1. **ABSTRACT**

Implementation of Data Repository for Human Development Indicators is a web-based Data Analysis project. The purpose of this project is to analyse the Human Development Indicators by the organisations over the different analysis parameters like different areas or years with the data provided by the census data of countries.

The data is mainly collected by demo countries and analysis is done on that data by organisations. This is a three-entity system which communicates with each other over this system and share the data between them.

This project was implemented using Web Technologies.

1. **PROBLEM STATEMENT**

Information and Technology are the two faces of the same coin; ones which have become an indispensable part of the digital world. As the cyber world has grown so has the volume of Data that is being churned at a daily and monumental pace. Data itself is not sufficient to seek the information that organizations need to make informed decisions. This data needs to be closely managed, monitored, processed, summarized, collated, documented and secured so that it can be used effectively by the rightful owners. From time to time, we all face the same problem - data analysis. How to find useful information in a shapeless mass of data. This is quite a typical problem, when we face retrofit from **spreadsheet-like data** storage. Importing into relational database is quite easy, but then data quality and *queryability* is usually very low.

I have faced the exact same problem recently. In my case it was analyzing feedback responses. Essentially, there was substantial quantity of entries that was mainly plain text.

Managing this huge volume of data has made it mandatory to have Data Repository for easy discovery, Visualization and Analysis to have an Automatized flow of Data from the Data Provider to the Data Consumer. This Data Repository Systems amalgamate people and procedures to process Data and involves Discovery, Analysis, Visualization, Manipulation and Retrieval of Data.

**2.1 Providing, Maintaining and Acquiring Data for analysis:**

The Data System includes the entire lifecycle of the conversion of raw data into its meaningful counterpart, information. This is a Data Repository System which is basically a Data System in the form of a Dashboard for Easy Discovery, Visualization and Analysis of Human Development Indicators like Literacy Rate, Poverty Rate, and Mortality Rate etc. which facilitates the automatic flow of Data from the Provider to the Consumer with the Maintainer in between, maintaining the Data. This is a 3-way process in which there are 3-Entities: Data Provider, Data Repository Maintainer, and Data Consumer, with which the Data flows is the System.

1. Provider: This is the Data Provider which provides the Data to the Data Repository System according the measures set by the Data Maintainer. Data provider, to provide the data has to register with the System so to provide the Data to the system so that Consumer can consume the Data. Moreover, System will generate a notification on the Registered Provider’s System when the Data Consumer generates a request for the Data.
2. Maintainer: This is the Data Maintainer which maintains the flow of Data from the Provider to the Consumer. This Maintainer Entity sets the measures with which the Data provider provides the Data and the Data Consumer can consume the Data with the help of the Data Repository System. And also, this Maintainer takes the request from the Data Consumer to consume the Data and notify the Provider for the same. Similarly, when a Provider provides the Data to the System, this Maintainer will generate a notification on the Consumer’s System so that he can consume the provided Information.
3. Consumer: This is the Data Consumer consumes the Data, provided by the Data Provider to the System. This Entity constitutes various organizations which require Analyzed Data in the Visual form so as to work on it. For this, these various organizations have to register with the System so as to get the Analyzed, visualized Data from the Provider.
4. **INTRODUCTION**

Data permeates everything we do – that’s why we often hear data referred to as the lifeblood of businesses and organizations. However, it’s true that not all data is equal, but it’s also true that data is so critical we couldn’t live without it. That’s why numerous experts have put a dollar value on Data.

In this Era of Information Technology, when Data is in such an abundant form, when each and every day we are generating Millions of GBs of Data, this much data has to be managed very carefully and analyzed very critically so that this millions of GBs of data can be used crucially, Organizations are investing millions of dollars, if not billions of dollars to create, store, maintain and protect data. So, if we combine the value of data, the necessity of data, and the amount of money being spent on data, it’s easy to see how important data is, but if you can’t access that data within an appropriate timeframe based on the need, then organizations are spending a lot of money without gaining any real value. After all, what do we really gain when data is just sitting on storage? Nothing.

The real value of data is derived when it’s used. Furthermore, the speed and efficiency in accessing and obtaining that data is even more important as it significantly impacts the value of the data and one of the way to use it crucially is Data Repository System. There is no denying that data is invaluable. It underpins everything we do, especially being in the digital age. If anything happens to any of our servers we use backup copies to restore lost data.

We use data to make decisions; some simple and others life changing e.g. using data to develop cures for illnesses. We use data to bring to market new applications. We use data to enable us to move forward.  Some businesses are literally built on data – e.g. Facebook (worth $28 billion) and Twitter (worth $10 billion). Other businesses can easily attribute how data is adding to their bottom line.

Data Repository System is an Automised Dashboard System to analyze and visualize the Data and here specifically Human Development Indicators, which can be used by various organizations such as UNO (United Nations Organisation), WHO (World Health Organisation) and various local organisations to prepare their results out of these Human Development Indicators accordingly which is provided by the various Countries on this Dashboard System in an automised way. This system is used to create an automized flow of Data between the Data Provider and Data Consumer, and forming a Data Repository out of this Data so that Consumer can also analyze this Data via text or via a more systematic visual form which is also get notified and delivered on the System with the help of Web Services so that, that respective system can also manage that data accordingly.

1. **DATA LOADING AND PROCESSING**

**4.1 Loading Files From JSON Into D3.js:**

For nested data, or for passing around data where you don't want to mess with data typing, its hard to beat [JSON](http://json.org/).

JSON has become the language of the internet for good reason. Its easy to understand, write, and parse. And with [d3.json](https://github.com/d3/d3-request/blob/master/README.md#json) - you too can harness its power.

Here is an example JSON file called employees.json:

[ {"name":"Andy Hunt",

"title":"Big Boss",

"age": 68,

"bonus": true

},

{"name":"Charles Mack",

"title":"Jr Dev",

"age":24,

"bonus": false

}]

Loading employees.json with d3.json:

d3.json("/data/employees.json", **function**(data) {

console.log(data[0]);

});

=> {name: "Andy Hunt", title: "Big Boss", age: 68, bonus: true}

We can see that, unlike our flat file parsing, numeric types stay numeric. Indeed, a JSON value can be a string, a number, a boolean value, an array, or another object. This allows nested data to be dealt with easily.

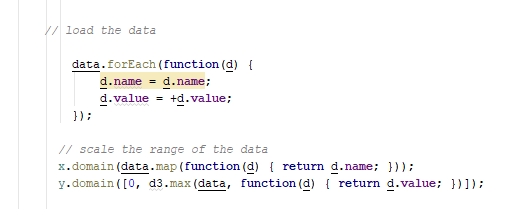


Fig.: Loading data from data object and mapping x & y domains

**4.2 Combining Files:**

**Using native Array functions**

We can implement a simple join (left outer join in database terms) using native, i.e., already existing Array functions as follows. The method presented here modifies the articles array in place by adding a new key-value-pair for brand.

articles.forEach(**function**(article) {

**var** result = brands.filter(**function**(brand) {

**return** brand.id === article.brand\_id;

});

**delete** article.brand\_id;

article.brand = (result[0] !== undefined) ? result[0].name : null;

});

console.log(articles);

If we want to join by more than one attribute, we can modify the filter function to achieve this. Hypothetically, this might look something like:

innerArray.filter(**function**(innerArrayItem) {

**return** innerArrayItem.idA === outerArrayItem.idA &&

innerArrayItem.idB === outerArrayItem.idB;

});

**4.3 Summarizing Data:**

With the data [loaded](http://learnjsdata.com/read_data.html), we want to take a quick look at what we have. D3 has a number of tools to use for quick data exploration.

To start, let's pretend we have loaded up a csv file - and have a dataset that looks something like:

**var** data = [

{"city":"seattle", "state":"WA", "population":652405, "land\_area":83.9},

{"city":"new york", "state":"NY", "population":8405837, "land\_area":302.6},

{"city":"boston", "state":"MA", "population":645966, "land\_area":48.3},

{"city":"kansas city", "state":"MO", "population":467007, "land\_area":315}

];

## Min & Max

As it turns out, D3 comes to the rescue again, with [d3.min](https://github.com/d3/d3-array#min) and [d3.max](https://github.com/d3/d3-array#max). Use the callback function to indicate which property (or computed value based on the properties) to access.

**var** minLand = d3.min(data, **function**(d) { **return** d.land\_area; });

console.log(minLand);

**=>** 48.3

*This code is using d3.js*

**var** maxLand = d3.max(data, **function**(d) { **return** d.land\_area; });

console.log(maxLand);

**=>** 315

*This code is using d3.js*

If you want both of them at the same time, you can use [d3.extent](https://github.com/d3/d3-array#extent)

**var** landExtent = d3.extent(data, **function**(d) { **return** d.land\_area; });

console.log(landExtent);

**=>** [48.3, 315]

*This code is using d3.js*

This returns an array with the first element the minimum value and the second element the maximum.

## Summary Statistics

D3 provides a few basic tools to analyze your data, all using the same format as the min and max functions. Simply provide the property you would like to analyze, and you are good to go.

[d3.mean](https://github.com/d3/d3-array#mean)

**var** landAvg = d3.mean(data, **function**(d) { **return** d.land\_area; });

console.log(landAvg);

**=>** 187.45

*This code is using d3.js*

[d3.median](https://github.com/d3/d3-array#median)

**var** landMed = d3.median(data, **function**(d) { **return** d.land\_area; });

console.log(landMed);

**=>** 193.25

*This code is using d3.js*

[d3.deviation](https://github.com/d3/d3-array#deviation) - for standard deviation

**var** landSD = d3.deviation(data, **function**(d) { **return** d.land\_area; });

console.log(landSD);

**=>** 140.96553952414519

**4.4 Iterating Over Data:**

## Iterating

First some basic iteration. We already saw this in the data loading task, but a common way to process each data object is by using [forEach](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach)

**var** count = 0;

data.forEach(**function**(d) {

count += 1;

});

console.log(count);

**=>** 4

## Mapping

JavaScript's [map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map) can be a very useful tool to implement this concept of a transformation on immutable data.

map takes an array and produces another array which is the result of the callback function being executed on each element in the array.

**var** smallData = data.map(**function**(d,i) {

**return** {

name: d.city.toUpperCase(),

index: i + 1,

rounded\_area: Math.round(d.land\_area)

};

});

console.log(data[0]);

console.log(smallData[0]);

=> {city: "seattle", state: "WA", population: 652405, land\_area: 83.9}

{name: "SEATTLE", index: 1, rounded\_area: 84}

## Filtering

Select a subset of the data using the built in [filter](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter) method. This creates a new array of data (again see transformation talk above) with only the values that the callback function returns true for.

**var** large\_land = data.filter(**function**(d) { **return** d.land\_area > 200; });

console.log(JSON.stringify(large\_land));

**=>** [{"city":"newyork","state":"NY","population":8405837,"land\_area":302.6},

{"city":"kansas city","state":"MO","population":467007,"land\_area":315}]

**4.5 Grouping Data:**

## Nesting

**var** expenses = [{"name":"jim","amount":34,"date":"11/12/2015"},

{"name":"carl","amount":120.11,"date":"11/12/2015"},

{"name":"jim","amount":45,"date":"12/01/2015"},

{"name":"stacy","amount":12.00,"date":"01/04/2016"},

{"name":"stacy","amount":34.10,"date":"01/04/2016"},

{"name":"stacy","amount":44.80,"date":"01/05/2016"}

];

And now we want to slice up this data in different ways.

First, let's use nest to group by name:

**var** expensesByName = d3.nest()

.key(**function**(d) { **return** d.name; })

.entries(expenses);

*This code is using d3.js*

Which results in a nested data structure:

expensesByName = [

{"key":"jim","values":[

{"name":"jim","amount":34,"date":"11/12/2015"},

{"name":"jim","amount":45,"date":"12/01/2015"}

]},

{"key":"carl","values":[

{"name":"carl","amount":120.11,"date":"11/12/2015"}

]},

{"key":"stacy","values":[

{"name":"stacy","amount":12.00,"date":"01/04/2016"},

{"name":"stacy","amount":34.10,"date":"01/04/2016"},

{"name":"stacy","amount":44.80,"date":"01/05/2016"}

]}

];

## Summarizing Groups

The nested structure can be great for visualizing your data, but might be a little underwhelming for analytical applications. Never fear! [d3.rollup](https://github.com/d3/d3-collection#nest_rollup) is here!

With rollup, you provide a function that takes the array of values for each group and it produces a value based on that array. This provides for some very flexible group by functionality.

Here is a simple one to get back the counts for each name:

**var** expensesCount = d3.nest()

.key(**function**(d) { **return** d.name; })

.rollup(**function**(v) { **return** v.length; })

.entries(expenses);

console.log(JSON.stringify(expensesCount));

**=>** [{"key":"jim","values":2},{"key":"carl","values":1},{"key":"stacy","values":3}]

1. **DATA ENTITIES**
   1. **Data Provider**

Data Provider, the first entity in the three-entity communication system. From this data provider, the system communication and sharing of data starts taking place. This provides the data onto the system. Data provider, as an entity itself, firstly gets the input data from the providers (i.e. countries in this case) on the following parameters:

1. Indicators

2. Areas

3. Units

4. Subgroups

For inputting the data each country has a specific login with which it has to login onto the system and maintain the session, this has been done with node and express middleware, where after login a session is created for a specific country for a specific time, where no activity results in session termination. Then after login into the country’s portal one creates the data against the parameters, which will be available into the Datasets tab, where the country entity combines the data according to the data stored with respective parameters. With data input into the respective parameters, data provider also provides the option to edit and delete the input data in those respective parameters. When all the data is uploaded according to the parameters then dataset is created. Then that dataset in the datasets table gets shared across the Repository to the subscribed Consumers which consumes and analyze that dataset.

* 1. **Data Registry**

Data Registry, it is the second entity in the three-entity communication system. This entity is responsible for managing the system and the entities. It controls the communication between these entities. It uses a **Pub-Sub** design pattern, where the providers and consumers both registers themselves onto this entity, and with-it they also state their specifications. Consumers, after stating their specifications subscribes to the registered providers from where they need the data according to the specifications they stated.

After both the providers and subscribers are registered then, this middle entity send a request to both the entities i.e. provider and consumer so as to establish the link between the two so that communication and data can be shared between them. With this communication link, data is shared with the consumers (i.e. organizations) which has been put onto this system by the data providers. When the data has been put in by the providers on to the respective systems, data Registry fetches that data from the provider (or data is provided by the provider) and then that data is shared with the data consumer with a notification, that data has been put into the system by its subscribed provider. In this way consumers gets their subscribed providers data. This sharing via pub-sub, linking and the maintenance of the entities is the responsibility of the Data Registry.

* 1. **Data Consumer:**

This is the third entity in the three-entity Data communication system. This entity is responsible for the consuming of the data supplied onto the system by the subscribed registered data providers and then finally to analyze the system via the bar graphs. This entity is the last step of this whole process. In this it collects the data in the form of a Datasheet, where all the data is collected in a tabular form. And at the same there is this another option(charting) to visualize the data graphically by the means of bar graphs. Charting by the Bar graphs has been done by the D3.js which stands for basically Data Driven Documents which is basically used to represent the textual data into the visual and more of a graphical form as data in the graphical form is more easily readable and understandable.

For consumption of the data and finally analyzing of it, starts from where the consumer has to go onto the Data Registry entity and register itself onto the system and state it data specifications for which it needs data and also its URL to where it needs that data. And at the same time with specification and URL, consumer also has to subscribe to list of providers which have registered onto the Data Registry system. This completes the Subscription part of the Pub-Sub pattern. After this, whenever data has been put by the specific provider, the subscribed consumers will get the data which has been put into the system by that respective provider.

1. **APPLICATIONS**

Data Repository technology can be used where there is a communication need between these or like these entities, to share data or to communicate with each other. This project links these three entities in the simplest way for the purpose of their communication and sharing of the data and to analyze the data further by the consuming entity. Thus, this can be applied to various applications from the simple to the complex ones.

Data Repository for Human Development Indicators provides easy and convenient access to real-time data generated by censuses conducted by governments in various nations, this data ranges from a country’s population to its literacy rate. These indicators are then used by organizations like United Nations(UN) or World Health Organization(WHO) to rank nations for lifestyle quality.

One such example is Human Development Index which is calculated on the basis of a country’s life expectancy for health, expected years of schooling, mean of years of schooling for education and Gross National Income per capita for standard of living by United Nations Development Program. Every year UNDP ranks countries based on the HDI report released in their annual report. HDI is one of the best tools to keep track of the level of development of a country, as it combines all major social and economic indicators that are responsible for economic development.

Access to this data is paramount and is made easily possible through Data Repository for Human Development Indicators.

Other than for Human Development Indices, it (Data Repository) can also be used to implement simpler system like to maintain the stock of goods and services at the consumer (i.e. shop owner) by the provider (i.e. supplier) with which both the entities are updated simultaneously, when something is consumed or in the same way when something is produced. With the help of the Data Repository, both the entities get to be updated all the time of their goods and services in the simplest manner with which they can fill their inventories timely.

**Uses in the Public Sector**

Groups in the public sector include all levels of government (those in it and those trying to get in it), as well as police, military, transportation agencies, and educational and healthcare facilities. Just a few folks, right? Oh, and let’s add philanthropy and philanthropic projects, a.k.a. charities, into the mix, just for fun.

All these organizations have a key interest in discovering what is happening (the data) and then conveying that information internally to others in their own group and/or externally to the broader public (the visual). Many such efforts are mandated and essential to the organization’s existence. Take, for example, the U.S. census. The data is collected on a massive scale every 10 year—by law—and then impacts multiple facets of American life such as state and regional funding and, of course, congressional representation. The U.S. Census Bureau maintains a treasure trove of the aggregate data, now visually accessible to everyone through its online presence at www.census.gov. Not only are there government-sanctioned representations of the collected census information but the site also makes APIs available (api.census.gov) for public web developer access.

**Business-to-Business and Intrabusiness Uses**

If the business of business is business, how do you do business? Mostly through marketing, whether you’re a vendor targeting another company or one department lobbying internally for increased resources. And the heart of marketing is persuasion—which is often bolstered, if not solely accomplished, by making your case through the compelling presentation of data.

As with the public sector, many such presentations are required. Look through any annual report to see the latest encapsulation of the company’s standing, graphically depicted in quickly graspable charts. Today, creating an online report is standard practice. Similar data visualizations are undertaken daily in department and division meetings to plot sales progress, reveal public reaction to products, and adjust business direction.

1. **USE CASE DIAGRAM:**

A use case diagram is used to model functionalities and requirements of a system. A use case diagram shows the interaction of various users with the system, it has actors and the actions they can perform with the system. An actor is any person or external system that interacts with the system to achieve a user goal. The Use Case Diagram is a visualization of a use case. The Use Case Diagram for the actions that the Users (provider, maintainer and consumer) can perform in the system. Usually referred to as behavior diagrams they are used to describe a set of actions that some system or systems should or can perform in collaboration with one or more external users of the system. Each use case should provide some observable and valuable result to the actors or other stakeholders of the.

The use case diagrams shown here are for the two most common users of the system namely provider and data consumers.

**Figure1** on the following page shows the use case diagram for a data provider with provider as an actor and its various functionalities listed in the diagram.

**Figure2** on the page after that shows the use case diagram for a data consumer and lists the functionalities provided by the system to a data consumer. The whole interface is designed simplistic in nature and self-explanatory for a better user experience.

**Figure 3** shows the use case diagram for the Data Maintainer or Registry Entity which describes about the Maintainer and its responsibilities to add Provider and Consumer to the system and manage both the entities. It also acts as a middle entity between the two, which notify the consumer when data is uploaded onto the system so that consumer can fetch the data uploaded by the provider.

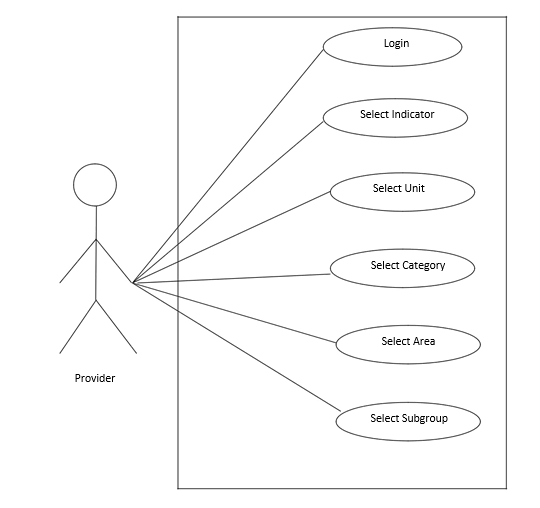
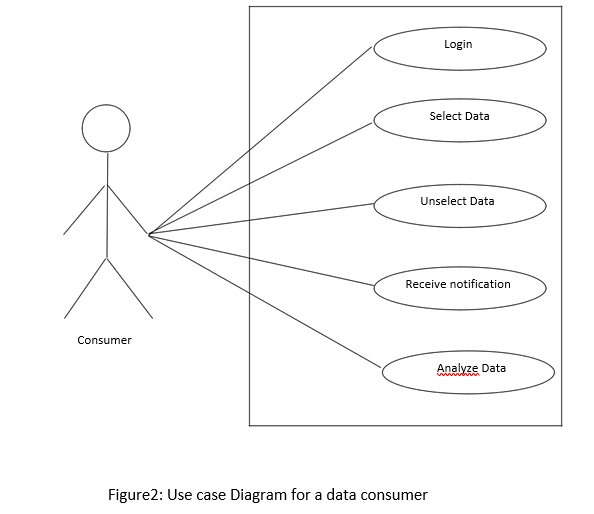
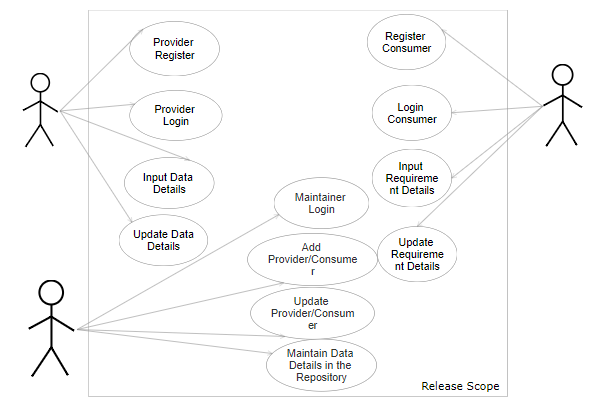


Figure 1: Data Provider





Data Maintainer

Data Consumer

Data Provider

Figure 3: Use case diagram of Registry

**8. REQUIREMENTS**

* 1. **TECHNICAL REQUIREMENTS:**
* **D3.js:** D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3’s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

D3.js helps you attach your data to DOM (Document Object Model) elements. Then you can use CSS3, HTML, and/or SVG showcase this data. Finally, you can make the data interactive through the use of D3.js data-driven transformations and transitions. This D3.js Framework will be used to generate visual representation to the Analysed Data on the Data Repository Dashboard which is provided by the Provider so that the consumer can analyse and consume the Data accordingly.

* **Node.js:** **Node.js** is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [run-time environment](https://en.wikipedia.org/wiki/Runtime_system) that executes JavaScript code [server-side](https://en.wikipedia.org/wiki/Server-side). Historically, JavaScript was used primarily for [client-side scripting](https://en.wikipedia.org/wiki/Client-side_scripting), in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting)—running scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content *before* the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm

Node.js is a platform built on Chrome’s JavaScript Runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications.

* **Express.js:** Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework

1. Allows to set up middleware to respond to HTTP Requests.
2. Defines a routing table which is used to perform different actions based on HTTP Method and URL.
3. Allows to dynamically rendering HTML Pages based on passing arguments to templates.

This Node framework with the help of Node is used to make the server side of the Dashboard, which is used to generate the notifications to the provider and the consumer’s system.

* **MySQL:** MySQL is a fast, easy-to-use RDBMS being used for many small and

big businesses.

1. MySQL is released under an open-source license.
2. MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
3. MySQL uses a standard form of the well-known SQL data language.
4. MySQL is very friendly to PHP, the most appreciated language for web development.
5. MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
6. MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

**Storage Engine Used:**

**InnoDB**is a general-purpose storage engine that balances high reliability and high performance. In MySQL 8.0, InnoDB is the default MySQL storage engine.

1. Its [DML](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_dml) operations follow the [ACID](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_acid) model, with [transactions](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_transaction) featuring [commit](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_commit), [rollback](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_rollback), and [crash-recovery](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_crash_recovery) capabilities to protect user data.
2. Row-level [locking](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_locking) and Oracle-style [consistent reads](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_consistent_read) increase multi-user concurrency and performance
3. InnoDB tables arrange your data on disk to optimize queries based on [primary keys](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_primary_key). Each InnoDB table has a primary key index called the [clustered index](https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_clustered_index) that organizes the data to minimize I/O for primary key lookups

* **HTML:** HTML is the standard mark-up language for creating Web pages.

1. HTML stands for Hyper Text Mark-up Language
2. HTML describes the structure of Web pages using mark-up
3. HTML elements are the building blocks of HTML pages
4. HTML elements are represented by tags
5. HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
6. Browsers do not display the HTML tags, but use them to render the content of the page

**CSS:** CSS stands for Cascading Style Sheets:

1. CSS describes how HTML elements are to be displayed on screen, paper, or in other media

1. CSS saves a lot of work. It can control the layout of multiple web pages all at once
2. External stylesheets are stored in CSS files
   1. **SYSTEM REQUIREMENTS:**

* Operating System:

1. Windows: Windows 7 or later
2. Mac: Mac OS X 10.9.x or later
3. Linux: 64-bit Ubuntu 12.04+, Debian 8+, OpenSuSE 12.2++, or  
   Fedora Linux 17

* Processor:

1. Windows: Intel Pentium 4 or later
2. Mac: Intel
3. Linux: Intel Pentium 3 / Athlon 64 or later

* Browser:

1. Chrome
2. Internet Explorer: 9.0 and above
3. Mozilla Firefox: 1.5 and above
4. Safari

* Memory: 2 GB minimum, 4 GB recommended
* Screen Resolution: 1280x1024 or larger
* Application Window size:1024x680 or larger
* Internet Connectivity: Required

1. **METHODOLOGY**

The project was made in following four phases:

**Phase 1** - Developed Data Provider Interface to input the data in the system.

**Phase 2** - Login of Data provider entity to login into the Data Provider

system.

**Phase 3** - Development of data maintainer’s side.

**Phase 4** - Developed Data Consumer Interface to register the request and to

analyze the data.

**Phase 5**- Testing

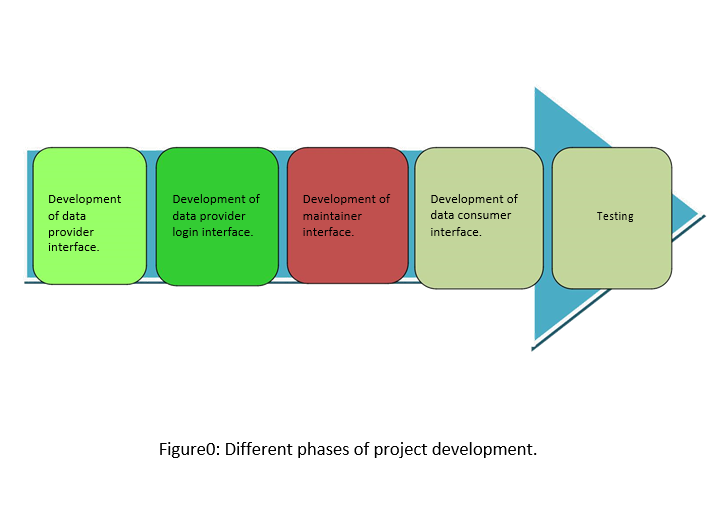
In phase1 data provider interface was developed, functionalities like selection of category, region, unit, subgroup and indicators were implemented for data input by the provider.

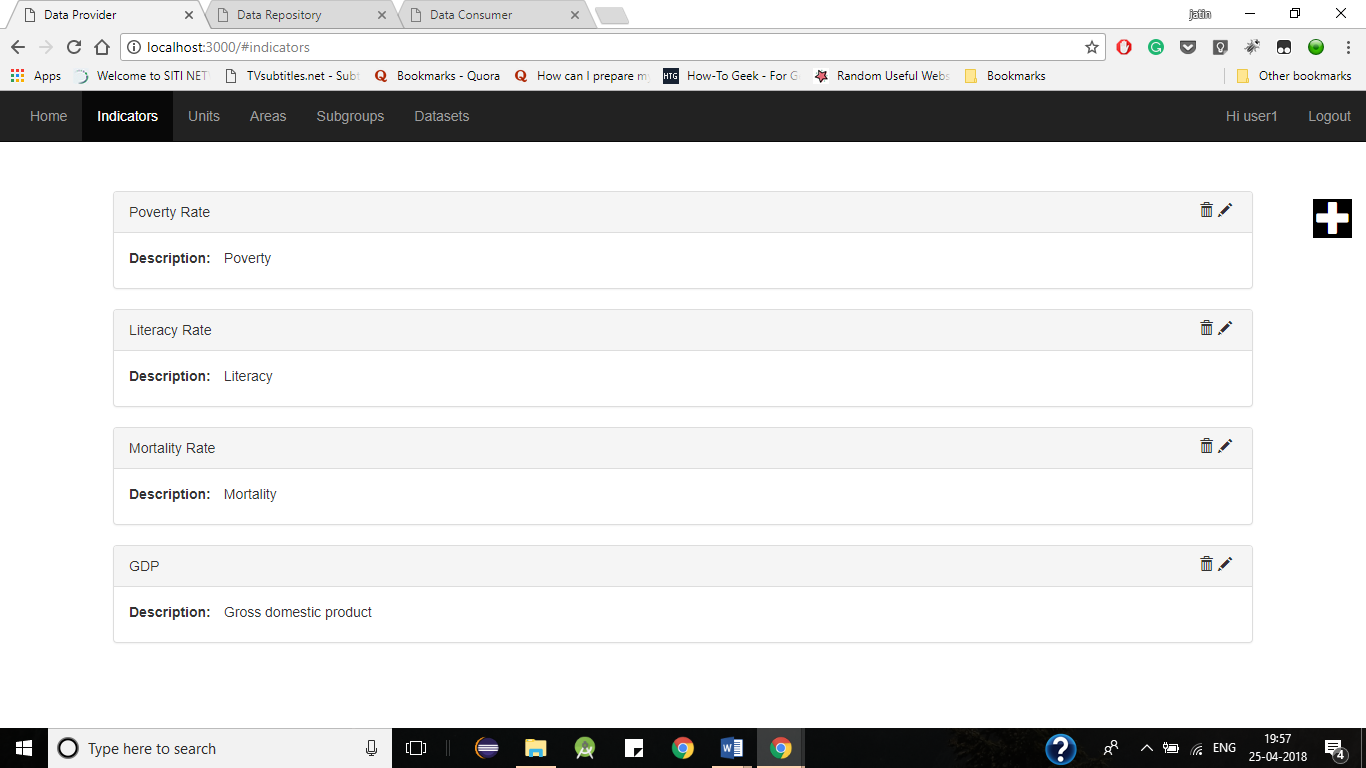
In phase2 login facility for the data provider was developed.

In phase3 data maintainer’s side was developed, now a data maintainer can keep the system sorted, whether or not the posted data matches it’s category can be checked by the maintainer.

In phase4 data consumers interface was developed, functionalities like subscribing to different indicators, regions, subgroups, units etc were implemented.

Finally, in phase4 the system was locally hosted and tested whether or not working properly.



1. **SCREENSHOTS**

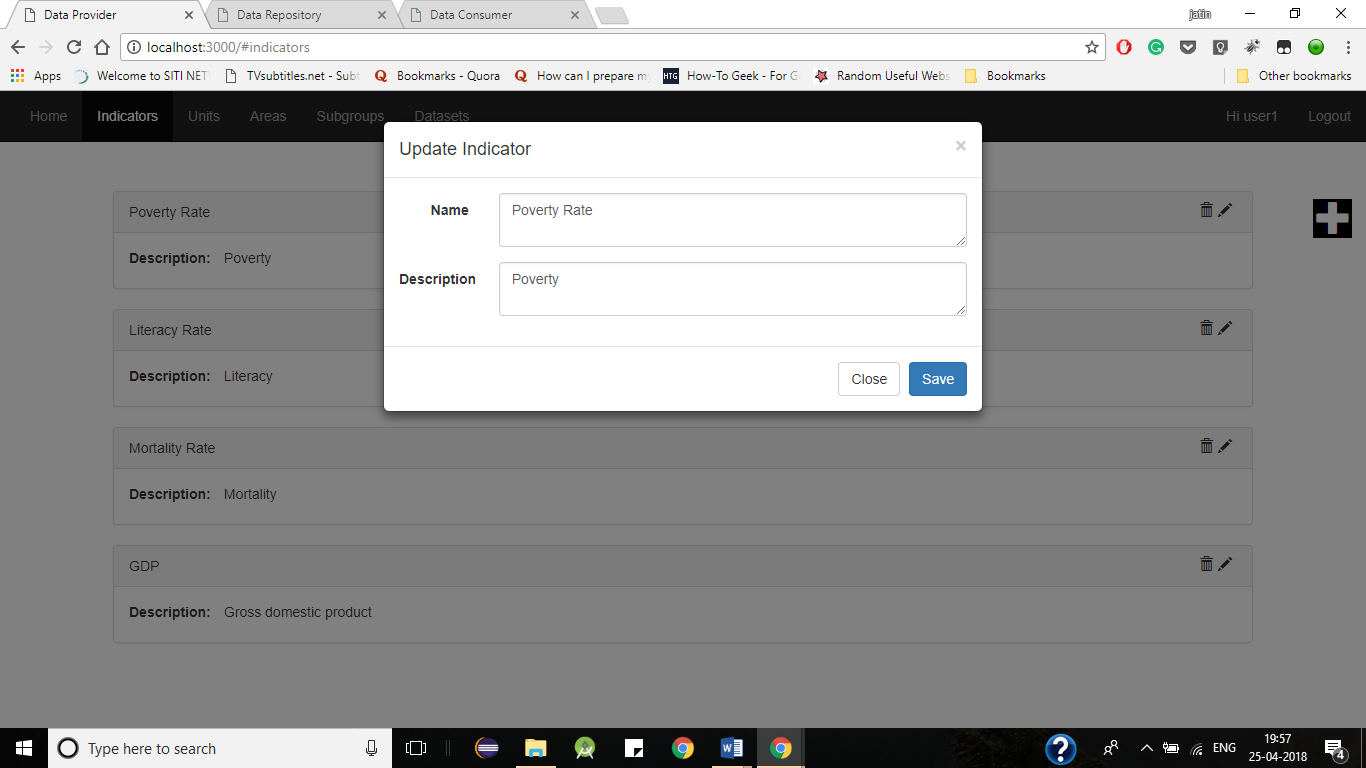
Fig. Values in Parameters (At Provider)

Fig. Editing the Values in Parameters

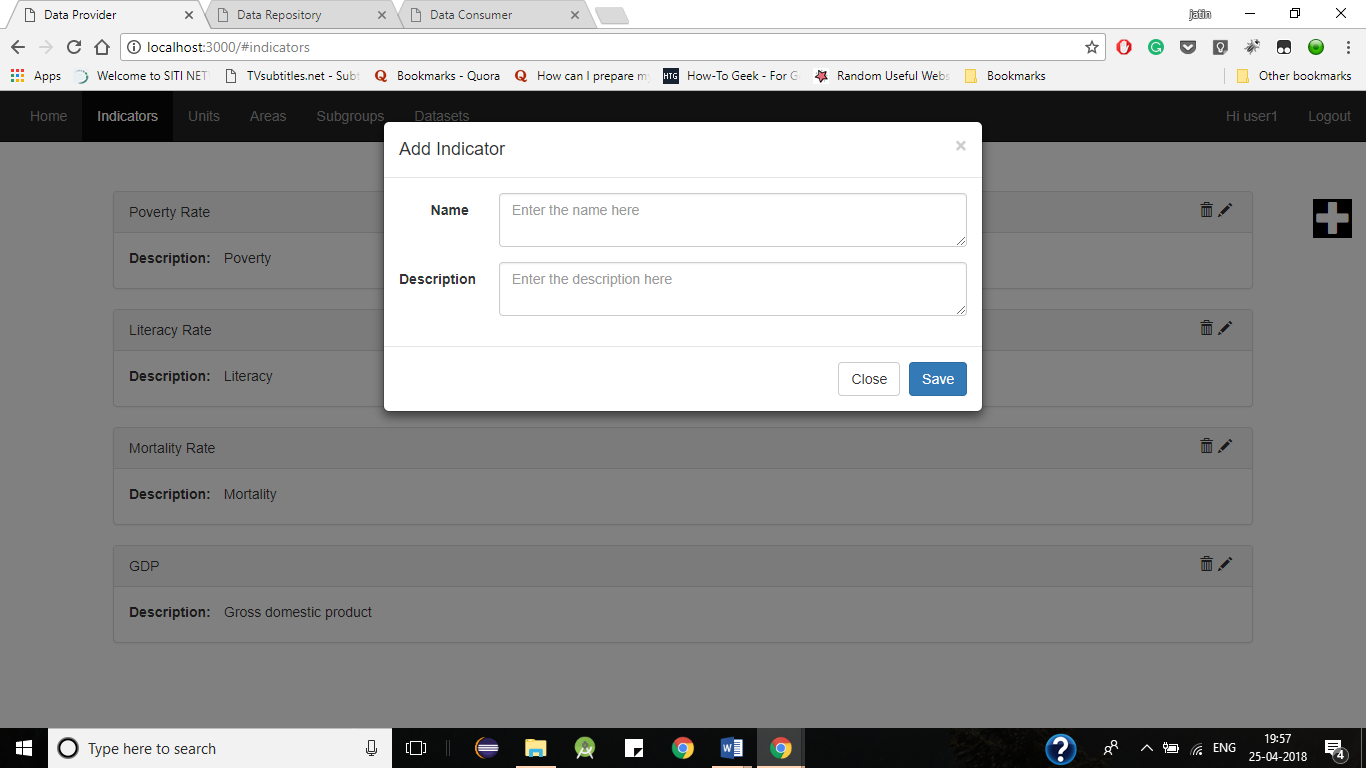
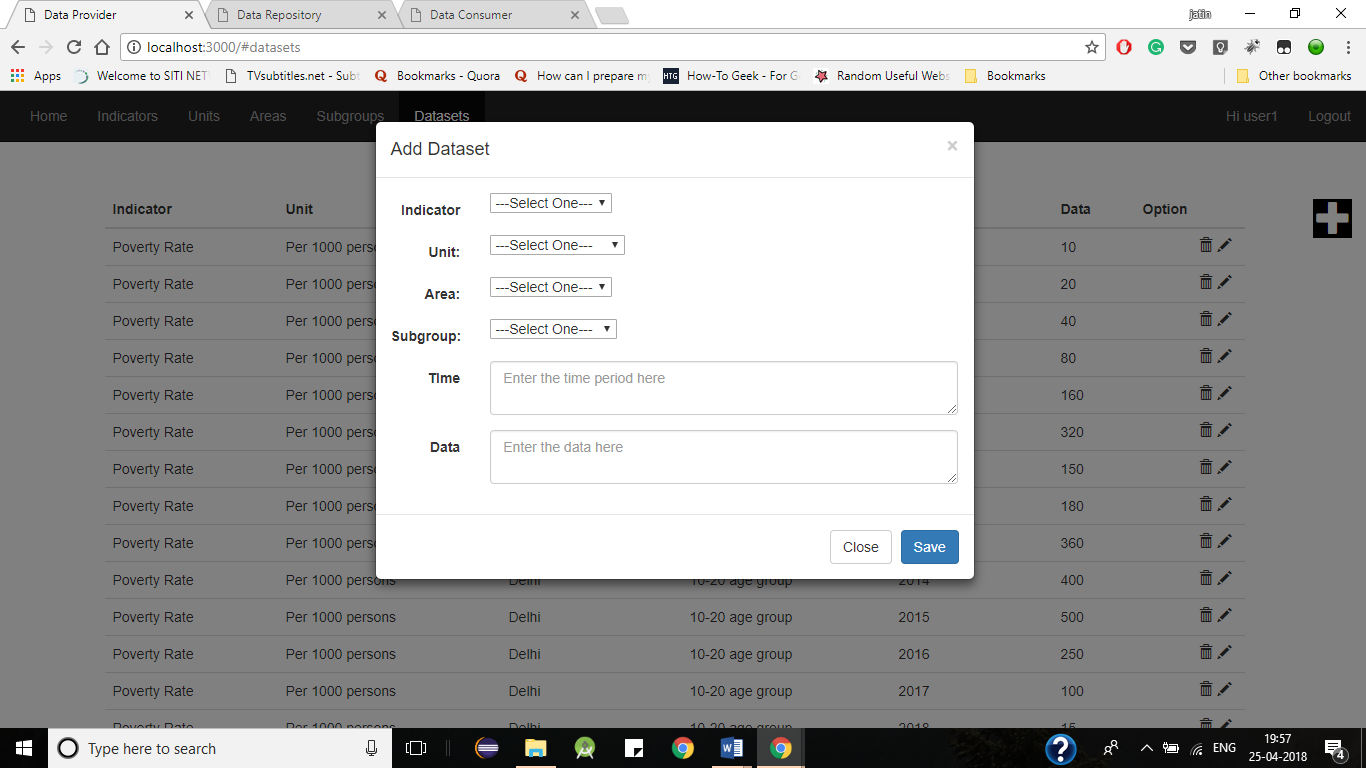
Fig. Adding the Value in the parameters

Fig. Creating Dataset selecting values from different Parameters

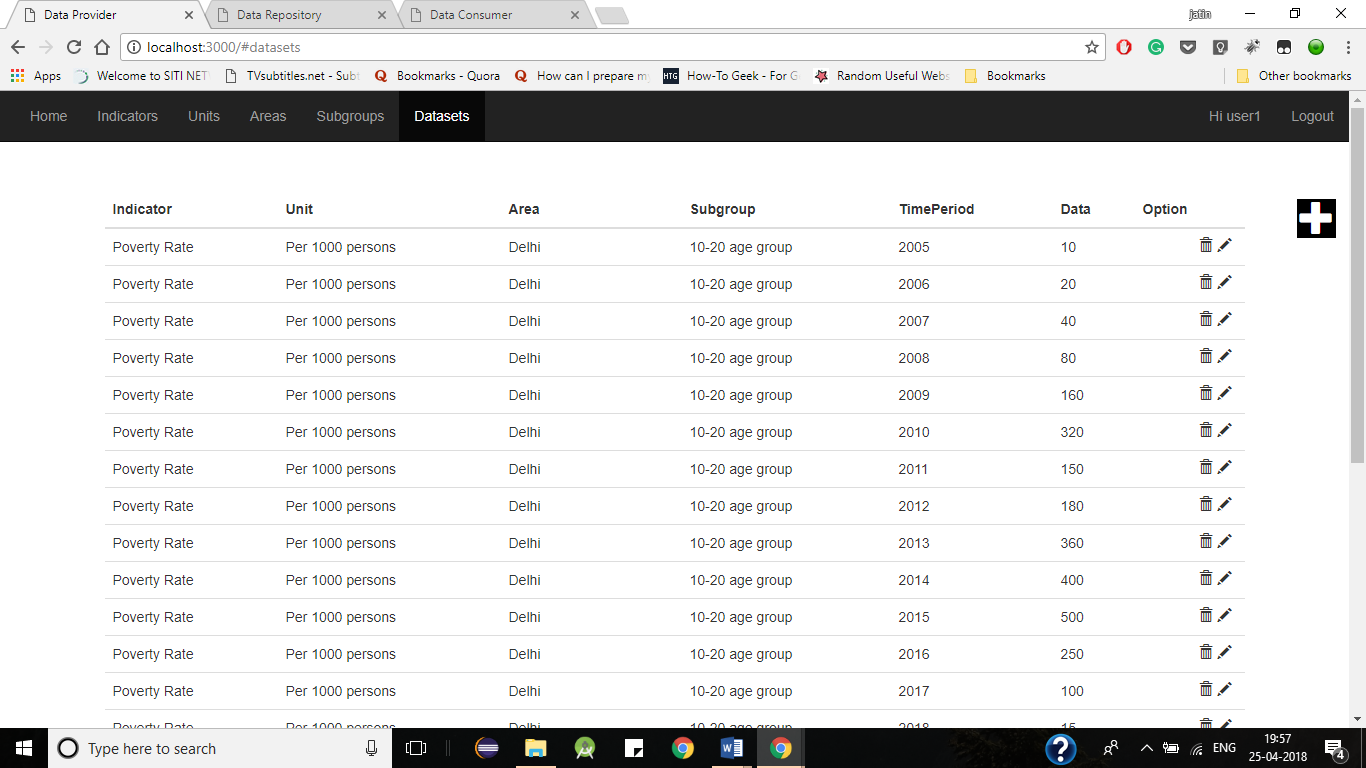
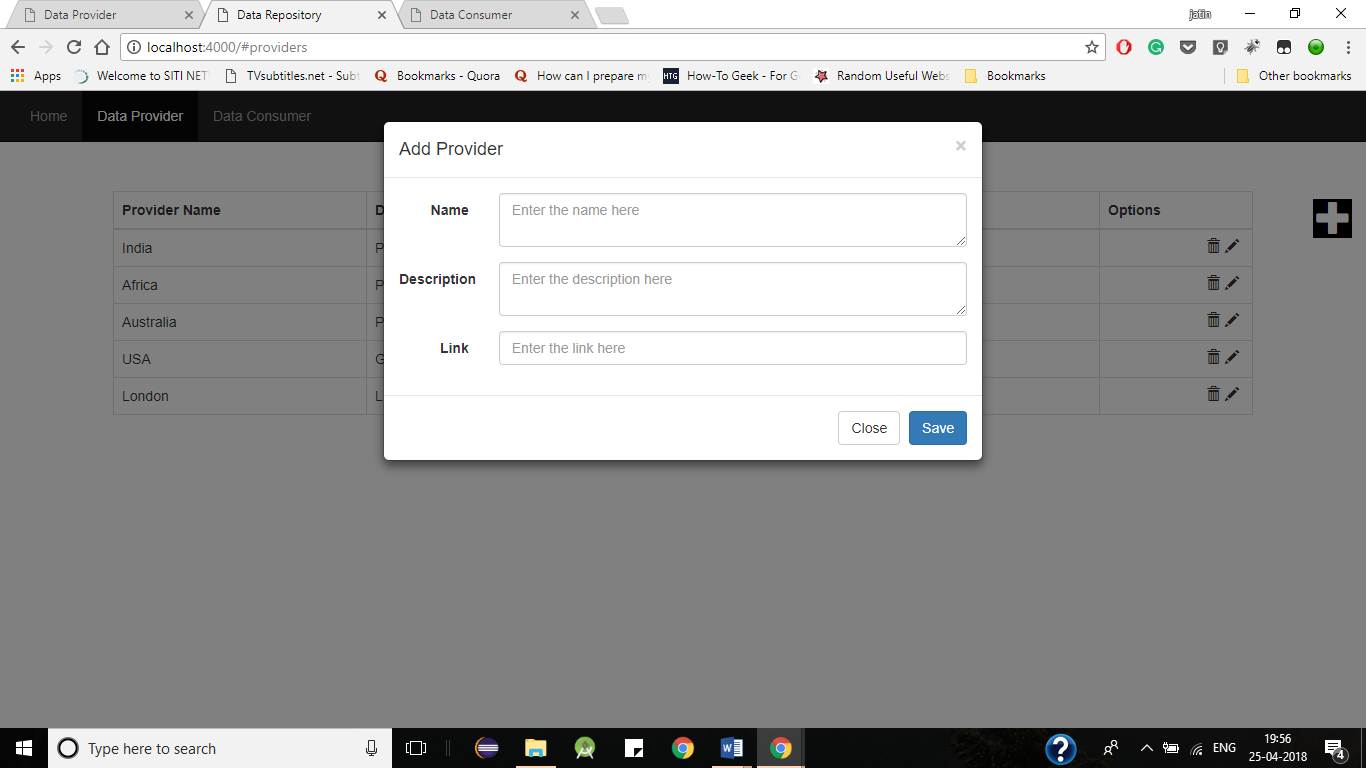
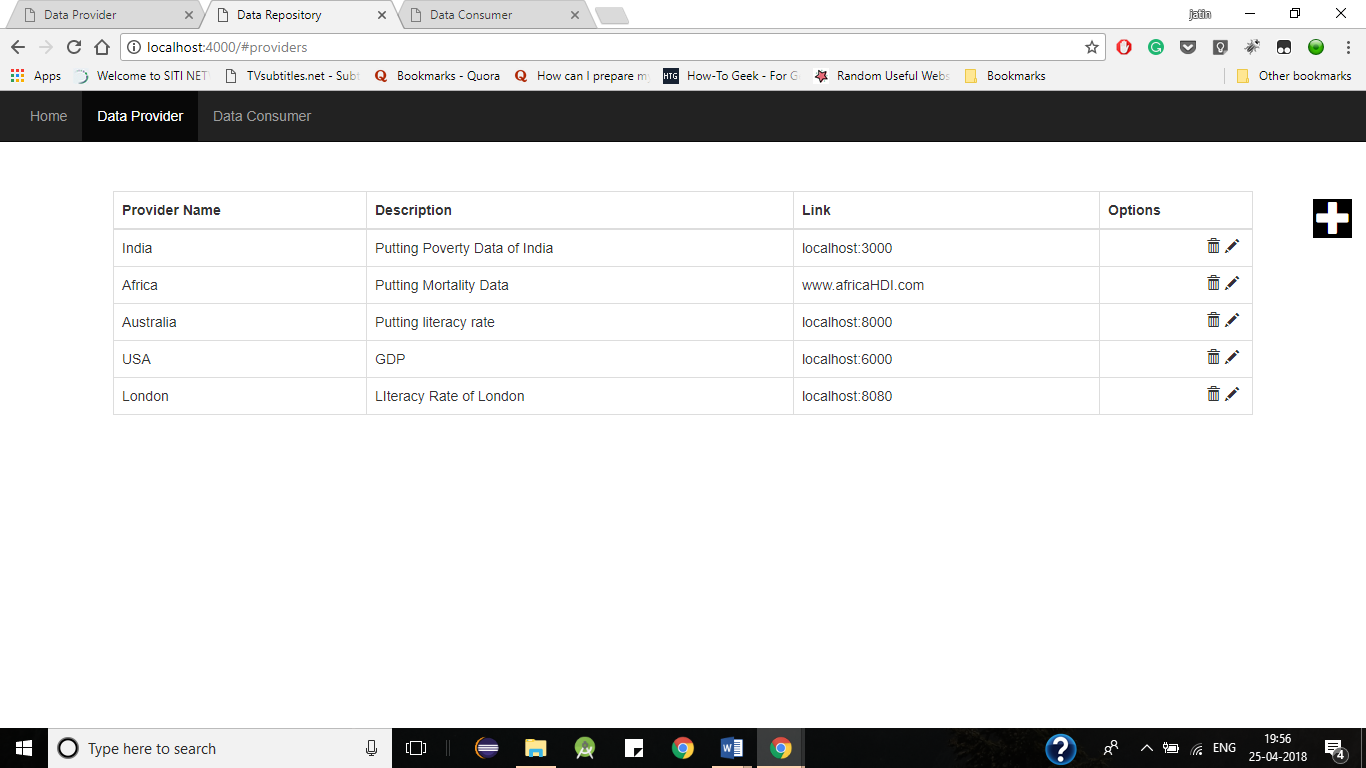
 Fig. All Dataset Values

Fig. Registering Data Provider at Data Registry (At Data Registry)

 Fig. Data Providers at Data Resgistry

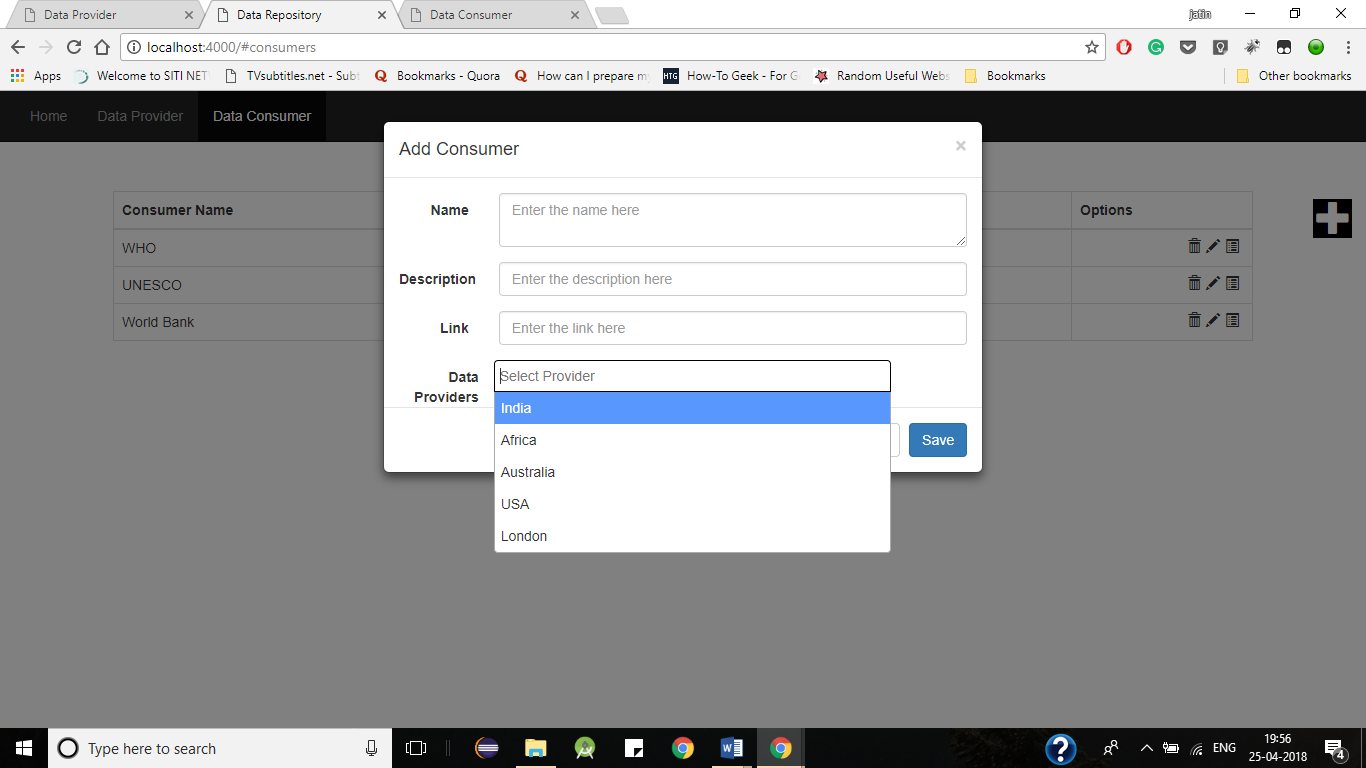
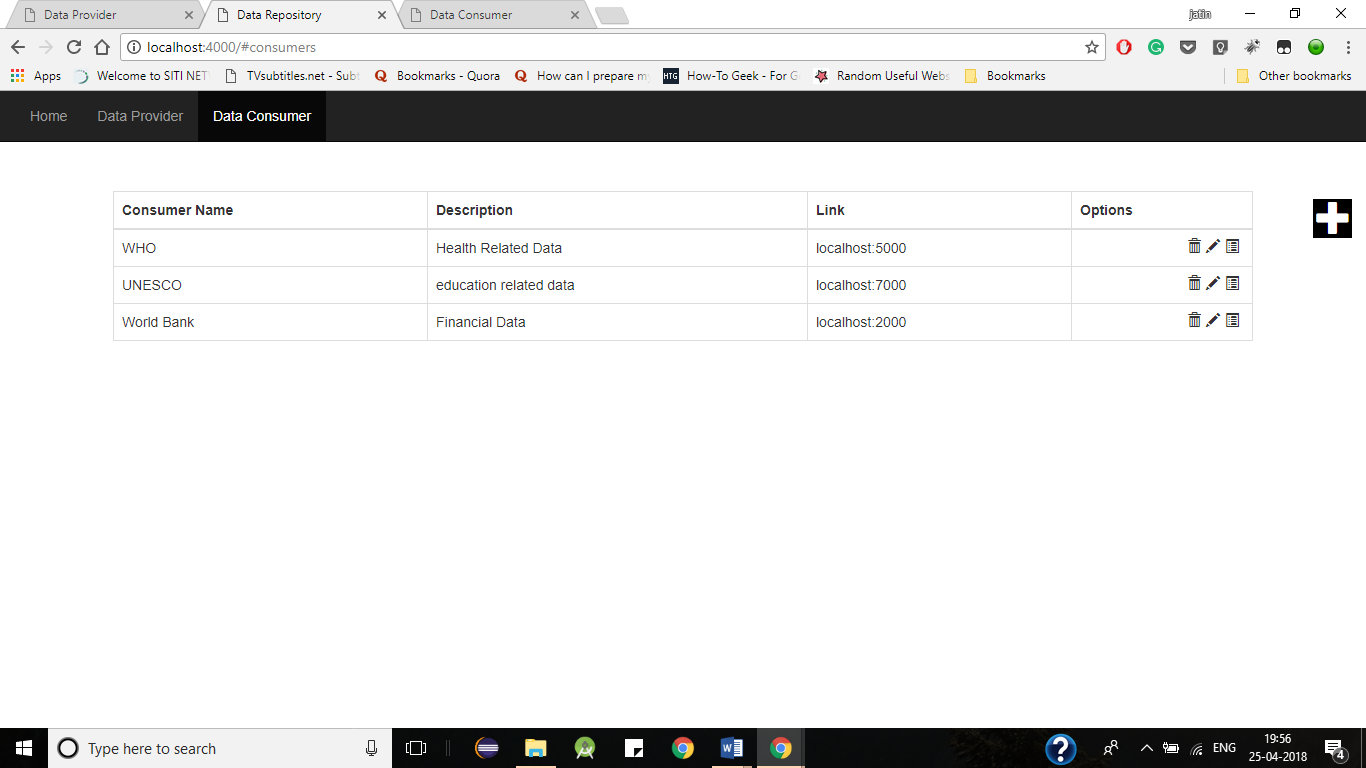


Fig. Registering Consumers while subscribing Providers

Fig. List of Data Consumers

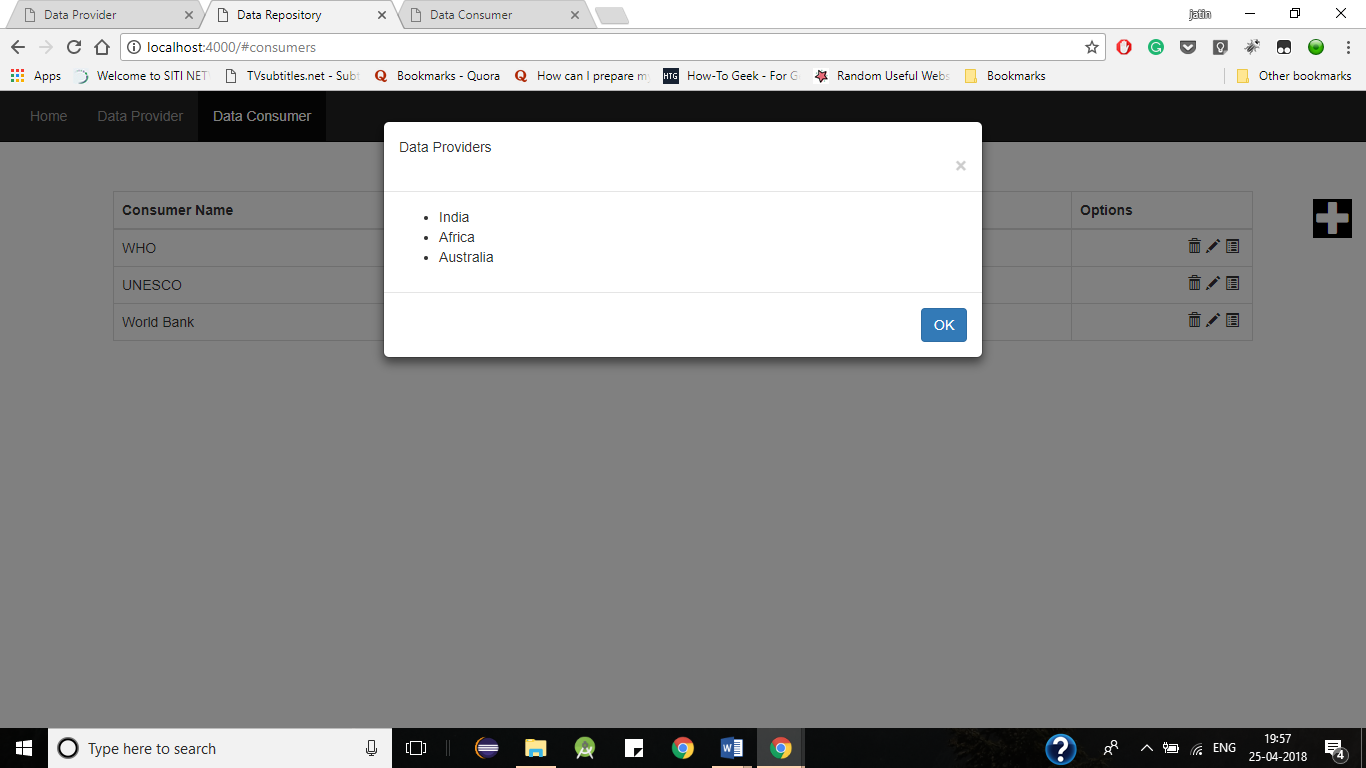
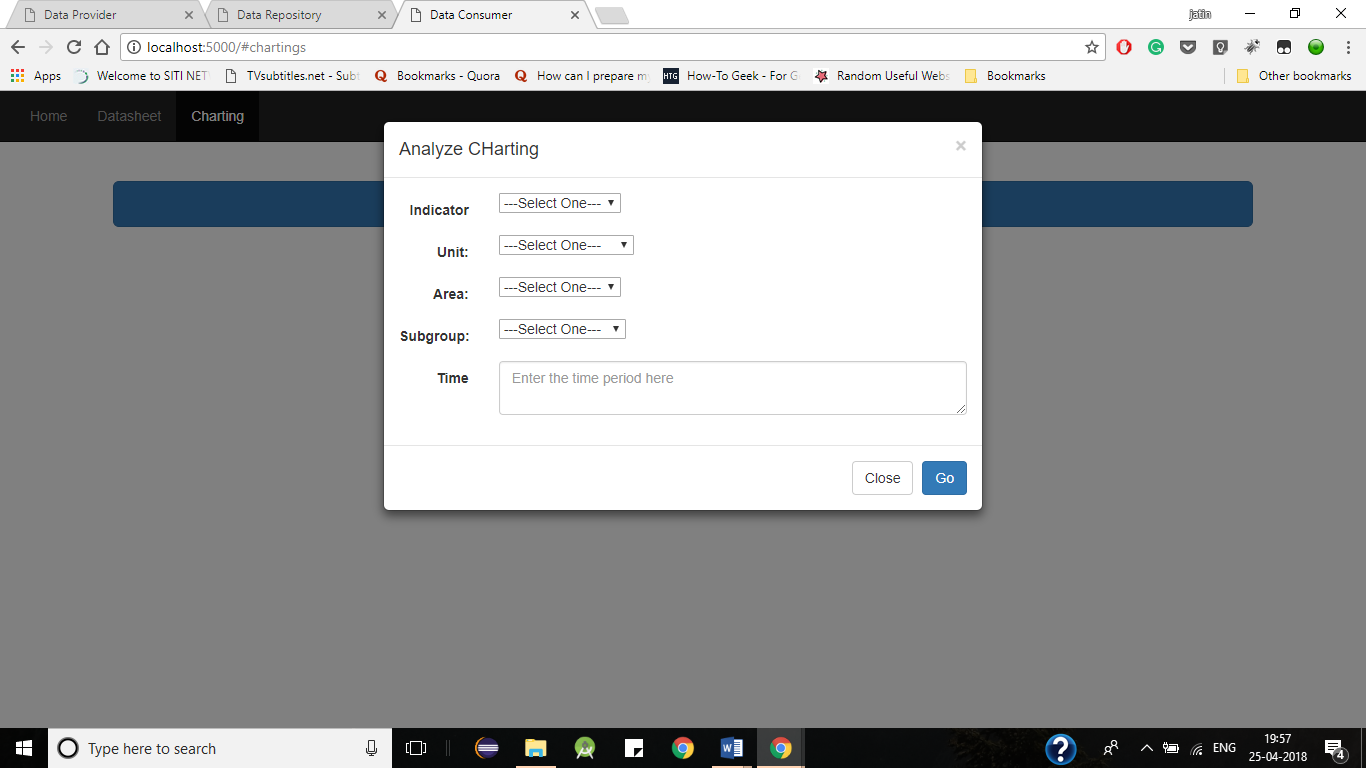


Fig. Subscribed list of Providers

 Fig. Selecting Parameters to Analyze Data (At Consumer)

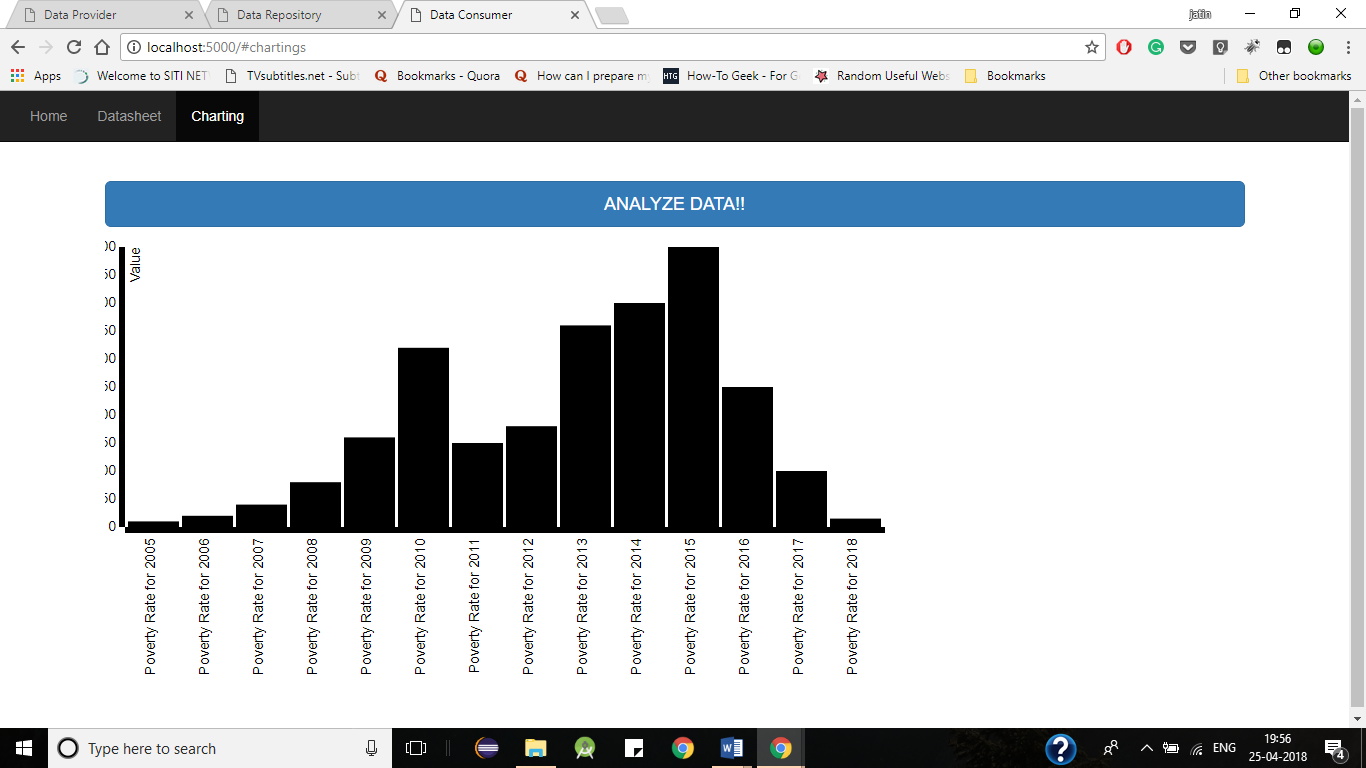


Fig. Analysis of Data of Poverty Rate across diff years (in bar graphs)

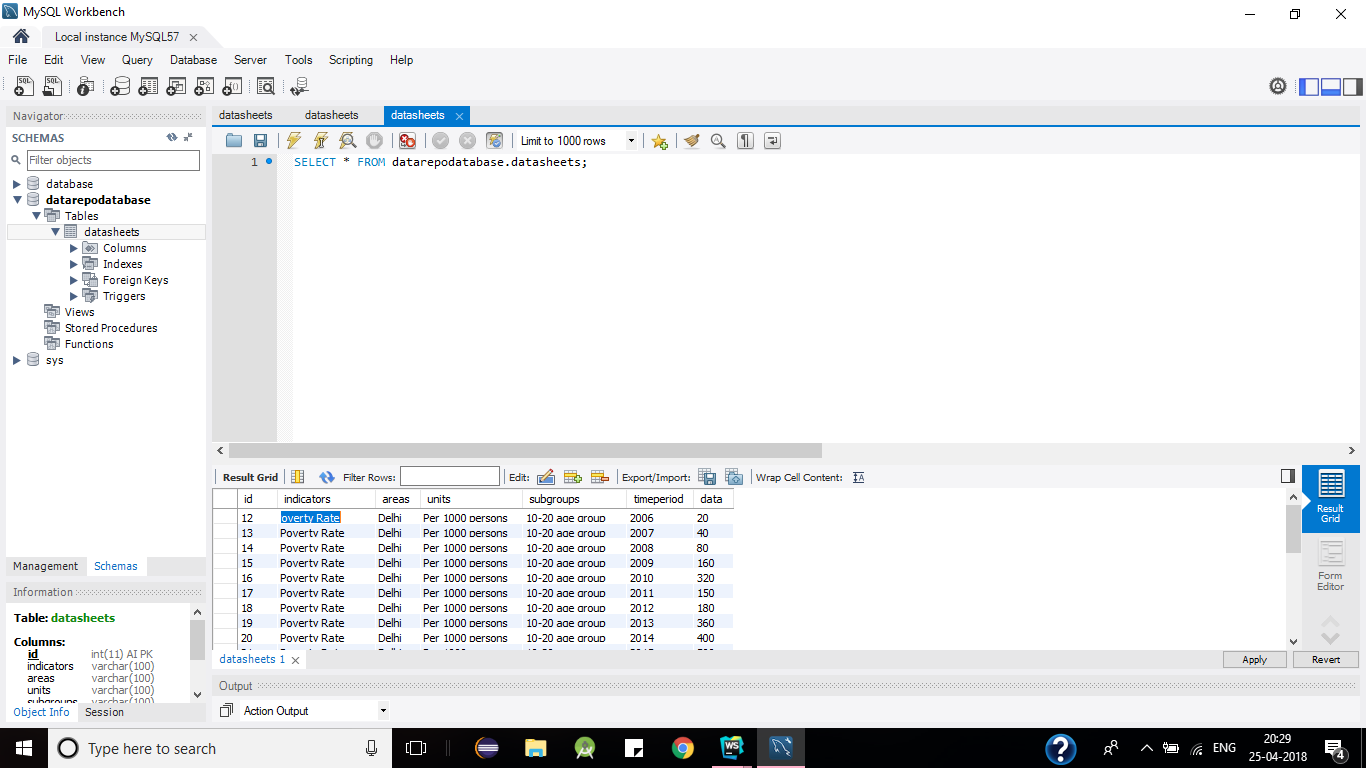
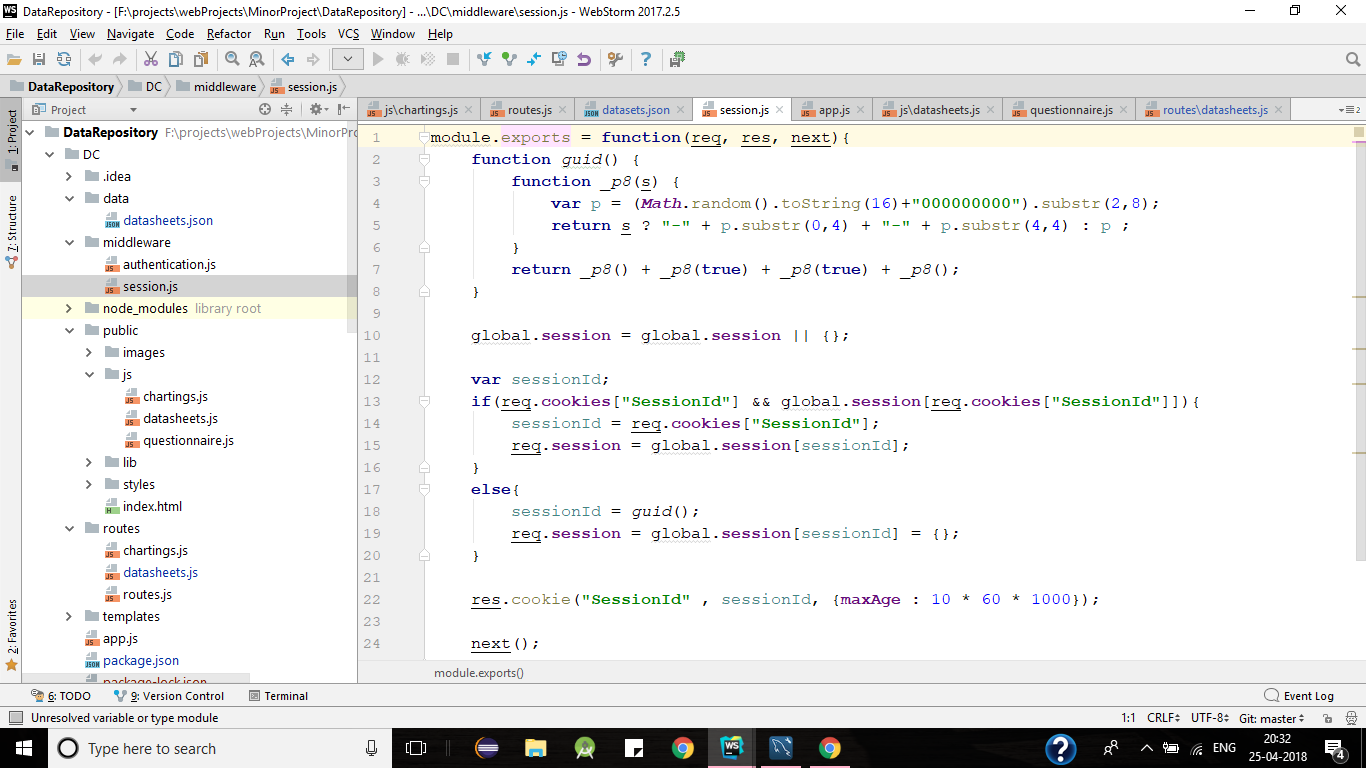
Fig. Data in the Database (MySQL)

Fig. Session Code to Login Data Provider

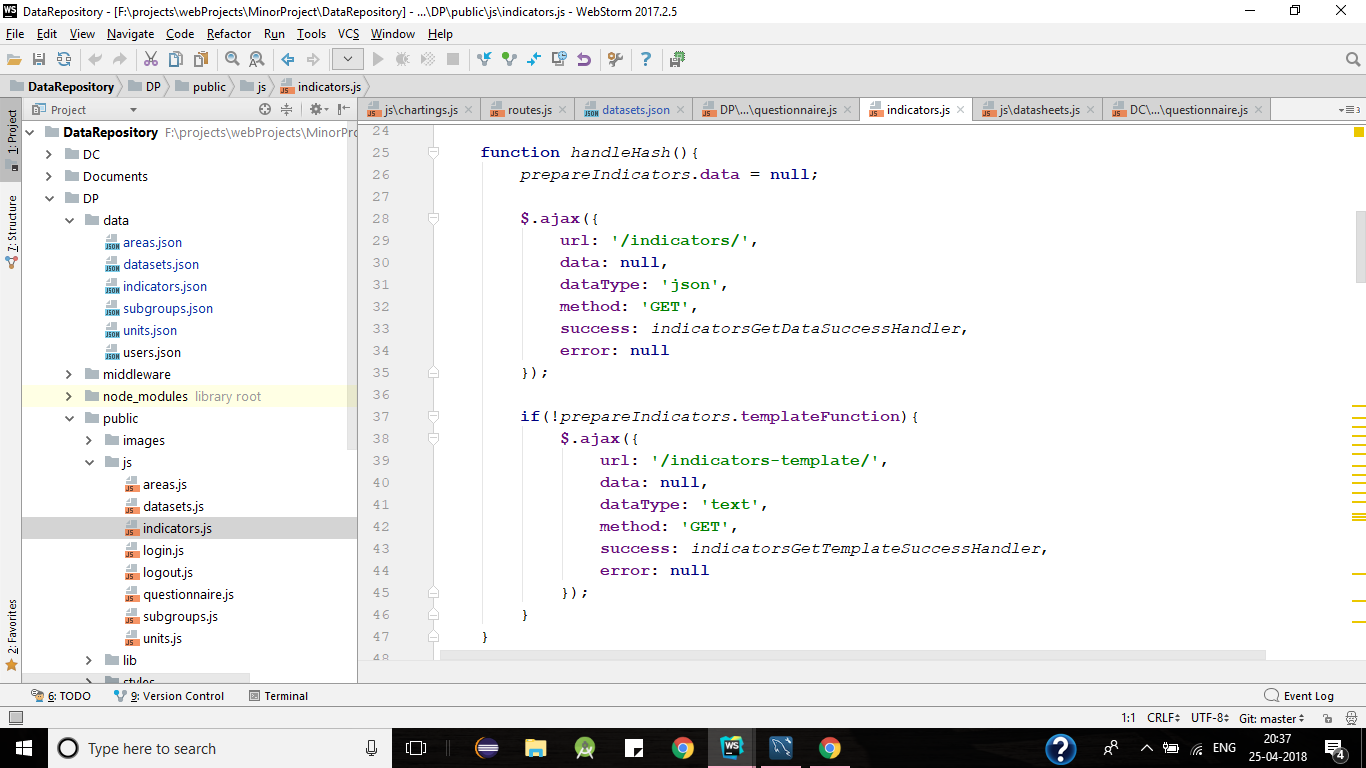
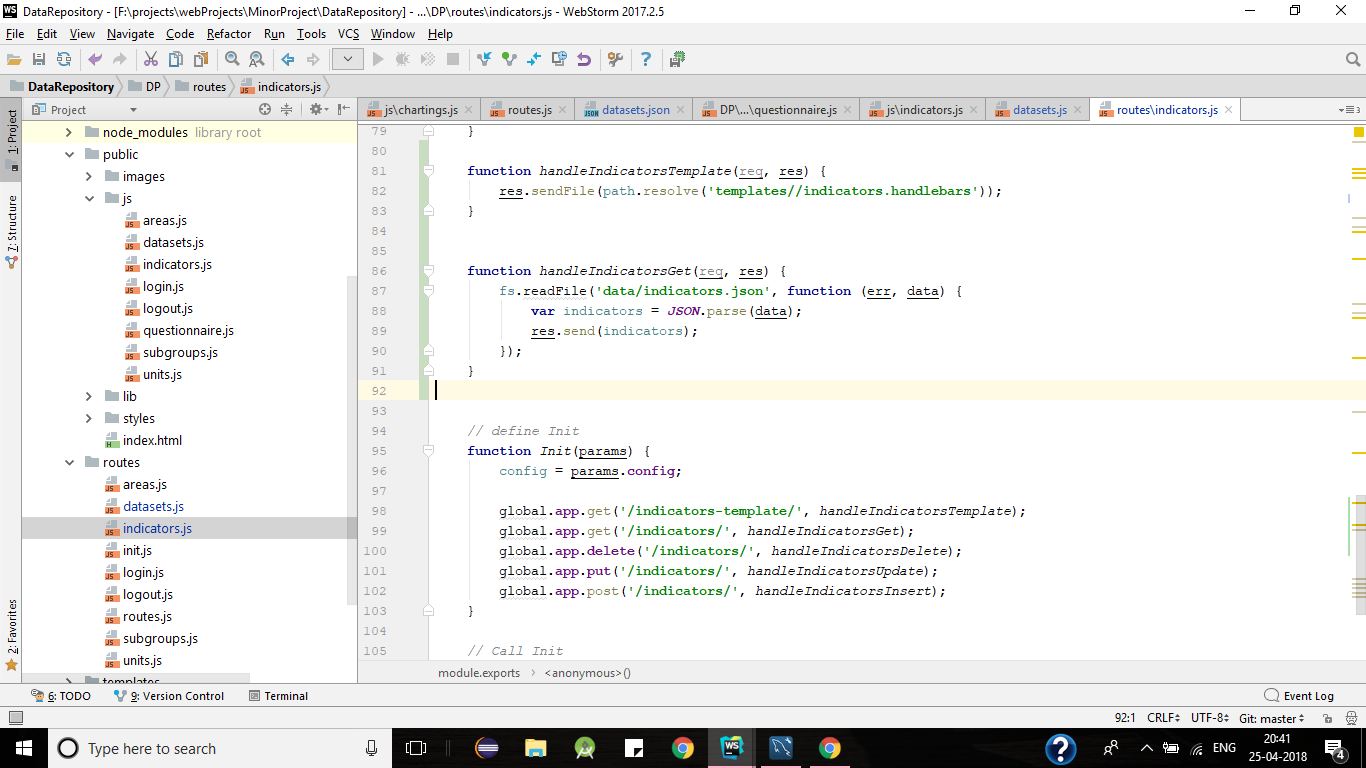
Fig. Code to make requests at server (public/js/indicators.js)

Fig. Code at Server Handling Requests (routes/indicators.js)

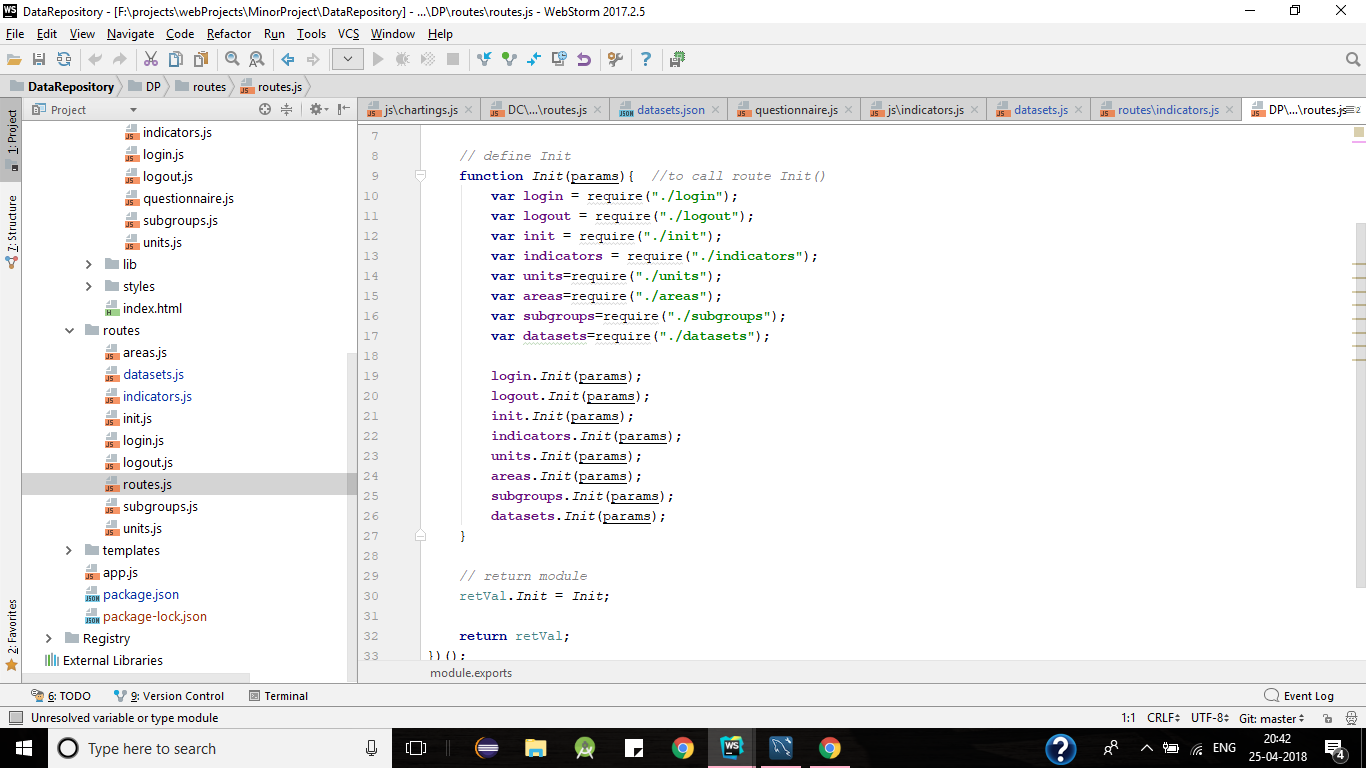
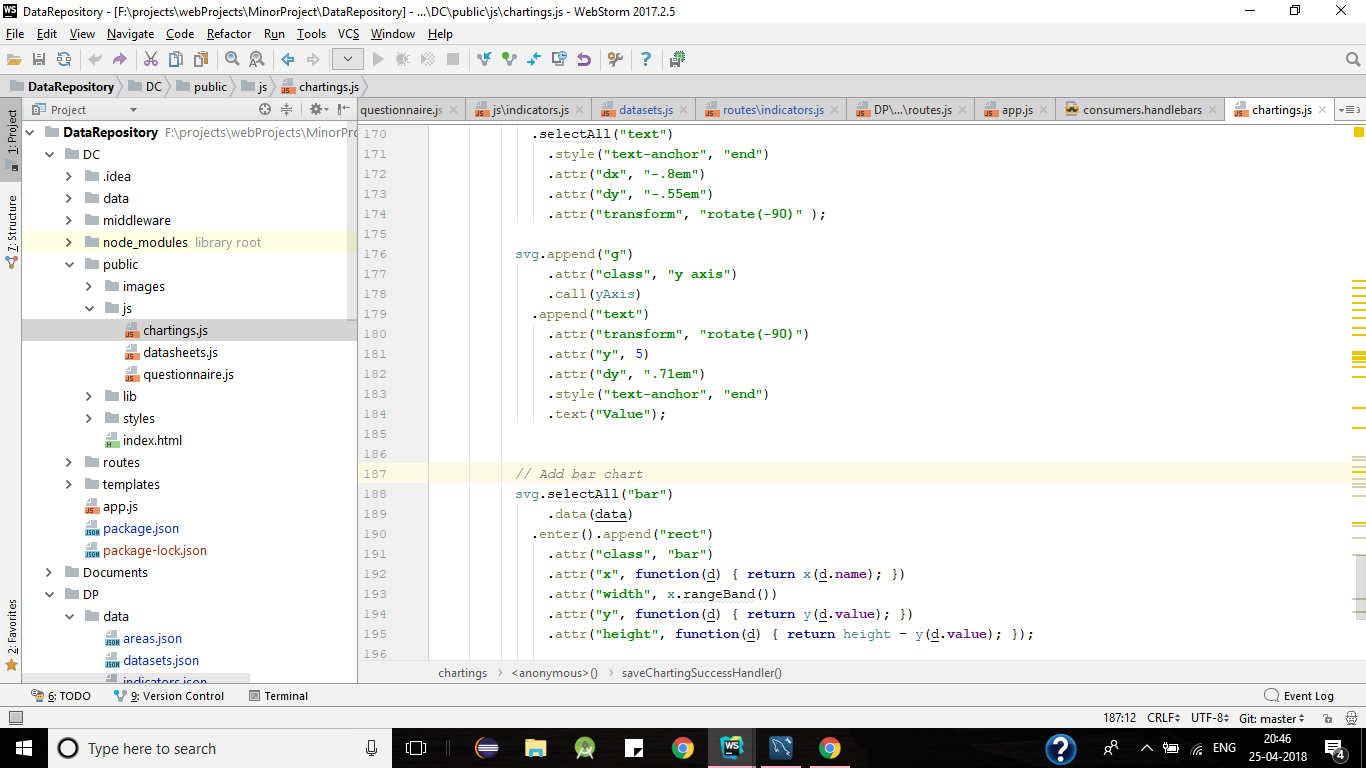
 Fig. Setting routing at Server (routes.js)

Fig. Code to show bar-graphs on screen (chartings.js)

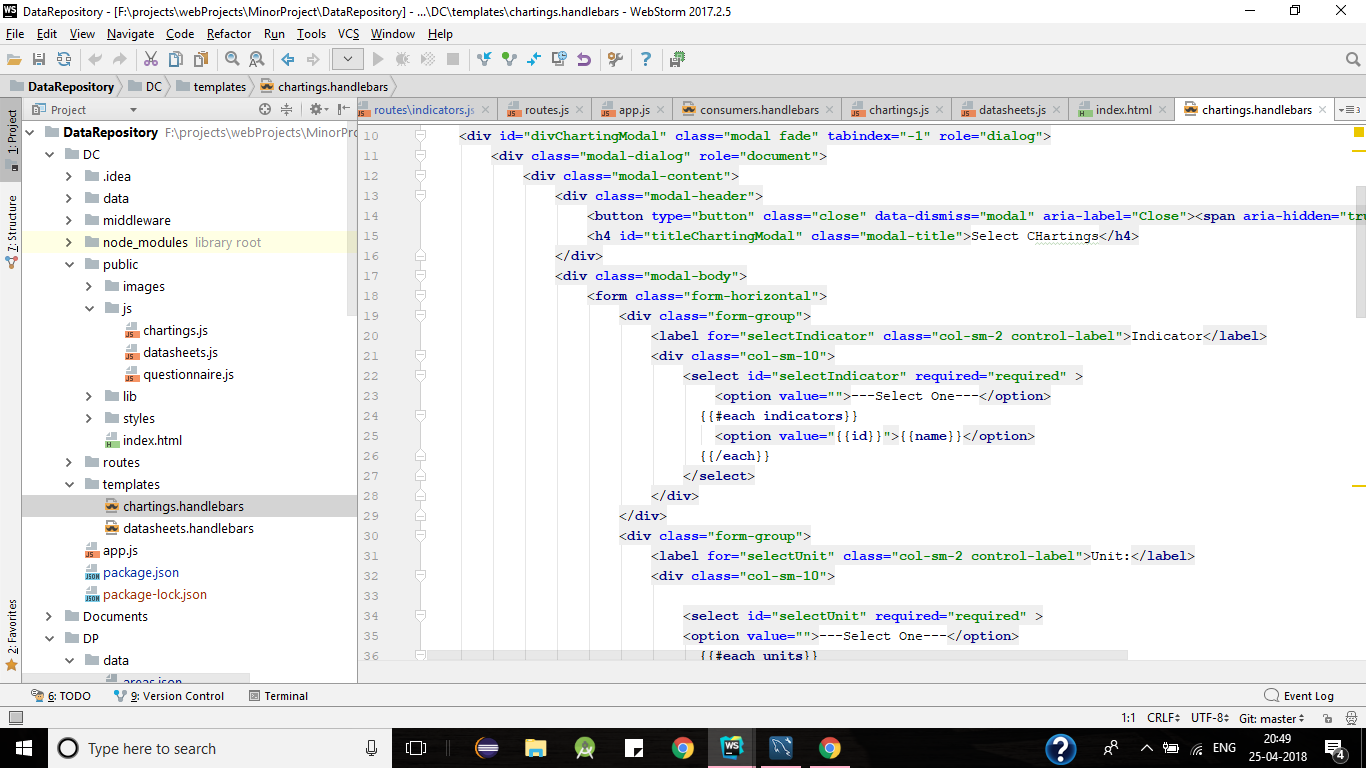
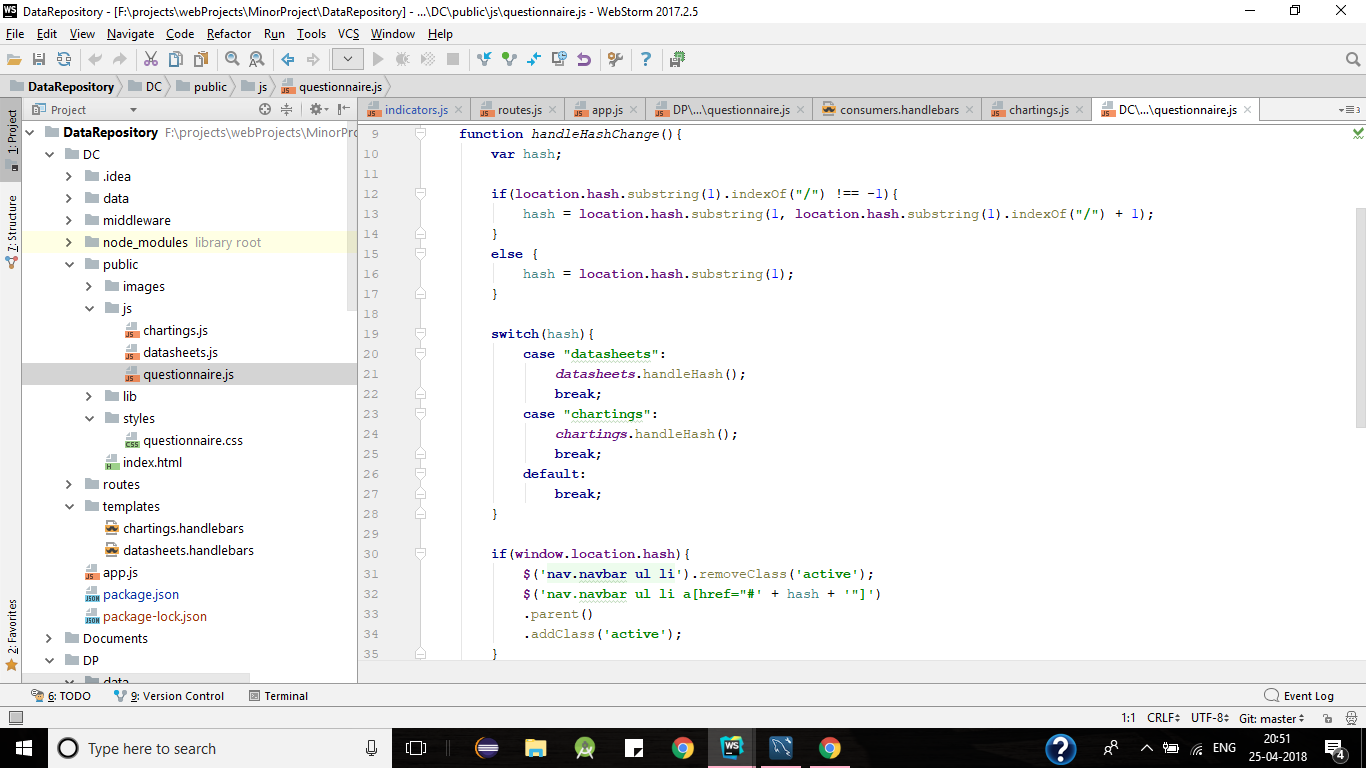
Fig. Template file for chartings(chartings.handlebars)

Fig. Code to Handle Hash Change

**11. CONCLUSION**

Data Repository for Human Development Indicators (HDI) was successfully built and hosted locally on to the local computer. All the functionalities i.e. Data Provider’s Data input in the system, Data Registry’s maintenance of data provider and data consumer and at last Data Consumer’s data consumption and finally analyzing the data are working properly in a successful manner without any glitches. Communication and Data sharing between the entities is working fine. One can create a link with pub-sub method at data Registry (or data maintainer) and share data successfully between these three entities, and can analyze the Data stored in the system Graphically (using Bar-graphs).

Pub-Sub Design pattern is implemented successfully and is working efficiently in subscribing providers and then providing data to the subscribed consumers. Textual Data provided by the Providers can be successfully observed in the graphical bar-graph form which is easy to read and easy to understand to the user. Also, Data is maintained by the Registry in an efficient manner between the entities such that communication and data sharing is fast and without any glitches or errors.

The project can be deployed live and has the potential to be used on a global platform as a powerful tool for real time data analysis between three entities for communication and data sharing between them.

**12. REFERENCES**

[1] D3.js basics (Online): http://learnjsdata.com/index.html

[2] Data Visualization fundamentals by BenSullins(Online): https://app.pluralsight.com/player?course=d3js-data-visualization-fundamentals&author=ben-sullins&name=d3js-data-visualization-fundamentals-m8-pulling-it-all-together&clip=2&mode=live

[3] Advances Web Developer Bootcamp (Online): <https://www.udemy.com/the-advanced-web-developer-bootcamp/>

[4] Real time web-app with Node.js (Online): https://www.codeschool.com/courses/real-time-web-with-node-js

[5] Building Blocks of Express.js (Online): https://www.codeschool.com/courses/building-blocks-of-express-js

[6] MySQL Documentation (Online): https://dev.mysql.com/doc/

[7] Javascript and jQuery for Data Analysis and visualization by Jon Raasch, Graham Murray, Vadim Ogievetsky, Joseph Lowery, 2014

[8] Mastering Node.js by Sandro Pasquoli, 2013

[9] Web Development with Node and Express by Ethan Brown, 2014

[10] Express Web Application Development by Hage Yappa

[11] Node Up and Running by Tom-Hughes Croucher