Voting Demographic Analyzer

## By: Navtejinder Singh & Savio Moonnumackel

Table of Contents

[1 Application Overview 2](#_Toc53221502)

[1.1 Voting demographic analyzer 2](#_Toc53221503)

[1.2 User interaction 2](#_Toc53221504)

[2 Components used 2](#_Toc53221505)

[2.1 Google Products used 2](#_Toc53221506)

[2.1.1 Kubernetes Enging 2](#_Toc53221507)

[2.1.2 Cloud SQL 2](#_Toc53221508)

[2.1.3 BigQuery 2](#_Toc53221509)

[2.2 Languages Used 2](#_Toc53221510)

[3 Architecture 2](#_Toc53221511)

[4 Design 2](#_Toc53221512)

[5 Implementation Plan 2](#_Toc53221513)

[6 Test Plan 2](#_Toc53221514)

# Application Overview

## Voting demographic analyzer

“Explain what the project does”

## User interaction

“Explain how the user is going to interact with the UI. Try to include a use case diagram as well”

# Components used

## Google Products used

### Kubernetes Enging

“Explain general use of kubernetes “

### Cloud SQL

“Explain general use of Cloud SQL “

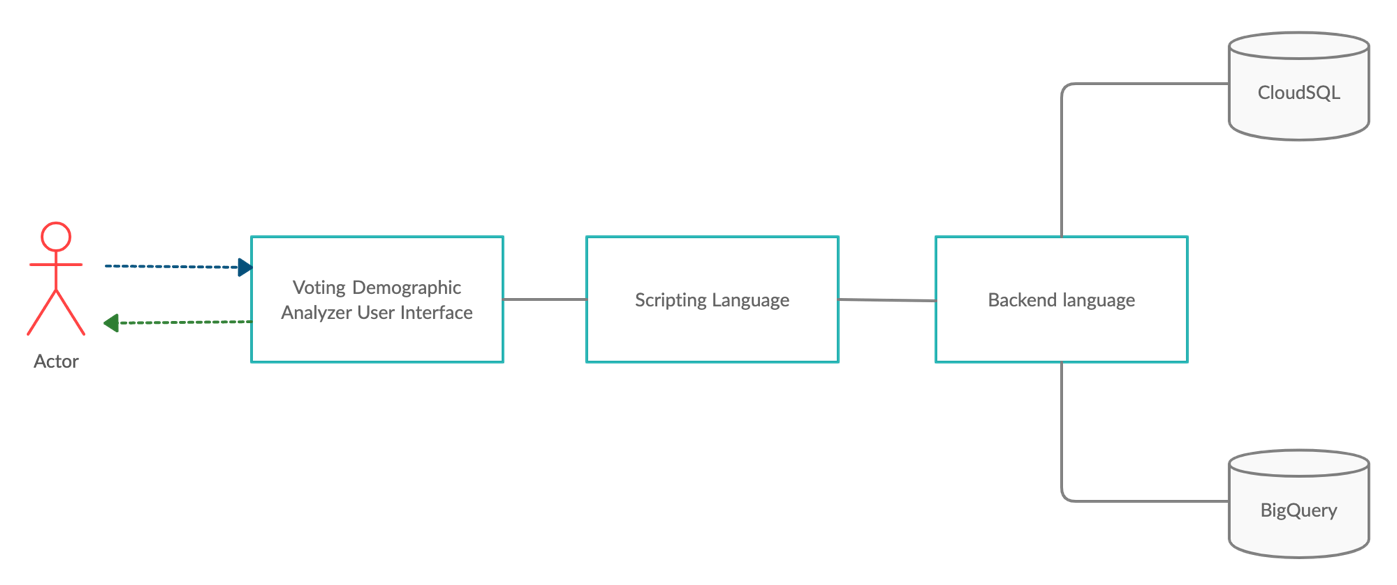
### BigQuery

“Explain general use of BigQuery “

# Architecture

Voting demographic analyzer accepts data from a front end user through HTML forms. This data is then parsed using the scripting language. This parsed data is then stored in the data base and we use a backend language to hook the application logic with the database persistence layer. Periodically, based on a configurable time interval or delta change to the demographic data, BigQuery will pull transactional data from the DataBase and provide analytical data. This data then travels in the reverse direction through the backend logic to the front end and is provided to the UI user.

This flow can be better visualized as below,

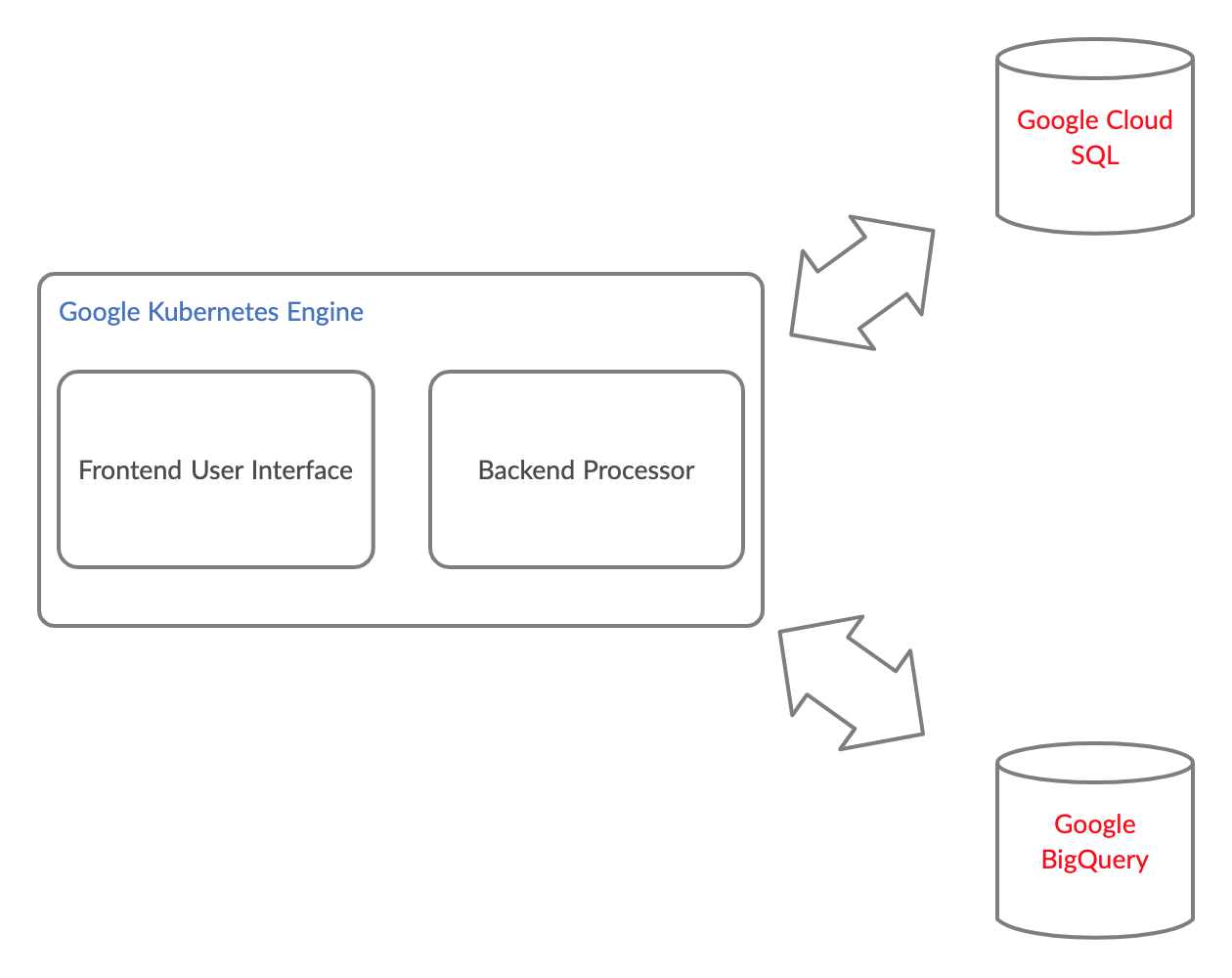


The architecture of Voting Demographic Analyzer can be visualized as 3 main blocks for simplicity,

1. Front End User Interface
2. Backend Processing
3. Data Persistence layer

The front end is responsible for accepting input data from the user and allowing a seamless interaction with the user. The backend processor will process the data entered by the user and will know what to do with it. It will store the data in a database and based on configurable values, it will interact with BigQuery to allow for data analytics. This configurable parameter was chosen as BiqQuery can tend to be expensive and charges per interaction. Now based on financial requirements we can configure the frequency of these transactions.

The Frontend User Interface and the Backend processor will be packaged as a single container which will be hosted via Google Kubernetes Engine. This container will then interface with Google Cloud SQL and Google BigQuery.



# Design

## Languages

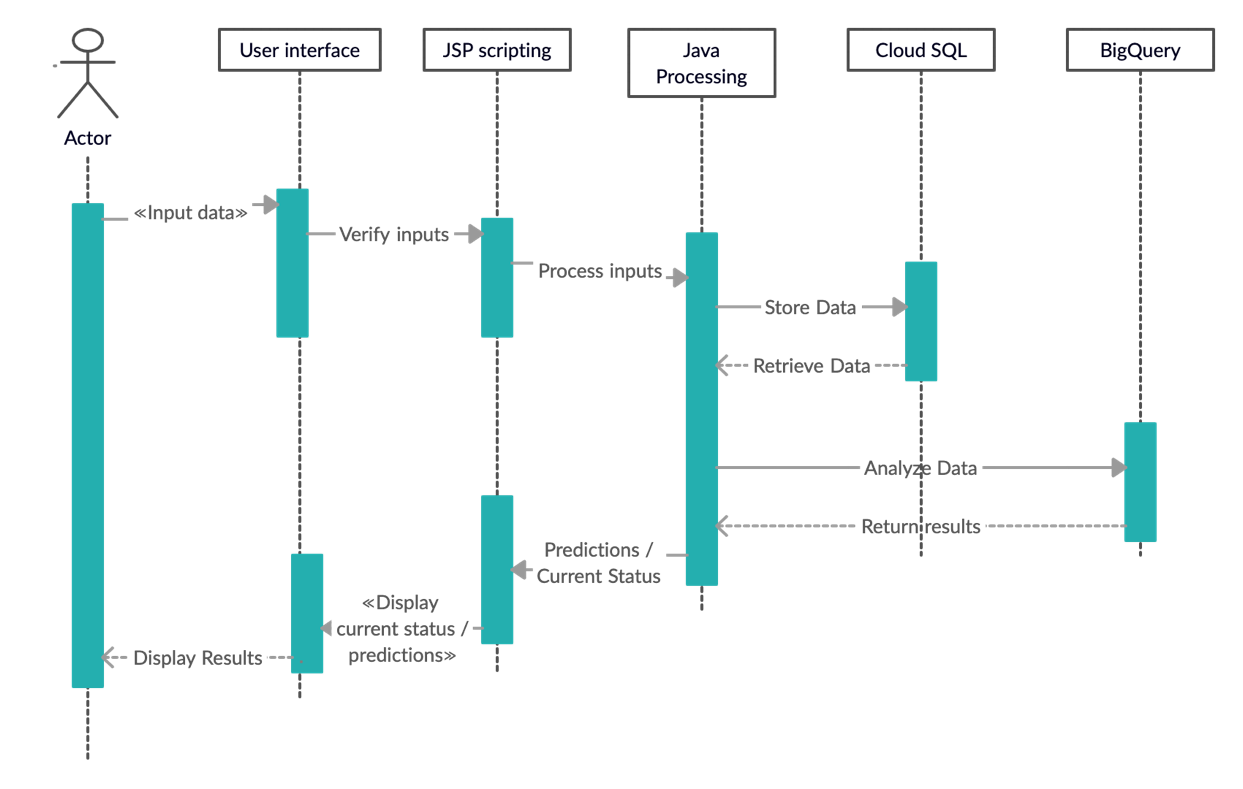
The proposed languages decided for each portion can be seen in the table below.

|  |  |  |
| --- | --- | --- |
| SI No | Functionality/Component | Language/Tool |
| 1. | Frontend | HTML, CSS, Bootstrap |
| 2. | Backend scripting | JSP |
| 3. | Backend processing | Java |
| 4. | Persistence Layer | SQL |

## Workflows

The major workflows in the application can be described as follows:

1. User inputs data to user interface, as a HTML form
2. JSP pulls this data and verifies that the data is actually in the correct format and no erroneous input was received
3. Java processes this data and hooks it with the cloud SQL
4. The java processor can store data and retrieve data from the cloud SQL
5. The Java processor requests BigQuery to analyze data, BigQuery returns the results
6. Java processor returns data to JSP
7. JSP returns data to a HTML page



# Implementation Plan

We intend to implement the project in a collaborative manner however due to the current circumstances of Covid-19 we will try to do as much as possible virtually and hold personal meetings only at milestones.

To harbor a collaborative approach, we will be storing all our data in github at the below address:

<https://github.com/saviomoon/CloudVotingAnalyticsProject>

The above project is currently private and access can be granted when requested.

|  |  |  |
| --- | --- | --- |
| Component | Owner | Date |
| Frontend UI | Navtijender Singh | Oct 28 |
| JSP scripting | Savio Moonnumackel | Oct 28 |
| Java Processing | Savio Moonnumackel | Oct 28 |
| Cloud SQL | Savio Moonnumackel | Oct 28 |
| BigQuery | Navtijender Singh and Savio Moonnumackel | Nov 12 |
| Additional Features | Navtijender Singh and Savio Moonnumackel | Nov 26 |
| Testing | Navtijender Singh and Savio Moonnumackel | Nov 30 |

# Test Plan

|  |  |
| --- | --- |
| Requirement | Test Procedure |
|  |  |
|  |  |
|  |  |

# Teammate Contribution

|  |  |  |
| --- | --- | --- |
| Section | Author | Approval |
| Application Overview | Navtijender Singh | Savio Moonnumackel approves. |
| Components Used | Navtijender Singh | Savio Moonnumackel approves. |
| Architecture | Savio Moonnumackel | Savio Moonnumackel approves. |
| Design | Savio Moonnumackel | Savio Moonnumackel approves. |
| Implementation Plan | Savio Moonnumackel | Savio Moonnumackel approves. |
| Test Plan | Navtijender Singh | Savio Moonnumackel approves. |