every other record in the database share the same structure [6].

2.3.3 Columnar Databases

These were created to store and process very large amounts of data distributed over many machines. There are still keys but they point to multiple columns. The columns are arranged by column family. Examples are Cassandra, HBase [4].
A column family is a NoSQL object that contains columns of related data. It is a tuple (pair) that consists of a key-value pair, where the key is mapped to a value that is a set of columns. In analogy with relational databases, a column family is as a "table", each key-value pair being a "row". Each column is a tuple (triplet) consisting of a column name, a value, and a timestamp. In a relational database table, this data would be grouped together within a table with other non-related data [7].
Two types of column families exist:

- Standard column family: contains only columns
- Super column family: contains a map of super columns.

2.3.4 Key-Value Store Databases

The main idea here is using a hash table where there is a unique key and a pointer to a particular item of data. The Key/value model is the simplest and easiest to implement. But it is inefficient when you are only interested in querying or updating part of a value, among other disadvantages. Examples: Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB, Amazon SimpleDB, Riak [4].

Key-value stores allow the application to store its data in a schema-less (key, value) pairs. This data is stored in a hash table like data-types, so that each value can be accessed by its major or minor key. Even though such storage facility might not be much effective as they provide single way to access the values, but excludes the need for a fixed data model.

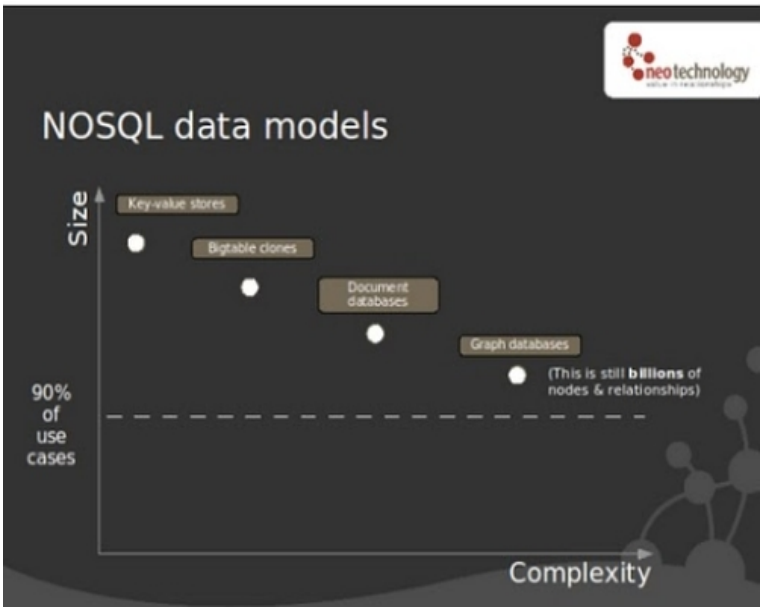

Figure 2.4: NoSQL Data Models

- *Oracle Key Value Pair NoSQL Database*

Oracle NoSQL Database provides multi-terabyte distributed key/value pair storage that offers scalable throughput and performance. That is, it services network requests to store and retrieve data which is organized into key-value pairs. Oracle NoSQL Database services these types of data requests with a latency, throughput, and data consistency that is predictable based on how the store is configured. Oracle NoSQL Database offers full Create, Read, Update and Delete (CRUD) operations with0 adjustable durability guarantees.

Oracle NoSQL Database is designed to be highly available, with excellent throughput and latency, while requiring minimal administrative interaction.

Oracle NoSQL Database provides performance scalability. If you require better performance, you use more hardware. If your performance requirements are not very steep, you can purchase and manage fewer hardware resources. Oracle NoSQL Database is meant for any application that requires network-accessible key-value data with user-definable read/write performance levels. The typical application is a web application which is servicing requests across the traditional three-tier architecture: web server, application server, and back-end database. In this configuration, Oracle NoSQL Database is meant to be installed behind the application server, causing it to either take the place of the back-end database, or work alongside it. To make use of Oracle NoSQL Database, code must be written (using Java or C) that runs on the application server. An application makes use of Oracle NoSQL Database by performing network requests against Oracle NoSQL Database's key-value store, which is referred to as the KVStore. The requests are made using the Oracle NoSQL Database Driver, which is linked into your application as a Java library (.jar file), and then accessed using a series of Java APIs.

- The KVStore :

The KVStore is a collection of Storage Nodes which host a set of Replication Nodes. Data is spread across the Replication Nodes. The store contains multiple Storage Nodes. Every Storage Node hosts one or more Replication Nodes as determined by its capacity.

- Replication Nodes and Shards :

At a very high level, a Replication Node can be thought of as a single database which contains key-value pairs. Replication Nodes are organized into shards. A shard contains a single Replication Node, called the master node, which is responsible for performing database writes. The master node copies those writes to the other Replication Nodes in the shard, called the replicas. These replicas obtain a full copy of the data from the corresponding master node and are used to service read-only operations. Although there can be only one master node at any given time, any of the members of the shard are capable of becoming a master node.

- Replication Factor :

The number of nodes belonging to a shard is called its Replication Factor. The larger a shard's Replication Factor, the faster its read throughput (because there are more machines to service the read requests) but the slower its write performance (because there are more machines to which writes must be copied).

- Partitions :

Each shard contains one or more partitions. Key-value pairs in the store are organized according to the key. Keys, in turn, are assigned to a partition. Once a key is placed in a partition, it cannot be moved to a different partition.

- KVLite :

KVLite is a simplified version of Oracle NoSQL Database. It provides a single-node store that is not replicated. It runs in a single process without requiring any administrative interface. You configure, start, and stop KVLite using a command line interface.

- Architecture of Key Value Pair NoSQL Database

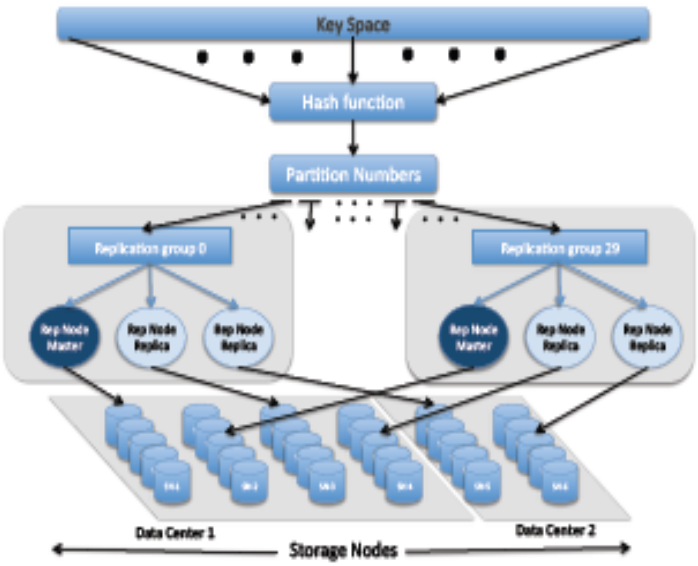

Figure 2.5: Architecture of Key Value Pair NoSQL Database

- The CAP Theorem of NoSQL Database

Despite the high demand in recent years for massively distributed databases with high partition fault-tolerance, the CAP theorem stipulates that it is actually impossible for a distributed system to provide consistency, availability and partition fault-tolerance guarantees simultaneously; a distributed system can satisfy at most any two of these guarantees at the same time, but not all three. These guarantees can be understood as follows:

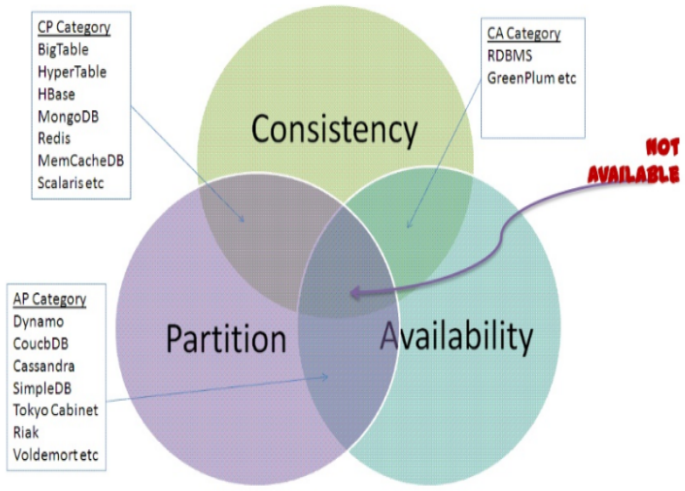

Figure 2.6: CAP Theorem of NoSQL

Consistency : Concurrently executing queries see the same valid and consistent data at the same time.
Availability : This is a guarantee that every request receives a response about whether it succeeded or failed.
Partition-tolerance : Also known as fault-tolerance, this is a guarantee that the system continues to operate despite arbitrary message loss.

Because no distributed system is capable of satisfying all three guarantees at the same time, a trade-off must be made. While traditional databases make that decision for us, NoSQL databases provide these guarantees as tuning options. Database vendors must always decide which two to prioritize. The options are as follows:

1. Availability is compromised in favour of consistency and partition-tolerance.
2. Partition-tolerance is forfeited in favour of consistency and availability.
3. Consistency is compromised but systems are always available and can work when parts are partitioned.

Traditional SQL databases place a high priority on consistency and fault-tolerance and have generally as a result chosen to go with the first option above and forfeit high availability. NoSQL databases frequently leave that decision to the application operations team and provide configuration options so that the preferred options can be chosen based on the application use case

- NoSQL in Practice

There are many products that now claim to be part of the NoSQL database market, far too many to mention here or describe in any detail.

	MongoDB	**CouchDB**	**Riak**	**Redis**	**Voldemort**	**Cassandra**	**HBase**
Language	C++	Erlang	Erlang	C++	Java	Java	Java
License	AGPL	Apache	Apache	BSD	Apache	Apache	Apache
Model	Document	Document	Key/value	Key/value	Key/value	Wide Column	Wide Column
Protocol	BSON	HTTP/REST	HTTP/REST or TCP/ Protobufs	TCP		TCP/Thrift	HTTP/REST or TPC/Thrift
Storage	Memory mapped b-trees	COW-BTree	Pluggable: InnoDB, LevelDB, Bitcask	In memory, snapshot to disk	Pluggable: BDB, MySQL, in-memory	Memtable/ SSTable	HDFS
Inspiration		Dynamo	Dynamo		Dynamo	BigTable, Dynamo	BigTable
Search	Yes	No	Yes	No	No	Yes	Yes
MapReduce	Yes	No	Yes	No	No	Yes	Yes

Figure 2.7: NoSQL in Practice

2.4 Characteristics of NoSQL Databases

- Auto-sharding - A NoSQL database automatically spreads data across servers, without requiring applications to participate. Servers can be added or removed from the data layer without application downtime, with data (and I/O) automatically spread across the servers. Most NoSQL databases also support data replication, storing multiple copies of data across the cluster, and even across data centers, to ensure high availability and support disaster recovery. A properly managed NoSQL database system should never need to be taken offline, for any reason, supporting 24x365 continuous operations of applications.

- Unstructured Database.

- Distributed query support - "Sharding" a relational database can reduce, or eliminate in certain cases, the ability to perform complex data queries. NoSQL database systems retain their full query expressive power even when distributed across hundreds of servers.

- Integrated caching - To reduce latency and increase sustained data throughput, advanced NoSQL database technologies transparently cache data in system memory. This behavior is transparent to the application developer and the operations team, compared to relational technology where a caching tier is usually a separate infrastructure tier that must be developed to, deployed on separate servers, and explicitly managed by the ops team.

2.5 Conclusion

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

The purpose of this section is to describe "A Tool For Managing Key Value Pairs on Oracle NoSQL". This document contains the functional and non-functional requirements of the project. The recent problem of exponentially growing data and to store and manage is needed to be solved. As a solutions new technologies are been introduced in the market by various vendors. But again using these technologies needs trainings to users. As these technologies does not resemble in any way with the traditional technologies. So the purpose is to make this new introduced Oracle NoSQL, familiar with the user by providing various tools to manage this database for user's purpose.

3.1.2 Project Scope

A Tool for Managing Key Value Pairs on Oracle NoSQL is basically a application through which the user can make fundamental operations on Oracle NoSQL Database. Project provides various CRUD functions and Import and Export. Here user has to provide the key and its value either one by one or into chunk from CSV File. The project also provides RESTFul web services which provide all the functionalities for managing Oracle NoSQL Database. The project has an Android Application which is used for Attendance check for the users of particular organization. The application can be used in:

1. Performing basic operations on Oracle NoSQL Database.
2. Into an organization where Oracle NoSQL is used.
3. The RESTFul web services can be used by any application which needs to use the Oracle NoSQL Database functionalities.

4. The Attendance check application can be used by organizations having Biometric thumb scan machine for attendance.

3.1.3 Need

The system is needed because Oracle NoSQL database does not have any Command Line Interface or Graphical User Interface for managing the database. So the system provides GUI and RESTFul web services which make the ease to user, to handle the Oracle NoSQL database. The RESTFul web services are intended to provide project functionalities on any platform or any class of user.

3.1.4 Benefits

It is stated that any database should provide ease to users to make operations effort less and efficient. So this project is intended to reduce the user's efforts for managing Oracle NoSQL database by providing GUI and RESTFul web services for the same.

3.1.5 Objectives

The objective behind implementing this application is to make a tool which could make the interaction of user easier with Oracle NoSQL database. This application is is new access mechanism for managing database. The application provides interactive and intuitive interface to the user. The Project also provides RESTFul web services which can be used on any platform who needs to make use of Oracle NoSQL Database.

3.1.6 User Classes and Characteristics

Users of the system might be any person who needs to interact or use the Oracle NoSQL Database. The user must have basic knowledge of the Key Value Pair based Oracle NoSQL database and computer. Administrators of the system must have more knowledge about the internal modules of the system and are able to rectify small problems that may arise due to disk crashes, network failure, power failure and other catastrophes. Friendly user interface, help options and user guide must be sufficient to educate the user on how to use this product without any problem or difficulties.

3.1.7 Operating Environment

The project is an online application, which is deployed on the application server. So the application can be accessed and used from any platform having internet connectivity. The RESTFul web services are free to use on any platform, but mostly the RESTFul web

services are intended to use on mobile platform or on the platform where low data is to be handled. And the attendance check application works on Android Platform.

3.1.8 Assumptions and Dependencies

- The application is intended to be used for managing big data, the user must have fast internet connectivity on his/ her system.
- The user who wants to use the RESTFul web services must know how to use the web services on mobile or any other platform.
- The most important is that, any user who needs to use the system must have basic knowledge of the Oracle NoSQL Database.

3.2 System Features

- Connect To Store

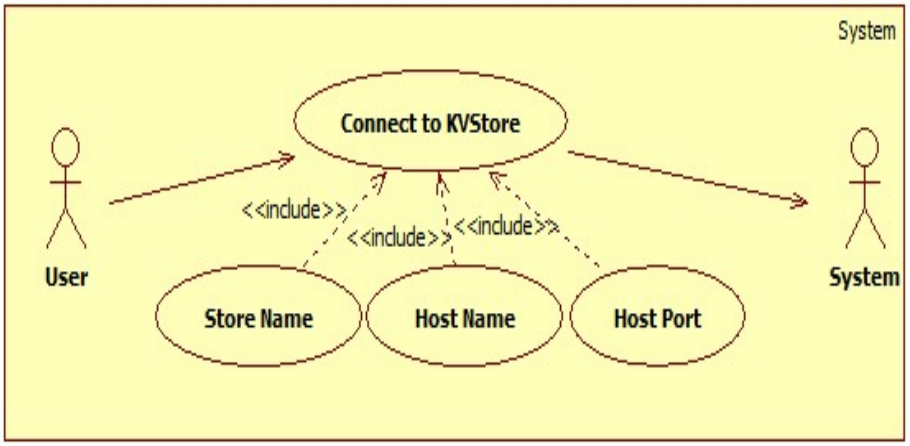

Figure 3.1: Connect To Store

- Insert Data Into Store

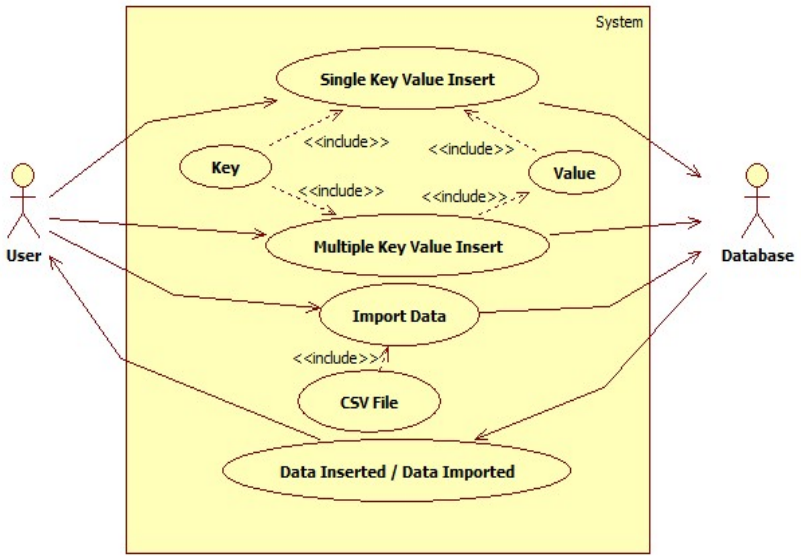

Figure 3.2: Insert Data Into Store

- Display Data

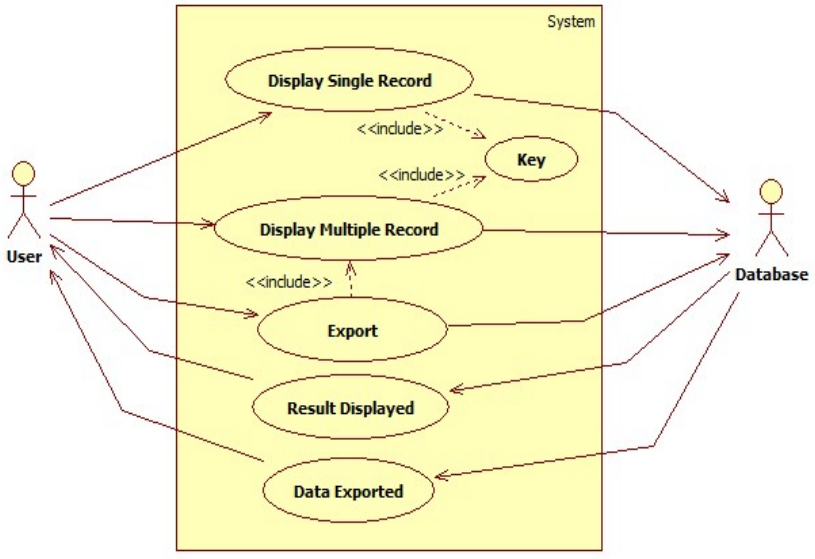

Figure 3.3: Display Data

- Delete Data

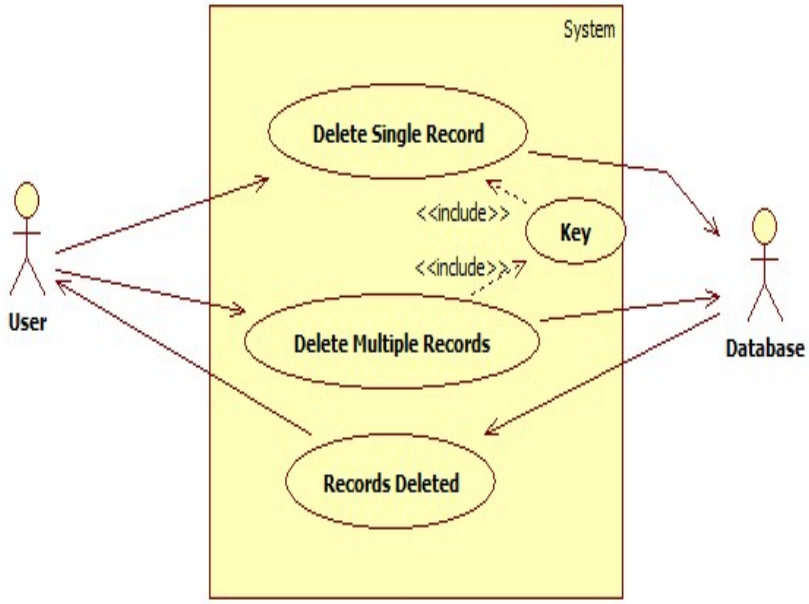

Figure 3.4: Delete Data

- Update Data

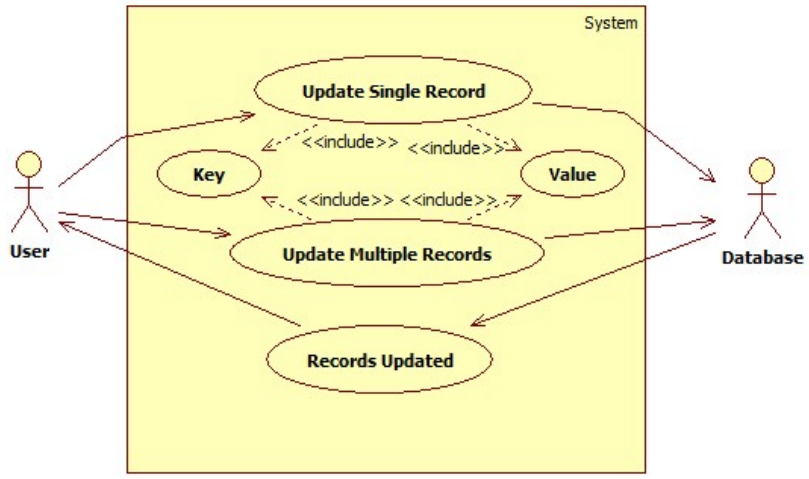

Figure 3.5: Update Data

- Import/Export

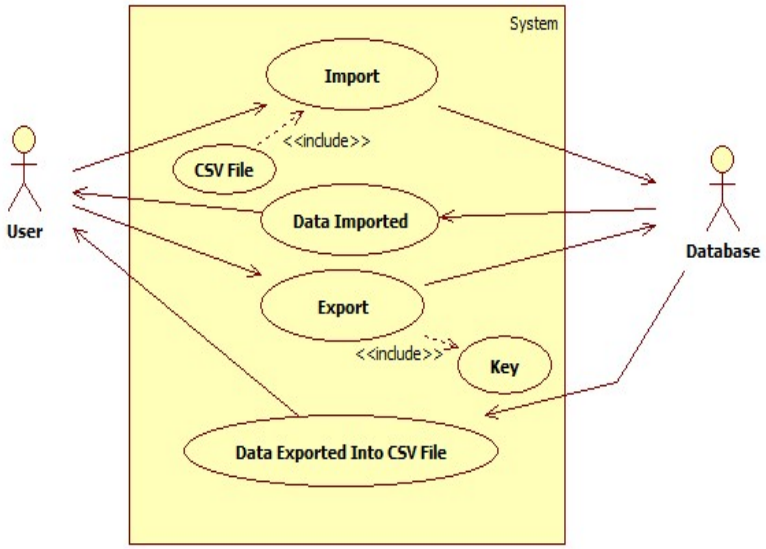

Figure 3.6: Import/Export

- Daily Attendance Check

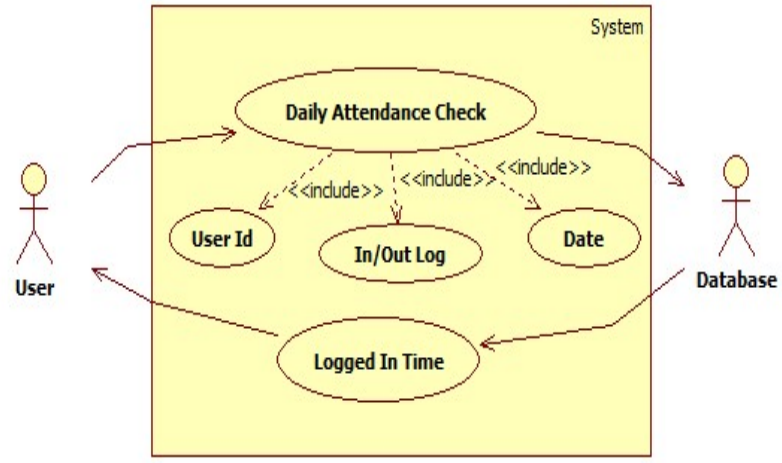

Figure 3.7: Daily Attendance Check

- Monthly Attendance Check

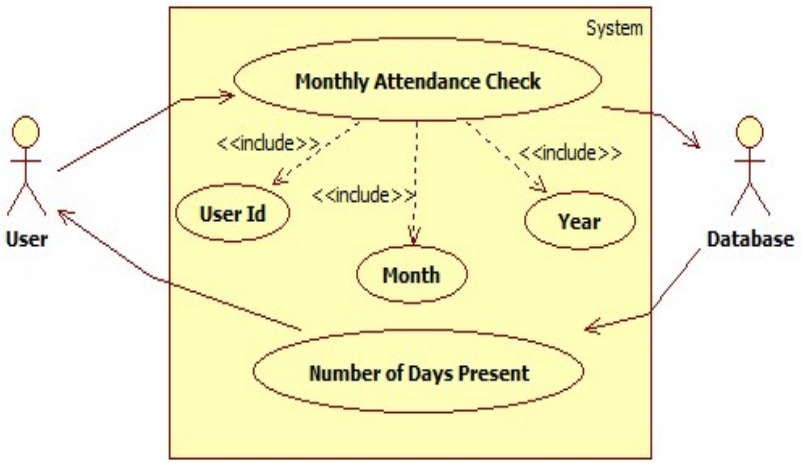

Figure 3.8: Monthly Attendance Check

3.3 External Interface Requirements

3.3.1 User Interfaces

- Home Screen :

After starting the application home screen should appears. Home screen has various options including insert, display, update, delete, import and export. But for performing any of this operation the application needs to be connected to the KVStore. For this purpose application asks to user for three options i.e. Store Name, Host Name on which the database is installed and the Host Port on which the database service is running. After getting these parameters the application would connect to database server. And the session for that user should be initialized.

- Insert Record :

Using this option the user can insert either single Key Value pair or Multiple Key Value pairs into the database. For single Key Value Pair insertion user needs to enter Major Key Component, Minor Key Component and Value. And for multiple Key Value Pairs insertion first user needs to enter Major Key Component and number of Minor Key Component for same, followed by this number of Minor Key Components would have to be inserted and Values for them respectively.

- Display Record :

This option is used for either single or multiple Key Value Pair display. For single Key Value Pair display user needs to enter the Major Key Component and Minor Key Component of which the value is to be displayed. And for multiple Key Value pairs display user only needs to enter Major Key Component.

- Delete Record :

This option is used to delete single or multiple Key Value pair from database. For single value deletion user needs to enter Major Key Component and Minor Key Component and for deletion of multiple records user has to enter Major Key Component only.

- Update Record :

Using this option the user can update the existing records into database. For updating single key value pair user has to enter Major Key Component, Minor Key Component and Value for them. And for updating multiple Key Value pair user has to enter Major Key Component its Minor Key Components and their respected values.

- Import :

This option should provide the functionality of importing data from CSV file to Oracle NoSQL Database. Here very firstly a CSV file is uploaded to the server. And after that user has to specify Major Key Component and Minor Key Component for that file and user also has to specify that which field from the file is to be relate with Minor Key Component.

- Export :

Here the user can export the data to CSV file from the database. Very firstly the user has to specify Major Key Component by which he/she would get multiple records and after that user can download that file.

- Android Interface for Attendance Check :

Using this application the user can check daily as well as monthly attendance. Here the user first needs to connect to database and after that user has to specify userid, In/Out log and date for daily attendance check, and userid, month and year for monthly attendance check.

3.3.2 Hardware Interfaces

- Intel Core 2 Duo or higher processor.
- Android device with Ginger bread version or higher.
- Android device with more than 125 mb RAM.
- For deployment of server application the server should be good configuration.

3.3.3 Software Interfaces

- Any operating system where java can be installed.
- Apache tomcat or Glassfish server.
- Any Internet browser.
- Android operating system.

3.4 Nonfunctional Requirements

3.4.1 Performance Requirements

- The server must be able to handle all the requests incoming from the clients.
- Network bandwidth must be properly managed while the big CSV file is uploaded to the server.
- While retrieving multiple Key Value from the database server, the records must be retrieved into batches so that network would not get jammed.
- On android device the application must not take more space into internal memory and RAM.
- The client must operate at less internet speed.

3.4.2 Safety Requirements

- UPS: in case of power failure the system requires UPS.
- Hard Disk: the database server must have enough space to store the Data.
- The network at the database server side must be properly managed so that it should not crash during the operations are in progress.

3.4.3 Software Quality Attributes

- Data Security : The data stored at the database server is safe as the Oracle NoSQL takes care of it.
- Adaptability : the system must be able to adapt the changes made onto the device. In case if the system software is updated then the application must also adapt those changes.

- Availability : availability defines that whenever the application is needed that time user must get the service.
- Correctness : every time the data is accessed that time correct data should be provided.
- Flexibility : the application should be flexible with any class of user.
- Maintainability : the system should not need more maintenance as the end user is not expertise to do so.
- Reliable: the user can be reliable on the system, as the system is reliable.
- Reusability: the code of application is reusable. The RESTFul web can be reused into other applications.
- Robustness: the application can be operate into any device environment.
- Testability: the system can be exposed to the test all its functionalities.

3.5 Analysis Model

3.5.1 Class Diagrams

Class diagram in the unified modeling language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes ,and the relationship between the classes .It is the main building block in object oriented modeling. It is being used for both general conceptual modeling of the systematic of the application, and for detailed modeling translating into programming code

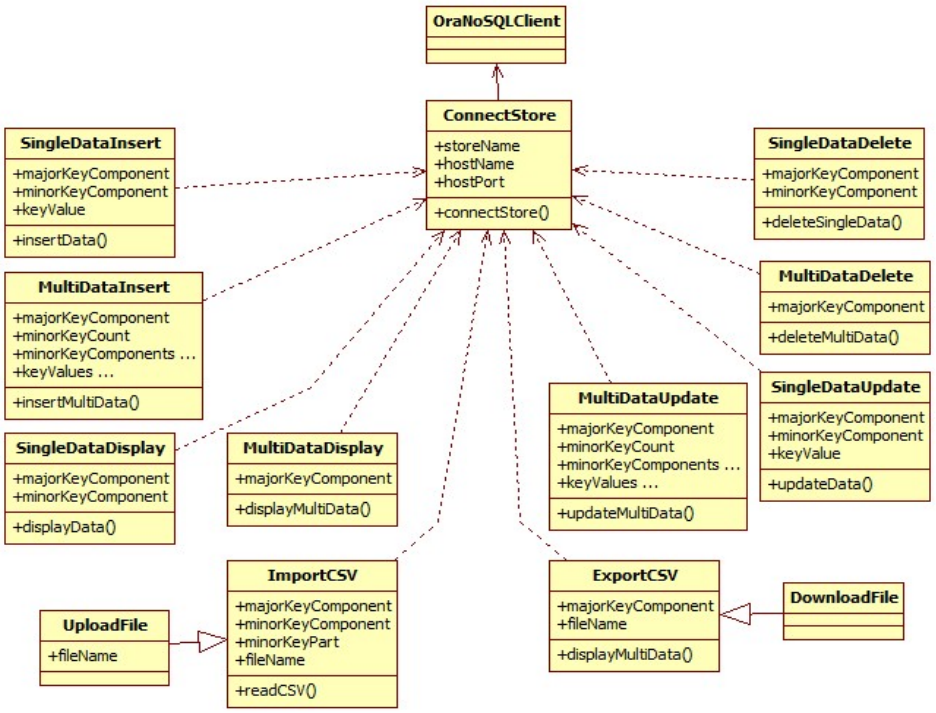

Figure 3.9: Class Diagrams

3.6 System Implementation Plan

There are many important phases for implementation. Following are some important phases and task of implementation.

- The main phase to study the NoSQL standard.
- Followed by that to Study the Oracle NoSQL and its Architecture in detail.
- Understand the use case for which the system is to be implemented.
- After understanding the use case proper plan is to created.
- So according to plan very first the GUI is to be decided. Various functions will be decided which are to be performed by the system. Based on this function further implementation will be done.
- Taking consideration the implementation phase, all the necessary software required for implementation should be installed on the machine.
- After the GUI based application is completed, test its functionalities. If all functionalities are up to the mark then proceed to the next phase or remove the errors or bugs.

- The next phase is to implement the RESTFul web services. For this purpose, first understand the use case and accordingly start the implementation.
- Test the web services, if they are according to use case they move further or again check the functionalities and correct it.
- Next step is to implement the Android Application which will consume the RESTFul web services.
- After completing the entire above tasks the application is ready to be used.

3.6.1 Description of Implementation

- The system must be developed into phases as described above.
- The system should follow prototype model.
- Here none of the prototype depends upon each other, so can be developed in parallel. Development in parallel will reduce the time to develop the project and also the implementation cost.

Chapter 4

System Design

4.1 System Architecture

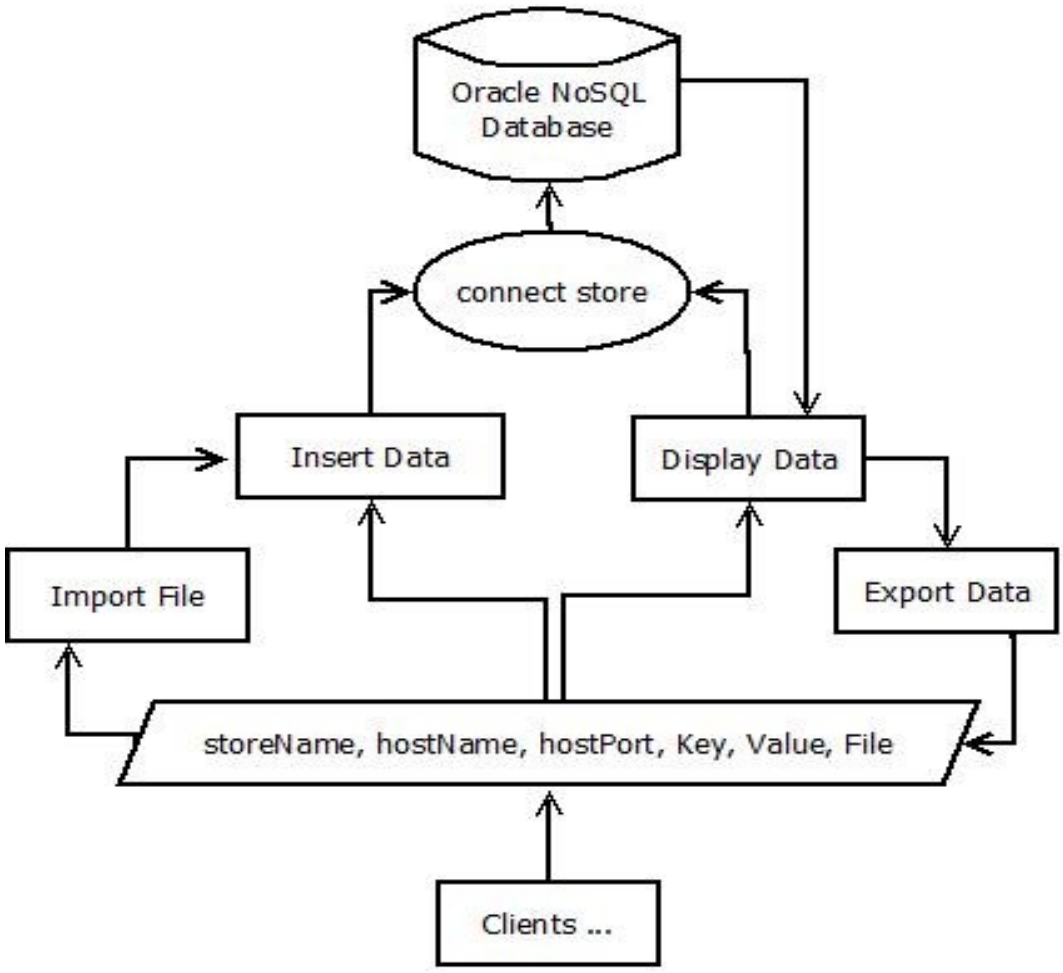

Figure 4.1: System Architecture

Brief summary of the software architecture is described below in the form of following UML Diagrams :

1. State transition diagram
2. Sequence diagram
3. Collaboration diagram
4. Package diagram
5. Component diagram
6. Deployment diagram

State transition diagram

A state machine diagram models the behavior of a single object specifying the sequence of an event that an object goes through during lifetime in response to events. Elements used in state diagram are Initial state, state, transition, history state, final state, signals etc. The most common purpose for which you will use state machines is to model the lifetime of an object, especially instance of classes, use cases and the system as a whole.

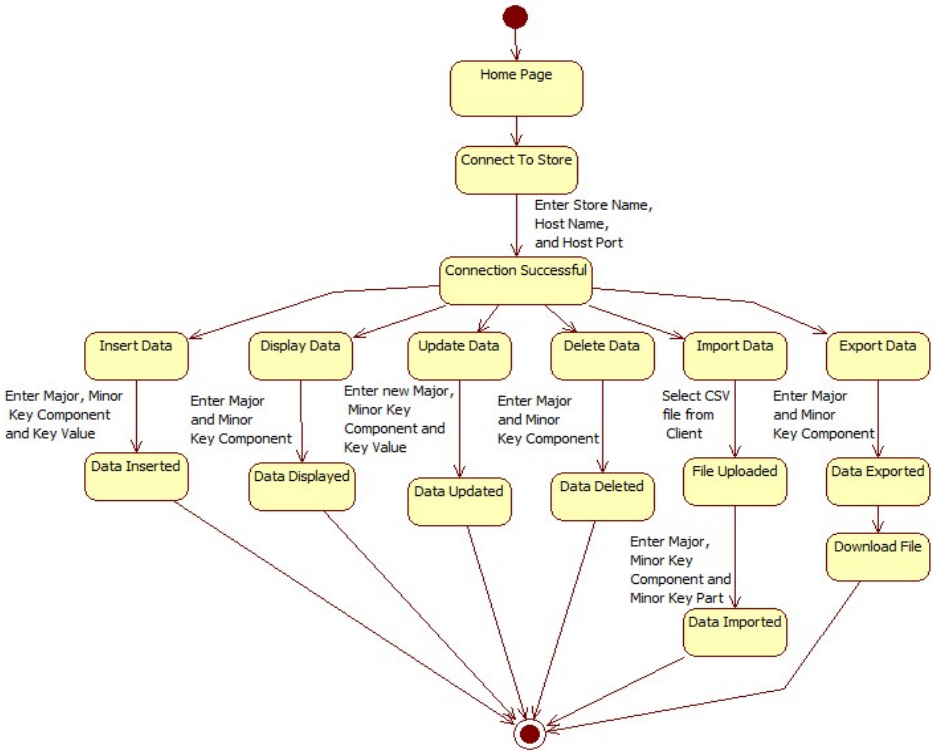

Figure 4.2: State Transition For User

- GUI based System :

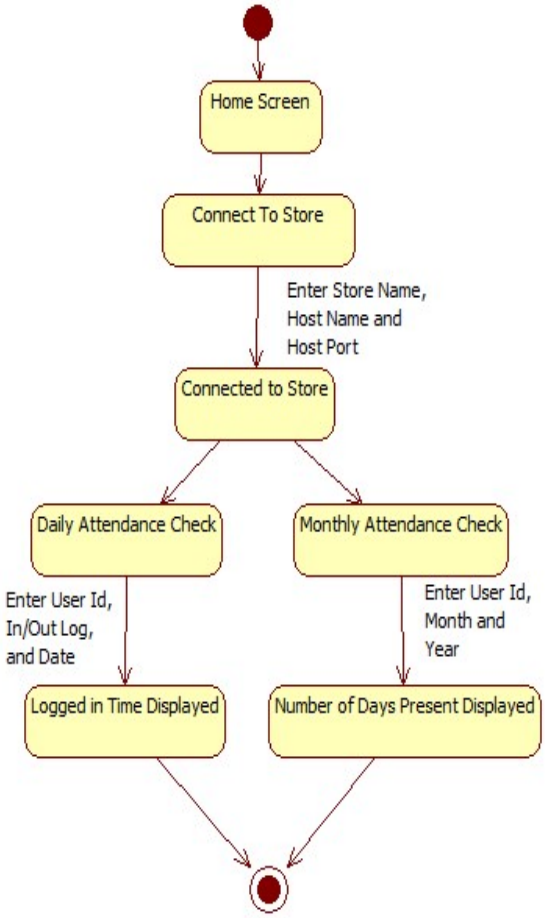

Figure 4.3: Attendance check System

Sequence diagram

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

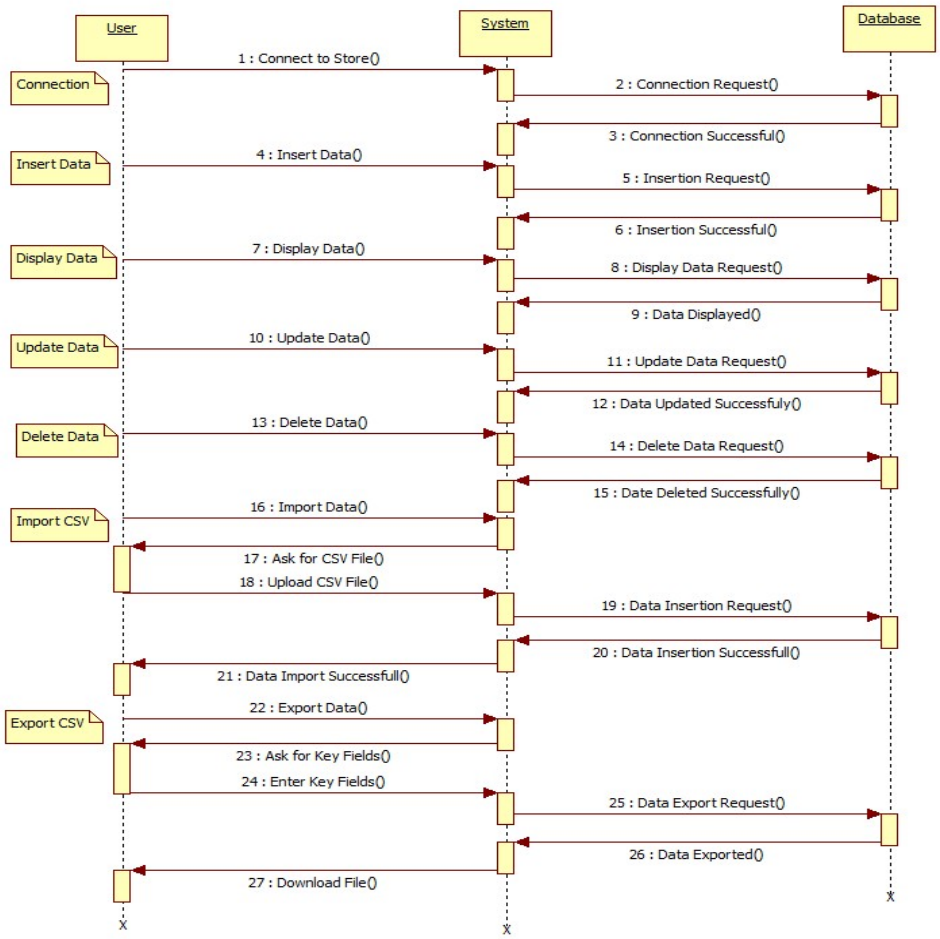

Figure 4.4: Sequence Diagram

- GUI based System :

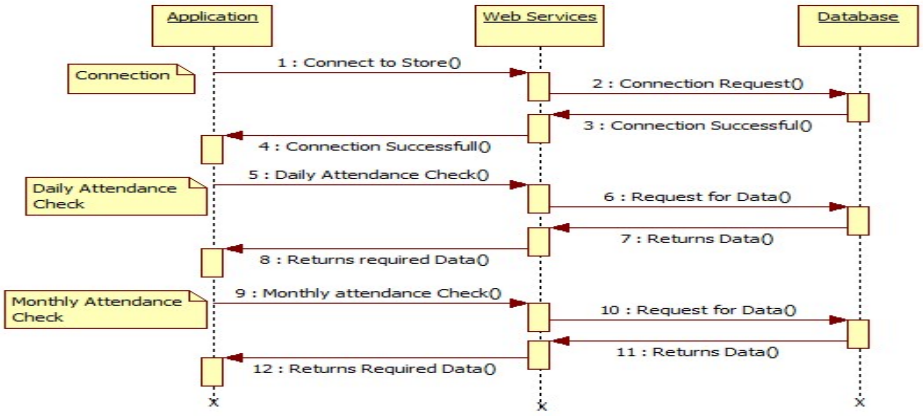

Figure 4.5: Attendance check System

Collaboration diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

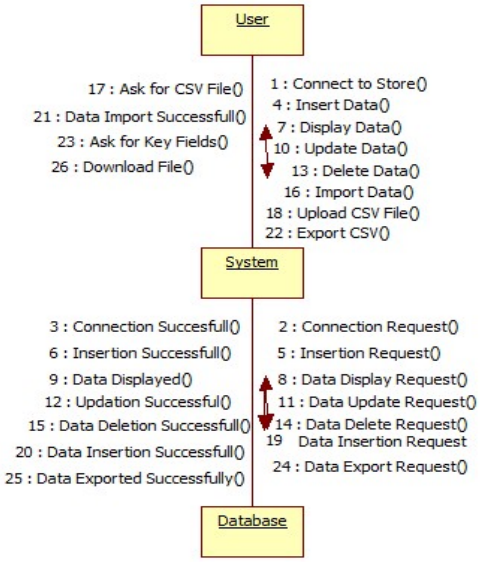

Figure 4.6: Collaboration Diagram

- GUI based System :

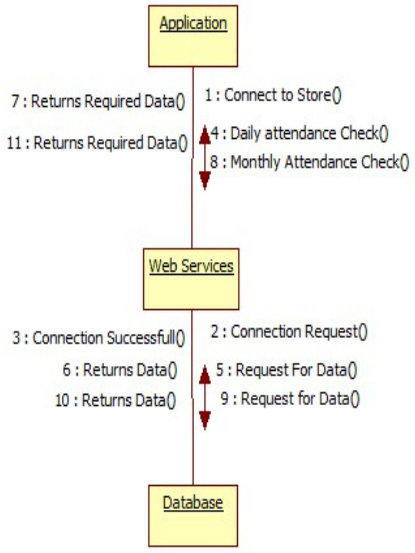

Figure 4.7: Attendance check System

Package diagram

Package is general purpose mechanism for organizing elements into groups. Every package must have a name that distinguishes it from other packages. Package name must be unique within its enclosing package .A name is a textual string .That name alone is known is known as simple name. A path name is package name prefixed by name of package lives.

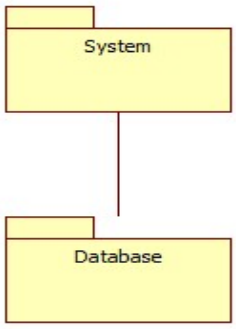

Figure 4.8: Package Diagram

- GUI based System :

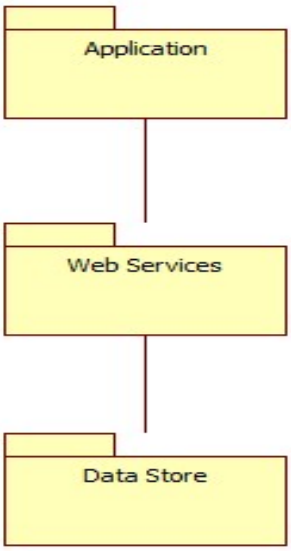

Figure 4.9: Attendance check System

Deployment Diagram

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.

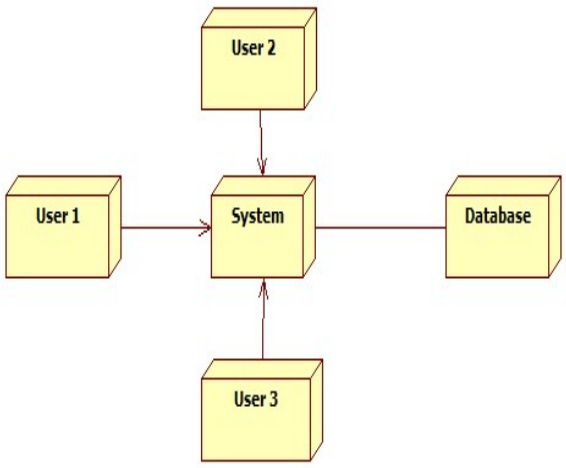

Figure 4.10: Deployment Diagram

- GUI based System :

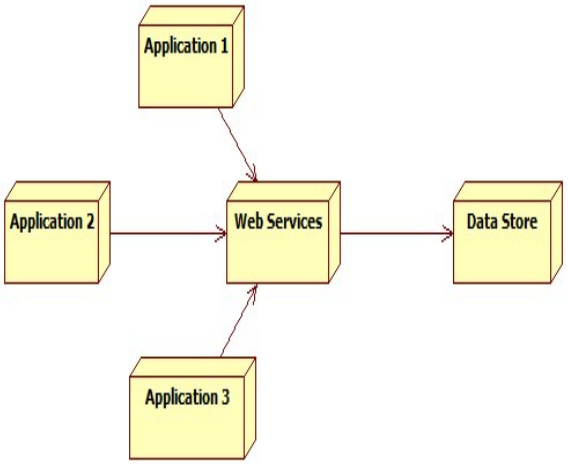

Figure 4.11: Attendance check System

Chapter 5

Technical Specification

5.1 Technology used for the project

- Java 7 :

Java is a complete package for application and generic programming. Java is used to develop various kind of application from console to high end applications. We have used Java Software Development Toolkit Version 7 for our project development.

- Java Servlet 3.0 :

Java servlet is a part of Java EE community project. Servlet provide an full package for web application development. Java servlets is a technology which works at the server in web applications. Servlets are used for handling the reuests from various clients. Servlets is having strong set of functions for managing the web applications. Every functionality of java SE and java EE is embedded or can be used with java servlet. Servlet has two functions namely doGet() and doPost() which manage all the request from clients. Servlets can be of type Generic servlet or HTTP servlets. We have used HTTP servlets for our project development. Java servlets comes into the package i.e. javax.servlet.*;

- Glassfish Server 4.0:

Glassfish is world's first java EE 7 application server. Application server is for deployment of various applications which are developed for web. Glassfish server is used to host to deploy the application which is developed under java EE. Glassfish has excellent support for Java EE 7 applications. Our project is tested and deployed on Glassfish server 4.0 which is provided along with Netbeans IDE 7.4.

- Android SDK:

Android Software development toolkit provides platform for android developers where they can develop android applications. Android SDK comes along with all the necessary library files which are used for android application development. SDK provides development from very older version to latest version. The application developed can work on Android version 2.2 to latest version 4.4.

- HTML 5:

Hyper Text Markup Language is an language which is used to develop Graphical user interface for web applications. HTML 5 is latest version which has all new features. The GUI for web application is this project is developed under HTML 5.

- RESTful web services:

REpresentational State Transfer (REST) is a technology which provides a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URLs. Web service clients that want to use these resources access a particular representation by transferring application content using a small globally defined set of remote methods that describe the action to be performed on the resource. In Simple language RESTful web services are a kind of web applications which acen be accssed form any type of client, RESTful web services provide platform independent service to all clients. We developed web services which are used to for managing KVStore Oracle NoSQL Database.

Chapter 6

Project Estimate, Schedule And Team Structure

6.1 System Implementation Plan

There are many important phases for implementation. Following are important phases and tasks of implementation:
The main phase is to study existing related application. All the necessary software for implementing the system must be installed on machine where the system is to be implemented.

Software project management begins with set of activities that are collectively called project planning. Before the project can begins the team must estimate the work to be done, the resources that will be requiring, and the time that will elapse from start to finish. Planning involves estimation your attempt to determine how much money, how much effort, how many resource and how much time it will take to build to specific software base to system or product.

Would one build a house without knowing how much you were about to spend for construction? Of course not and since most computer based system and products cost considerably more to build than a large house, it would seems reasonable to develop and estimate before you start creating a software.

- Project Plan for semester I and II:

Table 6.1: Project Plan for Semester I and II

Sr.No	Phase	Start Date	End Date	Status
1	Topic Selection	26 June 2013	1 July 2013	Completed
2	Introduction	1 July 2013	20 July 2013	Completed
3	Literature Survey	21 July 2013	10 Aug 2013	Completed
4	System Requirement Specification	11 Aug 2013	31 Aug 2013	Completed
5	System Design	1 Sept 2013	7 Sept 2013	Completed
6	Technical Specification	8 Sept 2013	21 Sept 2013	Completed
7	Project Estimate, Schedule, Team Structure	22 Sept 2013	28 Sept 2013	Completed
8	Software Implementation	29 Sept 2013	25 Apr 2014	Completed
9	Software Testing	1 Jan 2014	15 Feb 2014	Completed
10	Experimental Results And Discussion	16 Feb 2014	5 Apr 2014	Completed
11	Conclusion And Future Scope	4 Apr 2014	5 Apr 2014	Completed

6.1.1 Major Task

• Major task 1:

Provide overall planning and coordination for the implementation:

This task refers to plan for the system implementation. It also provides the guideline for implementing the system. The tasks which are planned must be completed within date specified in schedule. This will lead to successful completion of the project.

• Major task 2:

Provide appropriate training for personnel:

This task refers to the training of the system developer on java,html and android framework. For successful completion of the scheduled task in estimated time, this is required.

• Major task 3:

Provide all needed technical assistance:

The technical assistance must be provided whenever the work gets halt due to technical barriers.

Table 6.2: Major Task

Phase	Task	Description
Phase1	Requirement	Gather all the information for the selected topic.
Phase2	Analysis	Analyze all information on selected topic.
Phase3	Design	Assign the module.

6.1.2 Description of Implementation

For the implementation purpose the machine on which the system to be implemented, the Operating System must be Windows. For the implementation purpose developer will require Java knowledge as well aw HTML and Android SDK's knowledge. The implementation must be carried out in phases as described above. But the implementation can go Prototype wise development. One by one prototype must be implemented and further the functionalities must be implemented into that prototype without affecting previous prototype.

6.1.3 Points-of-Contact

Table 6.3: Points-of-Contact

Role	Name	Contact Number
Project/Program Manager	Sushank Dahiwadkar Anjali Kapadni	9730971651, 9422499677
System Developer or System Maintainer	Sushank Dahiwadkar, Anjali Kapadni	9730971651, 9422499677,
Quality Assurance Manager	Sushank Dahiwadkar, Anjali Kapadni	9730971651, 9422499677,
Database Administrator	Sushank Dahiwadkar	9730971651,

6.1.4 Security and Privacy

For performing operations with NoSQL KVstore user need to Connect with KV-Client for that user need to provide Store Name, Port Number and Host Name to logged into UI to perform these operations. If user is not able to connect to store then he/she may not be able to perform tha operations.

6.2 Project Scheduling

Project scheduling is the process of putting together a time line for all the activities in the project. This involves examining the interdependencies of all of the activities, and coordinating all the tasks to ensure a smooth transition from the beginning to the end of the project. There are many different methods of scheduling, which can address the requirements of the type of project resulting in different pictorial representations of the schedules. For example use Gantt charts to represent the project scheduling.

• Gantt Chart of Scheduling for Semester I is as follows :

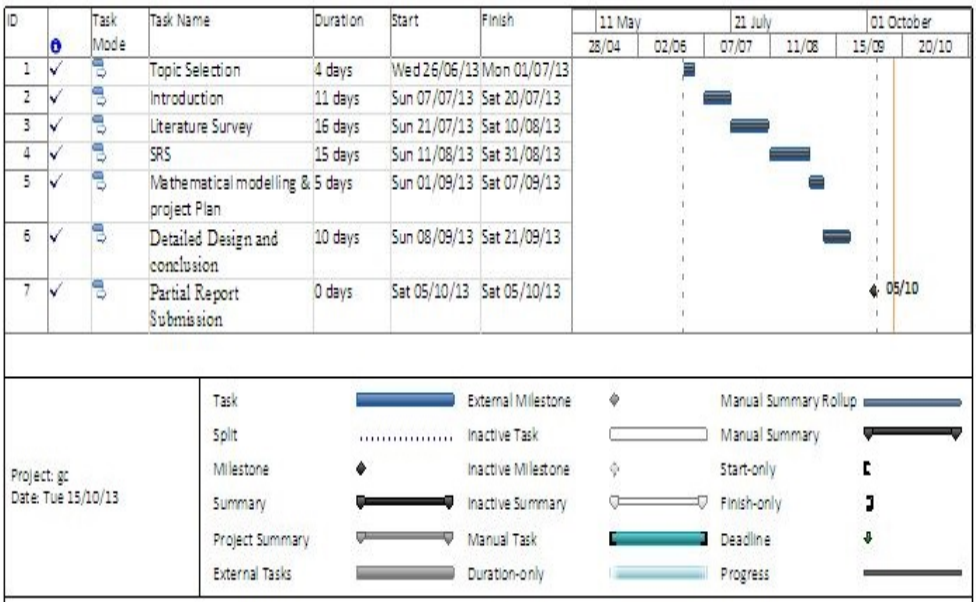

Figure 6.1: Gantt Chart of Scheduling of Sem I

• Gantt Chart of Scheduling for Semester II is as follows :

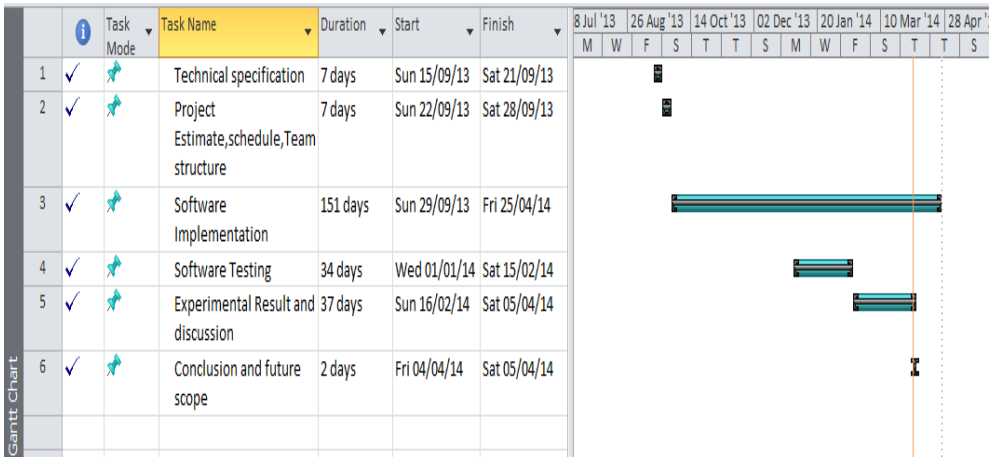

Figure 6.2: Gantt Chart of Scheduling Sem II

6.3 Feasibility Study

6.3.1 COCOMO Model

Constructive cost model is one of the most widely used and discussed software cost estimation model. The Query by example based Video Retrieval System uses COCOMO MODEL. The basic COCOMO MODEL computes software development efforts and cost as a function of program size express in estimated line of code.

Table 6.4: Analysis and Estimation Cocomo model

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

"Query by Example based Video Retrieval System" is an *Organic* project.

✠ Effort = $a_b KLOC^{b_b}$

✠ Duration = $c_b KLOC^{d_b}$

- $a_b = 2.4$

Table 6.5: Calculation

Functions (Modules)	Estimated LOC
Video Input	400
Preprocess	700
Feature extraction And Copmarison	300
Result generation	200
Total	1600

- $b_b = 1.05$
- $c_b = 2.5$
- $d_b = 0.38$
- KLOC=1.6

✠ Effort = $a_b KLOC^{b_b}$
= 2.4 * ($1.6^{1.05}$)
= 3.931 persons/month.

✠ Duration = $c_b KLOC^{d_b}$
= 2.5 * ($1.6^{0.38}$)
= 2.989 months

✠ Number of people recommended = $\frac{Efforts}{Duration}$
= $\frac{3.931}{2.989}$
= 1.315 persons
= $\sim$ 1 person

1 person doing work in 4 months. Hence work duration for 3 people is 1 and half month.

LOC based Estimation:

◇ The average productivity for our product is : 1000 LOC/Month

◇ The labour rate estimated is : Rs. 1000/- per month

◇ The cost per line of code is : approximately Rs. 6

◇ Base on LOC estimated and historical productivity data, **Total estimated project cost is : Rs. 13600/-**

6.4 Team Structure

The team structure for this project is as follows :

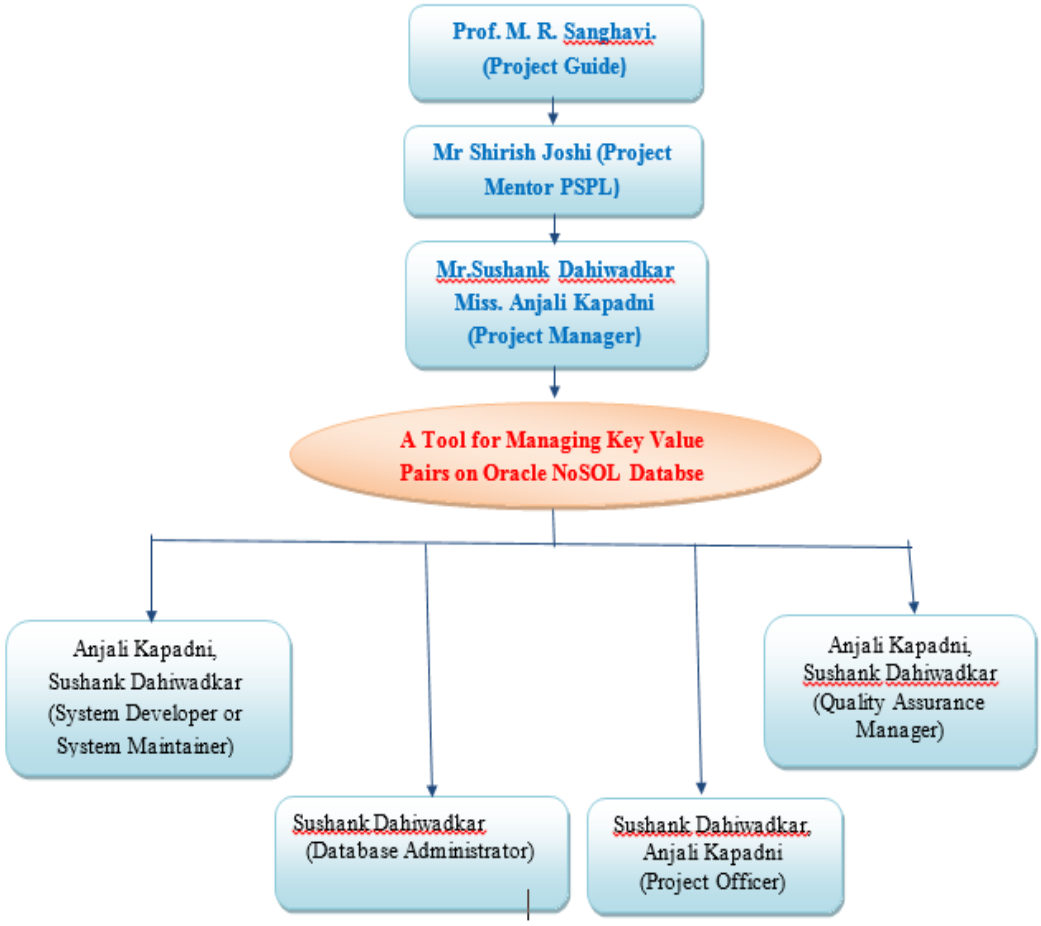

Figure 6.3: Team Structure

Chapter 7

Software Implementation

7.1 System Implementation

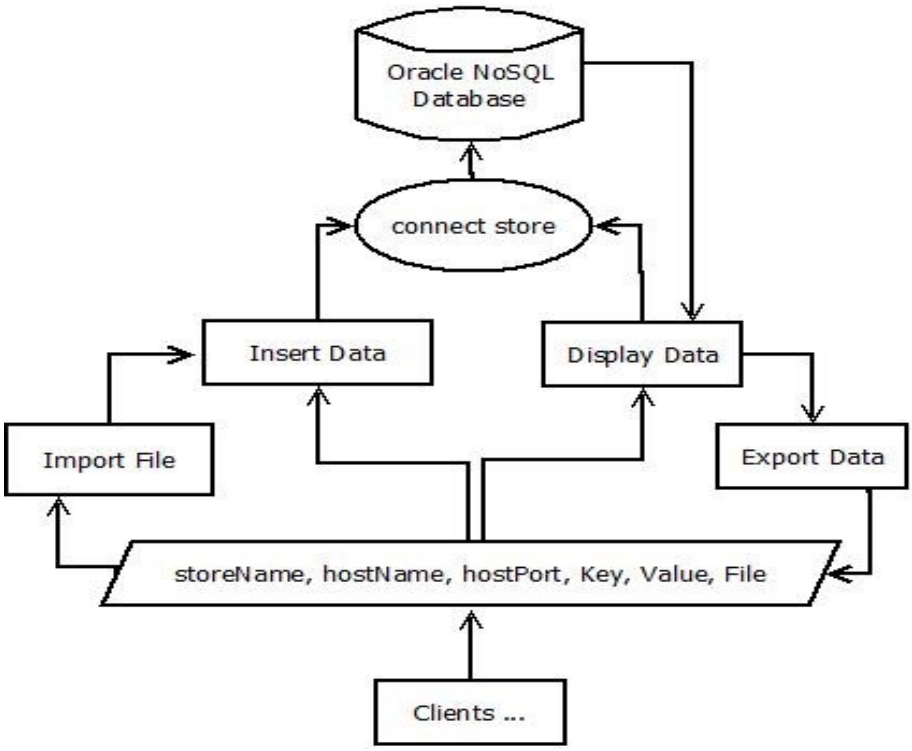

Figure 7.1: System Architecture

- **Part1**

The system is a web application which fits into the application logic layer of three tier Architecture. System provides the ease to access key-value paired Oracle NoSQL database. System is a web interface through which the user can connect to the database and perform required operations. Position of the System in Three Tier Architecture is shown in figure

System's interface provides with various functionalities on database which includes CRUD operations and Import Export. The System follows disconnected architecture. in disconnected architecture the system is not synchronized with another system. System works in three stages i.e. connect to data store, perform operations and disconnect from data store. For performing every new operation the system needs to follow above three steps. System follows disconnected architecture then too system maintains session, in sessions Store Name, Host Name and Host Port are stored. Because foe every new operation user should not have to enter again those parameters.

The system interface is kept simple, so that any user having basic knowledge of key value based NoSQL database can use the system. The System is developed as a User Interface for Oracle NoSQL Database. System uses KVLITE which is a Single Node Data Store by Oracle NoSQL. The System uses kvclient.jar file which has all the necessary library files for using KVLITE or KVSTORE.
Following operations can be performed using the Developed System:

1. Connect to Store :

   This function connects the client to Data Store on Remote Server. this functions takes three parameters namely Store Name, Host Name i.e. host address on which the data store is installed and Host Port i.e. on which the Data Store's service is started. When the system connects to data store a new session is created and this session is maintained until the user completes specific operation or logs out from system.

2. Insert Data :

   This function is used to insert Single or Multiple Key Value pairs in the Data Store. The function comes in two forms i.e. Single and Multiple Key Value Pair Insert. For Single Key Value Pair insert system requires three parameters, namely Major Key Component, Minor Key Component and Key Value. For Multi Key Value Pair Insertion the system first asks for Major Key Component and Minor Key Component Count, followed by that the user has to enter Minor Key Components and Key Values.

3. Display Data :

   This function gets the data from the store and displays to user. Functions come in two forms i.e. Single and Multiple Key Value pair display. For Single Key Value display system takes Major Key and Minor Key Component and parameters. And for Multi Key Value pair display System takes only Major Key Value Component.

4. Update Data :

   This function is used to update the existing Key Value pairs. This function also comes in two forms Single and Multiple Key Value Pairs Update. For Single Key Value Pairs update user has to enter Major Key Component, Minor Key Component and Key Value. And for Multi Key Value pair update user has to enter one Common Major Key Component and its Minor Key Components along with Key Values.

5. Delete Data :

   Delete functions is used to delete existing Key value Pairs. The function has two forms. The first form is Single Key Value Pair delete where user has to enter Major and Minor Key Component. And for Multiple Key Value pair delete user has to enter only Major Key Component.

6. Import Data :

   The function provides Import functionality. Here the data can be imported from a Comma Separated value file into the database. Here firstly user has to upload a file to server which is to be imported into the data store. After the file is been uploaded the system ask user to mention Major Key Component and Minor Key Component for that file along with the Minor Key Part. Here the Minor Key Component is a composite form of Minor Key Component specified by the user and a field from CSV file. These two parts makes a new Minor Key which is used to identify the Key Value Pair uniquely. This approach is used so that the Relational Database can be mapped with NoSQL Database.

7. Export Data :

   Here the data from database is directly exported into the CSV (Comma Separated Value) file. The data which is to be exported is firstly fetched from database and if user wish to export that data into file then that data is exported into the CSV file. This function is only available for multiple Key Value pairs Export.

- **Part2**

The Part 2 of project consists of Creating Web Services to access the NoSQL database. Web services are developed so that the any client independent of any platform can access the database. Here in our application RESTful webservices are developed. RESTful, because this are stateless and lightweight. This web services are meant to access from mobile devices.

In our case to demonstrate the use application of big data, Sensor generated data is stored and is access from the KVStore. This sensor data is gathered from the Bio Metrics

Thumb machine use for daily attendance record.

Here two web services are created:

1. To get the daily attendance of user :

   Here User gets the date wise attendance report. The user has to enter his/her user id In/Out log and Date and Log in or Log out Time is displayed to the user.

2. To get the monthly attendance of the user :

   Here user gets the month wise attendance report. The user has to enter his/her user id and the respective month and year and the monthly attendance is displayed by calculating it from working days.

- **Part3**

Android application is been developed through which the user gets his/her attendance report. The above explained web services are been consumed using this application.

7.2 Mathematical Modeling and algorithm

This mathematical model represents the proposed system described in the previous section.
$\bigodot S = \{\Sigma, q0, \delta, F, Q\}$
Where
Σ = { storeName, hostName, hostPort, majorKeyComponent, minorKeyComponent, keyValue }
q0 = Connect to Store
δ = f1 , f2 , f3 , f4 , f5 , f6 , f7
f1= connectStore
f2 = insertSingleData
f3 = insertMultiData
f4 = displaySingleData
f5 = displayMultiData
f6= updateSingleRecord
f7= deleteSingleRecord
f8= deleteMultiRecord
f6 = importCSV
f7 = exportCSV
Q = Connect to Store, Perform Operation, Disconnect from Store (All functions from f1 to f6 are already explained in previous section)

7.2.1 Applications

- For managing key value paired oracle nosql database.
- For importing and exporting data from Oracle NoSQL Database.
- For attendance check to employees in organizations.
- The GUI based application can be used anywhere, where Oracle NoSQL databse comes into focus.

7.2.2 Advantages and Disvantages

- Reduces efforts and time of user for managing Oracle NoSQL Database.
- Direct Import Export Functionality.
- REST Ful web services whose functionalities can be extended on any platform for any use.

Chapter 8

Software Testing

8.1 Introduction to Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1.1 Types of Tests

- **Level of Testing**

1. Unit testing tests the minimal software component or modules. Each unit of the software is tested to verify that the details designed for unit has been correctly implemented.

2. Integration testing exposes defects in the interface and interaction between integrated components. Progressively larger groups of tested software components corresponding to elements of architectural design are integrated and tested until the software works as whole.

3. System testing tests an integrated system to verify that it meets its requirements.

4. System integration testing verifies that system is integrated to any external or third party systems defined in the system requirements.

5. Acceptance testing can be conducted by the end users customers of client to validate whether or not to accept the product. Acceptance testing may be performed after the testing and the before the implementation phase.

6. Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at developers site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

7. Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to limited audience outside of the company. The software is released to the groups of people so that further testing can ensure the product has few faults and bugs. Sometimes, beta versions are made available to open public to increase the feedback field to maximal number of future users.

As a tester, it is always advisable to use manual white box testing and black box testing techniques on the test software. Manual testing helps discover and record any software bug or discrepancies related to the functionality of the product.

Manual testing can be replaced by test automation. It is possible to record and playback manual steps and write automated test script(s) using test automation tools. Although, Test automation tools will only help execute test scripts written primarily for executing a particular specification and functionality. Test automation tools lack the ability of decision-making and recording any unscripted discrepancies during program execution. It is recommended that one should perform manual testing of the entire product at least a couple of times before actually deciding to automate the more mundane activities of the product.

Manual testing helps discover defects related to the usability testing and GUI testing area. While performing manual tests the software application can be validated whether it meets the various standards defined for effective and effective usage and accessibility. For example, the standard location of the OK button on the screen is on the left and of CANCEL button on the right. During manual testing you might discover that on some screen, it is not. This is a new defect related to usability of screen.

A manual tester would typically perform the following steps for manual testing:

- Understand the functionality of program
- Prepare a test environment
- Execute test case(s) manually
- Verify the actual result
- Record the result as PASS or FAIL
- Make a summary report of the PASS and FAIL test cases

- Publish the report
- Record any new defects uncovered during the test case execution

There is no complete substitute for manual testing. Manual testing is crucial for testing software application more thoroughly. Test automation has become a necessity mainly due to shorter deadline for performing test activities, such as regression testing, performance testing and load testing.

8.2 Manual Test Cases

Following question answering provides an overview referred from the test plan built for the project being implemented:

Table 8.1: test cases for connect to store

Sr.No.	Test Cases	Expected Results	Actual results
1	Blank Store Name, Host Name and Host Port	Error Message: "Must Enter Store Name, Host Name and Host Port"	Error Message: "Must Enter Store Name, Host Name and Host Port"
2	Blank Store Name, Incorrect/Correct Host Name and Host Port	Error Message: "Must Enter Store Name"	Error Message: "Must Enter Store Name, Host Name and Host Port"
3	Blank Host Name, Incorrect/Correct Store Name and Host Port	Error Message: "Must Enter Host Name"	Error Message: "Must Enter Store Name, Host Name and Host Port"
4	Blank Host Port, Incorrect/Correct Store Name and Host Name	Error Message: "Must Enter Host Name"	Error Message: "Must Enter Store Name, Host Name and Host Port"
5	Incorrect Store Name, Correct Host Name and Host Port	Error Message: "Enter Correct Store Name"	Error Message: "Cannot Connect to Store"
6	Incorrect Host Name, Correct Store Name and Host Port	Error Message: "Enter Correct Host Name"	Error Message: "Cannot Connect to Store"
7	Incorrect Host Port, Correct Store Name and Host Name	Error Message: "Enter Correct Host Port"	Error Message: "Cannot Connect to Store"
8	Incorrect Store Name, Host Name and Correct Host Port	Error Message: "Enter Correct Store Name and Host Name"	Error Message: "Cannot Connect to Store"
9	Incorrect Store Name, Host Port and Correct Host Name	Error Message: "Enter Correct Store Name and Host Port"	Error Message: "Cannot Connect to Store"
10	Incorrect Host Name, Host Port and Correct Store Name	Error Message: "Enter Correct Host Name and Host Port"	Error Message: "Cannot Connect to Store"
11	Incorrect Store Name, Host Name and Host Port	Error Message: "Enter Correct Store Name, Host Name and Host Port"	Error Message: "Cannot Connect to Store"
12	Correct Store Name, Host Name and Hot Port	Message: "Welcome to KVStore"	Message: "Welcome to KV-Store"

Table 8.2: test cases for Insert Data

1	Blank Major Key Component, Minor Key Component and Key Value	Error Message: "Please Enter all Parameters"	Error Message: "Data Insertion Failed"
2	Blank Major Key Component and Valid Minor Key Component and Key Value	Error Message: "Please Enter Major Key Component"	Error Message: "Please Enter all Parameters"
3	Blank Minor Key Component and Valid Major Key Component and Key Value	Error Message: "Please Enter Minor Key Component"	Error Message: "Please Enter all Parameters"
4	Blank Key Value and Valid Major Key Component and Minor Key Component	Error Message: "Please Enter Key Value"	Message: "Data Inserted Successfully"
5	Valid Major Key Component, Minor Key Component and Key Value	Message: " Data Inserted Successfully"	Message: "Data Inserted Successfully"

Table 8.3: test cases for Display Data

1	Blank Major and Minor Key Component	Error Message: "Please Enter All Parameters"	Error Message: "Error"
2	Blank Major Key Component and Valid Minor Key Component	Error Message: "Please Enter All Parameters"	Error Message: "Please Enter All Parameters"
3	Blank Minor Key Component and Valid Major Key Component	Error Message: "Please Enter All Parameters"	Error Message: "Please Enter All Parameters"
4	Incorrect Major and Minor Key Component	Error Message: "Data not Available for given Parameters"	Error Message: "Error"
5	Incorrect Major Key Component and Correct Minor Key Component	Error Message: "Data not Available for given Parameters"	Error Message: "Error"
6	Incorrect Minor Key Component and Correct Major Key Component	Error Message: "Data not Available for given Parameters"	Error Message: "Error"
7	Correct Major and Minor Key Component	Data Displayed	Data Displayed

Table 8.4: test cases for Update Data

1	Blank Major Key Component, Minor Key Component and Key Value	Error Message: "Please Enter all Parameters"	Error Message: "Data Insertion Failed"
2	Blank Major Key Component and Correct Minor Key Component and Key Value	Error Message: "Please Enter Major Key Component"	Error Message: "Please Enter all Parameters"
3	Blank Minor Key Component and Correct Major Key Component and Key Value	Error Message: "Please Enter Minor Key Component"	Error Message: "Please Enter all Parameters"
4	Blank Key Value and Correct Major Key Component and Minor Key Component	Error Message: "Please Enter Key Value"	Error Message: "Data Updation Failed"
5	Incorrect Major Key Component and Correct Minor key Component and Key Value	Error Message: "Please Enter Major Key Component"	Error Message: "Data Updation Failed"
6	Incorrect Minor Key Component and Correct Major key Component and Key value	Error Message: "Please Enter Minor Key Component"	Error Message: "Data Updation Failed"
7	Correct Major Key Component, Minor Key Component and Key Value	Message: " Data Updated Successfully"	Message: "Data Updated Successfully"

Table 8.5: test cases for Delete Data

1	Blank Major Key and Minor Key Component	Error Message: "Please Enter all Parameters"	Error Message: "Error"
2	Blank Major Key Component and Valid Minor Key Component	Error Message: "Please Enter all Parameters"	Error Message: "Please Enter all Parameters"
3	Blank Minor Key Component and Valid Major Key Component	Error Message: "Please Enter all Parameters"	Error Message: "Please Enter all Parameters"
4	Incorrect Major Key Component and Correct Minor Key Component	Error Message: "Please Enter Correct Major Key Component"	Error Message: "Error"
5	Incorrect Minor Key Component and Correct Major Key Component	Error Message: "Please Enter Correct Minor Key Component"	Error Message: "Error"
6	Correct Major and Minor Key Component	Message: "Data Deleted Successfully"	Message: "Data Deleted Successfully"

Table 8.6: test cases for Import Data

1	Blank Select File Option	Error Message: "Please Select File"	Error Message: "Please Select Valid File"
2	Other than CSV file Selected	Error Message: "Please Select File"	Error Message: "Please Select Valid File"
3	Input CSV File	Message: "File Uploaded, Please Select Options for Import"	Message: "File Uploaded, Please Select Options for Import"
4	Blank Major, Minor Key Component and Minor Key Component Part Field	Error Message: "Please Enter all the Parameters"	Error Message: "Please Enter all the Parameters
5	Blank Major Key Component and Valid Minor Key Component and Minor Key Component Part Field	Error Message: "Please Enter all the Parameters"	Error Message: "Please Enter all the Parameters
6	Blank Minor Key Component and Valid Major Key Component and Minor Key Component Part Field	Error Message: "Please Enter all the Parameters"	Error Message: "Please Enter all the Parameters
7	Blank Minor Key Component Part Field and Valid Major Key Component and Minor Key Component	Error Message: "Please Enter all the Parameters	Error Message: "Please Enter all the Parameters
8	Valid Major, Minor Key Component and Minor Key Component Part Field	Message: "Data Imported Successfully"	Message: "Data Imported Successfully"

Table 8.7: test cases for Export Data

1	Blank Major Key Component	Error Message: "Please Enter Major Key Component"	Error Message: "Error"
2	Incorrect Major Key Component	Error Message: "Please Enter Major Key Component"	Error Message: "Error"
3	Correct Major Key Component	Data Displayed and File Download Option Appears	Data Displayed and File Download Option Appears

Chapter 9

Experimental Results And Discussion

Graphical user interface is designed and which is expressed through some screen shots using HTML and Java. The experiments carried out with the system are shown below with the help of graphical user interface of the system.

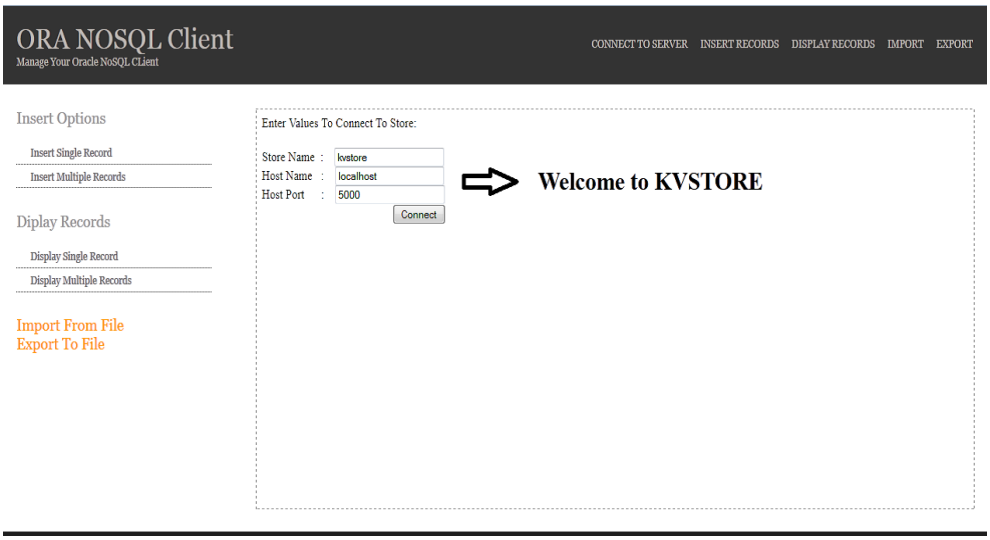

Figure 9.1: Working Model of System.

As the system was tested locally, it is capable of performing every operation listed above. The system is deployed on Glassfish Server and the results were good to compare with. Simultaneously various clients were connected and operations were performed and the System worked fine with that.

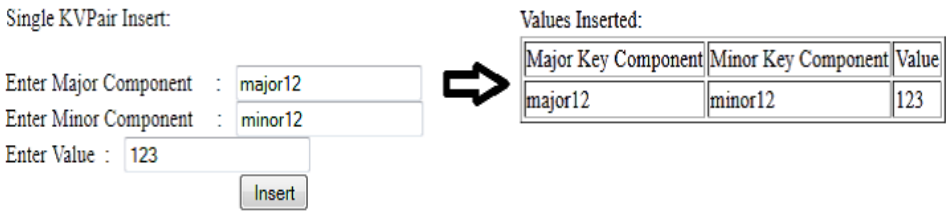

Figure 9.2: Single KVPair Insert Operation

Above Figure shows Single KVPair insert and the same is applied for Multiple KVPair insert, but only first time the Major Key component and Number of Minor Key Components is asked and that number of fields is generated for user to insert value.

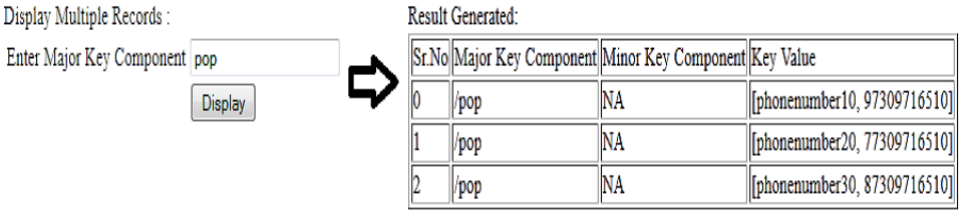

Figure 9.3: Multiple KVPair Display Operation

Above Figure shows Multiple KVPair Display operation, same applies for Single KVPair display, but there user has to specify Major as well as Minor Key Component.

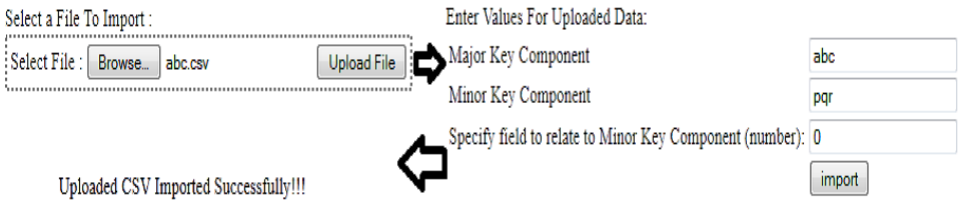

Figure 9.4: Import CSV Operation.

Above Figure shows Import Data from CSV operation, here the CSV file is uploaded to server and then user has to specify Major and Minor Key Component along with the index of field which is to be associated with Minor Key Component.
For Export Data to file, the user first has to perform Single or Multiple Display Operation and after that a link is provided, through which the user can download generated CSV file.

Below are the Screen shots of Android Application, which is developed to consume the web services. This application is used for getting attendance report of the user.

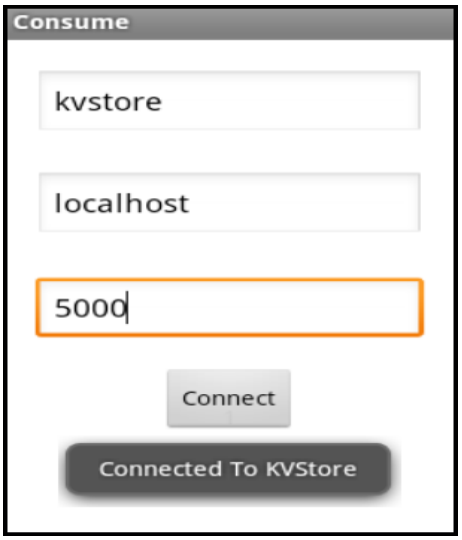

Figure 9.5: Interface for connection to Store.

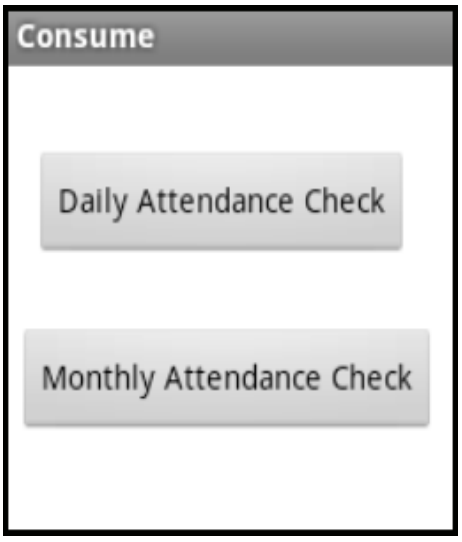

Figure 9.6: Options for Attendance Check.

Figure 9.7: Daily Attendance Check Interface.

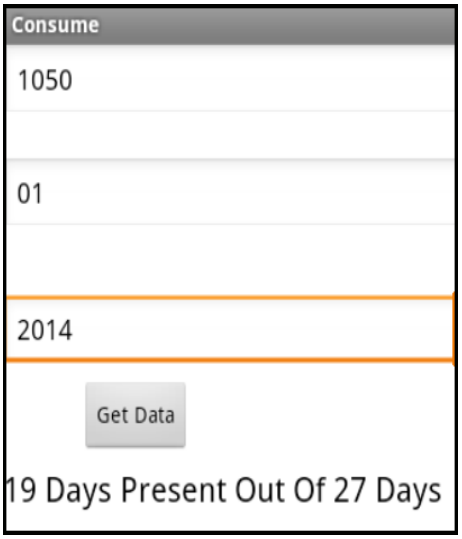

Figure 9.8: Monthly Attendance Check Report.

In this way the experiments are carried out on every functions of the system and we have found various results as shown in above screenshots. The system works well.

Chapter 10

Deployment And Maintenance

Chapter 11

Conclusion And Future Scope

REFERENCES

[1] [Online]. Available: http://en.wikipedia.org/wiki/Database

[2] [Online]. Available: http://

[3] [Online]. Available: http://

[4] [Online]. Available: http://rebelic.nl/2011/05/28/the-four-categories-of-nosql-databases/

[5] [Online]. Available: http://www.infoq.com/articles/graph-nosql-neo4j

[6] [Online]. Available: http://en.wikipedia.org/wiki/Document-oriented_database

[7] [Online]. Available: http://en.wikipedia.org/wiki/Column_family

[8] [Online]. Available: http://

Appendix

Authors Profile

Title : **Authors Profile**			
Report Code : **SNJBCOE/COMP/13-14/PROJECTID 01**			
Report Title : Query by Example based Video Retrieval System.			
Author Details			
Author Name	**Email-Id**	**Address**	**Photo**
Anjali Dattatray Kapadni	kapadni.anjali@gmail.com	739, Main road Chandwad Nashik (423101)	
Bhagyashri Shantilal Lalwani	bhagyashri.lalwani@gmail.com	916 B Somwar peth, Chandwad Nashik (423101)	
Prachi Vilas Danapure	prachidanapure@gmail.com	Pasaydan Colony, Waman bhau nagar, Pathardi, A'nagar (414102)	
Type of Report **Project Report**	**Time Covered** **From** / **To**	**Date of Report**	**Page Count** 100
Keywords : Video, Key-Frame, Feature Extraction, Query by Example.			
Report Checked By	**Report Checked Date**	**Guide Name** Prof. K. M. Sanghavi	**Total Copy**
CD/DVD : Yes / No Yes	**CD Checked By& Date**	**Content**	
Publication Details (if any) : NA			
Patent Details (if any) : NA			

Figure 11.1: Authors Profile