SEEK

(An Implementation of Elastic Search in CouchDB)

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering

By

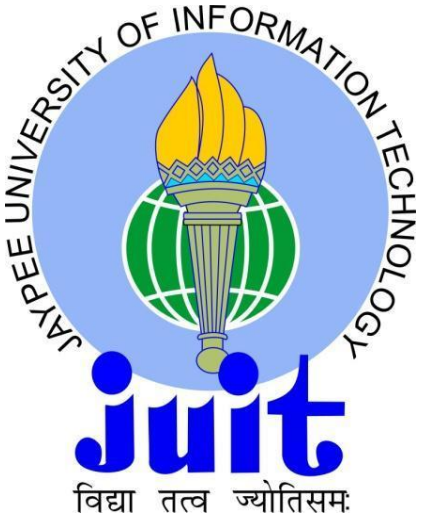
Prakhar Agarwal (121251)

Gagan Deep Singh (121281)

Under the supervision of

(Ms. Nishtha Ahuja)

To



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

Candidate's Declaration

This is to certify that the work which is being presented in the project title **"SEEK"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of work carried out by **Prakhar Agarwal(121251), Gagan Deep Singh(121281)** during a period from August 2015 to May 2016 under the supervision of **Ms. Nishtha Ahuja** (Assistant Professor of Computer Science Department).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. S P Ghrera	**Ms. Nishtha Ahuja**
Professor, Brig (Retd.) and Head, Dept. of CSE **JUIT Waknaghat**	**Assistant Professor** **JUIT Waknaghat**

ACKNOWLEDGEMENT

We owe our profound gratitude to our project supervisor **Ms. Nishtha Ahuja**, who took keen interest and guided us all along in our project work titled "**SEEK**" which is **an Implementation of Elasticsearch in CouchDB**, till the completion of our project by providing all the necessary information for developing the project. The project development helped us in research and we got to know a lot of new things in our domain. We are really thankful to her.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This project is intended to give an Implementation of Elasticsearch in Big Data. Since, Now a day's Organizations are generating, processing, and retaining data at a rate that often exceeds their ability to analyze it effectively and at the same time, the insights derived from these large data sets are often key to the success of the organizations, allowing them to better understand how to solve hard problems and thus gain competitive advantage. Because this data is so fast-moving and voluminous, it is increasingly impractical to analyze using traditional offline, read-only relational databases.

Recently, new big data technologies and architectures, including Hadoop and NoSQL databases, have evolved to better support the needs of organizations analyzing such data. In particular, Elasticsearch a distributed full-text search engine that explicitly addresses issues of scalability, big data search, and performance that relational databases were simply never designed to support.

1. INTRODUCTION

SEEK is a Web Application that runs on a web browser or is created in a browser-supported programming language (such as the combination of JavaScript, HTML and CSS) and relies on a common web browser to render the application.

This project is intended to give implementation of Elasticsearch on big data to search user query. The project highlights the advantages of using Elasticsearch mechanism to retrieve results against a user query. The major advantage of using Elasticsearch is its efficiency and hence the time taken to respond to user query is less than other mechanisms. The working of Elasticsearch is also described in this documentation. The documentation also lists out the requirements for the project implementation.

1.1) PROBLEM STATEMENT

Organizations are generating, processing, and retaining data at a rate that often exceeds their ability to analyze it effectively and at the same time, the insights derived from these large data sets are often key to the success of the organizations, allowing them to better understand how to solve hard problems and thus gain competitive advantage. Because this data is so fast-moving and voluminous, it is increasingly impractical to analyze using traditional offline, read-only relational databases.

Recently, new big data technologies and architectures, including Hadoop and NoSQL databases, have evolved to better support the needs of organizations analyzing such data. In particular, Elasticsearch a distributed full-text search engine that explicitly addresses issues of scalability, big data search, and performance that relational databases were simply never designed to support.

1.2) OBJECTIVE

Short-term – The short-term objective of the project is complete understanding of the project assigned. The concepts related to the project should be clear and the objective behind the project should be known to the entire team. The focus should not be only on theoretical concepts but also on practical implications of them. The technology used to implement the projects should be familiar and also one should be able to apply them on our respective project.

Long-term – Long-term objective includes that students are exposed to the industrial environment which should help us in future when we will work in real-time projects. One should gain experience of working with a team and learn to cooperate with our team members.

1.3) METHODOLOGY

- **Data Set Creation:** This module makes the logged user to post any question he wants that question will get stored in CouchDB with user id and enable the user to answer any question he wants after searching that question a user can answer multiple times which get stored in question database respectively.

- **UI Design:** We create a LOGIN page which has username and password of registered user, on entering the fields if it matches with the database stored data then user can use the application. Then we create a REGISTRATION page on which new user can create their account by entering their personal details like Email_id, Username, Password, which is directly stored in our database (CouchDB).

- **Indexing of Documents in Elasticsearch Cluster:** This module will insert questions and answer in elastic search server in particular index of particular document type so that a user can retrieve the questions and its respective answer from database based on some specified keyword searching.

- **Implementation of Elasticsearch in CouchDB**: This module will search out the content based on made indices and document type on single or multiple node.

- **Testing:** Various kinds of testing like black box testing, white box testing etc

2. LITERATURE SURVEY

2.1) GOOGLE WEB TOOLKIT

We can use Google Web Toolkit (GWT) for developing the front end of the project, it is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java. Other than a few native libraries, everything is Java source that can be built on any supported platform with the included GWT Ant build files. It is licensed under the Apache License version 2.0. GWT emphasizes reusable approaches to common web development tasks, namely asynchronous remote procedure calls, history management, bookmarking, UI abstraction, internationalization, and cross-browser portability.

2.1.1) ARCHITECTURE OF GWT

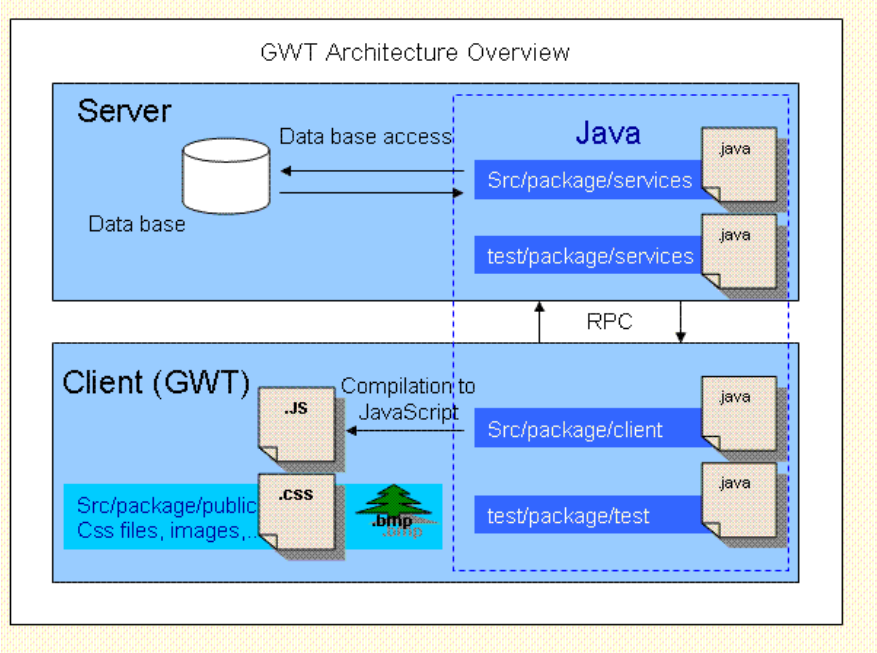


Fig 1

With Google Web Toolkit (GWT), we write our AJAX front-end in the Java programming language using the Java development tools of our choice. If GWT's class library doesn't meet our needs, we can mix handwritten JavaScript in our Java source code using the **JavaScript Native Interface (JSNI)**. GWT then cross-compiles this client-side code into optimized JavaScript that automatically works across all major browsers.

Using GWT you can build an entire web application from scratch or build single widgets and integrate them into existing web pages. GWT ships with a special *hosted mode* web browser to use while developing and debugging. You can iterate quickly in the same "edit - refresh - view" cycle you're accustomed to with JavaScript, with the added benefit of being able to debug and step through your Java code line by line. In production, your code is compiled to JavaScript, but in hosted mode it runs in the Java virtual machine. That means when your code performs an action like handling a mouse event, you get full-featured Java debugging, with exceptions and the advanced debugging features of IDEs like **Eclipse**.

2.2) SQL vs. NoSQL

SQL	**NoSQL**
Schema-oriented.	Schema-free.
Uses RDBMS which is a great tool for solving ACID problems.	NoSQL is a great tool for solving data availability problems.
We use SQL when data validity is important and we need to support dynamic queries.	We use NOSQL when it's more important to have fast data than right data and we need to scale based on changing requirements.
Scalability is limited.	Highly Scalable.

Table 1: SQL vs. NoSQL

2.3) CouchDB vs. MongoDB

CouchDB	**MongoDB**
Eventual Consistency	Strict Consistency
Master-Master Replication	Master-Slave Replication
Favours Availability (all clients can always read and write).	Favours Consistency (each client always has the same view of the data).
Couch maintains a different document for every update you make.	Mongo maintains a single document and update it after every update you make**.**
CouchDB is safer out of the box.	MongoDB is faster out of the box.

Table 2: CouchDB vs. MongoDB

2.4) What is CouchDB?

We can use couch as our database. Apache CouchDB is a NoSQL database which is written in Erlang. It uses JSON for documents. It is an open source database and is document-oriented as it used JSON for documents. Store your data with JSON documents. Access your documents with your web browser, via HTTP Query, combine and transform your documents with JavaScript. CouchDB works well with modern web and mobile apps. We can even serve web apps directly out of CouchDB. And we can distribute our data, or our apps, efficiently using CouchDB incremental replication. CouchDB supports master-master setups with automatic conflict detection.

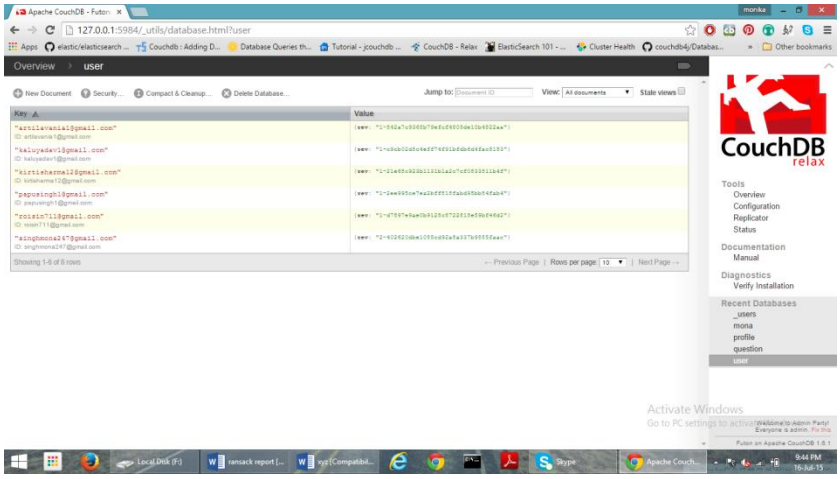


Fig 1

2.4.1) Concept of Map-Reduce in CouchDB

The concept of CouchDB views is actually very elegant. It's a purely functional map of the documents in the database. This means you can process the data any way you like using JavaScript, but CouchDB can make assumptions about data freshness, and safely cache and index the results to provide efficient queries and updates (at least in theory ;-).

Unfortunately you can only create a view from the original data, there is no way to create views whose input is other views. This means that you cannot do anything really interesting with values from multiple documents. You can aggregate data from several

documents using the reduce functionality into buckets, but you can't process that data further.

The reduce functionality alleviates this somewhat, but personally I feel this is a bit of a kludge (reduce is really just a special case of map: map takes data and outputs data to buckets using a key. Reduce is a map where the data is the resulting buckets from the previous pass and the output).

2.5) ECLIPSE IDE: Luna

In computer programming, **Eclipse** is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications.

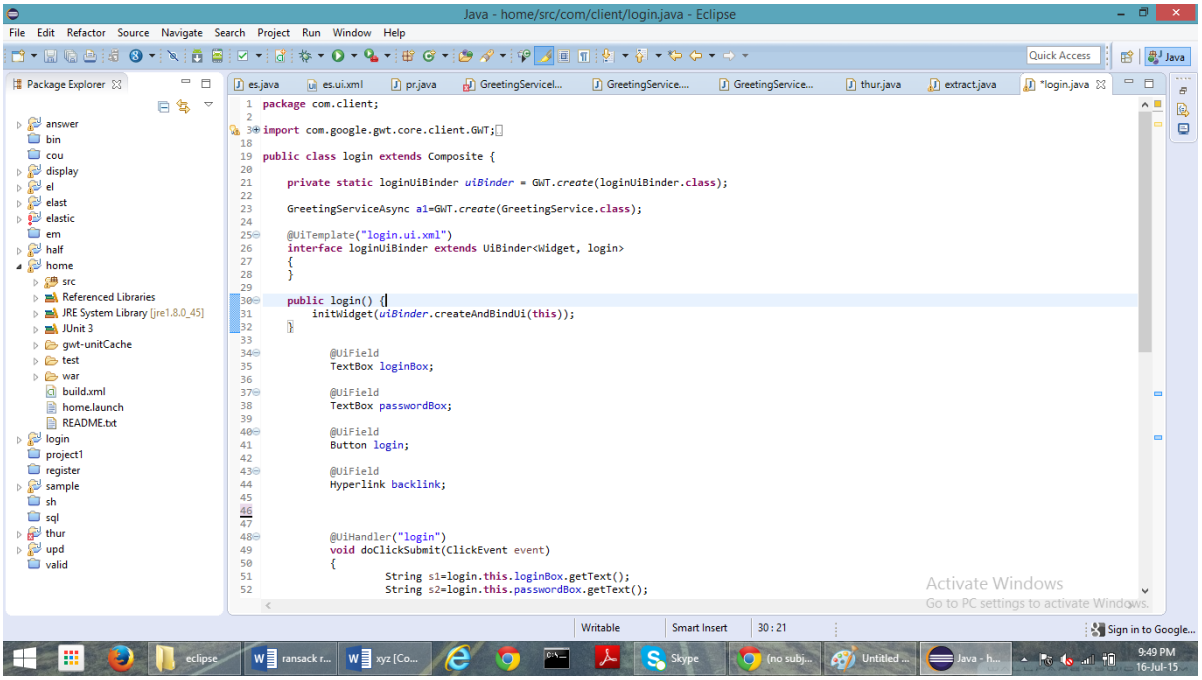


Fig 1

2.6) JETTY SERVER

Jetty is a pure Java-based HTTP (Web) server and Java Servlet container. While Web Servers are usually associated with serving documents to humans, Jetty is now often used for machine to machine communications, usually within larger software frameworks. Jetty is developed as a free and open source project as part of the Eclipse Foundation.

2.6.1) ARCHITECTURE OF JETTY

The Jetty Server is the plumbing between a collection of Connectors that accept HTTP connections and a collection of Handlers that service requests from the connections and produce responses, with threads from a thread pool doing the work.

While the Jetty request/responses are derived from the Servlet API, the full features of the Servlet API are only available if you configure the appropriate handlers. For example, the session API on the request is inactive unless the request has been passed to a Session Handler. The concept of a Servlet itself is implemented by a Servlet Handler. If Servlets are not required, there is very little overhead in the use of the Servlet request/response APIs. Thus you can build a Jetty server using only connectors and handlers, without using Servlets.

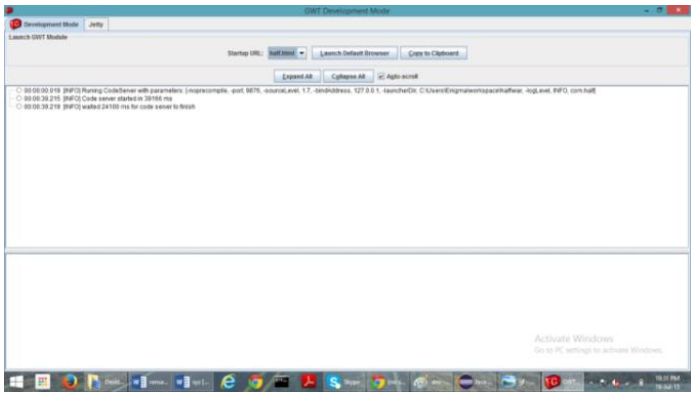


Fig 1

2.7) ELASTIC SEARCH

Have you ever needed to change mappings or gain performance benefits of one of the latest Elasticsearch data structures? Elasticsearch 2.3 is for you. In addition to our brand-new reindex API that can reindex data without it ever leaving the cluster, 2.3 include all of the latest advances in speed, security, scalability, and hardware efficiency. Plus it has a brand-new built-in task management API and is compatible with the 2.3 versions of Shield (security), Marvel (monitoring), Watcher (alerts), and more.

2.7.1) Features of Elasticsearch

2.7.1.1) Real-Time Data

How long can you wait for insights on your fast-moving data? With Elasticsearch, all data is immediately made available for search and analytics.

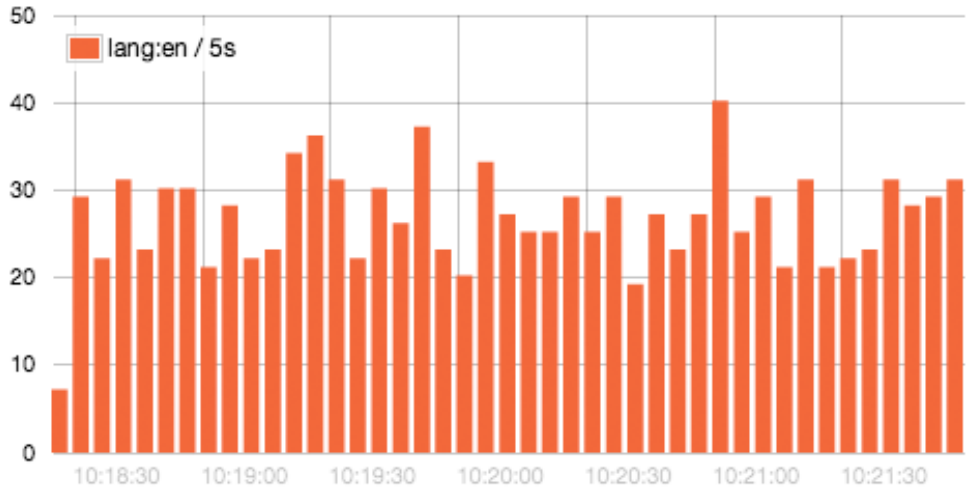


Fig 1

2.7.1.2) Real-Time Advanced Analytics

Combining the speed of search with the power of analytics changes your relationship with your data. Interactively search, discover, and analyze to gain insights that improve your products or streamline your business.

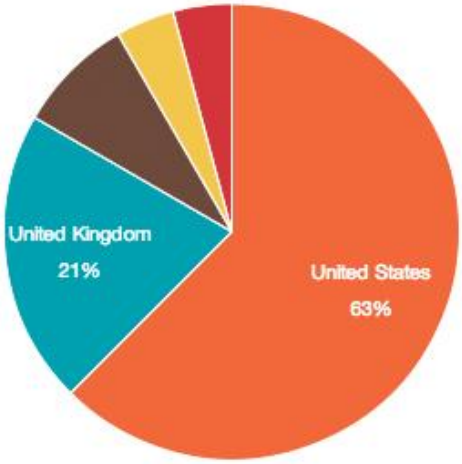


Fig 2

2.7.1.3) Massively Distributed

Elasticsearch allows you to start small and scale horizontally as you grow. Simply add more nodes, and let the cluster automatically take advantage of the extra hardware. Petabytes of data? Thousands of nodes? No problem.

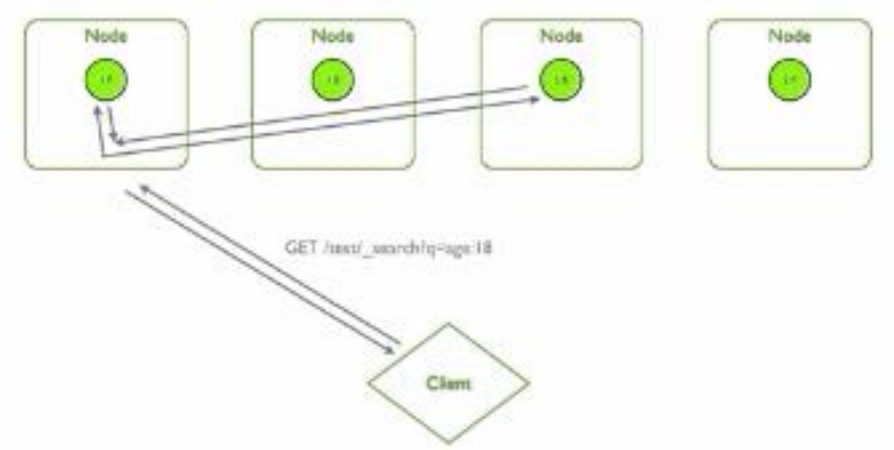


Fig 3

2.7.1.4) High Availability

Elasticsearch clusters are resilient — they will detect new or failed nodes, and reorganize and rebalance data automatically, to ensure that your data is safe and accessible.

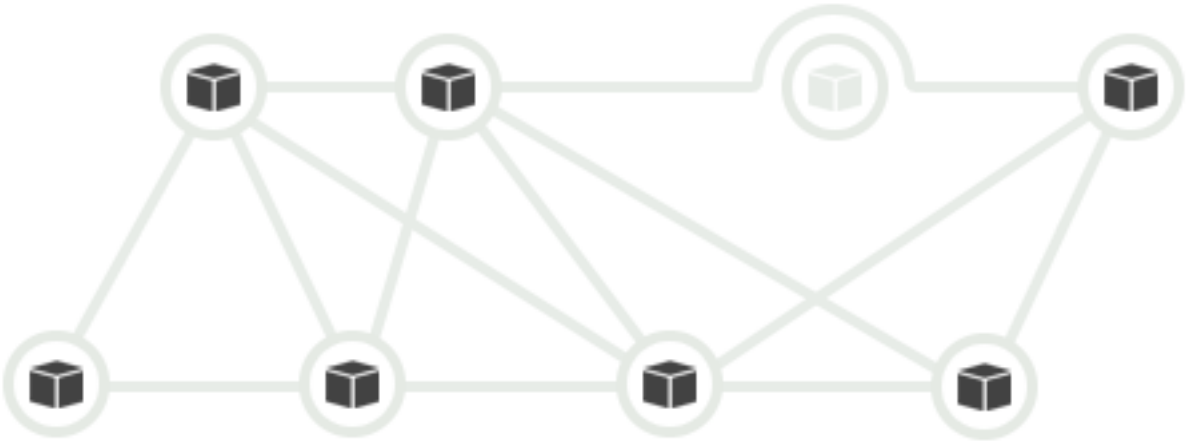


Fig 4

2.7.1.5) Multitenancy

A cluster may contain multiple indices that can be queried independently or as a group. Index aliases allow filtered views of an index, and may be updated transparently to your application.

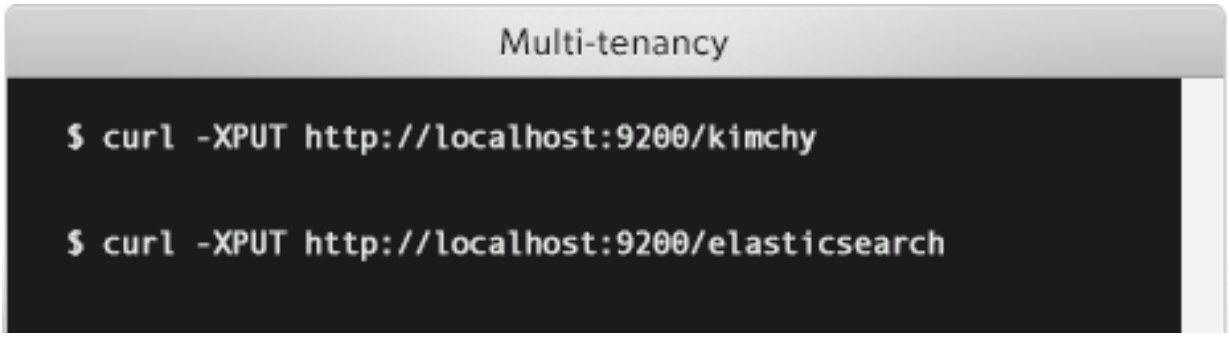


Fig 5

2.7.1.6) Full-Text Search

Elasticsearch builds distributed capabilities on top of Apache Lucene to provide the most powerful full- text search capabilities available. Powerful, developer-friendly query API supports multilingual search, geolocation, contextual did-you-mean suggestions, and auto complete, and result snippets.

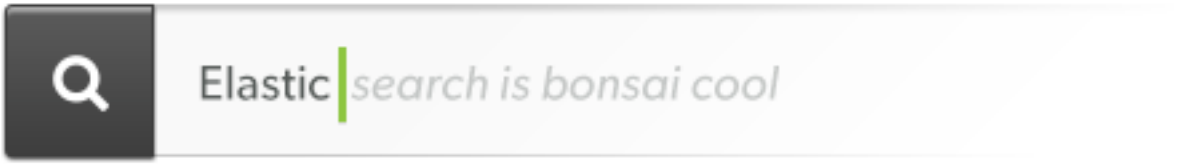


Fig 6

2.7.1.7) Document-Oriented

Store complex real world entities in Elasticsearch as structured JSON documents. All fields are indexed by default, and all the indices can be used in a single query, to easily return complex results at breathtaking speed.

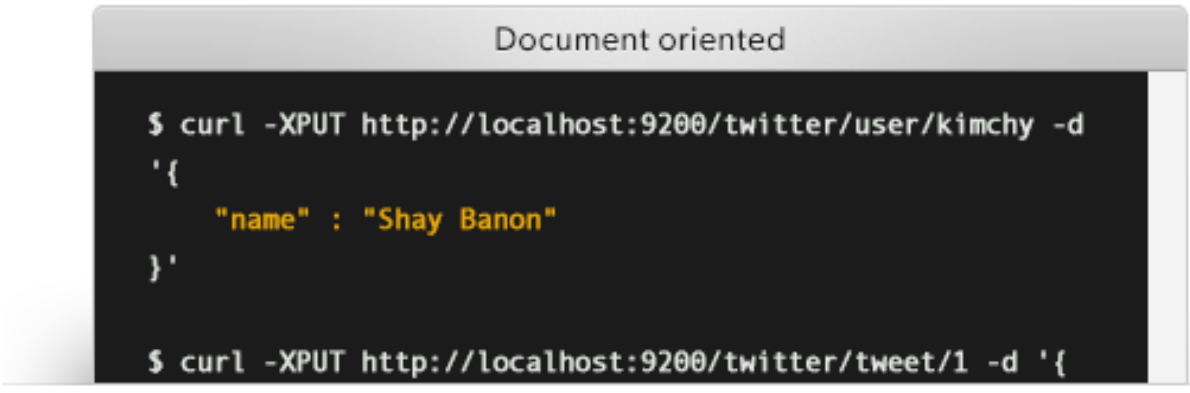


Fig 7

2.7.1.8) Schema-Free

Elasticsearch allows you to get started fast. Simply index a JSON document and it will automatically detect the data structure and types, create an index, and make your data searchable. You also have full control to customize how your data is indexed.

Fig 8

2.7.1.9) Developer-Friendly, RESTful API

Elasticsearch is API driven. Almost any action can be performed using a simple RESTful API using JSON over HTTP. Client libraries are available for many programming languages.

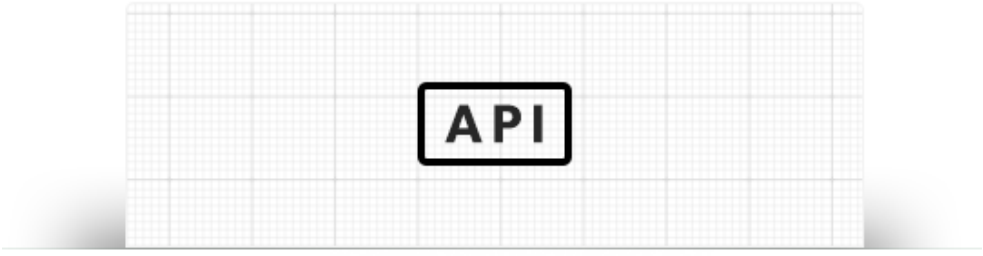


Fig 9

2.7.1.10) Per-Operation Persistence

Elasticsearch puts your data safety first. Document changes are recorded in transaction logs on multiple nodes in the cluster to minimize the chance of any data loss.

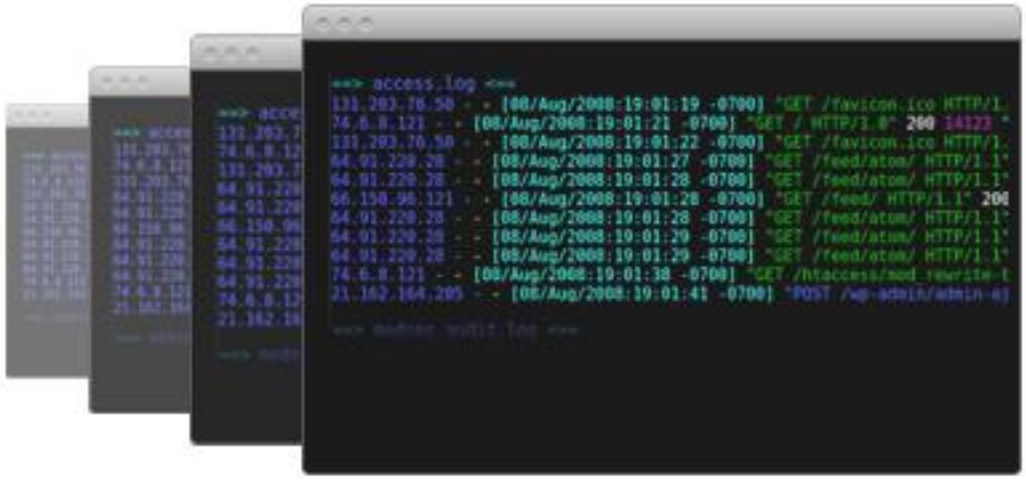


Fig 10

2.7.1.11) Apache 2 Open Source License

Elasticsearch can be downloaded, used, and modified free of charge. It is available under the Apache 2 license, one of the most flexible open source licenses available.

Fig 11

2.7.1.12) Build on top of Apache Lucene™

Apache Lucene is a high performance, full-featured Information Retrieval library, written in Java. Elasticsearch uses Lucene internally to build its state of the art distributed search and analytics capabilities.

Fig 12

2.7.1.13) WHO IS USING ELASTIC SEARCH?

Fig 13

2.8) Use Postman REST Client for HTTP Requests

Some find curl statements and the command line, used for most of the examples in this section of the documentation, difficult and intimidating. For those, there are numerous tools to send HTTP requests to a REST-based service.

2.8.1) Install Postman

In Chrome select **Window > Extensions**. Click the **Get More Extensions** link on the page and search for Postman. Follow the directions to install the extension.

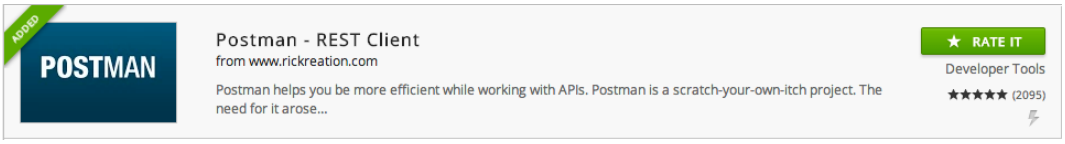


Fig 1

2.8.2) Start Postman

Once you have installed Postman, start the app by going to the **Apps** bookmark, find the **Postman - REST Client** link, and then click it.

2.8.3) Send GET request

To send a **GET** request, follow these steps (see screenshot following the steps for a look at the interface):

1. Be sure the **Normal** tab is selected, which is the default.
2. Supply the request URL. Be sure to place the actual values in the URL as the environment variables used in the curl statements will not be defined here.
3. Choose **GET** to be the selected HTTP method.
4. Click the **Headers** button, and supply the Header - Value pair **Content-Type - application/json**.
5. Click **Send**.
6. You will be presented with a pop-up window to enter your credentials. Do so and click **OK**.

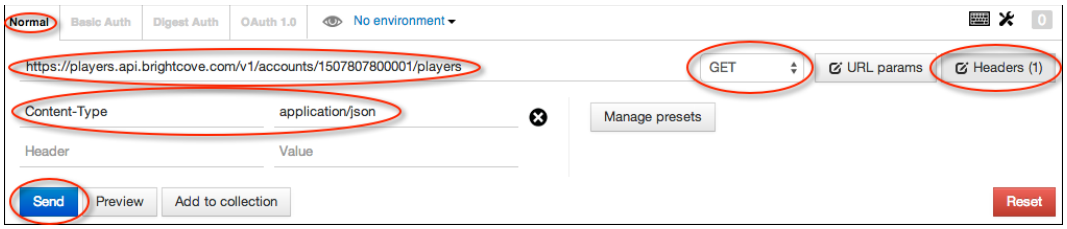


Fig 2

Your response will look similar to the following:

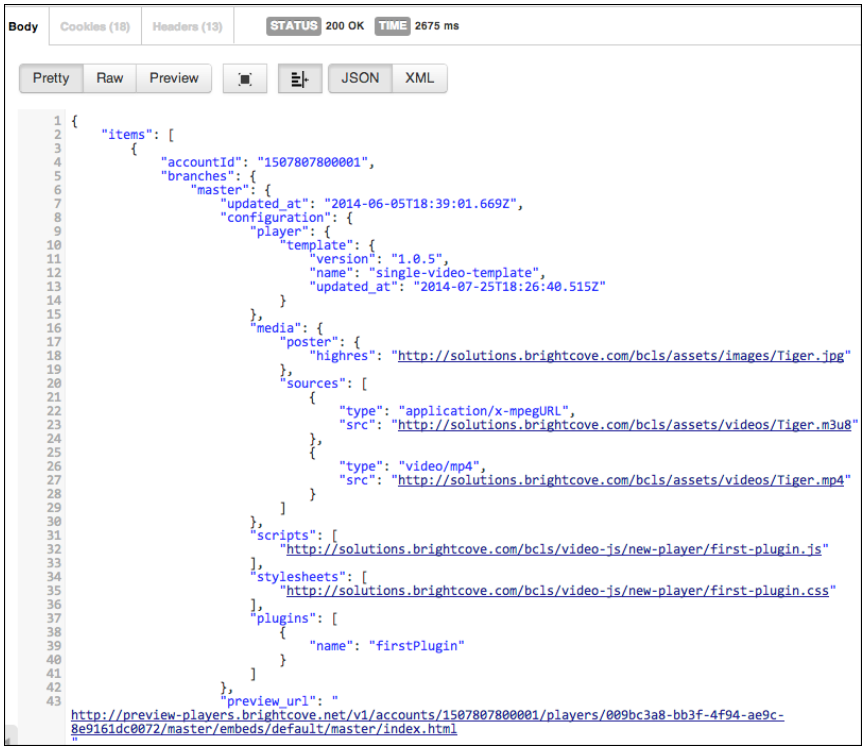


Fig 3

2.8.4) Send POST request

To send a **POST** request with data associated with it, follow these steps (see screenshot following the steps for a look at the interface):

1. Be sure the **Normal** tab is selected, which is the default.
2. Supply the request URL. Be sure to place the actual values in the URL as the environment variables used in the curl statements will not be defined here.
3. Choose **POST** to be the selected HTTP method.
4. Click the **Headers** button, and supply the Header - Value pair **Content-Type** - **application/json**.
5. Click the **URL params** button. For parameter configuration first select **raw**, then from the dropdown select **JSON**.
6. Paste the configuration JSON into the area provided.
7. Click **Send**.
8. (If you have recently submitted a request you may not be asked to authenticate again.) You will be presented with a pop-up window to enter your credentials. Do so and click **OK**.

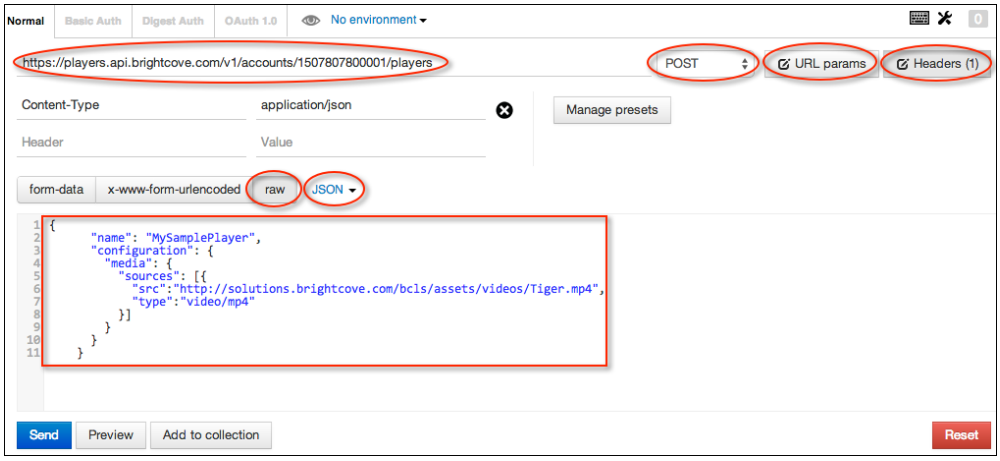


Fig 4

Your response will look similar to the following:

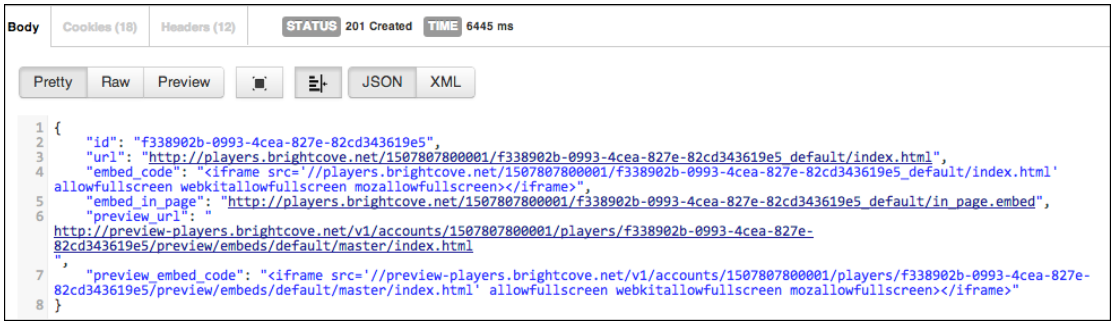


Fig 5

3. SYSTEM DEVELOPMENT

3.1) SOFTWARE REQUIREMENTS:

- **Developer side:**
- JDK 1.7.0 update u55 (required for ElasticSearch 0.90 or above).
- Elastic search version **0.90** or higher.
- Google web toolkit.
- Latest version of eclipse IDE.
- Postman REST Client.
- Composer.
- Wamp Server.

- **Server Side:**
- ElasticSearch head plugin.
- Postman REST Client.
- Composer.
- Wamp Server.

- **Client Side:**
- Internet Connection
- Web Browser

3.2) HARDWARE REQUIREMENTS:

- CPU: 2.2GHz Processor and above.
- CACHE: 2 MB or above.
- RAM: 2 GB or above.
- OS: Windows 7 or above, Ubuntu 12.04 or later.

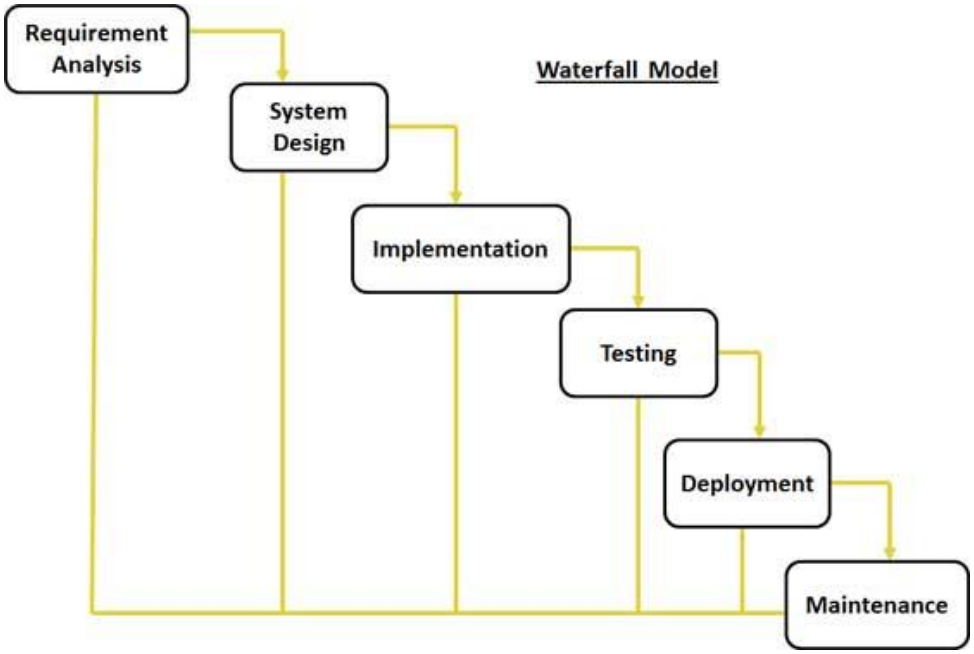
3.3) SDLC MODEL:

Software Development Life Cycle (SDLC) is a process used by software industry to design, develop and test high quality software. The SDLC aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software development process.
- The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

A typical Software Development life cycle consists of the following stages:

- Stage 1: Planning and Requirement Analysis
- Stage 2: Defining Requirements
- Stage 3: Designing the product architecture
- Stage 4: Building or Developing the Product
- Stage 5: Testing the Product

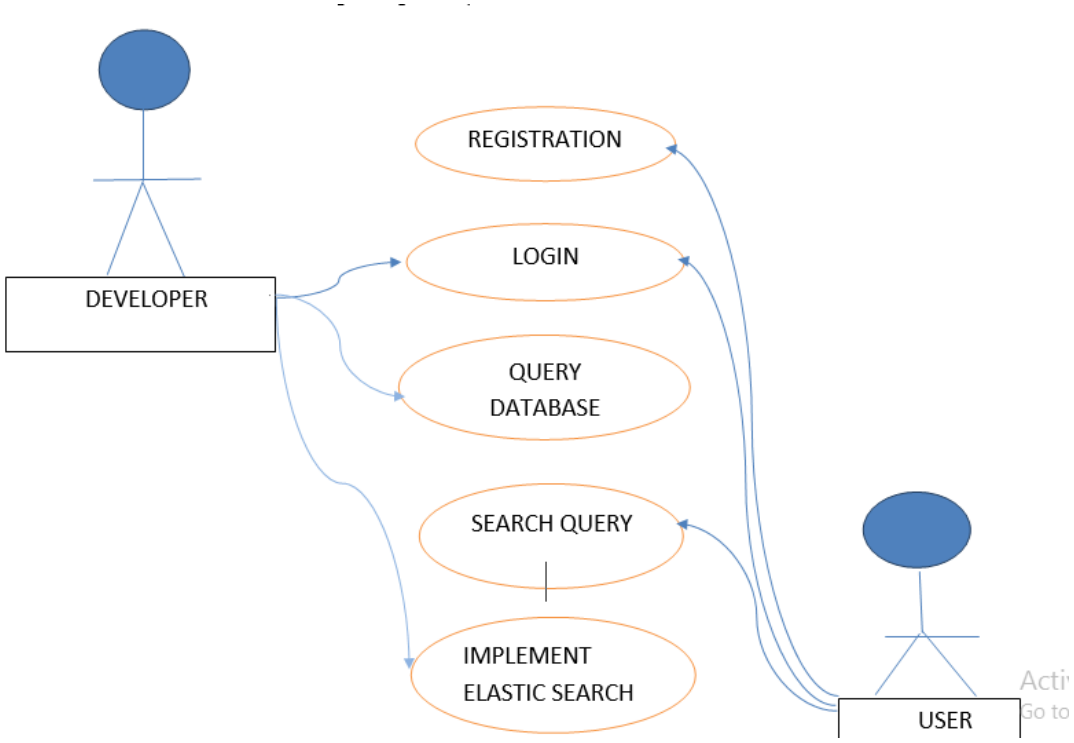


- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released.

3.4) SYSTEM DESIGN:

3.4.1) USECASE DIAGRAM

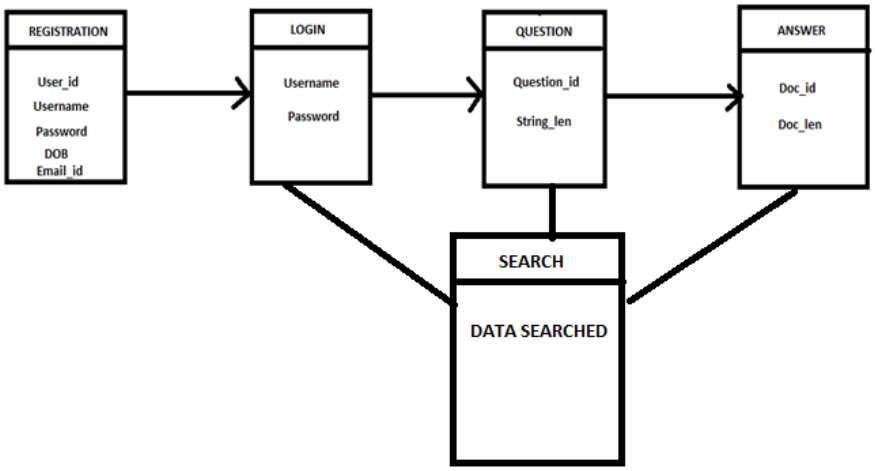
A use case diagram is a list of steps, typically defining interaction between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human or an external system.



3.4.2) SCHEMA DIAGRAM

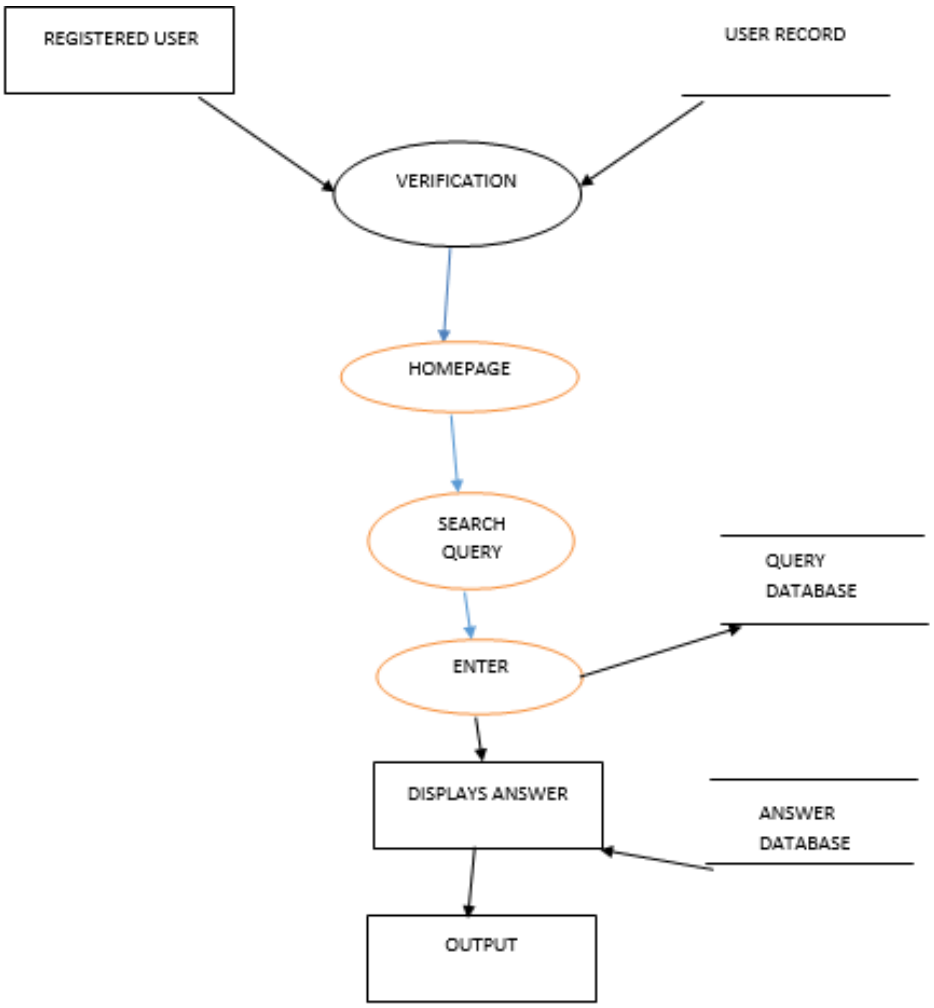
In a schema diagram, all database tables are designated with unique columns and special features, e.g., primary/foreign keys or not null, etc. Formats and symbols for expression are universally understood, eliminating the possibility of confusion. The table relationships also are expressed via a parent table's primary key lines when joined with the child table's corresponding foreign keys.

Schema diagrams have an important function because they force database developers to transpose ideas to paper. This provides an overview of the entire database, while facilitating future database administrator work.



3.4.3) DATA FLOW DIAGRAM

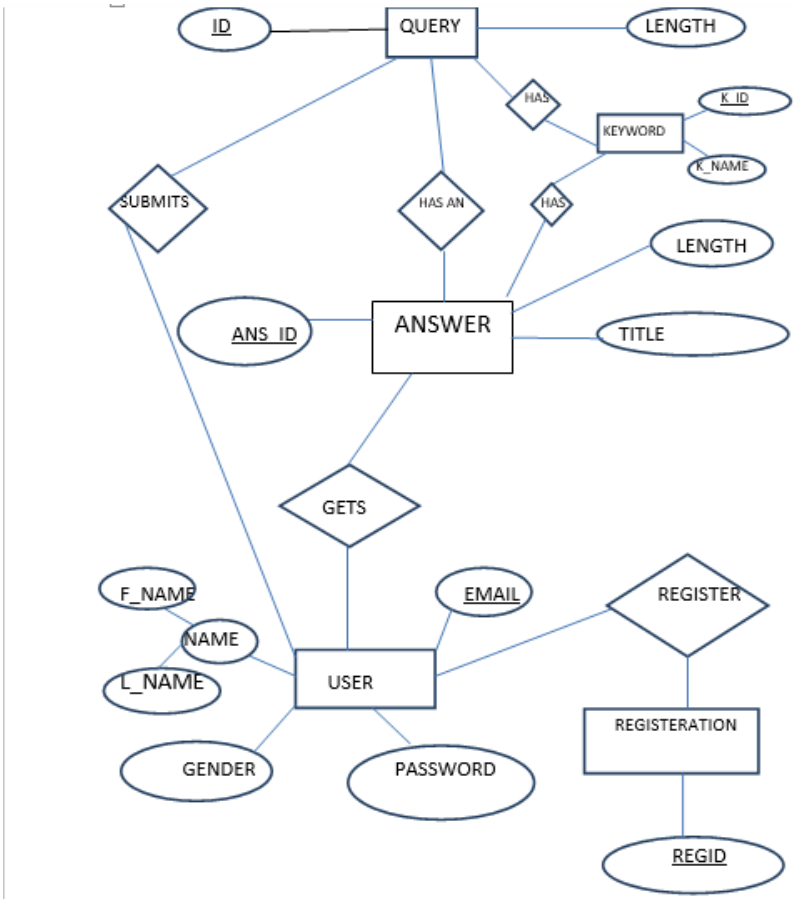
Data Flow Diagrams (DFD) helps us in identifying existing business processes. It is a technique we benefit from particularly before we go through business process re-engineering. At its simplest, a data flow diagram looks at how data flows through a system. It concerns things like where the data will come from and go to as well as where it will be stored. But you won't find information about the processing timing (e.g. whether the processes happen in sequence or in parallel).



3.4.4) ER DIAGRAM

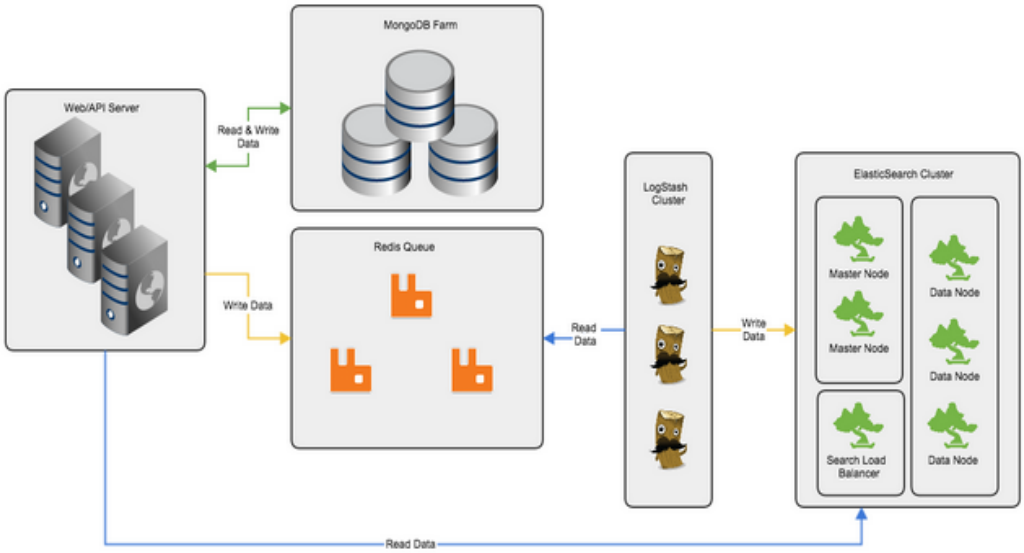
An entity–relationship model is a systematic way of describing and defining a business process. The process is modelled as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

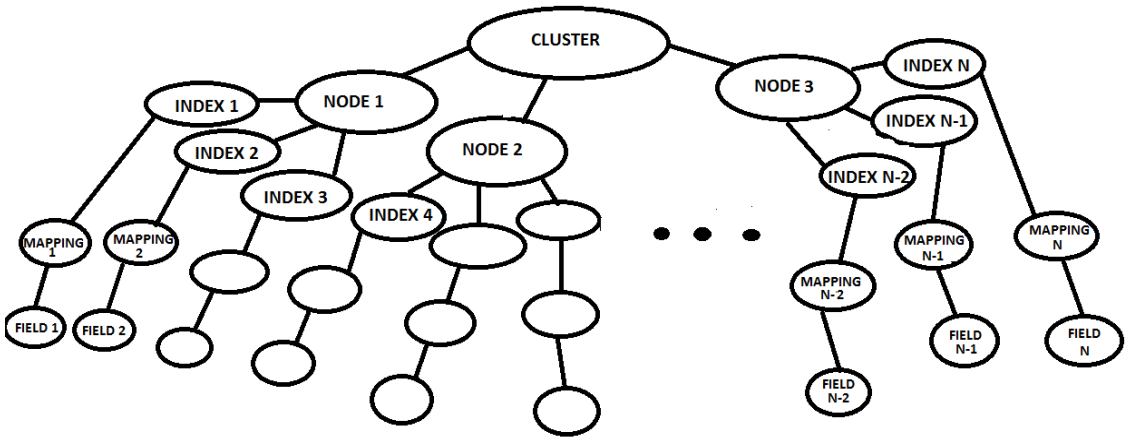
An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers represent the relationships.



3.4.5) ARCHITECTURE DIAGRAM

The API and UI servers push the data to both the database and the message queue simultaneously. Logstash subscribes to the channel to which the API server is writing, gets notified of incoming messages, and then finally pushes the messages to ElasticSearch. This is similar to adding data to Memcache for fast access and then persisting the same data in the database.



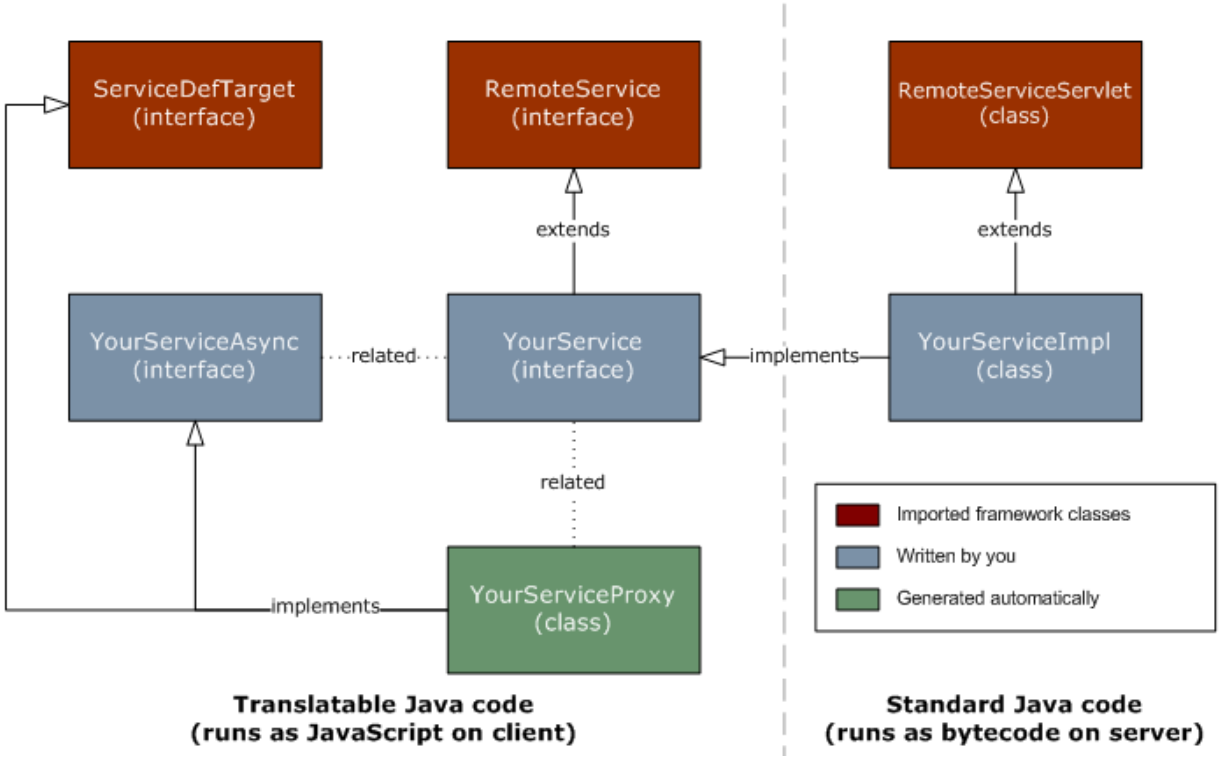


3.5) IMPLEMENTATION:

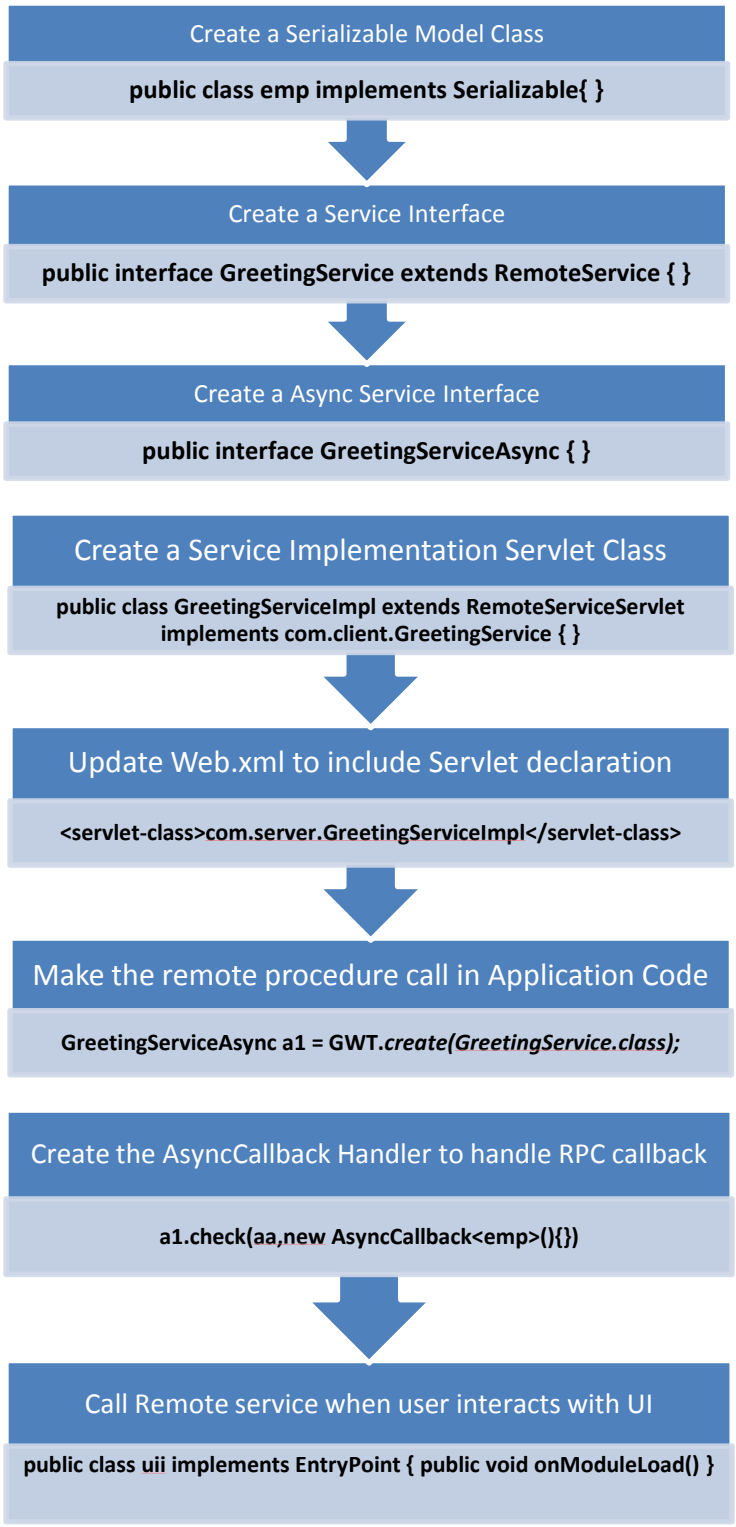
3.5.1) Remote Procedural Call (RPC)

- RPC, Remote Procedure Call is the mechanism used by GWT in which client code can directly executes the server side methods.
- GWT RPC is servlet based.
- GWT RPC is asynchronous and client is never blocked during communication.
- Server-side servlet is termed as service.
- Remote procedure call that is calling methods of server side servlet from client side code is referred to as invoking a service.

3.5.2) GWT-RPC Architecture



3.5.3) RPC Communication Workflow



3.5.4) Document validation (in CouchDb)

Design document is a place where you can define document validation function:

```
{
   "_id": "_design/assets",
   "language": "javascript",
   "validate_doc_update":"function(newDoc, oldDoc, userCtx) {
        if (newDoc.address === undefined) {
          throw({forbidden: 'Document must have an address.'});
        }"
   }
}
```

You should be aware of one thing that documents coming from replication are also validated, so the validation must be the same for all replicating instances. When validation for a single document fails, the replication process simply goes with the next documents. Effectively, the change is silently ignored, leading to hard to find and fix inconsistency.

3.5.5) Querying data (in CouchDb)

You can't simply query CouchDB with "dynamic" one. All possible queries should be defined earlier, which can be done also in a "design document". Such queries are named "views" and are actually B-tree indexes. Those indexes are defined in map/reduces manner.

```
Session dbSession = new Session("127.0.0.1",5984);
String dbname= "question";
ArrayList<emp> a= new ArrayList<emp>();
Database db = dbSession.getDatabase(dbname);
ViewResults vr = db.getAllDocuments();
List<Document> sd = vr.getResults();
for(Document cd:sd){
String id=
cd.getJSONObject().getString("id");
Document doc = db.getDocument(id);
String s1=doc.get("Ques").toString();
emp k = new emp();
k.setQues(s1);
k.setId(id);
```

3.5.6) Dealing with JSON Arrays and Objects in PHP (in Elasticsearch)

A common source of confusion with the client revolves around JSON arrays and objects, and how to specify them in PHP. In particular, problems are caused by empty objects and arrays of objects. This page will show you some common patterns used in Elasticsearch JSON API, and how to convert that to a PHP representation

3.5.6.1) Empty Objects

The Elasticsearch API uses empty JSON objects in several locations, and this can cause problems for PHP. Unlike other languages, PHP does not have a "short" notation for empty objects and so many developers are unaware how to specify an empty object.

Consider adding a Highlight to a query:

```
{
    "query" : {
        "match" : {
        "content" : "quick brown fox"
        }
    },
    "highlight" : {
        "fields" : {
        "content" : {}
      }
  }
  }
```

The problem is that PHP will automatically convert "content": {} into "content": [], which is no longer valid Elasticsearch DSL. We need to tell PHP that the empty object is explicitly an object, not an array. To define this query in PHP, you would do:

```
$params['body'] = array(
    'query' => array(
        'match' => array(
            'content' => 'quick brown fox'
        )
    ),
    'highlight' => array(
        'fields' => array(
```

3.5.7) Index Management Operations (in Elasticsearch)

Index management operations allow you to manage the indices in your Elasticsearch cluster, such as creating, deleting and updating indices and their mappings/settings.

3.5.7.1) Create an index

The index operations are all contained under a distinct namespace, separated from other methods that are on the root client object. As an example, let's create a new index:

```
$client = ClientBuilder::create()->build();
$params = [
    'index' => 'my_index'
];
// Create the index
$response = $client->indices()->create($params);
```

You can specify any parameters that would normally be included in a new index creation API. All parameters that would normally go in the request body are located in the body parameter:

```
$client = ClientBuilder::create()->build();
$params = [
    'index' => 'my_index',
    'body' => [
        'settings' => [
            'number_of_shards' => 3,
            'number_of_replicas' => 2
        ],
        'mappings' => [
            'my_type' => [
                '_source' => [
                    'enabled' => true
                ],
                'properties' => [
                    'first_name' => [
                        'type' => 'string',
                        'analyzer' => 'standard'
                    ]
```

3.5.8) Indexing Documents (in Elasticsearch)

When you add documents to Elasticsearch, you index JSON documents. This maps naturally to PHP associative arrays, since they can easily be encoded in JSON. Therefore, in Elasticsearch-PHP you create and pass associative arrays to the client for indexing. There are several methods of ingesting data into Elasticsearch, which we will cover here

3.5.8.1) Single document indexing

When indexing a document, you can either provide an ID or let Elasticsearch generate one for you.

Providing an ID value.

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'id' => 'my_id',
    'body' => [ 'testField' => 'abc']
];
```

Omitting an ID value.

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'body' => [ 'testField' => 'abc']
];
// Document will be indexed to
my_index/my_type/<autogenerated ID>
$response = $client->index($params);
```

If you need to set other parameters, such as a routing value, you specify those in the array alongside the index, type, etc. For example, let's set the routing and timestamp of this new document:

Additional parameters.

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'id' => 'my_id',
    'routing' => 'company_xyz',
    'timestamp' => strtotime("-1d"),
```

3.5.8.2) Bulk Indexing

Elasticsearch also supports bulk indexing of documents. The bulk API expects JSON action/metadata pairs, separated by newlines. When constructing your documents in PHP, the process is similar. You first create an action array object (e.g. index object), then you create a document body object. This process repeats for all your documents.

A simple example might look like this:

Bulk indexing with PHP arrays.

```
for($i = 0; $i < 100; $i++) {
    $params['body'][] = [
        'index' => [
            '_index' => 'my_index',
            '_type' => 'my_type',
        ]];$params['body'][] = [
        'my_field' => 'my_value',
        'second_field' => 'some more values'
    ];}$responses = $client->bulk($params);
```

In practice, you'll likely have more documents than you want to send in a single bulk request. In that case, you need to batch up the requests and periodically send them:

Bulk indexing with batches

```
$params = ['body' => []];
for ($i = 1; $i <= 1234567; $i++) {
    $params['body'][] = [
        'index' => [
            '_index' => 'my_index',
            '_type' => 'my_type',
            '_id' => $i
        ]
    ];
    $params['body'][] = [
        'my_field' => 'my_value',
        'second_field' => 'some more values'
    ];
    // Every 1000 documents stop and send the bulk request
    if ($i % 1000 == 0) {
        $responses = $client->bulk($params);
```

3.5.9) Getting Documents (in Elasticsearch)

Elasticsearch provides real-time GETs of documents. This means that as soon as the document has been indexed and your client receives an acknowledgement, you can immediately retrieve the document from any shard. Get operations are performed by requesting a document by its full index/type/id path:

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'id' => 'my_id'
];
```

3.5.10) Search Operations (in Elasticsearch)

Well…it isn't called Elasticsearch for nothing! Let's talk about search operations in the client.

The client gives you full access to every query and parameter exposed by the REST API, following the naming scheme as much as possible. Let's look at a few examples of the syntax.

3.5.10.1) Match Query

Here is a standard curl for a Match query:

```
curl-XGET
'localhost:9200/my_index/my_type/_search' -d '{
    "query" : {
        "match" : {
            "testField" : "abc"
        }
```

And here is the same query constructed in the client:

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'body' => [
        'query' => [
            'match' => [
                'testField' => 'abc'
            ]
```

Notice how the structure and layout of the PHP array is identical to that of the JSON request body. This makes it very simple to convert JSON examples into PHP. A quick method to check your PHP array (for more complex examples) is to encode it back to JSON and check by eye:

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'body' => [
        'query' => [
            'match' => [
                'testField' => 'abc'
            ]
```

3.5.10.2) Using Raw JSON

Sometimes it is convenient to use raw JSON for testing purposes, or when migrating from a different system. You can use raw JSON as a string in the body, and the client will detect this automatically:

```
            $json = '{
                "query" : {
                    "match" : {
                        "testField" : "abc"
                    }
                }
            }';
$params = [
                'index' => 'my_index',
                'type' => 'my_type',
                'body' => $json
            ];
$results = $client->search($params);
```

Search results follow the same format as Elasticsearch search response, the only difference is that the JSON response is serialized back into PHP arrays. Working with the search results is as simple as iterating over the array values:

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'body' => [
        'query' => [
            'match' => [
                'testField' => 'abc'
            ]
        ]
    ]
```

3.5.10.3) Bool Queries

Bool queries can be easily constructed using the client. For example, this query:

```
curl -XGET
'localhost:9200/my_index/my_type/_search' -d '{
    "query" : {
        "bool" : {
            "must": [
                {
                    "match" : { "testField" : "abc" }
                },
                {
                    "match" : { "testField2" : "xyz" }
```

Would be structured like this:

```
$params = [
    'index' => 'my_index',
    'type' => 'my_type',
    'body' => [
        'query' => [
            'bool' => [
                'must' => [
                    [ 'match' => [ 'testField' => 'abc' ] ],
                    [ 'match' => [ 'testField2' => 'xyz' ] ],
                ]
            ]
        ]
    ]
];
```

3.6) WORKING OF THE PROJECT

3.6.1) UI design using UI Binder

Fig 1

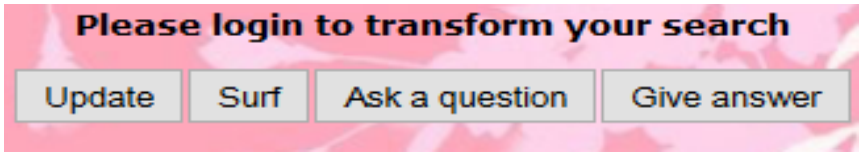


Fig 2

3.6.2) Post a Question in CouchDB

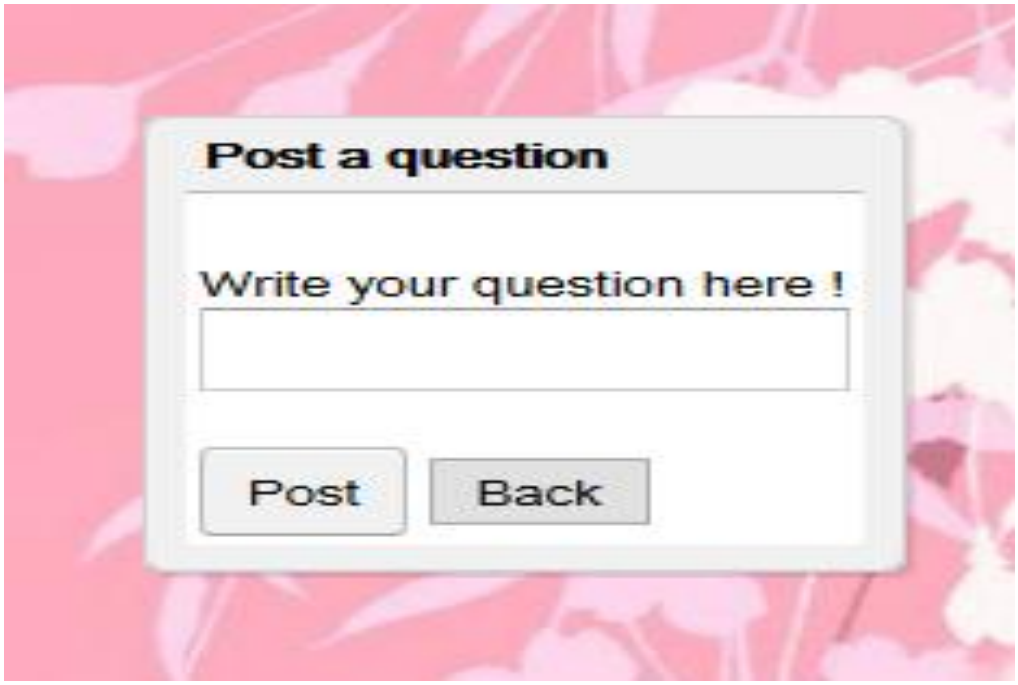


Fig 3

3.6.3) Search a Query in CouchDB

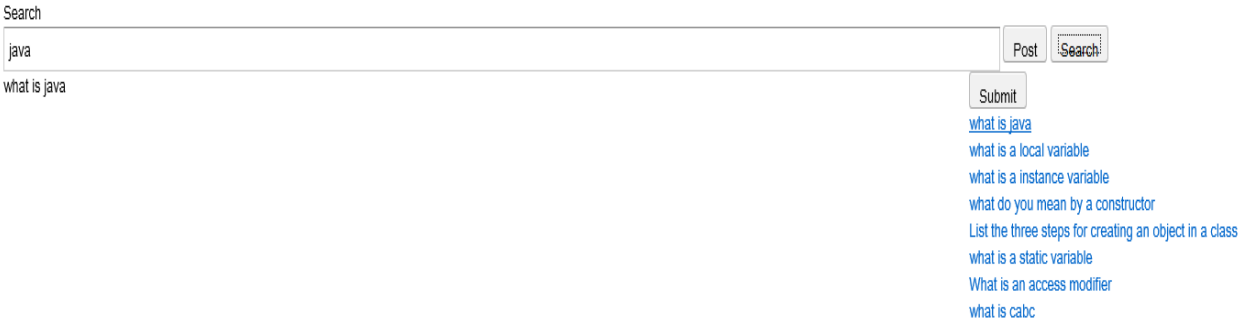


Fig 4

3.6.4) Add Text data to the Elastic Cluster

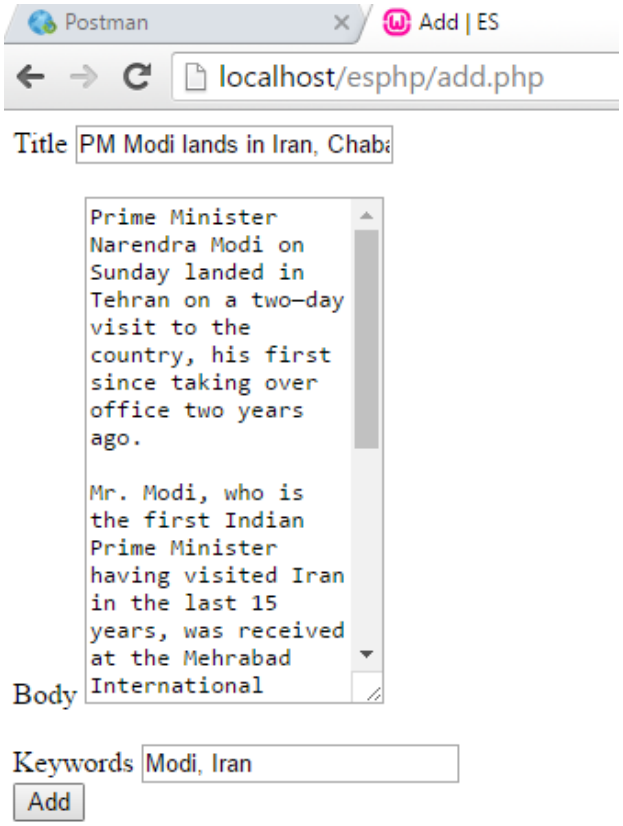


Fig 5

3.6.5) Text data is inserted in JSON format in the Elastic Cluster

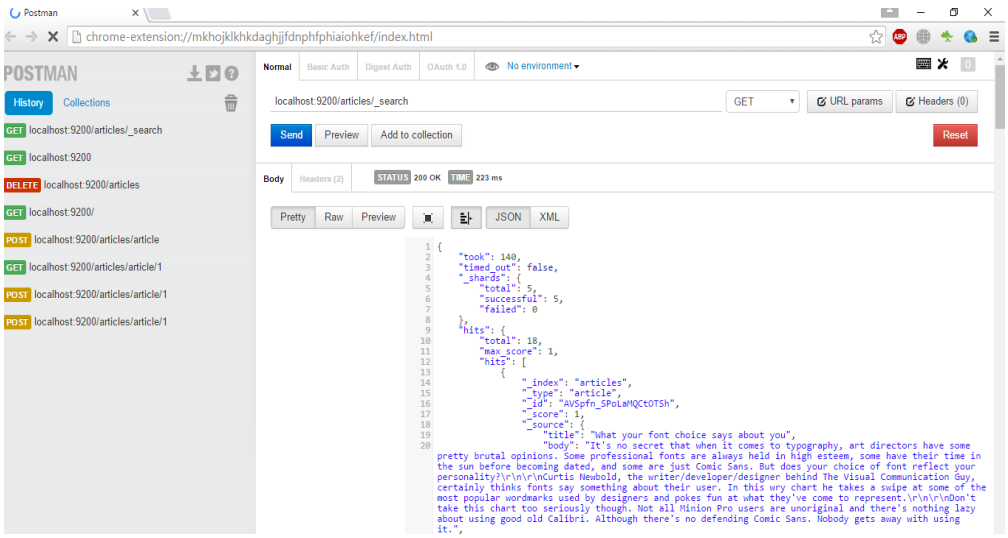


Fig 6

3.6.6) Retrieval of relevant Documents on the basis of Keywords

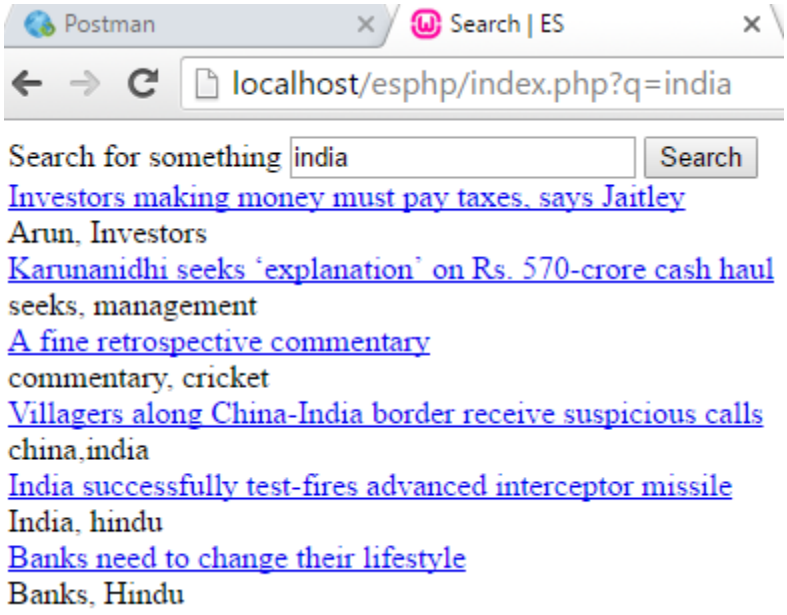


Fig 7

4. PERFORMANCE ANALYSIS

4.1) SYSTEM TESTING

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

It is very much similar functional test case writing. In test case writing you should write the test scenarios & use cases.

4.1.1) BLACK BOX TESTING:-

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings.

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

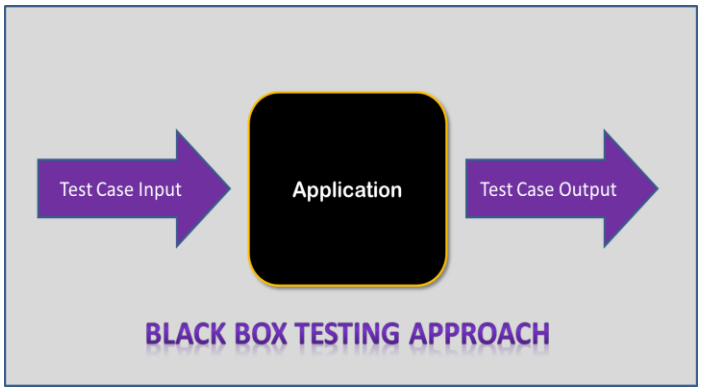


Fig 1

4.1.2) UNIT TESTING:-

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.

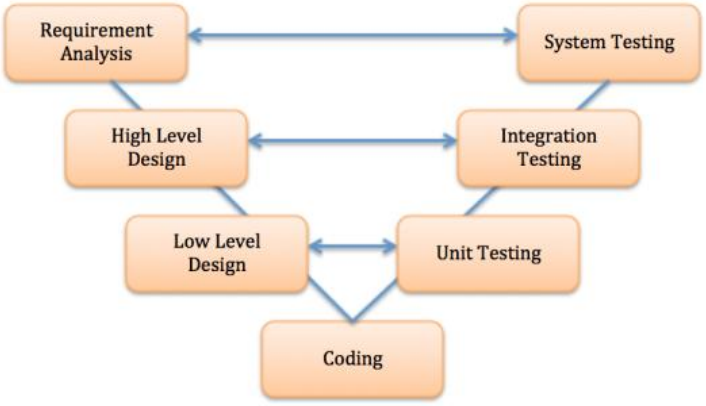


Fig 2

4.2) TEST CASES

Test Case 1

Module	Registration
Operating System	Windows 7
Environment	JDK and JRE version 1.7, CouchDB, Eclipse Luna with GWT plug-in, Chrome browser, Elastic search1.6.0
Input	Email, Password, Username
Output	All the fields get stored in CouchDB and login page appears.
Expected Output	All the fields get stored in CouchDB and login page appears.

Test Case 2

Module	Login
Operating System	Windows 7
Environment	JDK and JRE version 1.7, CouchDB, Eclipse Luna with GWT plug-in, Chrome browser, Elastic search1.6.0
Input	Username, Password
Output	Both the fields get validated with the data stored in CouchDB and homepage of user appears.
Expected Output	Login complete

Test Case 3

Module	Post Question and Multiple Answers
Operating System	Windows 7
Environment	JDK and JRE version 1.7, CouchDB, Eclipse Luna with GWT plug-in, Chrome browser, Elastic search1.6.0
Input	Question and Answers.
Output	Both the fields get validated with the data stored in CouchDB and popup appears to user.
Expected Output	Both the fields get validated with the data stored in CouchDB and popup appears to user.

Test Case 4

Module	Retrieve Questions and Answers
Operating System	Windows 7
Environment	JDK and JRE version 1.7, CouchDB, Eclipse Luna with GWT plug-in, Chrome browser, Elastic search1.6.0
Input	Question and Answer keywords
Output	Questions and all its answers are displayed to the user.
Expected Output	Questions and all its answers are displayed to the user.

Test Case 5

Module	Insert Questions and Answers in Elasticsearch
Operating System	Windows 7
Environment	JDK and JRE version 1.7, CouchDB, Eclipse Luna with GWT plug-in, Chrome browser, Elastic search1.6.0
Input	Questions and Answers
Output	Data is inserted in JSON format
Expected Output	Data is inserted in JSON format

Test Case 6

Module	Implementation of Elasticsearch on Q&A Data
Operating System	Windows 7
Environment	JDK and JRE version 1.7, CouchDB, Eclipse Luna with GWT plug-in, Chrome browser, Elastic search1.6.0
Input	Keywords
Output	Relevant documents are fetched
Expected Output	Relevant documents are fetched

4.3) MAINTENANCE

Software maintenance is widely accepted part of SDLC now a days. It stands for all the modifications and updates done after the delivery of software product. There are number of reasons, why modification is required, some of them are briefly mentioned below:

- **Market Conditions** - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification.
- **Client Requirements** - Over the time, customer may ask for new features or functions in the software.
- **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.
- **Organization Changes** - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

4.3.1) Types of Maintenance

In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

- **Corrective Maintenance** - This includes modifications and updates done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.

- **Adaptive Maintenance** - This includes modifications and updates applied to keep the software product up-to date and tuned to the ever changing world of technology and business environment.

- **Perfective Maintenance** - This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.

5. CONCLUSION

CouchDB is useful for many areas of an application. Because of its incremental MapReduce and replication characteristics, it is especially well suited to online interactive document and data management tasks. These are the sort of workloads experienced by the majority of web applications. This coupled with CouchDB HTTP interface make it a natural fit for the web.

There is no right answer about which application development framework you should use with CouchDB. We've seen successful applications in almost every commonly used language and framework. For this example application, we'll use two-layer architecture: CouchDB as the data layer and the browser for the user interface. We think this is a viable model for many document-oriented applications, and it makes a great way to teach CouchDB, because we can easily assume that all of you have a browser at hand without having to ensure that you're familiar with a particular server-side scripting language.

Elasticsearch can be used not only as a search engine, but also as NoSQL storage for your data, you can interface with it using REST API and this makes it easy to integrate it with any backend system in any language. It accepts documents for indexing in JSON format. It's schema-free, meaning it can automatically derived document mappings at indexing time. For search, Elasticsearch supports such nifty features as real-time search, faceted search, query suggest, filtered query, highlighting, and custom score functions.

REFERENCES

[1] The Windows Club, 'Difference between SQL and NoSQL: A Comparison', 2011. [Online]. Available: http://www.thewindowsclub.com/difference-sql-nosql-comparision. [Accessed: 06- Oct- 2015].

[2] Nathan Hurst's Blog, 'Visual Guide to NoSQL Systems', 2015. [Online]. Available: http://blog.nahurst.com/visual-guide-to-nosql-systems. [Accessed: 07- Oct- 2015].

[3] Quora.com, 'What are the advantages and disadvantages of using MongoDB vs CouchDB vs Cassandra vs Redis? - Quora', 2015. [Online]. Available: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-MongoDB-vs-CouchDB-vs-Cassandra-vs-Redis. [Accessed: 25- Oct- 2015].

[4] Quora.com, 'What are the advantages and disadvantages of using MongoDB vs CouchDB vs Cassandra vs Redis? - Quora', 2015. [Online]. Available: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-MongoDB-vs-CouchDB-vs-Cassandra-vs-Redis. [Accessed: 15- Nov- 2015].

[5]"GWT Project", *Gwtproject.org*, 2016. [Online]. Available: http://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideRemoteProcedureCalls. [Accessed: 13- Mar- 2016].

[6]"GWT tutorial", *www.tutorialspoint.com*, 2016. [Online]. Available: http://www.tutorialspoint.com/gwt/. [Accessed: 25- Mar- 2016].

[7]"CouchDB Tutorial", *www.tutorialspoint.com*, 2016. [Online]. Available: http://www.tutorialspoint.com/couchdb/. [Accessed: 10- Apr- 2016].

[8]"GWTLecturer", *YouTube*, 2016. [Online]. Available: https://www.youtube.com/user/GWTLecturer. [Accessed: 21- Apr- 2016].

[9]"Easy Learn Tutorial", *YouTube*, 2016. [Online]. Available: https://www.youtube.com/user/easylearntutorial. [Accessed: 21- Apr- 2016].

[10]"Search Engine with PHP & Elasticsearch", YouTube, 2016. [Online]. Available: https://www.youtube.com/watch?v=3xb1dHLg-Lk. [Accessed: 15- May- 2016].