**CS3219 Software Engineering Principles and Patterns**

Semester 1 2017/2018

**Developer’s Manual on**

**<Conference Information Retrieval>**

**Code Repository URL:** <https://github.com/destqh/CS3219-Assignment-5-Group-1.git>

**Web-based URL:** <https://github.com/destqh/CS3219-Assignment-5-Group-1>

|  |  |
| --- | --- |
| **Student Name** | **Matriculation Number** |
| Tan Qin Hui | A0140034B |
| Boh Tuang Hwee, Jehiel | A0139995E |
| Chong Ri Yang | A0140021J |
| Musa |  |

Contents

[1. Introduction 2](#_Toc497817497)

[2. Requirement Specification 2](#_Toc497817498)

[3. Software Development Process 3](#_Toc497817499)

[4. Visualization 3](#_Toc497817500)

[4.1. Trend 1: Transition over 3](#_Toc497817501)

[4.2. Trend 2: Contemporary comparison 3](#_Toc497817502)

[4.3. Trend 3: Top N X of Y 4](#_Toc497817503)

[4.4. Trend 4: Publication Relevance 4](#_Toc497817504)

[4.5. Citation Network 4](#_Toc497817505)

[4.6. Step Wise Method 4](#_Toc497817506)

[5. Architecture 4](#_Toc497817507)

[5.1. Web Server 5](#_Toc497817508)

[5.2. App Server 5](#_Toc497817509)

[5.3. Database Server 6](#_Toc497817510)

[6. Development Tools & Frameworks 6](#_Toc497817511)

[6.1. Node.js 6](#_Toc497817512)

[6.1.1. Express Framework 7](#_Toc497817513)

[6.1.2. Testing Framework 8](#_Toc497817514)

[6.2. GitHub 9](#_Toc497817515)

[6.3. Travis CI 9](#_Toc497817516)

[7. Additional Features… (Uploading of Data) 10](#_Toc497817517)

# Introduction

Every day, there are thousands of scientific documents being added to our knowledge base. It is thus of great interest in the scientific community to track the current trends on research topics and their future implications. However, it is a challenge to digest a big source of text and make meaning insights from data. Thus, the aim of this project is to come up with a tool, Conference Information Retrieval (CIR), to aid the process.

Since data is more easily understood when it is visually presented, patterns which often go unnoticed in a dataset, quickly becomes obvious when we visualise it on for example a graphical chart. Thus, CIR has been tailored to communicate information in a quick and visual way to ease the process of analysing the data.

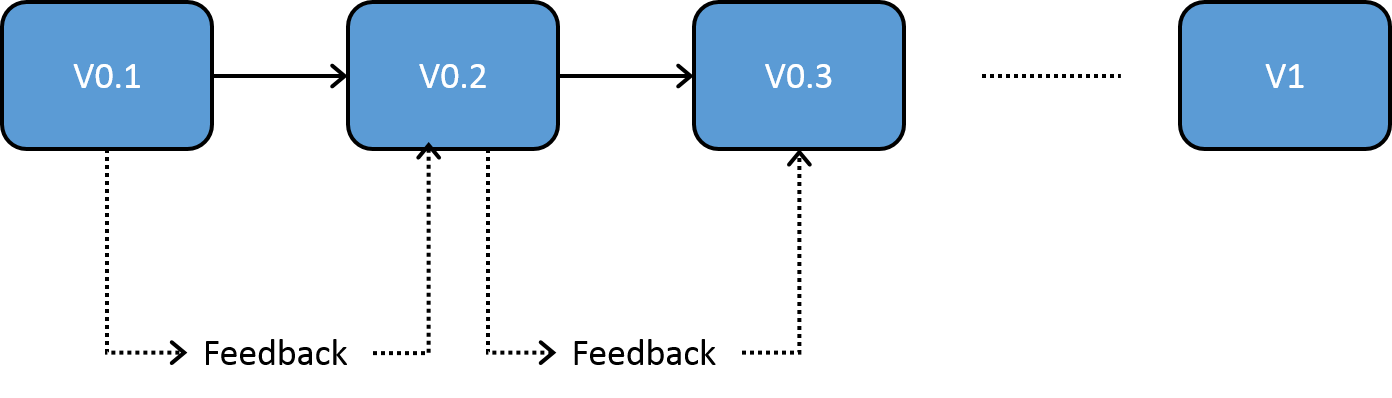
Individual Contribution…

# Requirement Specification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | **Description** | **Type** | **Priority** | **Done** |
| FR-1 | User is able to view the trend of the number of papers published overtime in a particular conference in a line chart. | FR | H |  |
| FR-2 | User is able to compare the trend of the number of papers published overtime between 2 or more conferences in a line chart. | FR | H |  |
| FR-3 | User is able to add new lines to the line chart to compare number of papers published overtime between different conferences. | FR | M |  |
| FR-4 | User is able to hover a particular line in the line chart to view number of papers for the corresponding year in a tooltip. | FR | M |  |
| FR-5 | User is able to view the top N X of Y in a bar chart, where  N is a number, X is any valid attributes of Y and Y can be authors, conference, cited authors, citing authors, etc | FR | H |  |
| FR-6 | User is able to hover a particular bar in the bar chart to view the attributes in a tooltip. | FR | M |  |
| FR-7 | User is able to filter the bar chart according to the years. | FR | L |  |
| FR-8 | User is able to view the incoming citations of any papers in a citation web chart. | FR | H |  |
| FR-9 | User is able to automatically layout the citation web. | FR | L | **X** |
| FR-10 | User is able to drag a node in the citation web to manually position it. | FR | L | **X** |
| FR-11 | User is able zoom in on any nodes in the citation web to display itself and its immediate neighbors. | FR | M |  |
| FR-12 | User is able to pan across the chart by dragging on the chart. | FR | L | **X** |
| FR-13 | User is able to zoom in/out on a particular location in the citation web. | FR | L |  |
| FR-14 | User is able hover a particular paper in the citation web to view the title and authors in a tooltip. | FR | M |  |
| FR-15 | User is able to upload json files to visualize information | FR | L |  |
| FR-16 | User is able to upload xml files to visualize information | FR | L |  |
| NFR-1 | Citation web should display base papers and citation papers in different colors. | NFR | M |  |
| NFR-2 | Citation Web should be able to display up to 50 nodes at once and maintain frame rate above 50fps. | NFR | L | **X** |
| NFR-3 | Citation Web load up to 50 nodes within 2 seconds after user enters query. | NFR | M |  |
| NFR-4 | Line chart should display different lines in different colors. | NFR | M |  |
| NFR-5 | Line chart should load within 1 sec after user enters query | NFR | M |  |
| NFR-5 | Bar chart should load within 1 sec after user enters query | NFR | M |  |

# Software Development Process

The Software Development Life Cycle process model of choice is the iterative model, where the whole process is broken up into iterations.



Our group meets every Monday and Friday to get feedback on each other’s progress, and to determine the tasks to be done until the next meeting. Feedback gathered during the meeting is fed to the next iteration as well. Typically, our iterations last about 3-4 days. We have also taken a breadth-first approach to our iterations planning in which we work on every major component in parallel every iteration.

# Visualization

The application uses d3 visualization tool to correlate conference papers data in JSON object and generate visualizations for easy data interpretation.

## Trend 1: Transition over Time

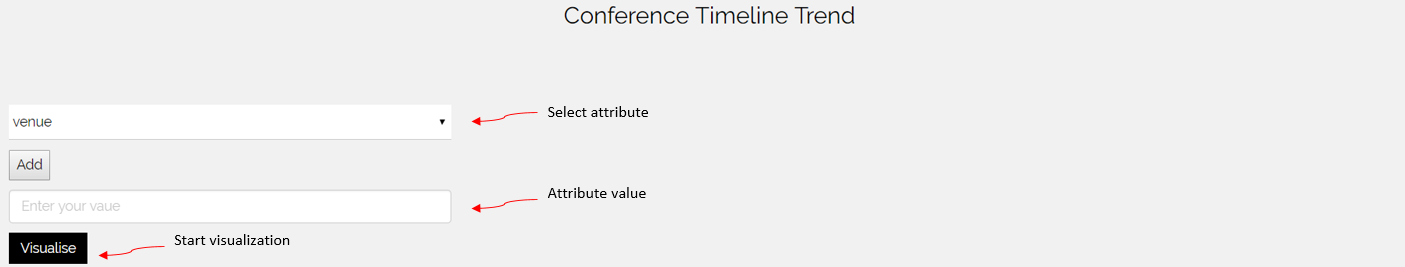
For transition over time query, the application will generate a graph to show a trend of particular attribute over the range of years.

### Graph Type:

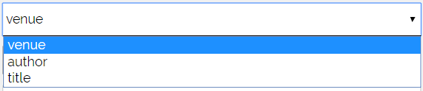
* Line Graph (Y-axis: Number of occurrence, X-axis: Year)

### UI

#### User input



#### Attribute selection (Dropdown)



#### D3 Visualization (Attribute ‘venue’, Attribute value: ‘ArXiv’)

## Trend 2: Contemporary comparison

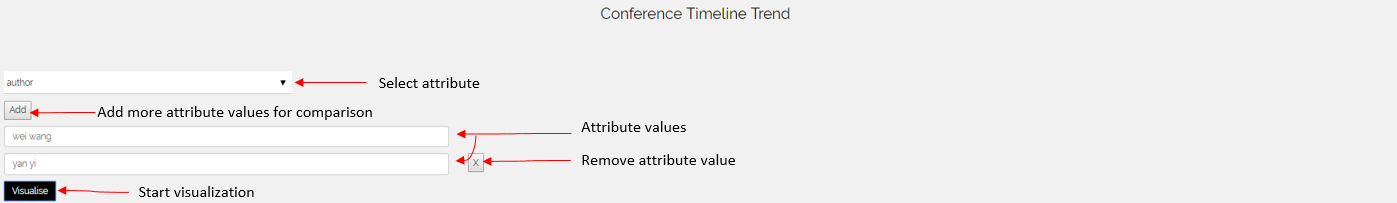
For contemporary comparison query, the application will generate a graph showing **overlapping** trends of particular attribute over the range of years.

### Graph Type:

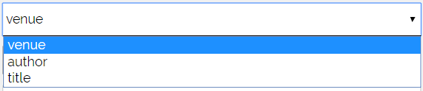
* Line Graph (Y-axis: Number of occurrence, X-axis: Year)

### UI

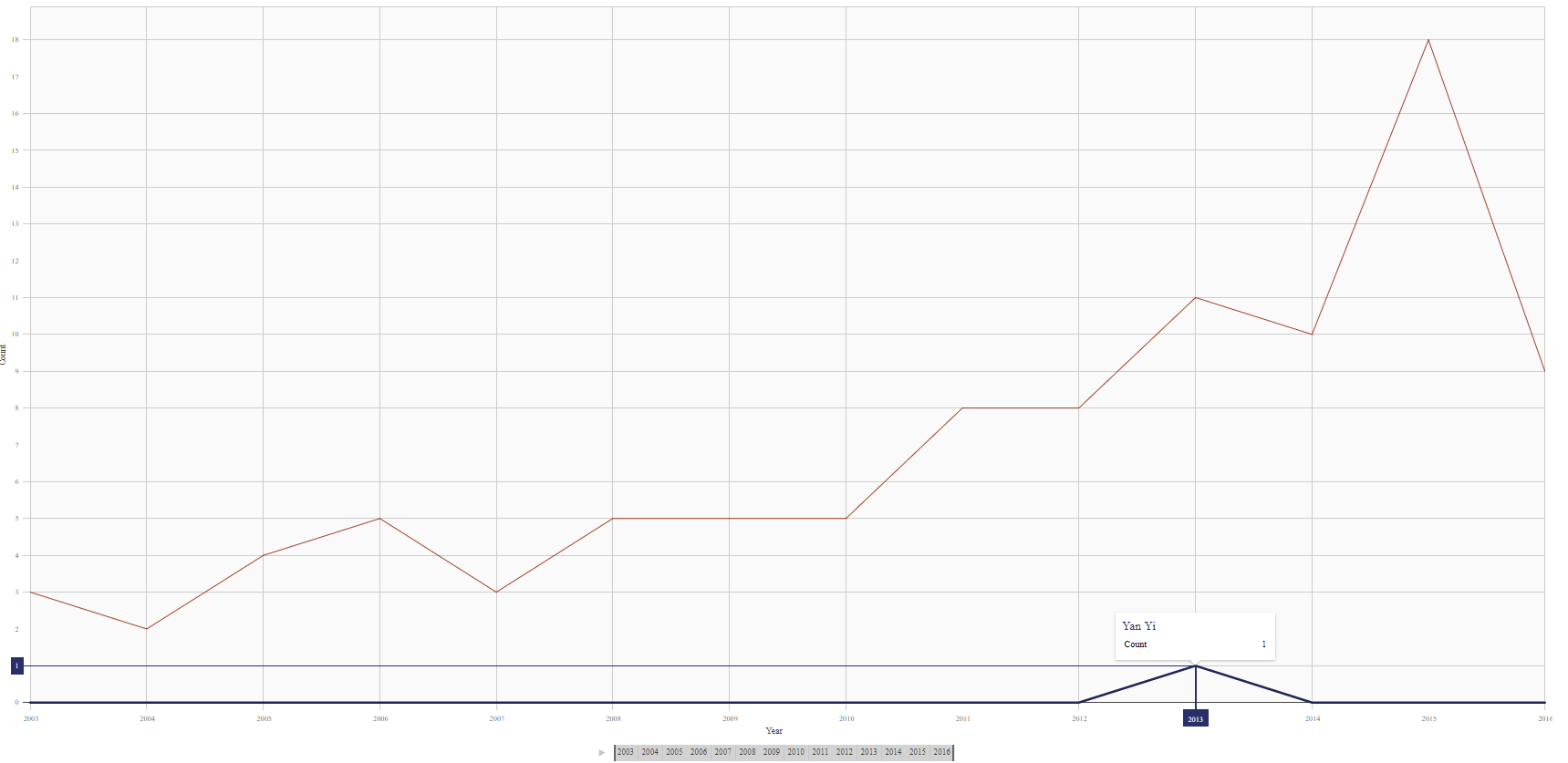
#### User input



#### Attribute selection (Dropdown)



#### D3 Visualization (Attribute: ‘author’, Attribute values: ‘wei wang’ & ‘yan yi’)



## Trend 3: Top N X of Y

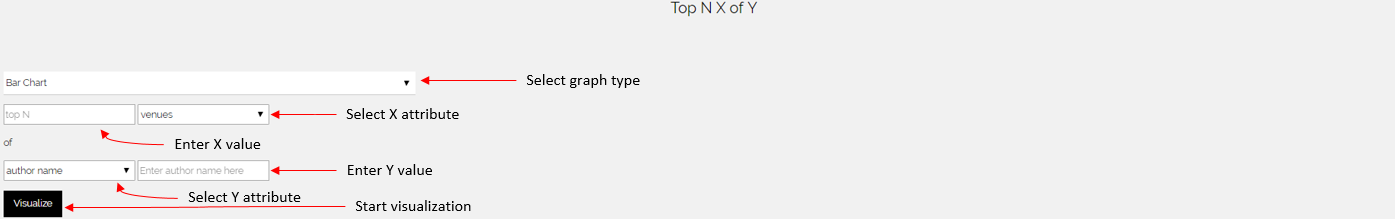
For Top N X of Y query, the application will generate a graph which will show and compare the top N X, where N is a number, X could be (venues, authors, cited papers or citing papers), and Y could be (author name or venue).

### Graph Type:

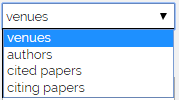
* Horizontal bar graph
* Treemap

### UI

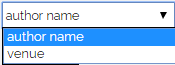
#### User input



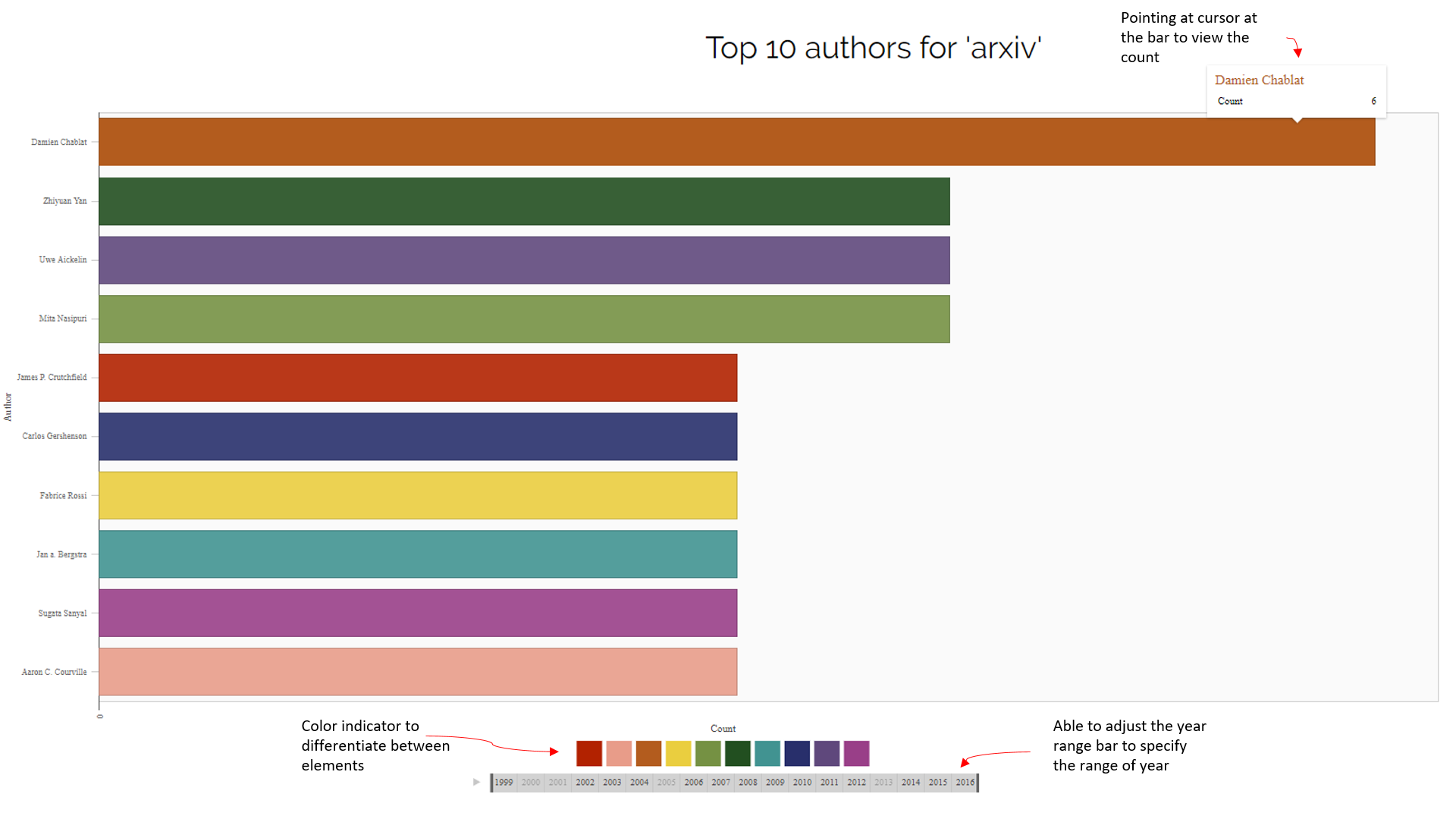
#### Attribute X selection (Dropdown)



#### Attribute Y selection (Dropdown)



#### D3 Visualization (Graph Type: ‘Bar chart’, N: ‘10’, X: ‘authors’, Y: ‘venue’, Y-value: ‘arXiv’)



## Trend 4: In-Citation Network

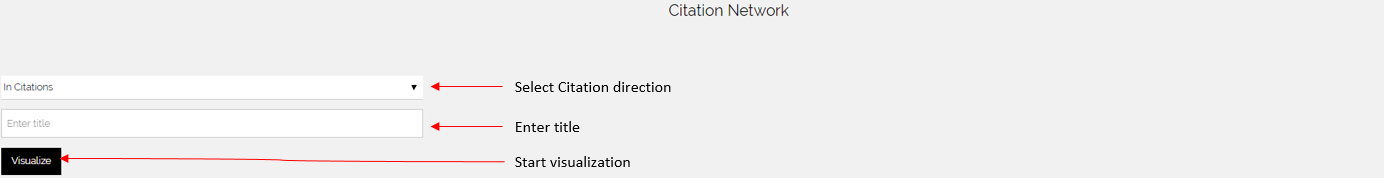
For In-Citation Network query, the application will generate a graph which will show a network graph up to 2 levels of base paper citations. If the base paper is A, for A <- B <- C, it means that C cites B, and B cites A.

### Graph Type:

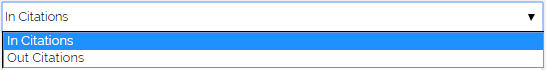
* Network Graph

### UI

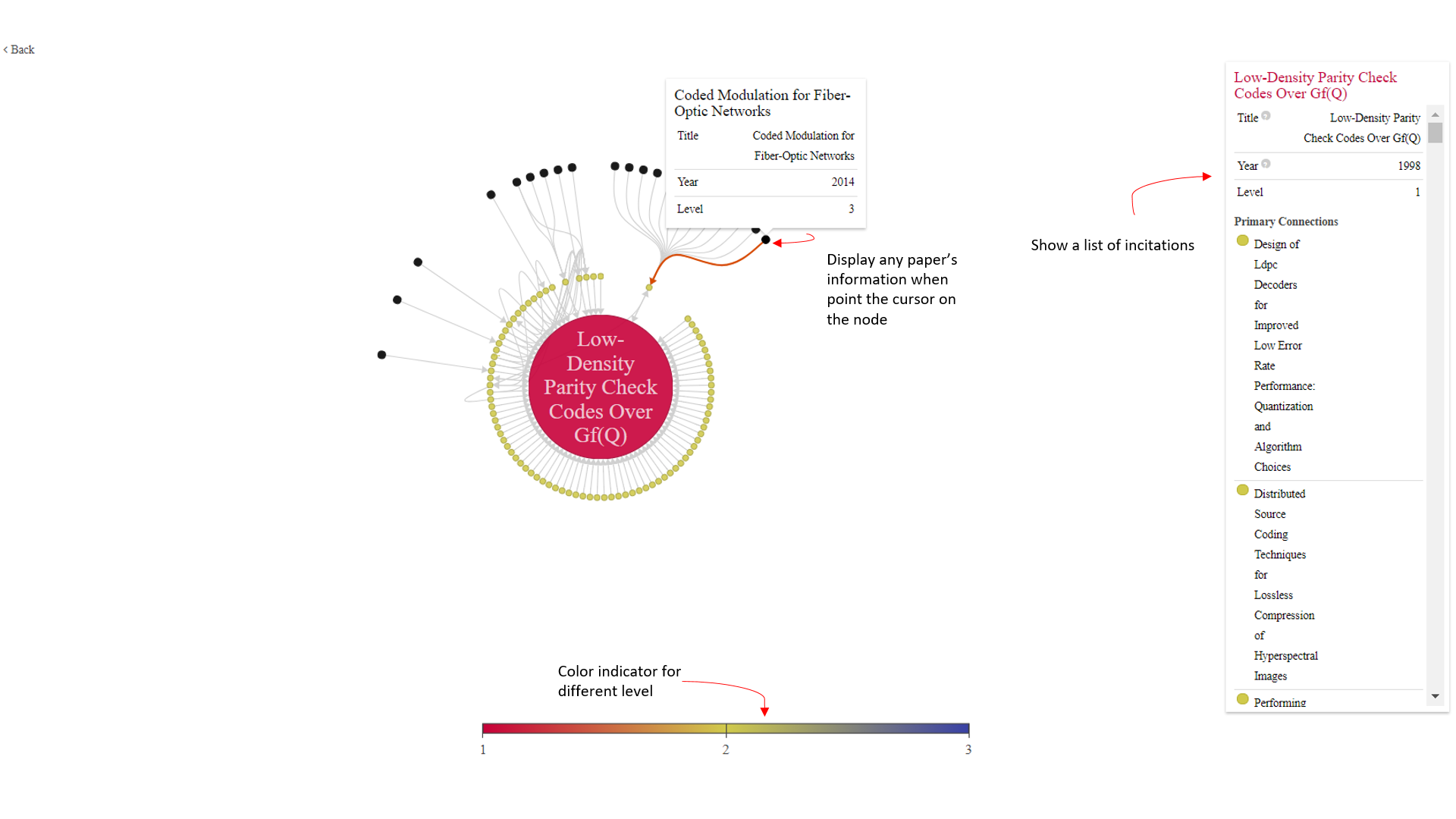
#### User input



#### Citation direction selection (Dropdown)



#### D3 Visualization (Citation direction: ‘In Citation’, Title: ‘Low-density parity check codes over GF(q)’)



## Trend 5: Out-Citation Network

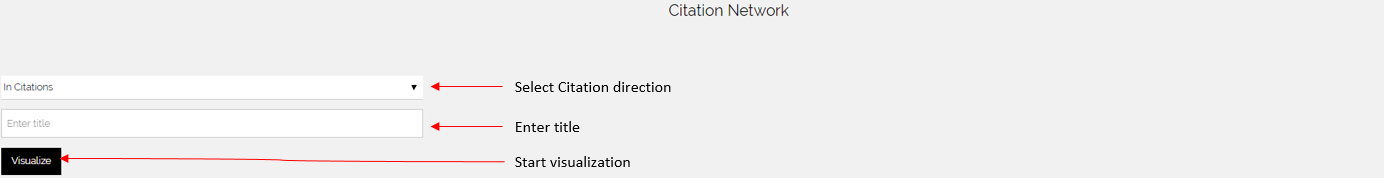
For Out-Citation Network query, the application will generate a graph which will show a network graph up to 2 levels of base paper citations. If the base paper is A, for A -> B -> C, it means that C is cited by B, and B is cited by A.

### Graph Type:

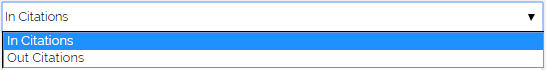
* Network Graph

### UI

#### User input



#### Citation direction selection (Dropdown)



### D3 Visualization (Citation direction: ‘Out Citation’, Title: ‘add new title here’)

Musa please check outCitation network have error on my side

## Step Wise Method

Musa please input here…

# Architecture

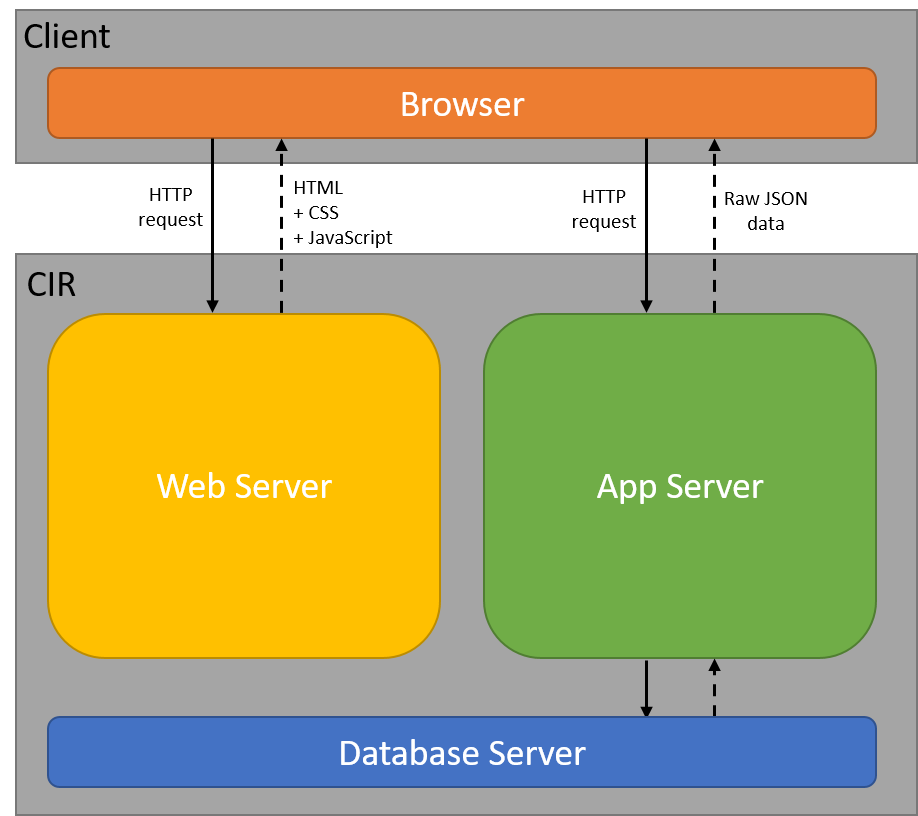


Figure 1 Architecture design of CIR

## Distributed System

The application is distributed system, where client access the application using a browser on front end-computer, and sends HTTP request to web server and app server which may be hosted on different physical machines or virtual machines at the back-end, as shown in figure 1 above.

Our system adopts a web application architecture where the client would query our servers for the HTML pages and JSON data for visualization of conference information.

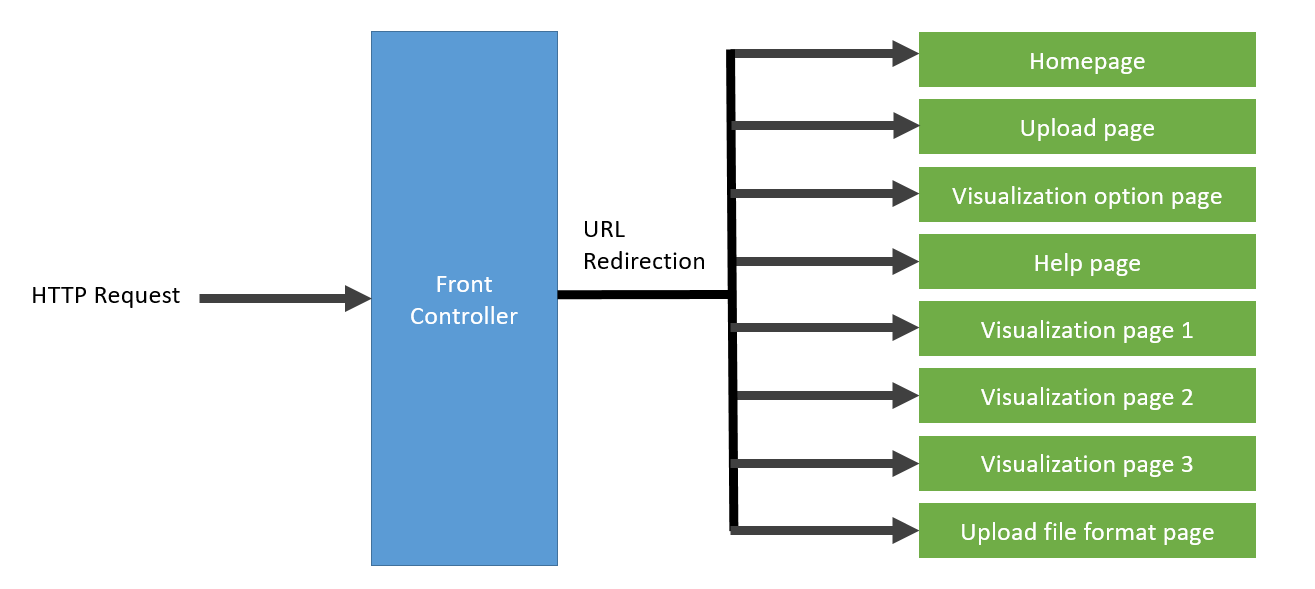
Client would first download the HTML pages from our web server’s front-end which contains JavaScript code to get JSON data from our web server’s back-end. The JSON data is then used by the JavaScript application to generate the conference information into the HTML content of the page.

This architecture increase the performance of our system as JavaScript for visualization is run on the client side which reduce processing on the servers. This allows the servers to be able to service more users and only app server is required to scale if there is a need to accommodate more users.

## Web Server

The Web Server provides Client with web pages to be displayed on the Client side with additional scripts for querying data from App Server. Web Server uses PHP, HTML and JavaScript languages.

#### Front Controller Design Pattern

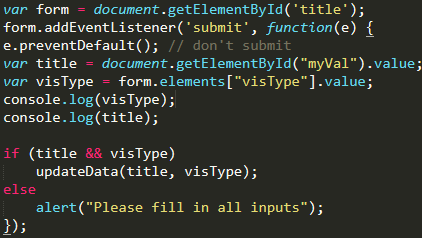


*Figure 2 Component diagram of web server’s front controller*

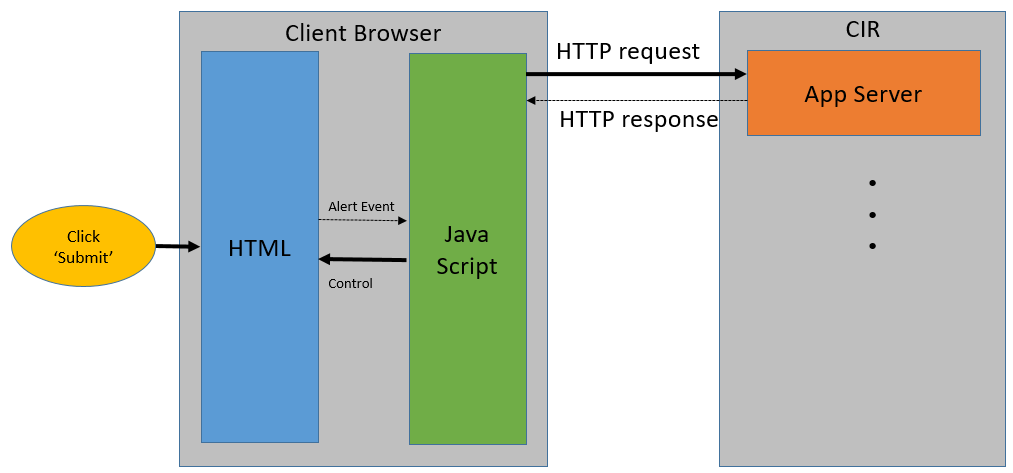
Front controller is implemented on the apache web server, which allows a single point of entry to handle all HTTP request and redirect to user to appropriate webpage. Front controller pattern improves modularity and a single location to easily modify and manage URL redirection. Webpages required to retrieve or update the information from database will communicate with the immediate layer, data access layer.

#### Observer Pattern

Observer pattern is implemented to respond to user’s input of HTML form and perform computations, followed by displaying the results. Figure 3 shows event listener is added to the document submit element. Upon clicking of the submit button on the form, the java script will make HTTP request to back-end application server, which will return the raw JSON data based on the parameters input by user, as shown in the interaction diagram in Figure 4.



*Figure 3 Code snippet of Event Listener visualizing citation network*



*Figure 4 Interaction between client browser and backend when user click button to start visualization*

## App Server

The App Server provides Client with the data of conference information for display on the Client side. App Server uses Node.js platform using express framework to provide RESTful API for the server. JSON object is used to bundle all data items to be transfer between components to reduce the number of method calls needed.

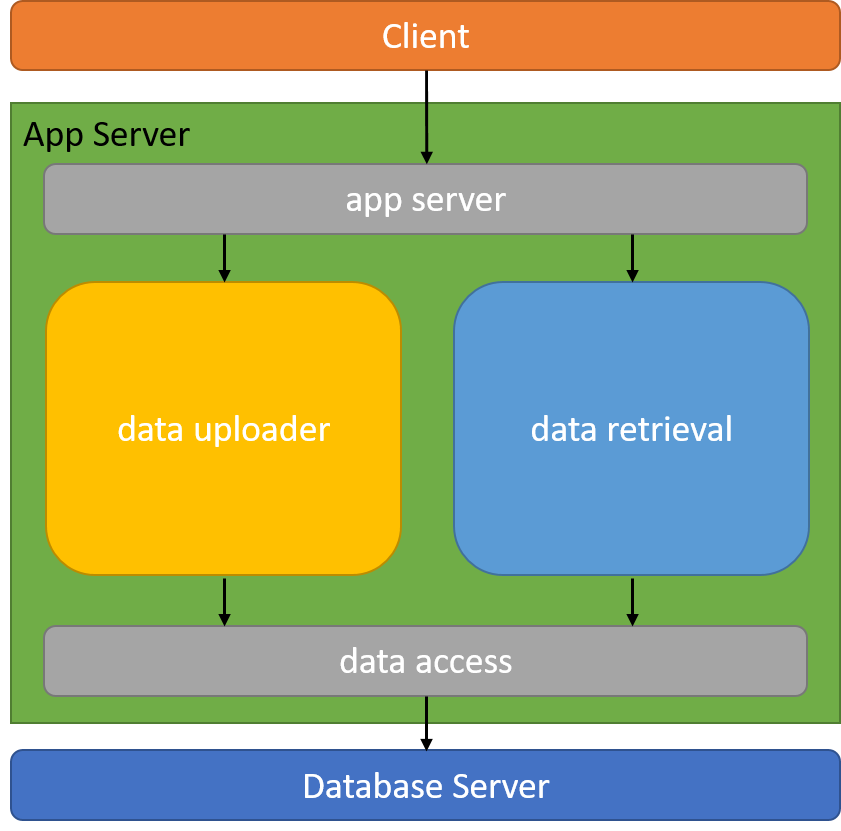


Figure 5 - Component Diagram for App Server

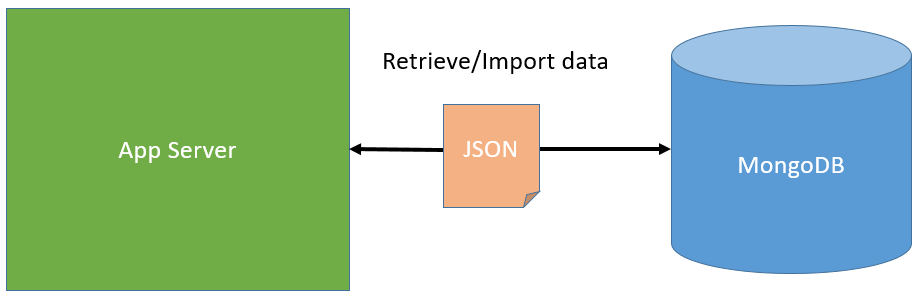
*app server*: *app server* serve as facade to our App Server where it would route the request to the data uploader or retrieval component using RESTful API. This reduces the coupling between the client and the App Server so that changes made on the App Server does not affect the Client.

*data uploader*: *data uploader* handles the uploading of data by the client. It will save the file uploaded by the client as a temporary file in the App Server, then convert the data in the file to the correct format and import the data to the Database Server.

*data retrieval*: *data retrieval* handles all data request from the client. It will query the Database Server for the data and format the data to answer the client request.

*data access*: *data access* serve as an adapter to the Database Server for App Server. This allows App Server logic to able to query the Database Server regardless of the database management system used by masking away the process of querying the Database Server.

### Data Transfer Object



*Figure 6 – Retrieval and importing JSON Object to mongoDB database*

JSON object is used to encapsulate the data of conference papers for visualization. All conference papers data are encapsulated in JSON object and stored in MongoDB database via import initially. For data retrieval, the App server will query document with filtering from MongoDB database and the data retrieved in JSON object will be sent to client browser java script to perform d3 virsualization.

|  |  |  |
| --- | --- | --- |
| **RESTful API** | **Parameters** | **Description** |
| The API of facade | venue, rank… | Do what?  Return what? |
| Musa please fill up this table | Just put the parameter it expect. No need put the type and so on… | Just put a brief overview of all the API. More details of the API like the one you did in assignment 4 can be put in appendix. |

## Database Server

MongoDB is used in our project to store our data so that our system can support a large dataset while being able to handle queries fast through the use of MongoDB query language. To interface with MongoDB, *mongojs* module is installed into Node.js in the App Server to provide the APIs for data queries by *data access.*

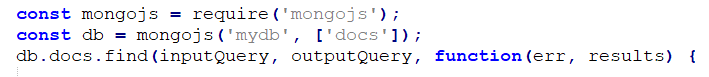


Figure 7 - Code snippet of a MongoDB query

# Development Tools & Frameworks

## Node.js

Our system is based on Node.js platform which provide an asynchronous event driven JavaScript runtime environment. This allow our server to handle concurrent connections with minimal overhead for answering queries fast and to unify our programming language of our backend and frontend development for seamless interfacing between the components and remove duplication of common codes.

Node.js’ package ecosystem, npm, helps make building and managing our project easy.

1. Project can be easily build and test using a single command:
   1. Build and run project: *npm start*
   2. Test project: *npm test*
2. Ease of managing project dependency with a single command:
   1. Install dependencies: *npm install*
3. Development process within the team is unified as all project dependencies and version is stored in *package.json* file which other developers would be able to install accordingly.

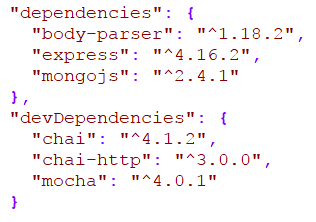


Figure 8 - Dependencies of project in package.json

1. Node.js also made it easy to integrate with continuous integration (CI) tool, Travis CI which is mentioned below, to build and run tests to verify the code and detect problems.

### Express Framework

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop our web application. Features used in our project includes:

1. Setting up middleware for preventing injection into database before handling any HTTP requests.

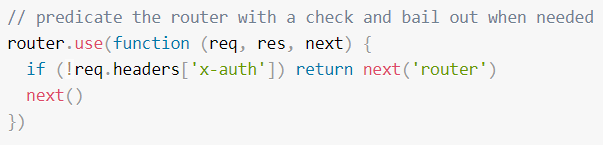


Figure 9 - Code snippet of parameters validation

1. Routes HTTP requests to front-end or back-end depending on the HTTP method and URL.

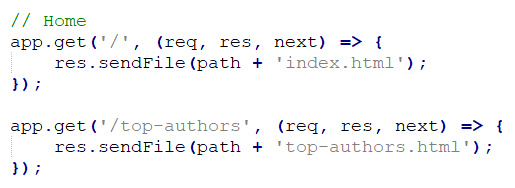


Figure 10 - Code snippet of routing requests

### Testing Framework

Our testing framework uses 2 Node.js modules, *Mocha* and *Chai*:

1. *Mocha* is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple.
2. *Chai* is an assertion library used in *Mocha* for readable interfaces for comparing expected and actual results.

*Mocha* command is added to *package.json* to allow *npm* to start test using *Mocha*.



Figure 11 - Test script in package.json

Test files are written and stored in test folder where Mocha would use by default.

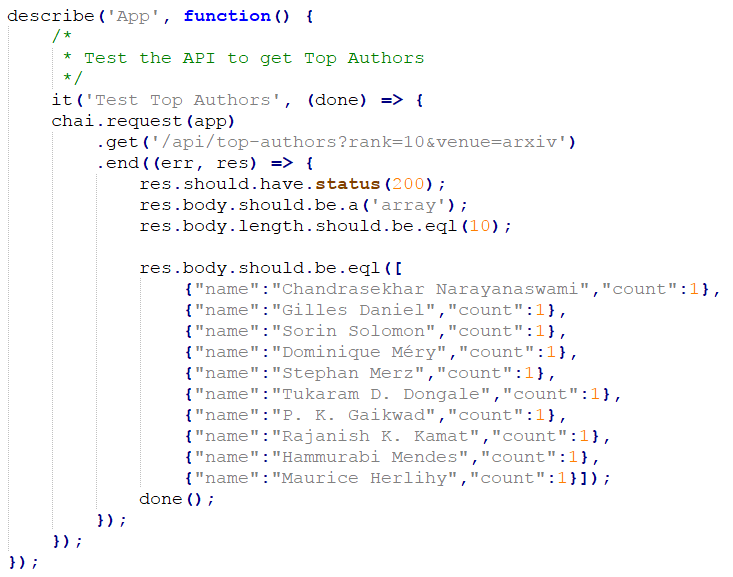


Figure 12 - Code snippet of a test case

Test cases can be run by using *npm test* as mentioned above.



Figure 13 - Screenshot of test run

## GitHub

GitHub is version control software to manage changes to our project without overwriting any part of that project. It allow us to make changes to the project simultaneously without losing any changes when conflict occurs and keep track of changes made throughout the development.

GitHub also provides a central repository which contain all our files except for the data dump and Node.js dependencies. This allow us to easily integrate our codes by pulling codes into our local repository and pushing any modifications to the central repository which can be pulled anytime by the other developers.

Node.js dependencies can be easily installed locally as mentioned above and the dependency files are not pushed to GitHub to allow Travis CI to install its own dependencies during testing. The actual data dump is also not pushed to GitHub as the data size is too large and exceed the GitHub’s maximum file size limit. Furthermore, as Git repository contains every version of every file, it is not practical to have multiple revisions of large files which increase the clone and fetch time for the other developers.

## Travis CI

Continuous Integration (CI) is implemented into our development process to allow us to detect problems at every integration phase by automatically building the project and running test cases to verify that the changes made. Travis CI is used for this purpose as it can be easily integrated with our project in GitHub.

Travis CI’s configuration file, *.travis.yml,* is configured to start Node.js and MongoDB service and run tests on our project using *npm test.*

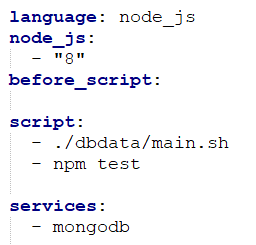


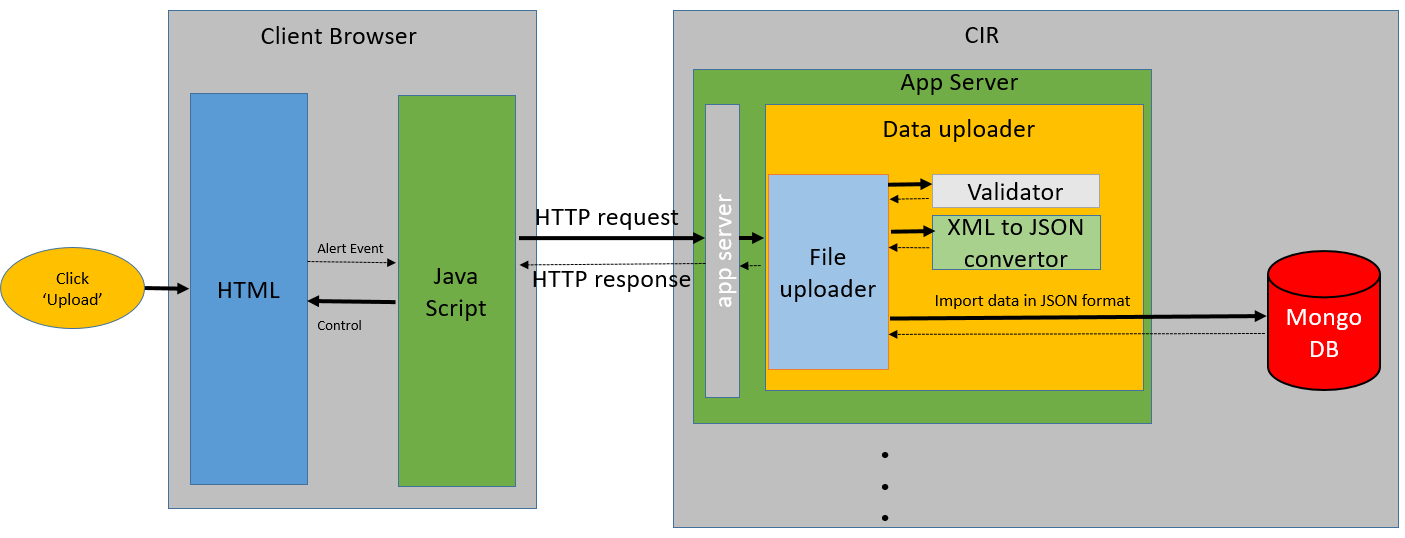
Figure 14 - Configuration of Travis CI

# Additional Features… (Uploading of Data)

## Uploading of data

User is able to upload new conference information data in JSON or XML format easily from the UI at browser.

Musa please check diagram if need change use p1.pptx



### UI for uploading

Musa put in the screen shot of upload