Data Analytics Workbench For Educational Data

Ankush Arora, Palak Agrawal, Prashant Gupta, Purva Bansal, Saatvik Shah

Fundamental Research Group
Project In-Charge : Mr. Nagesh Karmali

IIT Bombay

July 3, 2014

Outline

Introduction[1]

- Open EdX is an open source platform for building MOOCs with various advanced features
- EdX generates tremendous amounts of data(we'll get a glimpse in the next slide)
- Tremendous amount of data = Considerable opportunities for EDM
- Identify merituous and demeritiuous factors in MOOC learning

How big is tremendous?

Simple Number Crunching

1. At a mock test held on 14th May on IITBombayX[2]
   - ∼80 students participated
   - ∼2-3 hours long per student
   - ∼100-120 questions answered
   - ∼1.5GB of Data Generated
   - ∼$1.5/80 = 0.0188$GB generated per student interacting with the system in this period

How big is tremendous?

1. For an average EdX Course
   - ∼40000 students participate[3]
   - ∼2-4 hours long per student per week(take 2-3 hours for our convenience)
   - Interacts with problems,forums,videos,more..
   - ∼0.0188*40000 = 750GB generated per student interacting with the system in one week

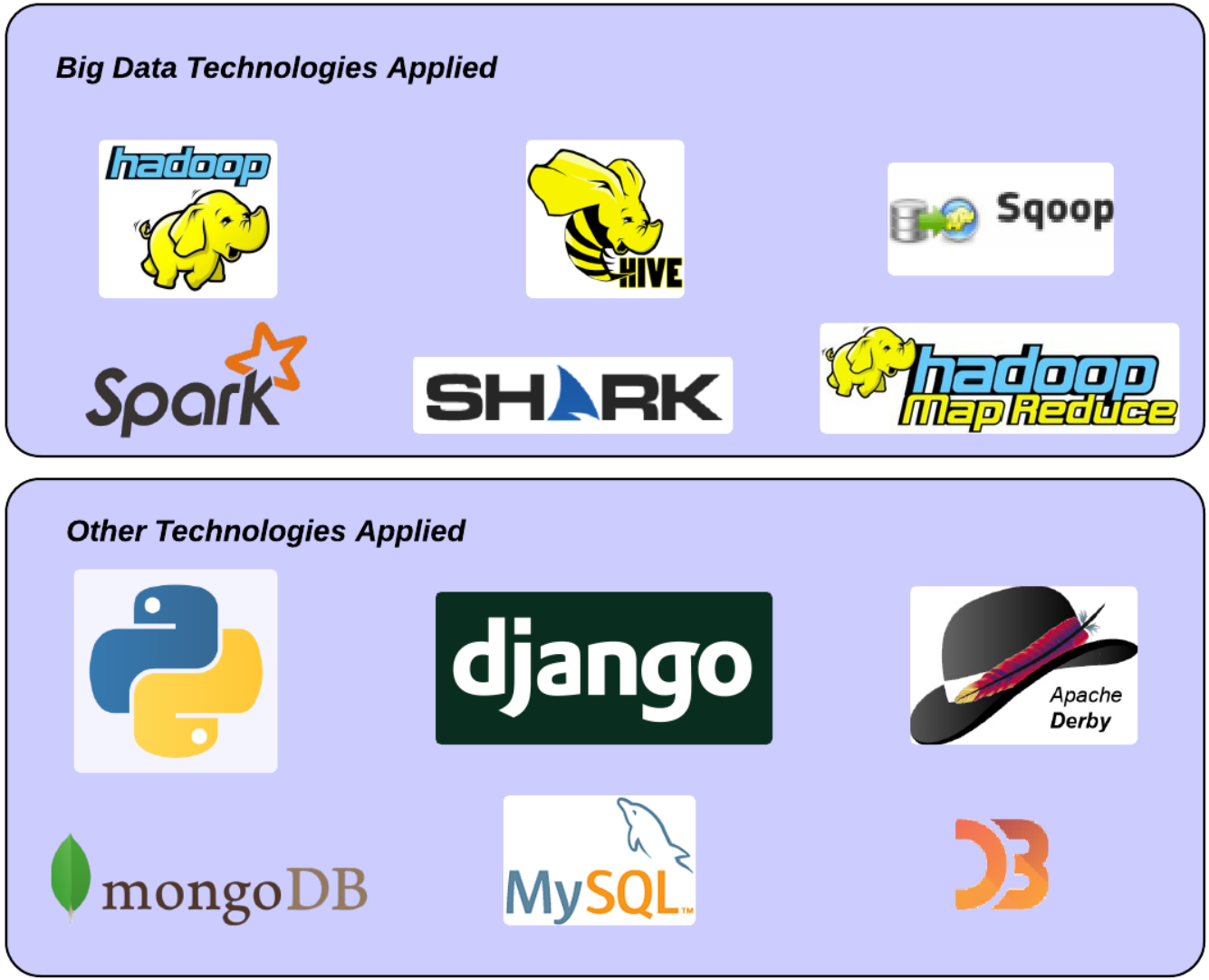
Question

So is this at the level of Big Data?

Objective

- Create a Data Analytics workbench for
  1. Automating ETL
     - Segregate and Organize relevant data from Data Source
     - Transform the Data by processing and providing output which is ready to load
     - Load the data onto Big Data Platform
  2. Visualization
     - Vital step of Data Analysis
     - Relevant Visualisations usable by MOOC staff as well as EDM researchers to make useful inferences
- Slick and User Friendly GUI for convenience of end-user

Tools for Big Data[4][5][6][7]



edX Data[9]

EdX provides two types of data to partner institutions who are running classes on edx.org and edge.edx.org:

- Log (event tracking) data
- Database data, including student information

edX conventions:

1. edX uses MySQL 5.1 relational database system with InnoDb storage engine
2. All datetimes are stored as UTC (Coordinated Universal Time)

Event tracking data

This data contains information about every interaction of every student. The tar file is cumulative.
Events that are logged for interactions with the LMS

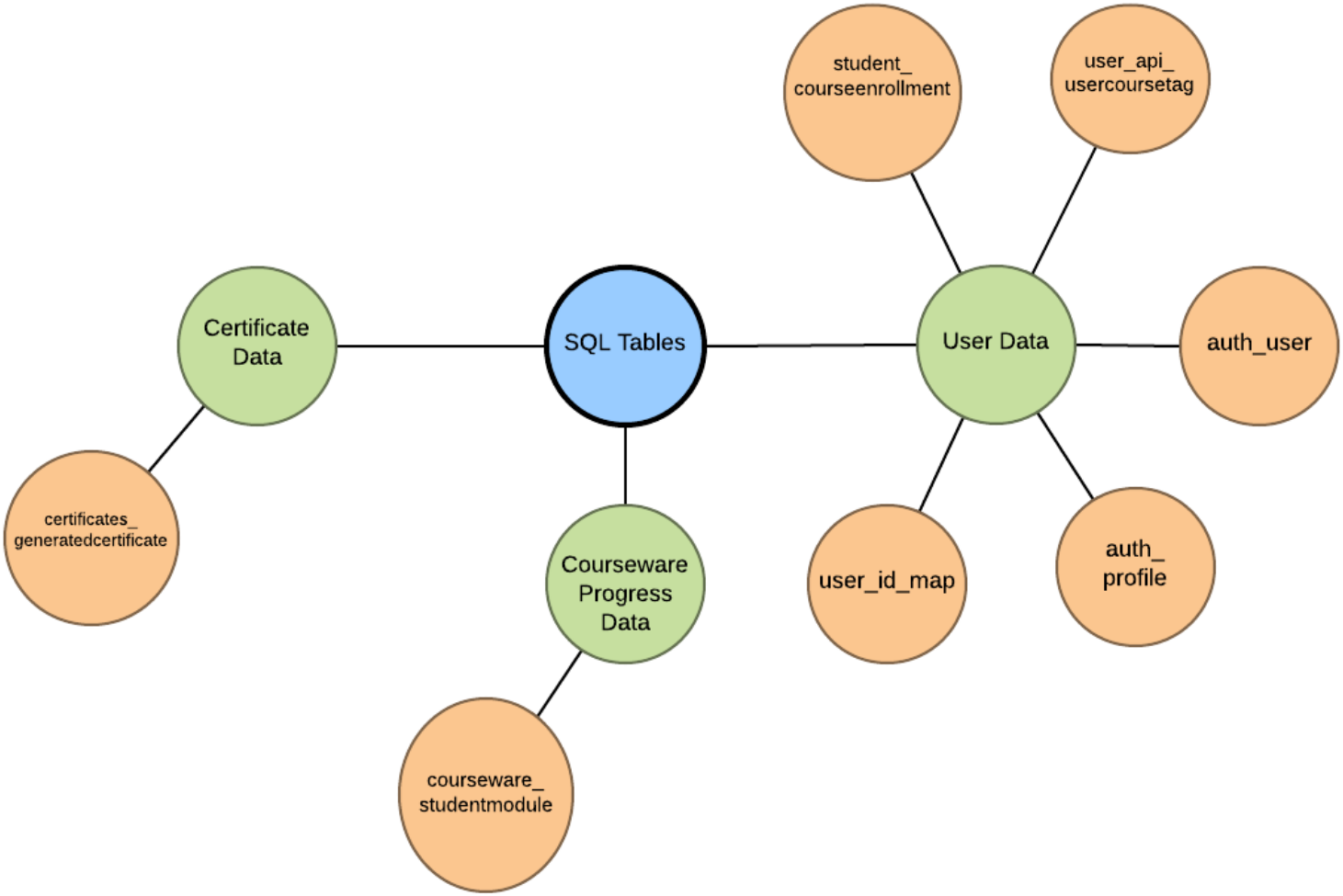
- Enrollment Events
- Navigational Events
- Video Interaction Events
- Textbook Interaction Events
- Problem Interaction Events
- Forum Events

Database data

Different types of data that edX delivers.

No.	Type
1	Authorized Users
2	Authorized User Profiles
3	Generated Certificates
4	Courseware
5	Forums
6	Course Enrollment
7	User IDs
8	Wiki articles

Sql Tables


student_
courseenrollment
user_api_
usercoursetag
Certificate
Data
SQL Tables
User Data
auth_user
certificates_
generatedcertificate
Courseware
Progress
Data
user_id_map
auth_
profile
courseware_
studentmodule

ETL
Receiving Input

1. GUI Input is required where the end user provides basic information about data to load

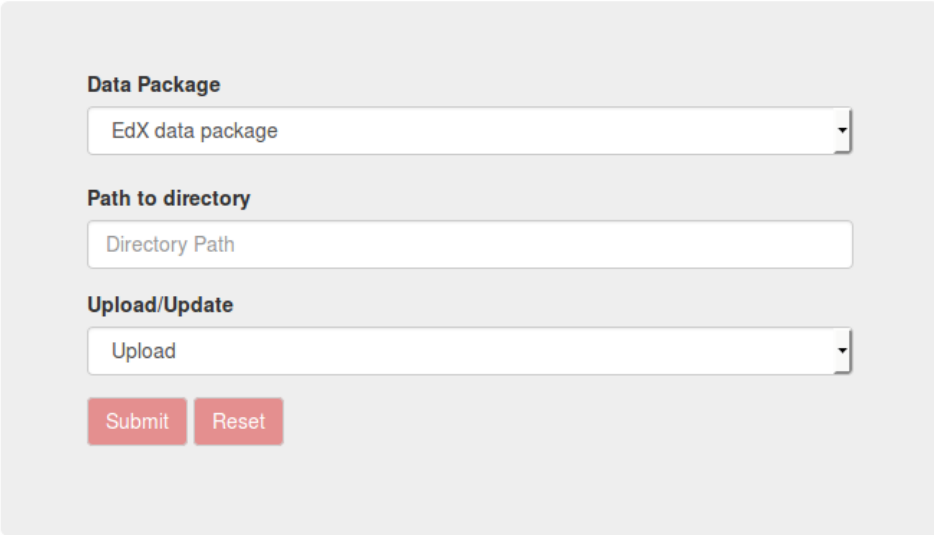
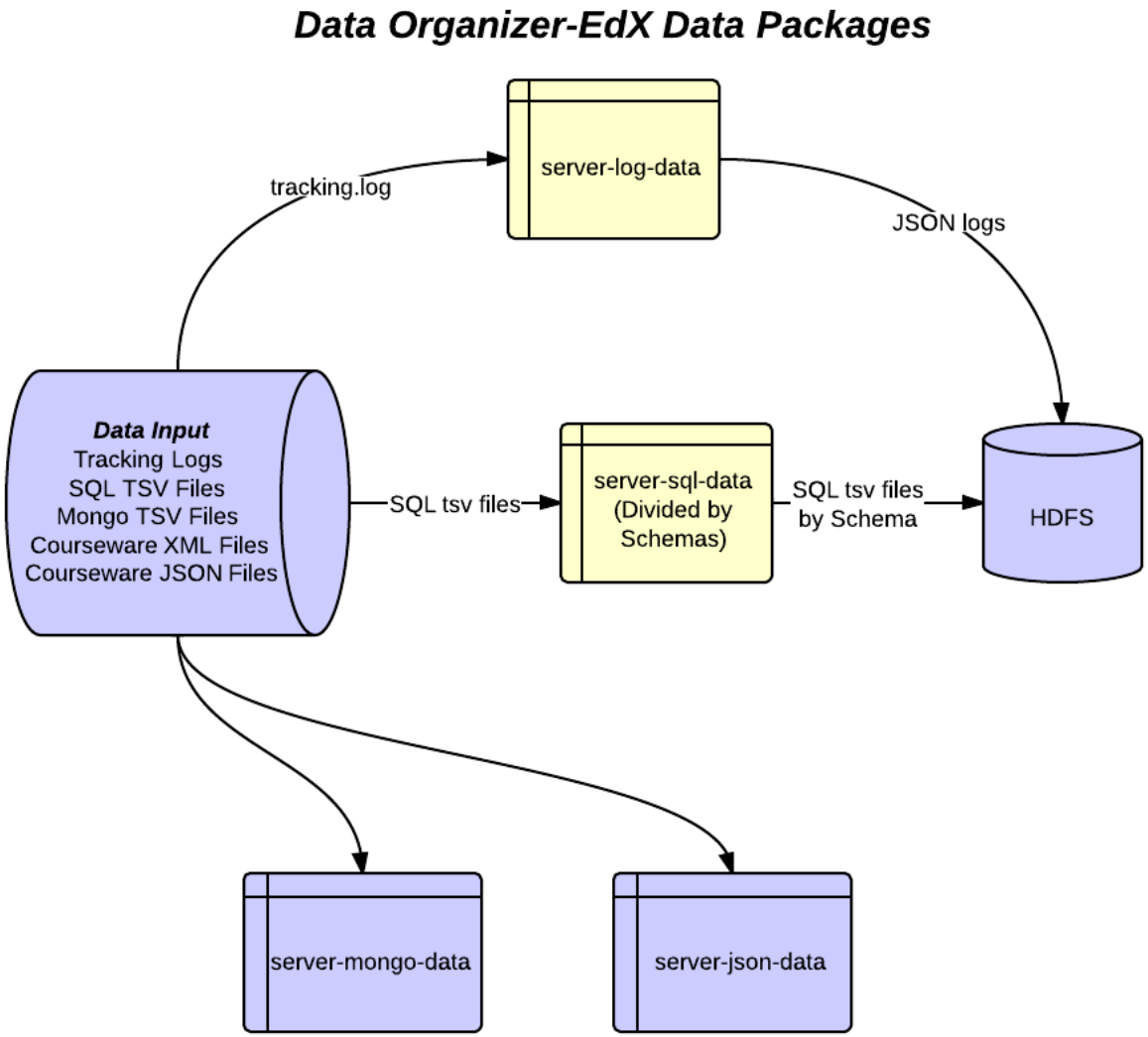
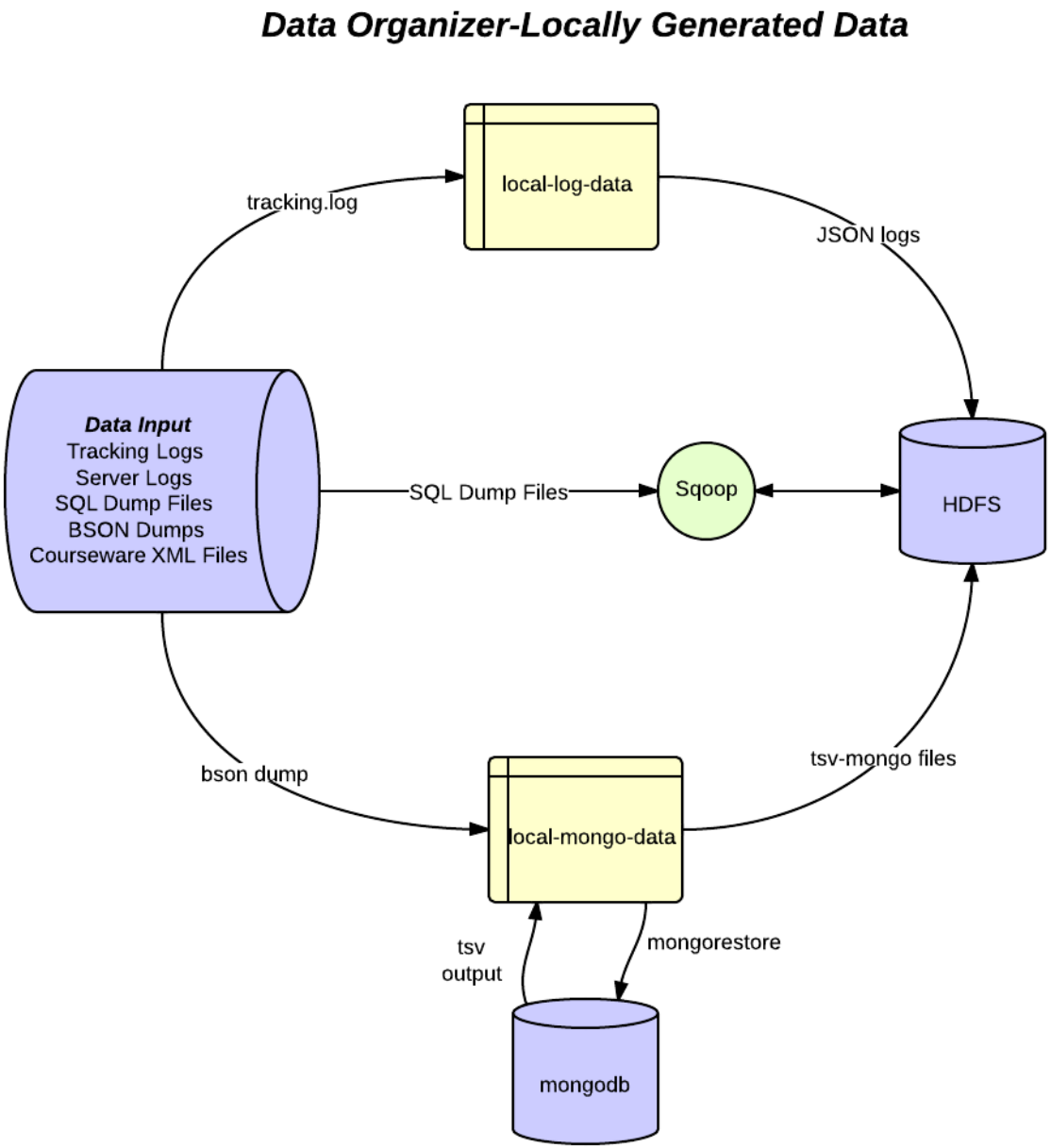


Figure: ETL GUI

ETL Step 2


Data Organizer-EdX Data Packages
server-log-data
tracking.log
JSON logs
Data Input
Tracking Logs
SQL TSV Files
Mongo TSV Files
Courseware XML Files
Courseware JSON Files
SQL tsv files
server-sql-data
(Divided by
Schemas)
SQL tsv files
by Schema
HDFS
server-mongo-data
server-json-data


Data Organizer-Locally Generated Data
local-log-data
tracking.log
JSON logs
Data Input
Tracking Logs
Server Logs
SQL Dump Files
BSON Dumps
Courseware XML Files
SQL Dump Files
Sqoop
HDFS
bson dump
local-mongo-data
tsv-mongo files
tsv
output
mongorestore
mongodb

ETL

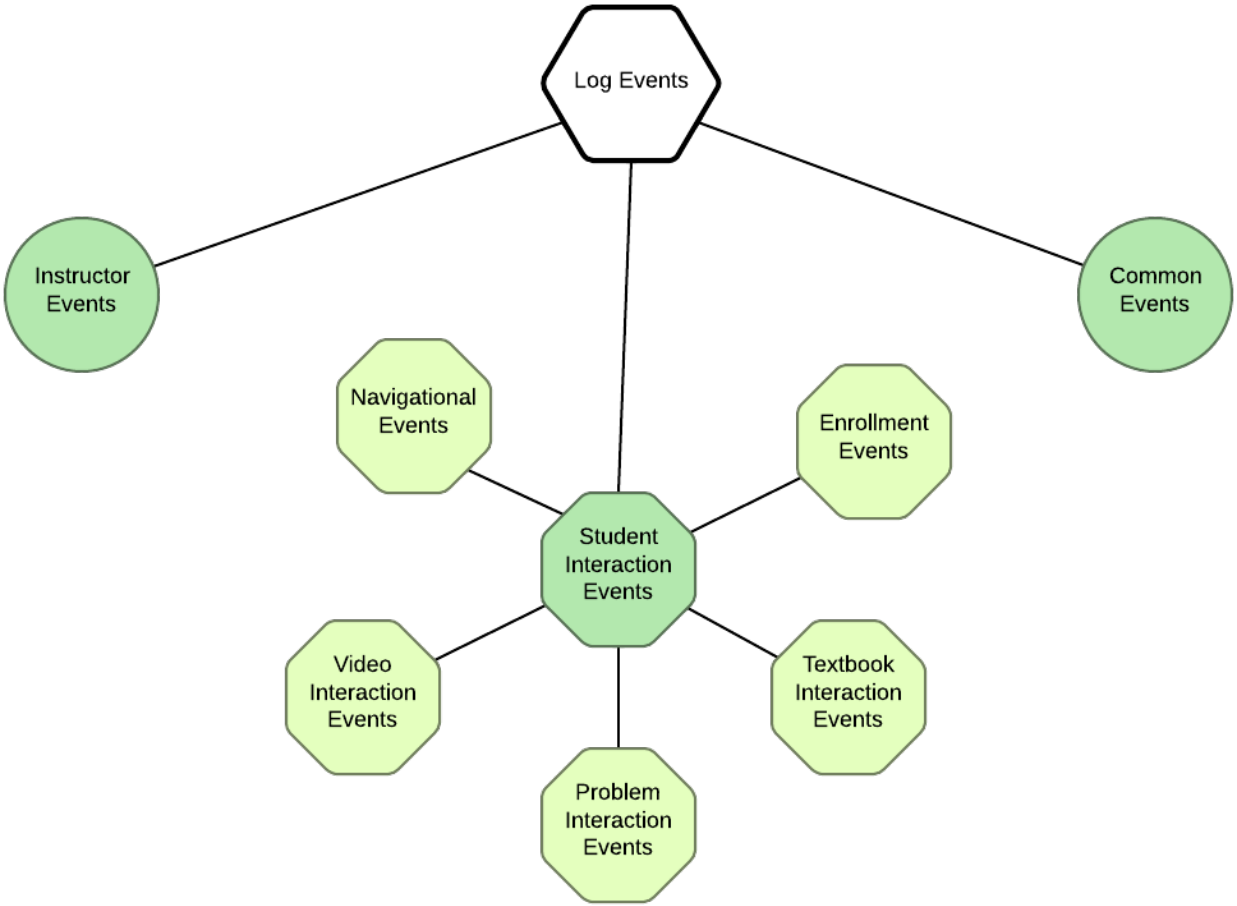
Step 3 : Loading SQL Data[8]

1. EdX Data Packages : SQL TSV files are directly loaded into Hive tables using `load data inpath` query
2. Locally Generated Data : SQL source files are loaded on MySQL and transported to Hive Tables using Sqoop

ETL

Problems in Structuring Log Data

Large Number of different types of logs generated[9]



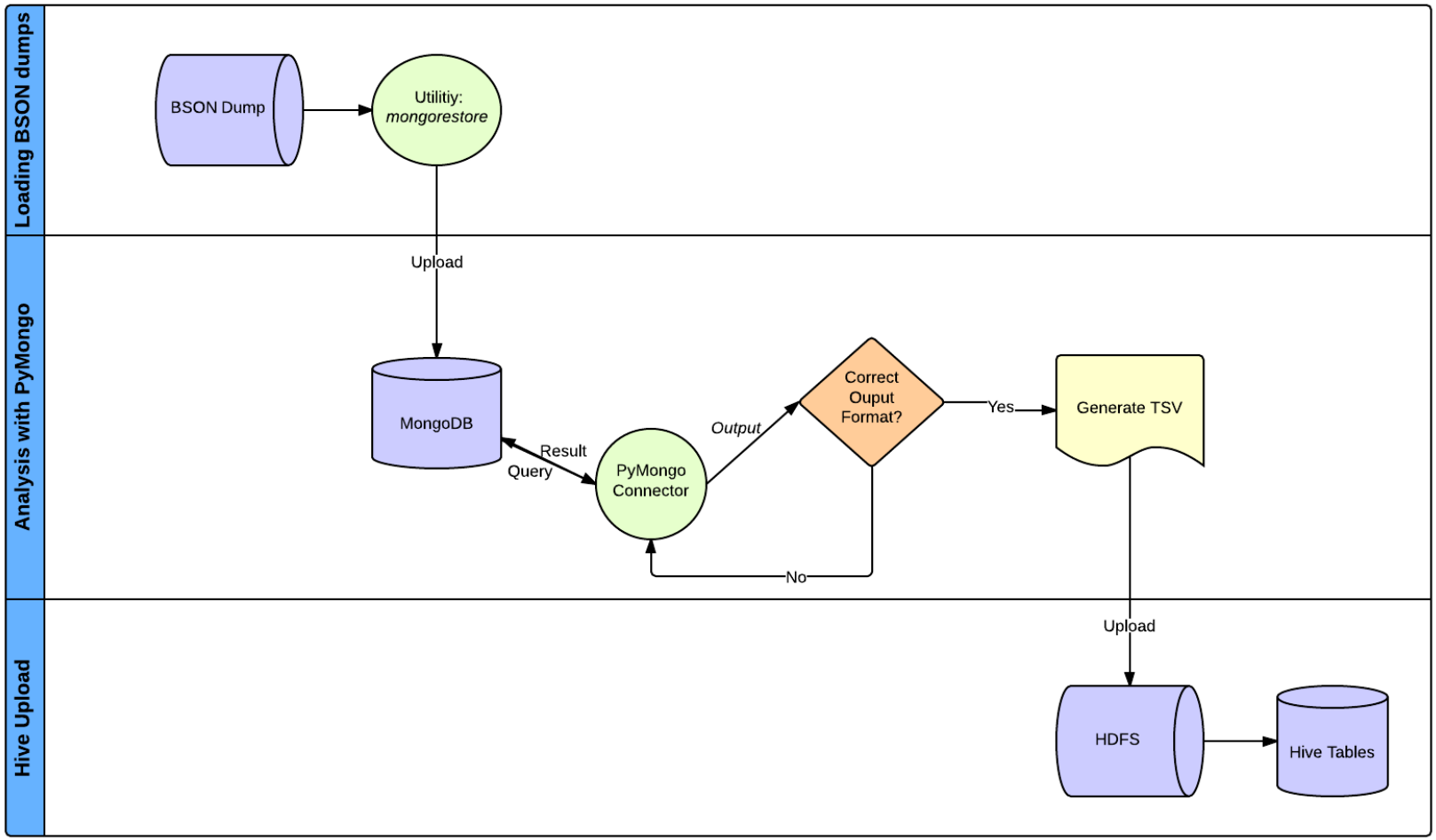
ETL

Step 3 : Structuring and Loading Logs

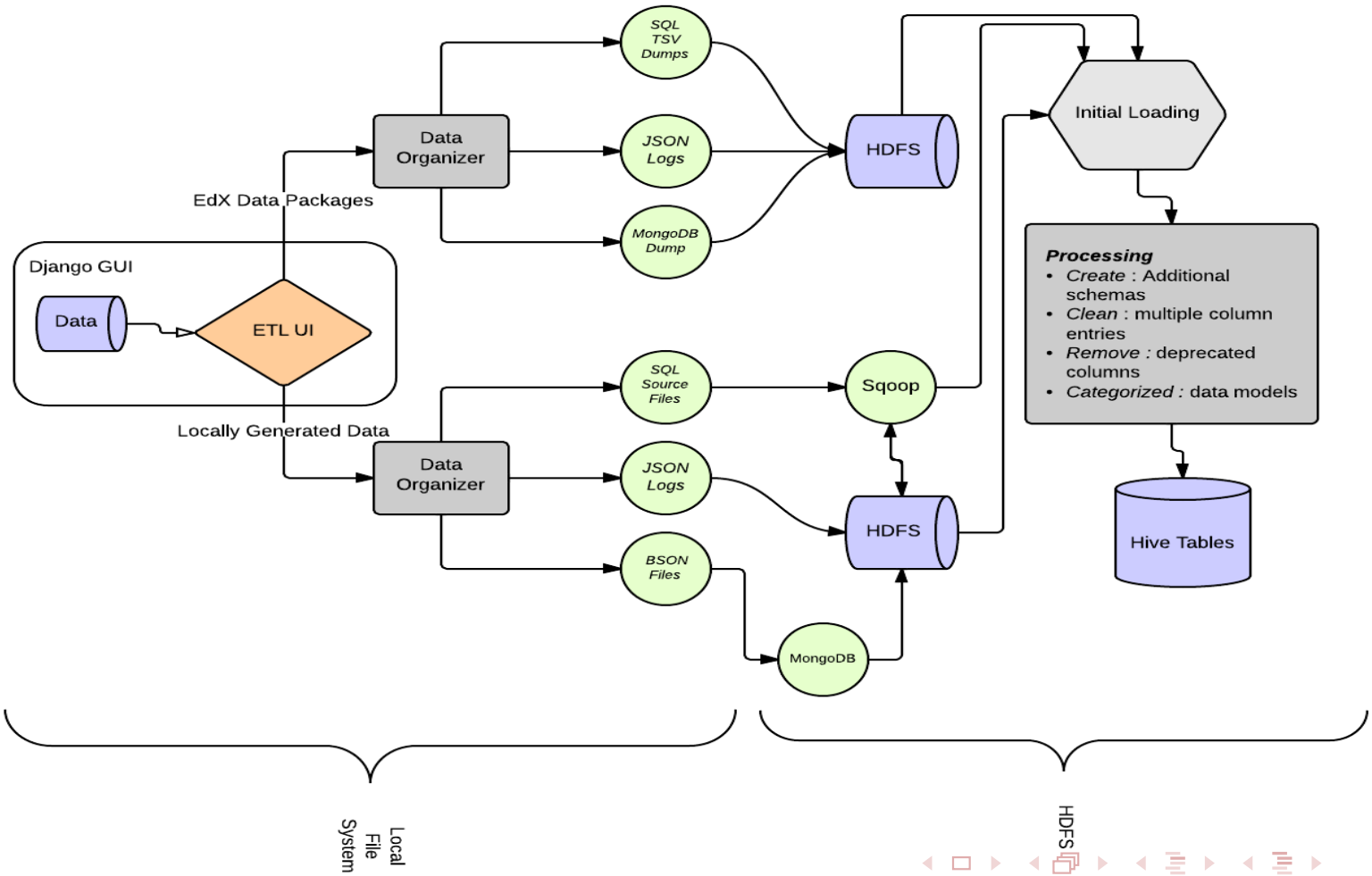
1. All the logs are first loaded onto the `log_table`
2. Segregation and subsetting is done on the basis of parameter `event_type`
3. Several columns are extracted via `json_tuple` and `get_json_object`
4. Extracted fields are cleaned by applying Hive String functions and conditionals
5. Finally these processed fields are loaded onto their respective `log_table`

ETL Step 3

MongoDB dumps to Hive



Overview of ETL



D3.js and Dimple.js

- Output of the hive queries are exported to tsv or csv files.
- These files are used as input for plotting the graphs.
- We used the D3.js and Dimple.js to plot the different kinds of charts.
- D3.js is a JavaScript library for manipulating documents based on data.
- Dimple.js is a library to aid in the creation of visualisations based on d3.js.

Queries implemented

From 1.4 GB of SummerIntern Test Data, 37 queries are formed and implemented.

Some of the important ones are mentioned below:-

Student Analytics

- Age Distribution of students along with the segregation depending on gender By Course
- Student Marks Statistics By Course
- Degree Distribution chart By Course

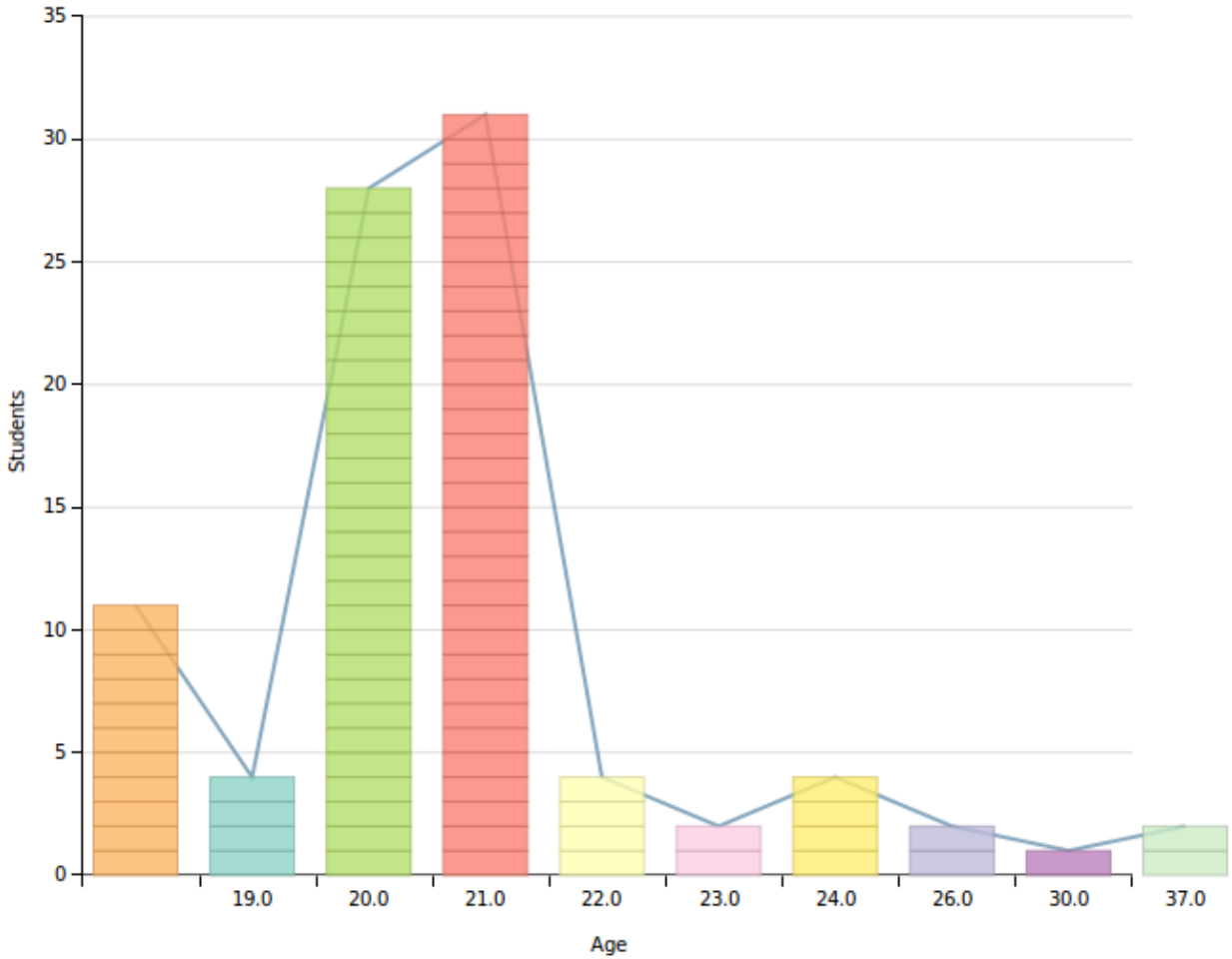


Figure: Age Distribution

Course Analytics

- No of active users per day by Course
- Sequence followed by maximum users to study a course

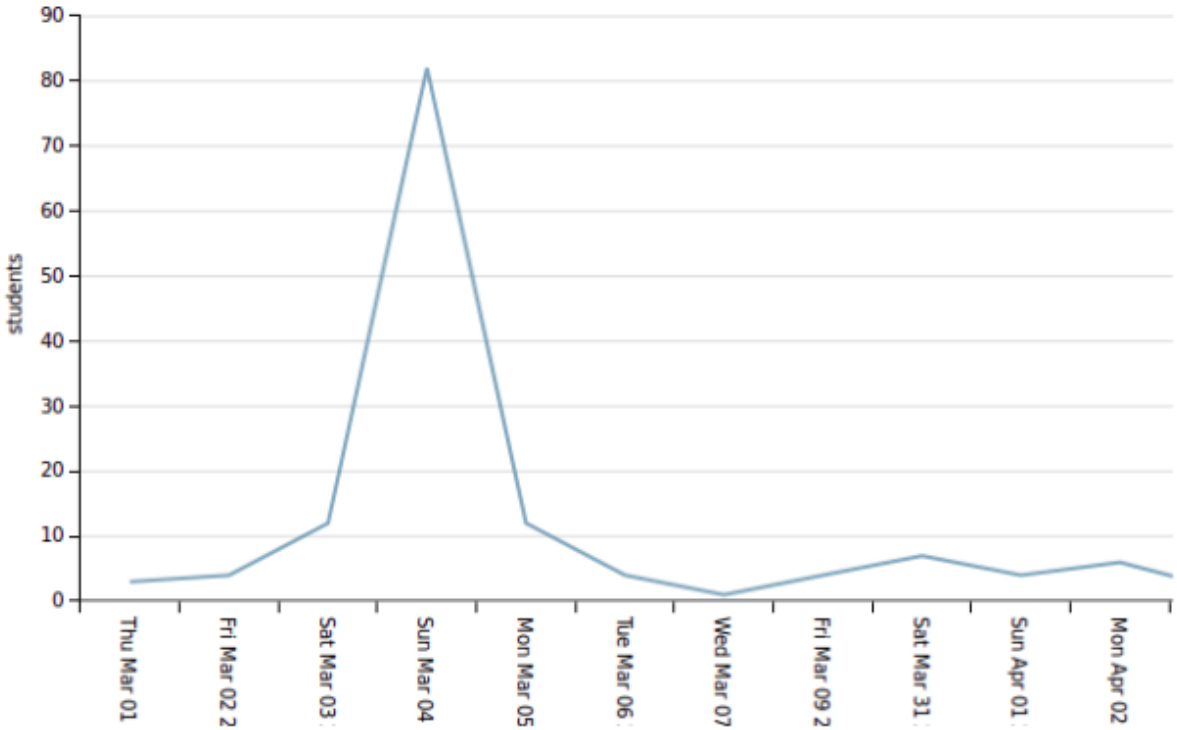


Figure: Active Users per day

Video Analytics

- Users who use transcript by Course
- Total videos watched by Course
- Statistics showing the number of students who have jumped the video
- Changes in video speed

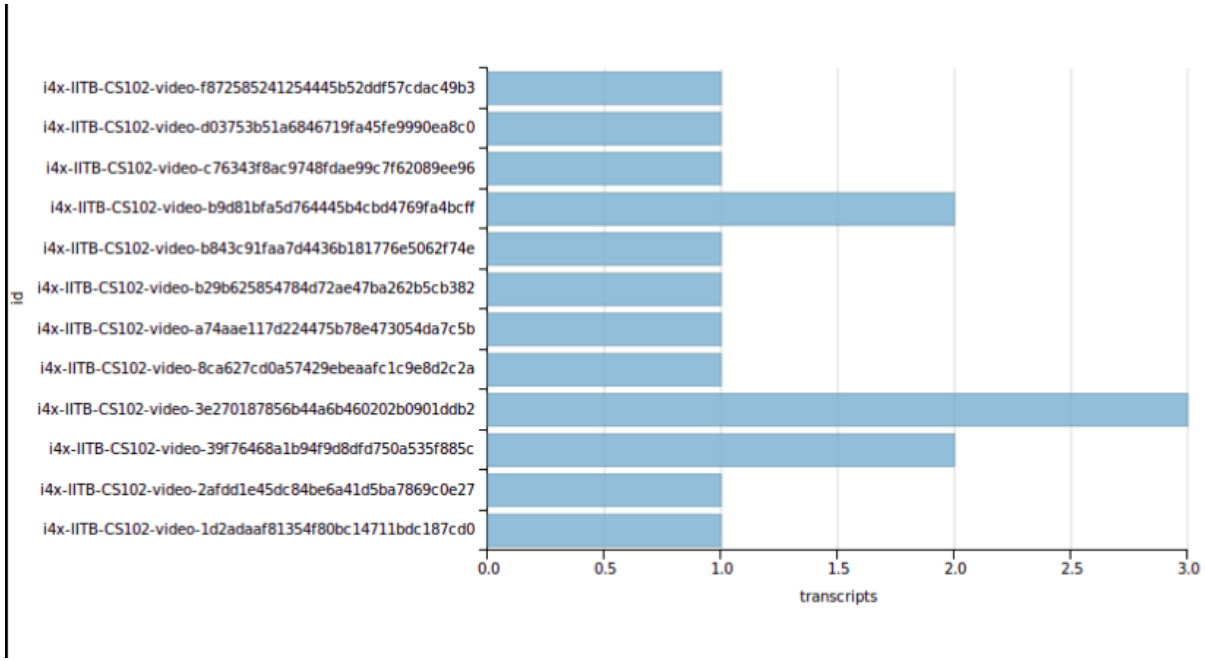


Figure: No of users who use transcript for video

Problem Analytics

- Chart showing the Response Time of students in solving questions of a quiz by course

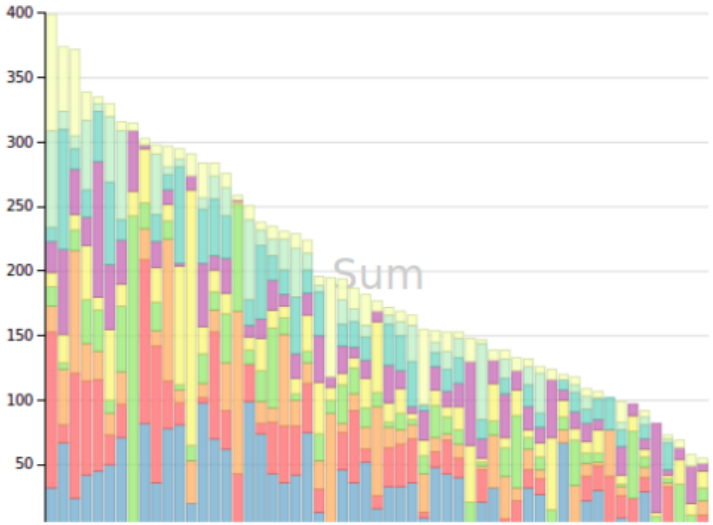


Figure: Response Time of students solving quiz questions by Course

Enrollment Analytics

- Students enrolled throughout the country based on their course shown in world map

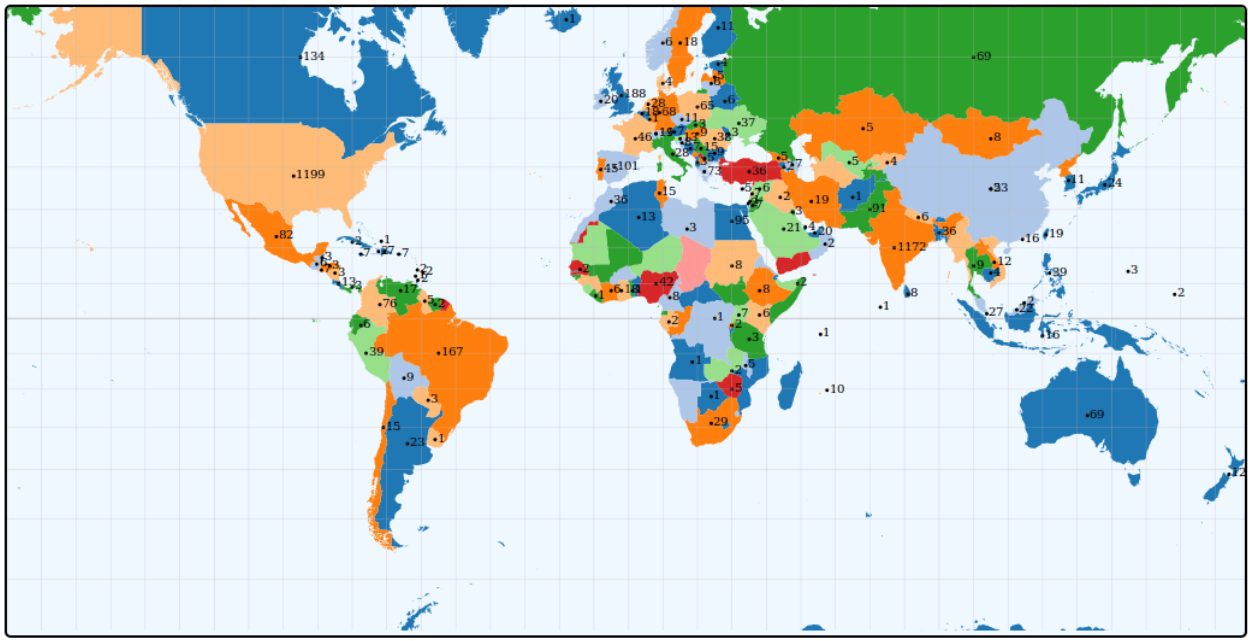


Figure: Students enrolled in a course throughout the world

Basic Statistics

- List of all the students enrolled by course

Some More Charts

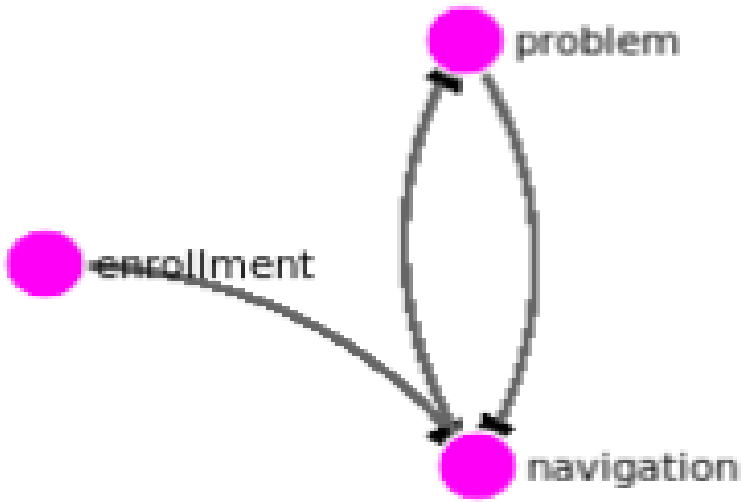


Figure: Event Sequence followed by majority of students

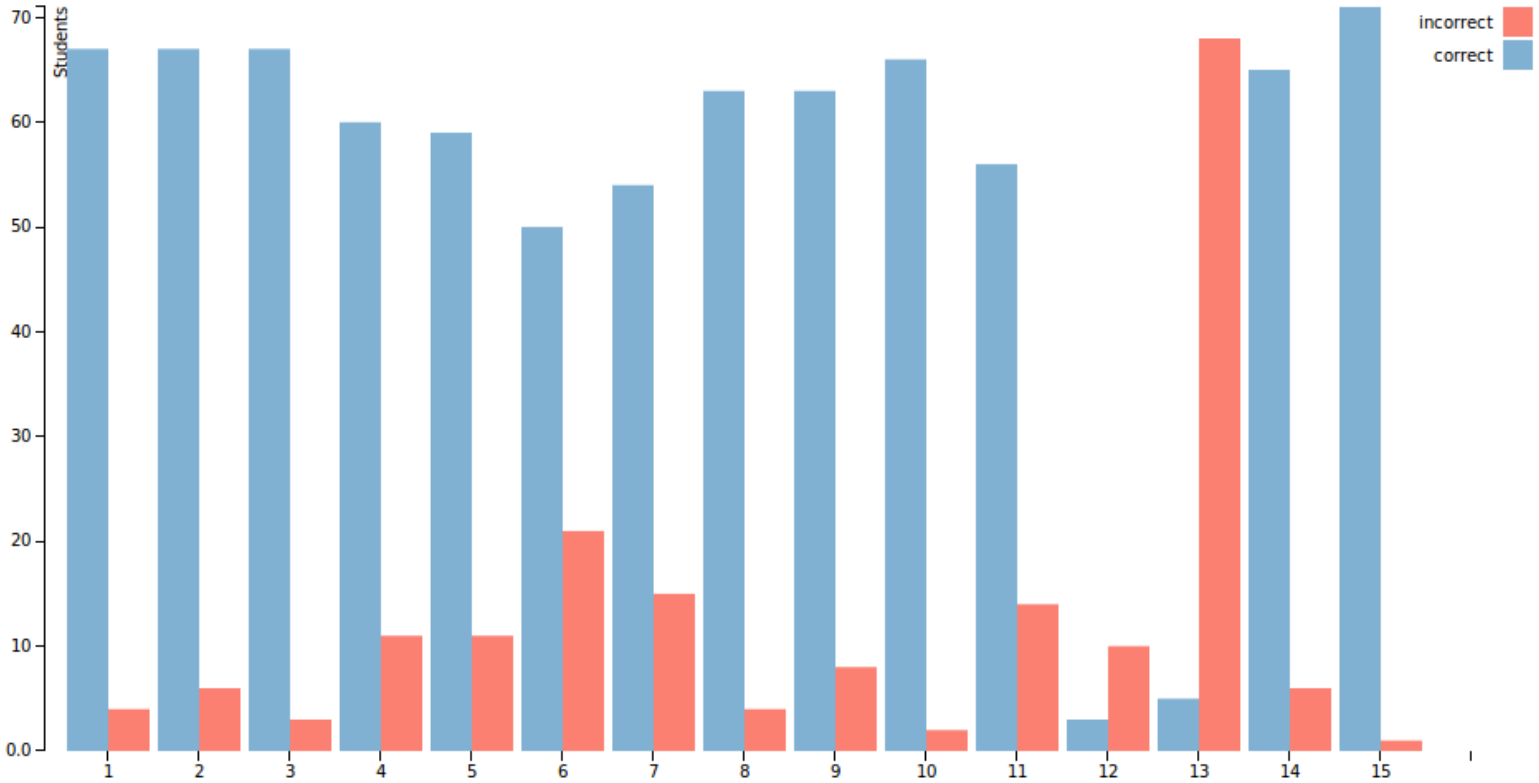


Figure: Correct/Incorrect responses for problems

Data Visualisation Using Google Charts

1. `Prerequisites:` Google JS API,Google Visualization library and library for the chart itself
2. Preparing data in form of datatables and dataviews
3. Customization by explicitly specifying the options and axis labels
4. Instantiating chart using google.visualization<charttype>
5. Drawing chart using `chart.draw()` and `drawChart()` functions
6. Data can be loaded in two ways into charts
   - Populating it manually
   - Loading it from local csv file
   - Loading it from google spreadsheets and querying it

Data Visualization Using Google charts

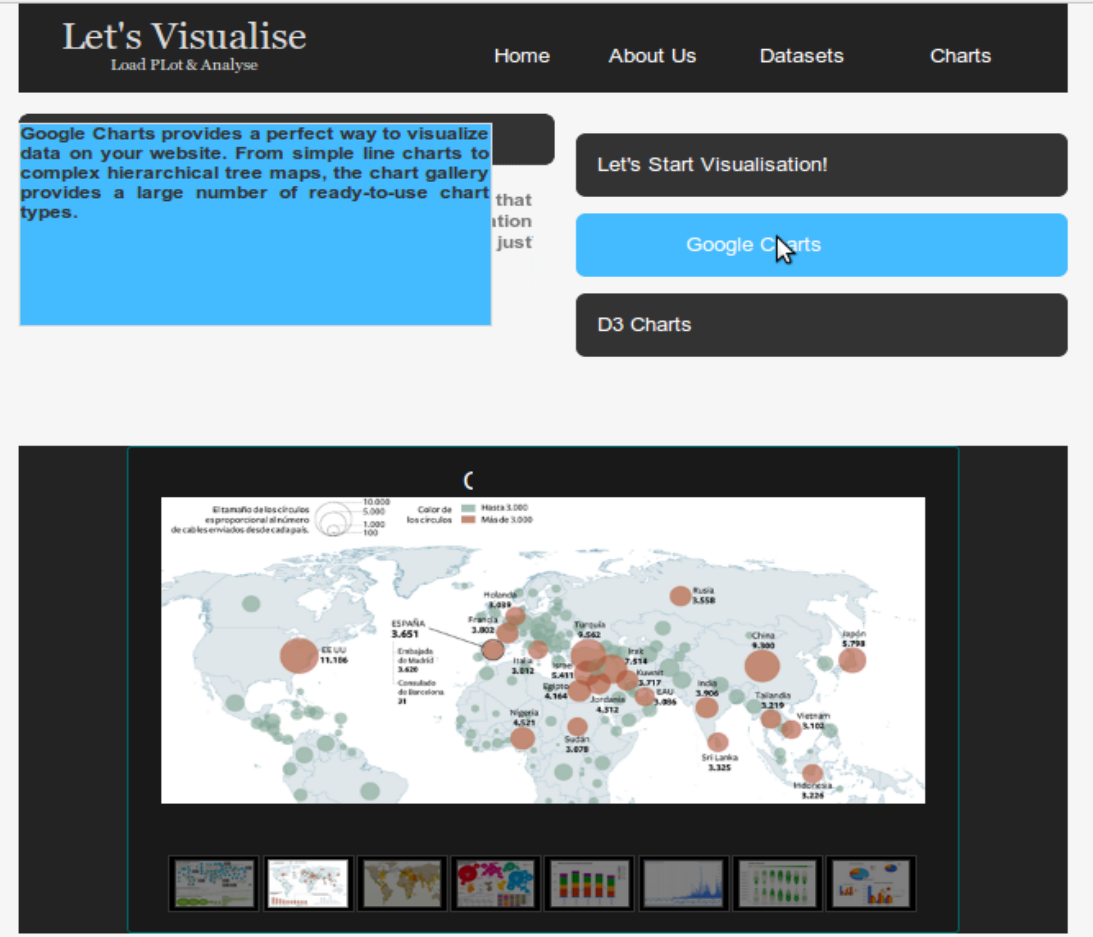


Figure: Data Visualization using Google Charts

HomePage for Google charts

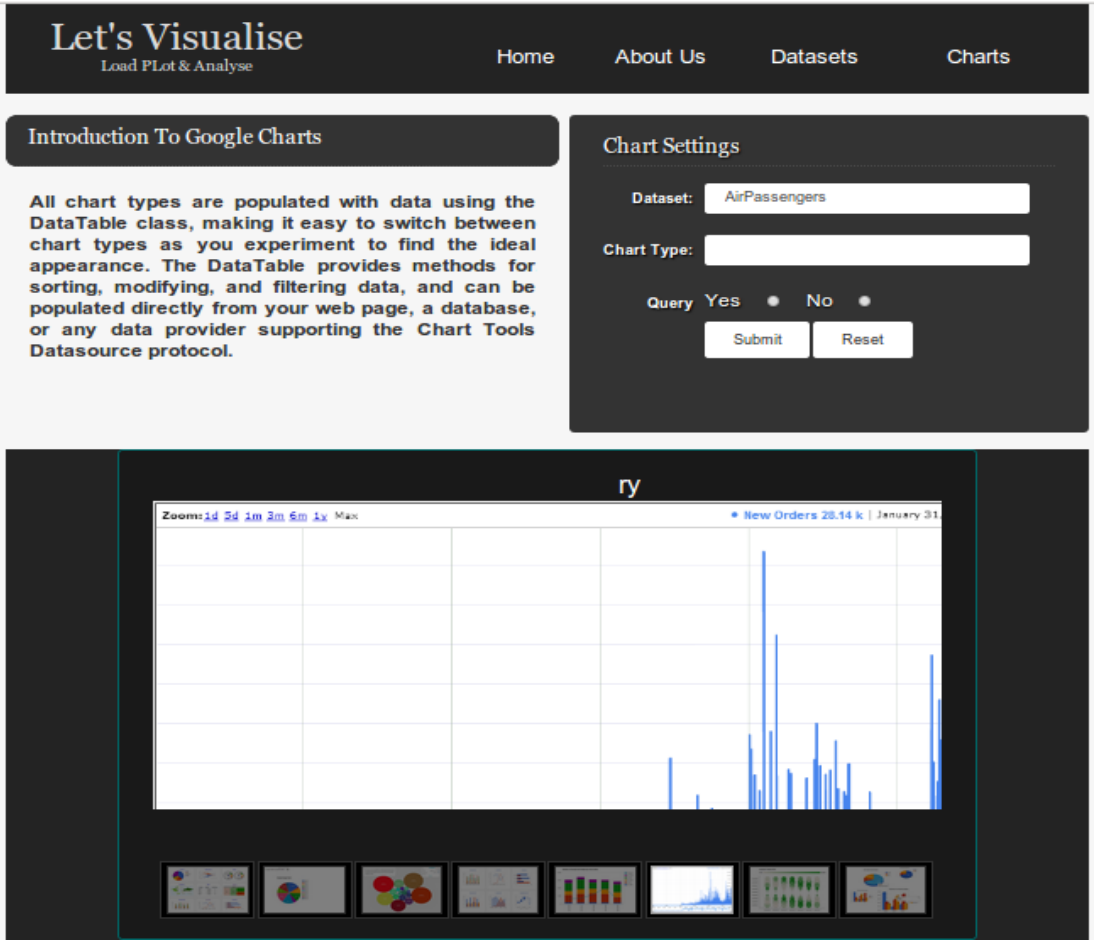


Figure: HomePage for Google Charts

Datasets used:

- AirPassengers(test dataset)
- ResponseTime dataset

Without querying:-

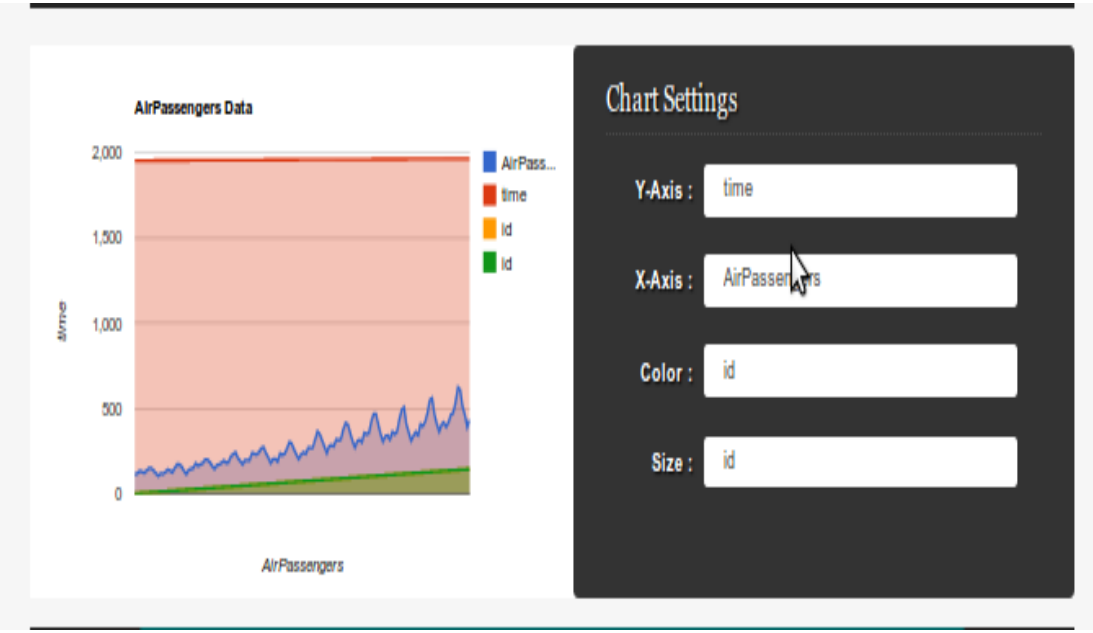


Figure: Areachart for AirPassengers dataset

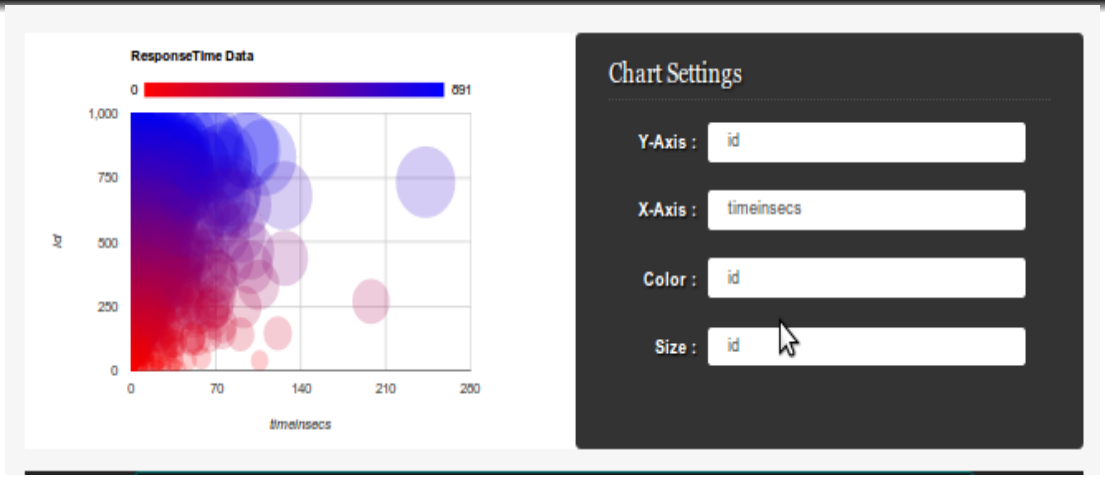


Figure: Bubblechart for Responsetime dataset

With Querying:-

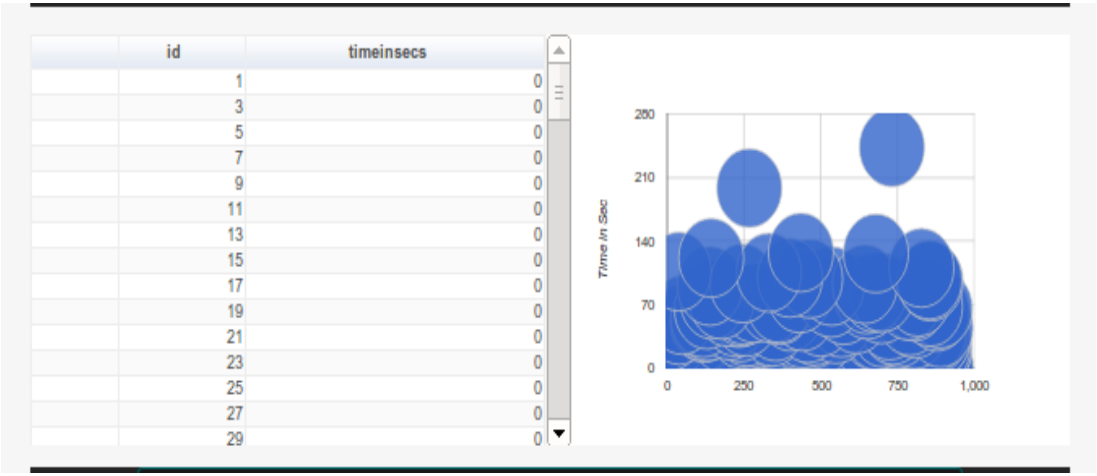
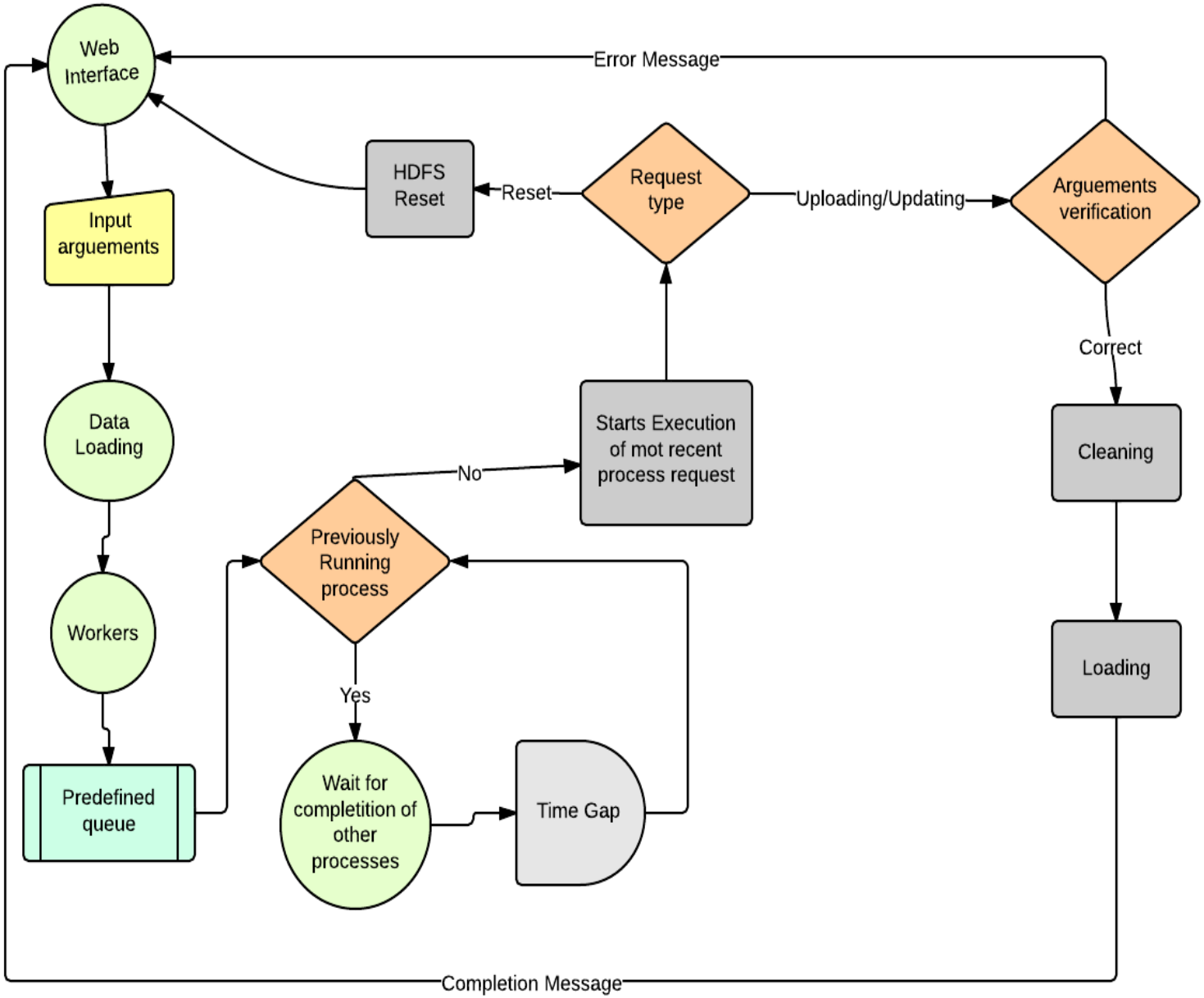
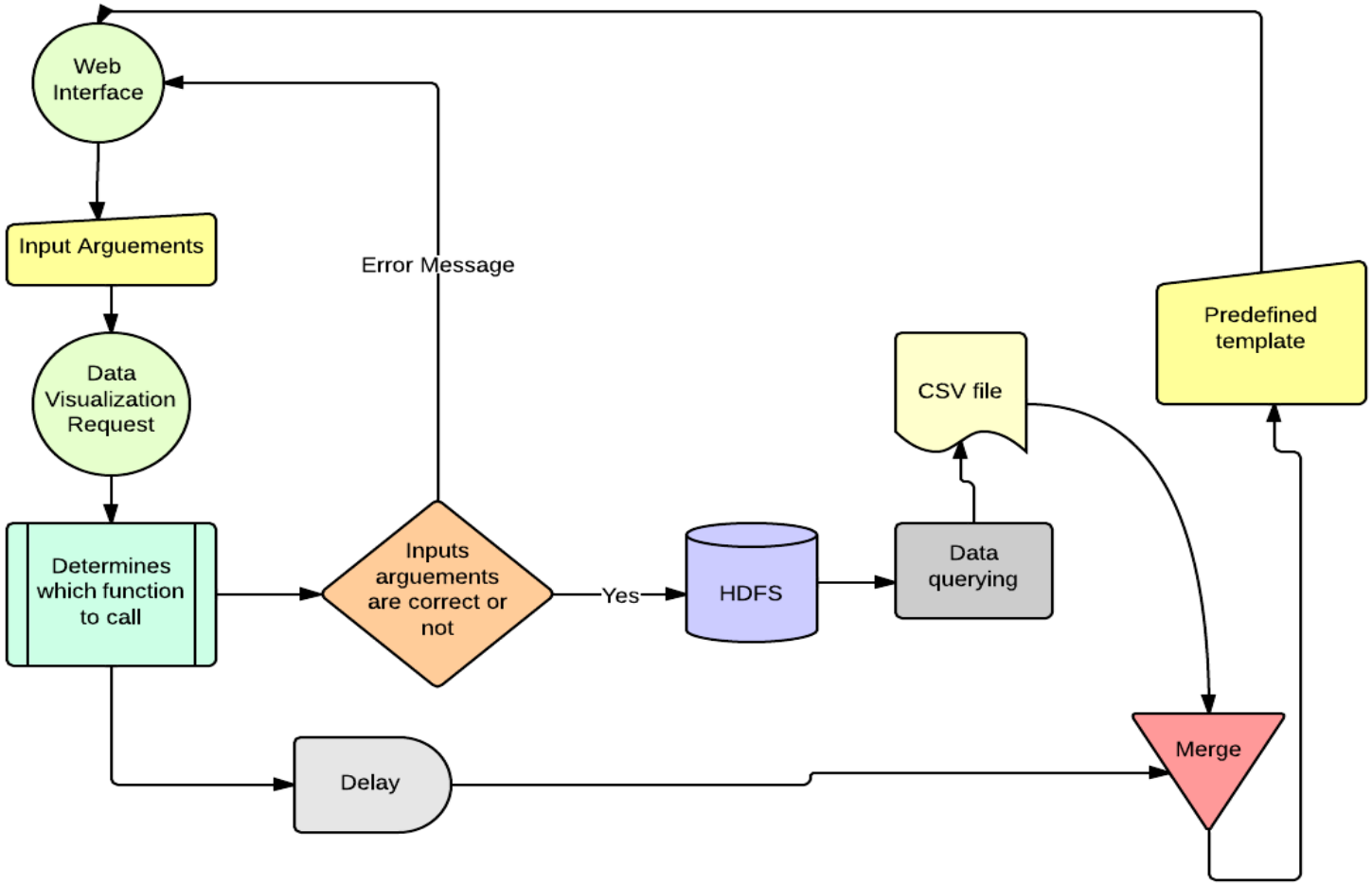


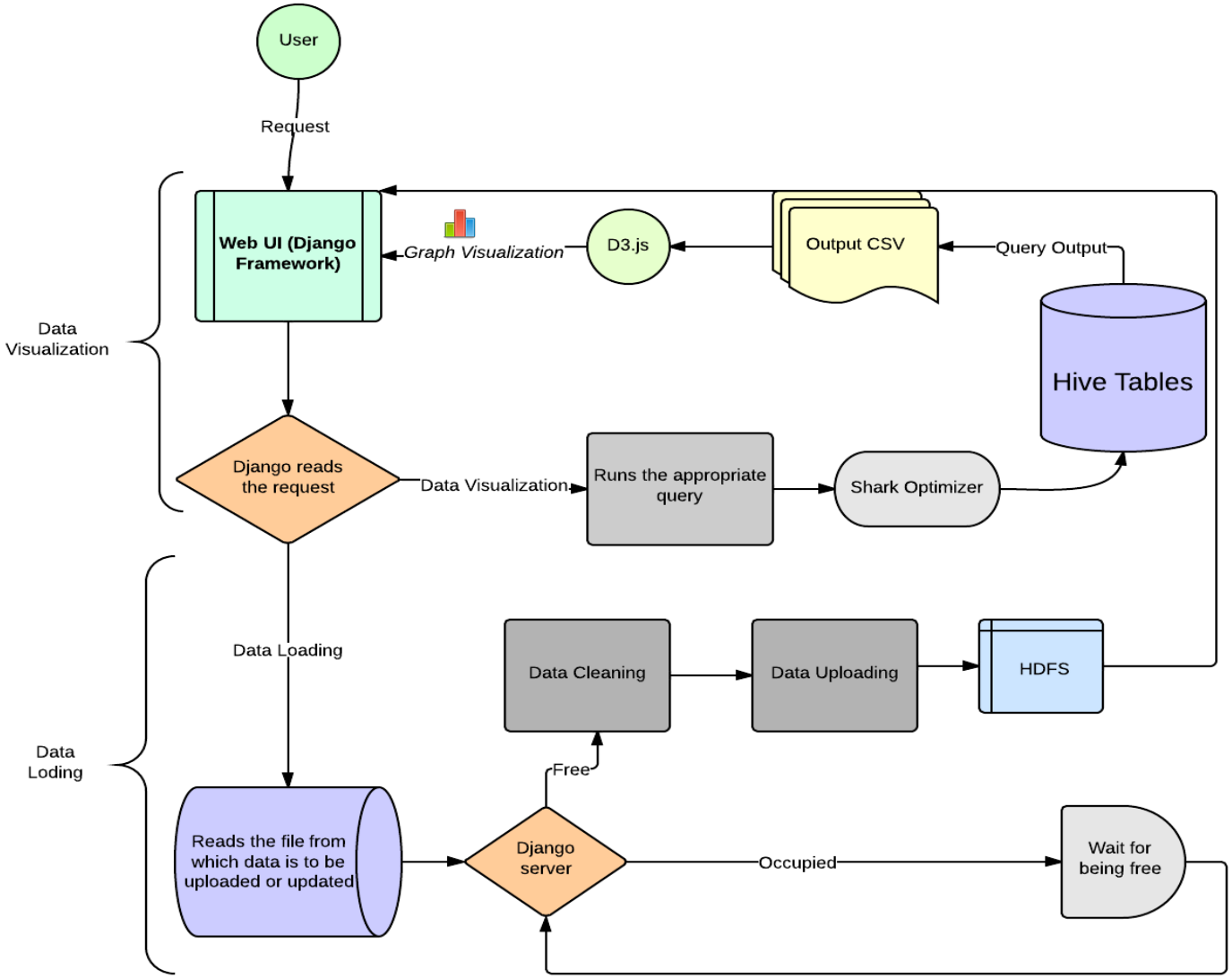
Figure: Alternate Bubblechart for Responsetime dataset

Developing a User Interface[11]

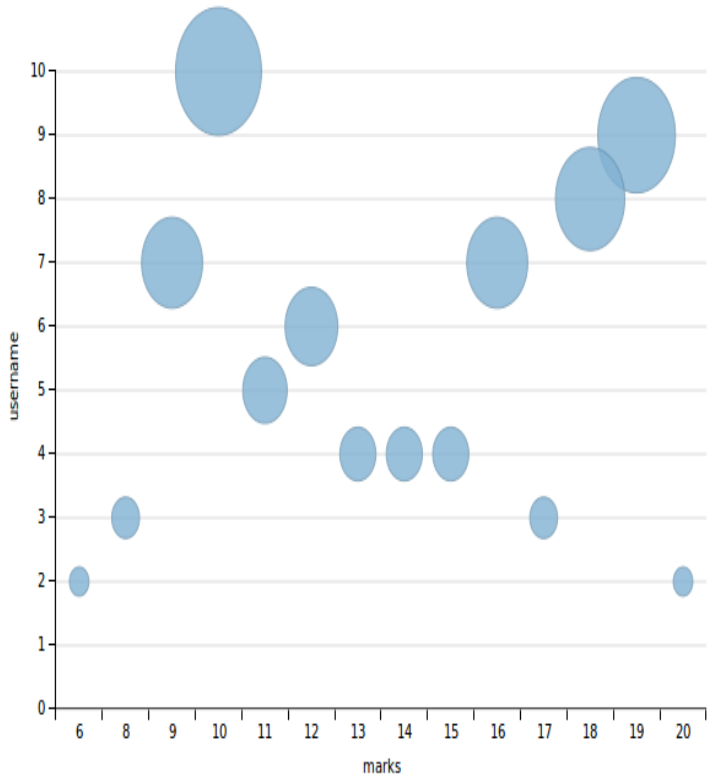
- *Django* has been used to integrate both data loading and data visualization in a web application
- A python based web framework.
- Can easily execute python queries, make a Django web template(interface) and has support of running Hive, R queries from it.
- Also capable of queuing the processes[10]


Web Interface
Error Message
HDFS Reset
Reset
Request type
Uploading/Updating
Arguements verification
Input arguements
Correct
Data Loading
Starts Execution of mot recent process request
Cleaning
No
Previously Running process
Workers
Loading
Yes
Predefined queue
Wait for completition of other processes
Time Gap
Completion Message


Web Interface
Input Arguements
Error Message
Data Visualization Request
Predefined template
CSV file
Determines which function to call
Inputs arguements are correct or not
Yes
HDFS
Data querying
Merge
Delay


User
Request
Web UI (Django Framework)
Graph Visualization
D3.js
Output CSV
Query Output
Hive Tables
Data Visualization
Django reads the request
Data Visualization
Runs the appropriate query
Shark Optimizer
Data Loading
Data Cleaning
Data Uploading
HDFS
Data Loding
Free
Reads the file from which data is to be uploaded or updated
Django server
Occupied
Wait for being free

A part of the Interface showing the form where user has to enter details along with its graph



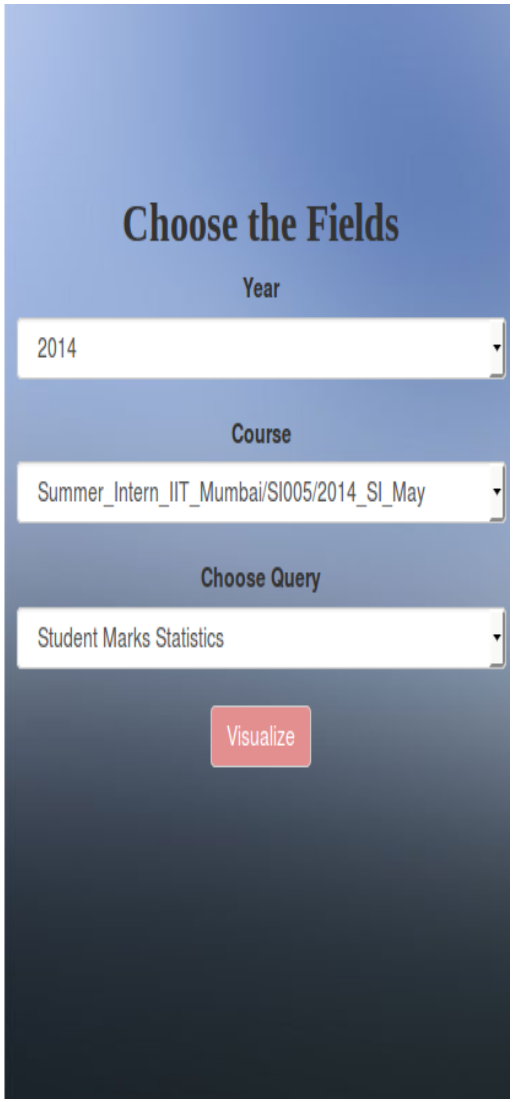


Figure: Input Form

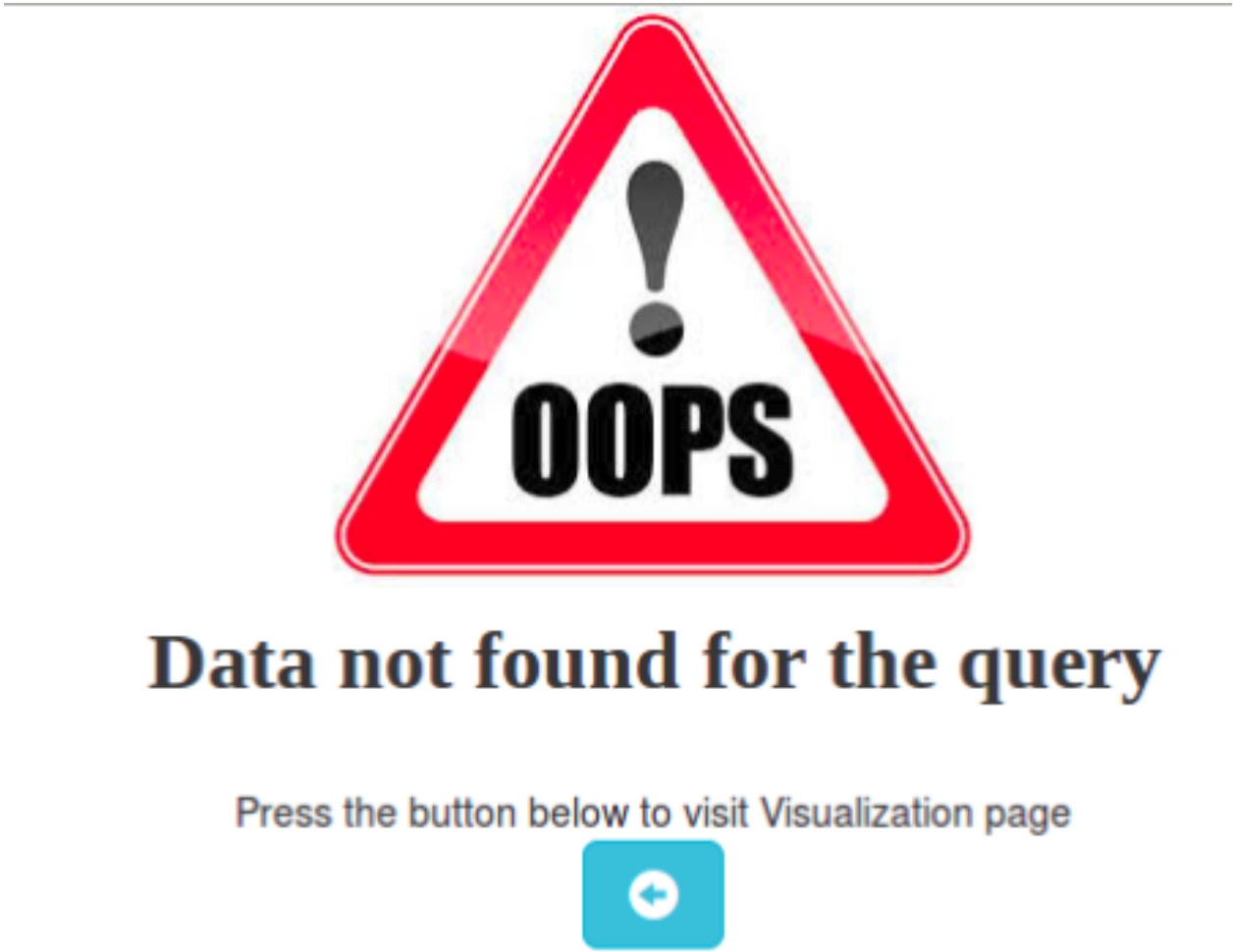


Figure: Error Message Page when no data is found for query

Optimization[12][13]

Spark and Shark clearly improve the performance of Hadoop by leaps and bounds. Some basic comparisons of simple queries are as follows:

- `select username,session,min(time) as tim from log_table where session is not null and username!="" and username is not null group by username,session order by tim`
  Hive : 48.923s
  Shark : 9.047s
- `select distinct year(time) as year from log_table where year(time) is not null`
  Hive:21.929s
  Shark:3.812s

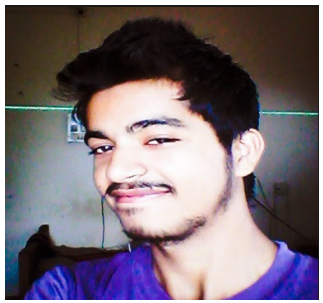
Results

- Technologies Used
- Architecture
- ETL Automater
- Data Visualization
- All-in-One UI

Future Work

1. Sequential Data Mining
2. Detecting Undesirable Student Behaviors
3. Latent Knowledge Estimation
4. Detecting Possibility of Student's Drop Out
5. Using the mongodb data of the EDX
6. Integrating the django frontend with multinode cluster

Our Team



Ankush Arora

Palak Agrawal

Prashant Gupta

Purva Bansal

Saatvik Shah

References I

[1] "edx research guide for researchers and data czars." https://devdata.readthedocs.org/en/latest/index.html, 2014.
[Online accessed 01-July-2014].

[2] "Test data." http://www.it.iitb.ac.in/frg/wiki/index.php/G7_-_Tools_for_Big_Data#edX_Logs, 2014.
[Online accessed 01-July-2014].

[3] "Average edx enrollments." http://techcrunch.com/2014/03/03/study-massive-online-courses-enroll-an-average-of-43000 2014.
[Online accessed 01-July-2014].

References II

[4] "Derby server documentation." https://cwiki.apache.org/confluence/display/Hive/HiveDerbyServerMode, 2014. [Online accessed 01-July-2014].

[5] "Hadoop installation for single node." http://hadoop.apache.org/docs/r1.2.1/single_node_setup.html, 2014. [Online accessed 01-July-2014].

[6] "Hadoop installation for multinode." http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/, 2014. [Online accessed 01-July-2014].

References III

[7] "Hive getting started." https://cwiki.apache.org/confluence/display/Hive/GettingStarted, 2014. [Online accessed 01-July-2014].

[8] "Sqoop documentation." http://sqoop.apache.org/docs/1.4.4SqoopUserGuide.html, 2014. [Online accessed 01-July-2014].

[9] "edx data package details." https://edx-wiki.atlassian.net/wiki/display/OA/Research+Data+Package+Details, 2014. [Online accessed 01-July-2014].

[10] "Django rq." http://python-rq.org/docs/, 2014. [Online accessed 01-July-2014].

References IV

[11] "Django book." http://www.djangobook.com/en/2.0/index.html, 2014. [Online accessed 01-July-2014].

[12] "Shark documentation." http://shark.cs.berkeley.edu/, 2014. [Online accessed 01-July-2014].

[13] "Spark documentation." http://spark.apache.org/docs/latest/configuration.html, 2014. [Online accessed 01-July-2014].