**PUCIT**

Punjab University College of Information Technology

# Project Final Documentation

Version (1.0)

# Political Social Network

*Team ID – BSEF12-514*

*Session: BS (SE) Fall 2012*

**Project Advisor:  Kamran Malik**

*Submitted By*

| | |
|---|---|
| Waleed Ashraf | BSEF12M515 |
| Zeeshan Awan | BSEF12M510 |
| Agha Ali Abbas | BSEF12M538 |
| Hassan Tariq | BSEF12M548 |

Punjab University College of Information Technology

University of the Punjab, Lahore.

# STATEMENT OF SUBMISSION

This is to certify that **Waleed Ashraf** Roll No. **BSEF12M515, Zeeshan Awan** Roll No. **BSEF12M510, Agha Ali Abbas** Roll No. **BSEF12M538 and Hassan Tariq** Roll No. **BSEF12M548** have successfully completed the final project titled as: **Political Social Network**, at the Punjab University College of Information Technology, University of The Punjab Lahore, to fulfill the partial requirement of the degree of Bachelor of Science in Information Technology.

______________________

Project Office Supervisor

PUCIT, Lahore

M.Bilal

____________________

Project Primary Advisor

Name: Kamran Malik

Designation: Lecturer

PUCIT

______________________

Project Examiner

Name: Bilal Javed

Designation: Visiting Lecturer

PUCIT

# Acknowledgement

We truly acknowledge the cooperation and help make by Name of Kamran Malik, Lecturer at PUCIT of PUCIT and Bilal Javed Visiting Lecturer of PUCIT. He has been a constant source of guidance throughout the course of this project. We are also thankful to our friends and families whose silent support led us to complete our project.

1- Mr. Hassan Tariq

2- Mr. Agha Ali Abbas

3- Mr. Zeeshan Awan

4. Mr. Waleed Ashraf

Date:

June 22, 2016

# Abstract

The purpose of this application is to provide a platform where people can promote the positives about their party and can argue on other parties' perspectives. To provide the fan following percentage of political parties on base different regions that will create a hype that which is the major party in which area. Where everyone can freely give opinion about his / her political views. They can also pass their view on different events happening around and can defend their political party. This application provide a place for everyone to freely express himself in the favor of or against any political party. Anyone can vote for a particular event to show consolidate the position of his party on that work. Online polls to predict the result of elections. User will be able to tell how much a user is sincere with his/her political party based on his/her profile and posts etc. As it is a social network so it will be open to everyone.

# Table of Contents

# 1. Functions/Modules descriptions:

## *1.1 API*

Political Social Network API's are totally RESTful. Base URL for this application is `http://nucleo.azurewebsites.net/api`

**Resources:**

- **Login:**

| Method | URL | Parameter | Return |
|---|---|---|---|
| Post | /login | Email<br>Password | User Object |

**e.g.**

**1.**

```
Request: http://nucleo.azurewebsites.net/api/login
Parameters: username: hassan.tariq66@yahoo.com,
password:123456
Response:
      id: 677997-777999-777788-787787-72778
      email: hassan.tariq66@yahoo.com
```

**2.**

```
Request: http://nucleo.azurewebsites.net/api/login
Parameters: username: hahdjh@yahoo.com, password:123456
Response:
      400 bad request
      User not found
```

- **User:**

| **Method** | **URL** | **Parameter** | **Return** |
|---|---|---|---|
| Get | /user | None | Array of user object. |
| Get | /user | Id | User Object |
| Get | /user | Name | User object |
| Post | /user | User Object | User object |
| Put | /user | 1. Id<br>2. User object | User object |
| Delete | /user | Id | Status 200 |

**e.g.**

**1. GET**

Request: `http://nucleo.azurewebsites.net/user`

```
Parameters: none
Response:
            [
        {
          "uid": "855f9d40-3735-11e6-b223-d7e5db589247",
          "password":
"1000:kwbhJ10aiHFbQ48SA3kQvsNysFUjBPI0:anKYPSGJWcS61QU7Oh4sCKwa8W
xsYNex",
          "email": "hassan@madfrequency.com"
        },
        {
          "uid": "187abfb6-37b4-11e6-b223-d7e5db589247",
          "password":
"1000:9+gNLhut7nN8TaAE3GRHJE52ApjpxoC9:pYuDwVWmlmA5IyHT4W+LcrcNQv
my9n4t",
          "email": "abc@live.com"
```

```
  }
]
```

### 2. POST

Request: `http://nucleo.azurewebsites.net/user`

```
JSON Object for Posting:
{
        "password": 123456
        "email": "hassan@madfrequency.com"
  }


Response:
  {
    "uid": "187abfb6-37b4-11e6-b223-d7e5db589247",
    "password":
"1000:9+gNLhut7nN8TaAE3GRHJE52ApjpxoC9:pYuDwVWmlmA5IyHT4W+LcrcNQv
my9n4t",
    "email": "abc@live.com"
  }
```

### 3. Delete

Request: `http://nucleo.azurewebsites.net/user`

```
Parameters: uid: 187abfb6-37b4-11e6-b223-d7e5db589247
Response: Status 200
```

## ➢ Party:

| **Method** | **URL** | **Parameter** | **Return** |
|---|---|---|---|
| Get | /party/getall | None | Array of party object. |
| Get | /party/getbyid | Id | Party Object |

| | | | |
|---|---|---|---|
| Get | /party/getbyname | Name | Party object |
| Post | /party/createparty | Party object | Party object |
| Put | /party/ | **1.** Id<br>**2.** User object | User object |
| Delete | /user | Id | Status 200 |

All the requests, parameters and responses are same as explained above.

- **Page:**

| **Method** | **URL** | **Parameter** | **Return** |
|---|---|---|---|
| Get | /page/getall | None | Array of page object. |
| Get | /page /getbyname | Name | Page object |
| Get | /page/ | Page | Page object |
| Put | /page / | **3.** Id<br>**4.** Page object | page object |
| Delete | /page | Id | Status 200 |

All the requests, parameters and responses are same as explained above.

There are extensive API's requests other than above, below are the all URL's for request.

### 1.2 User Request URLs:

- /userinfo
- /userfriend
- /userpoll
- /userpolloption
- /userpollVote
- /userPost
- /userpostcomment
- /userpostLike
- /userpostdeslike
- /userpostcommentlike
- /userpostcommentdislike

### 1.3Party Request URLs:

- /partyfollower
- /partyunfollower
- /partypost
- /partypostcomment
- /partypostLike
- /partypostdeslike
- /partyadmin
- /partymember
- /partypostcommentlike
- /partypostcommentdislike

### 1.4 Page Request URLs:

- /pagefollower
- /pageinfo
- /pagepost
- /pagepostcomment
- /pagepostLike
- /pagepostdeslike
- /pageadmin
- /pagepostcommentlike
- /pagepostcommentdislike
-

### 1.5 Phone Vote Request URLs:

- /phoneVote

### 1.6 Search Request URLs:

- /search

### 1.7Watch Panel request URLs:

- /watchpanel

There are various more requests for API’s in this application.

## 2. Database

We are using the Cassandra No-sql database for our application. Apache Cassandra is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.

Listed below are some of the notable points of Apache Cassandra:

- It is scalable, fault-tolerant, and consistent.
- It is a column-oriented database.
- Its distribution design is based on Amazon’s Dynamo and its data model on Google’s Bigtable.
- Created at Facebook, it differs sharply from relational database management systems.
- Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful “column family” data model.
- Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, eBay, Twitter, Netflix, and more.

Following is the description of some of the column families in our keyspace but before that let us explain two important terms partition key and clustering key used throughout our schema.

The primary key is a general concept to indicate one or more columns used to retrieve data from a Table.

The primary key may be ***SIMPLE***

```
create table stackoverflow (
    key text PRIMARY KEY,
    data text
 );
```

That means that it is made by a single column.

But the primary key can also be ***COMPOSITE*** (aka ***COMPOUND***), generated from more columns.

```
 create table stackoverflow (
     key_part_one text,
     key_part_two int,
     data text,
     PRIMARY KEY(key_part_one, key_part_two)
  );
```

In a situation of ***COMPOSITE*** primary key, the "first part" of the key is called ***PARTITION KEY*** (in this example **key_part_one** is the partition key) and the second part of the key is the ***CLUSTERING KEY*** (**key_part_two**)

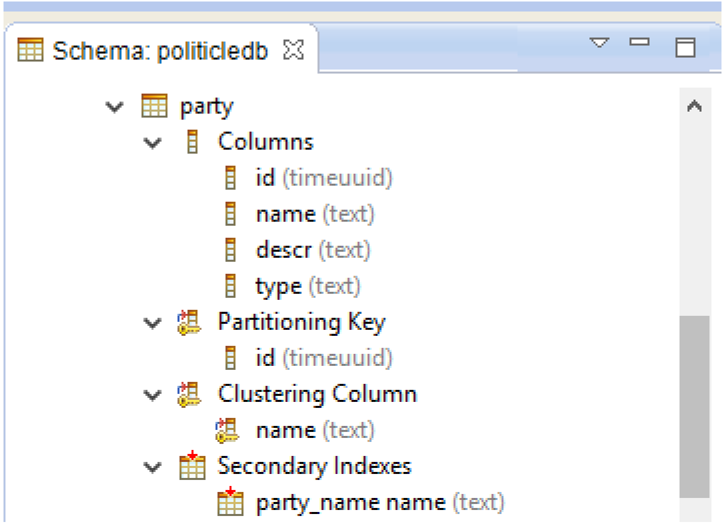
**Please note that the both partition and clustering key can be made by more columns**

```
 create table stackoverflow (
     k_part_one text,
     k_part_two int,
     k_clust_one text,
     k_clust_two int,
     k_clust_three uuid,
     data text,
     PRIMARY KEY((k_part_one,k_part_two), k_clust_one, k_clust_two,
k_clust_three)
  );
```

- The **Partition Key** is responsible for data distribution across your nodes.
- The **Clustering Key** is responsible for data sorting within the partition.

## 2.1 Party Column Family:

The table i.e column family in Cassandra for parties consists of id, name, description, type.

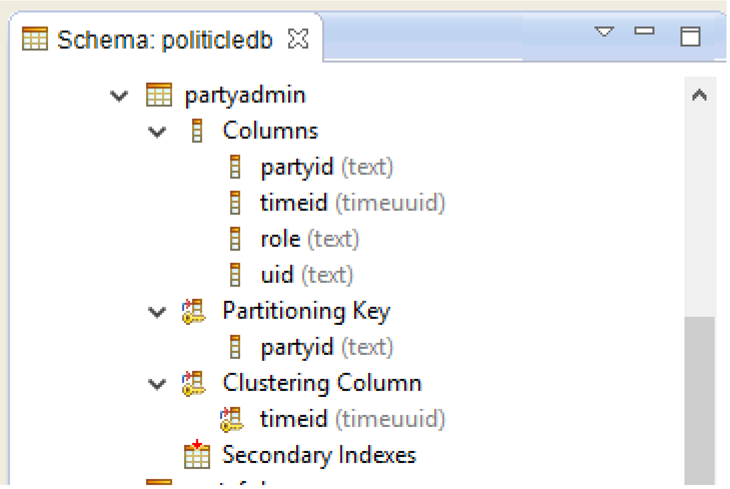


Schema: politicledb

- party
  - Columns
    - id (timeuuid)
    - name (text)
    - descr (text)
    - type (text)
  - Partitioning Key
    - id (timeuuid)
  - Clustering Column
    - name (text)
  - Secondary Indexes
    - party_name name (text)

## 2.2 Party Admin Column Family:

Party admin column family consists of party id, time id, role, user id.

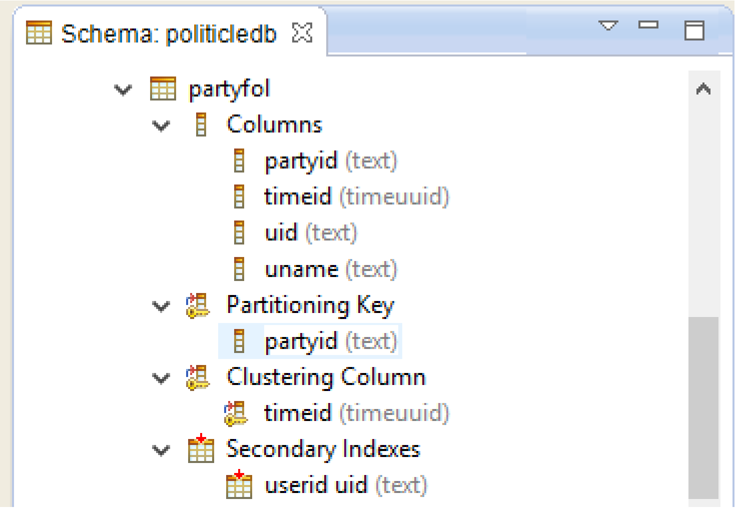


Schema: politicledb

- partyadmin
  - Columns
    - partyid (text)
    - timeid (timeuuid)
    - role (text)
    - uid (text)
  - Partitioning Key
    - partyid (text)
  - Clustering Column
    - timeid (timeuuid)
  - Secondary Indexes

### 2.3 Party Follower Column Family:

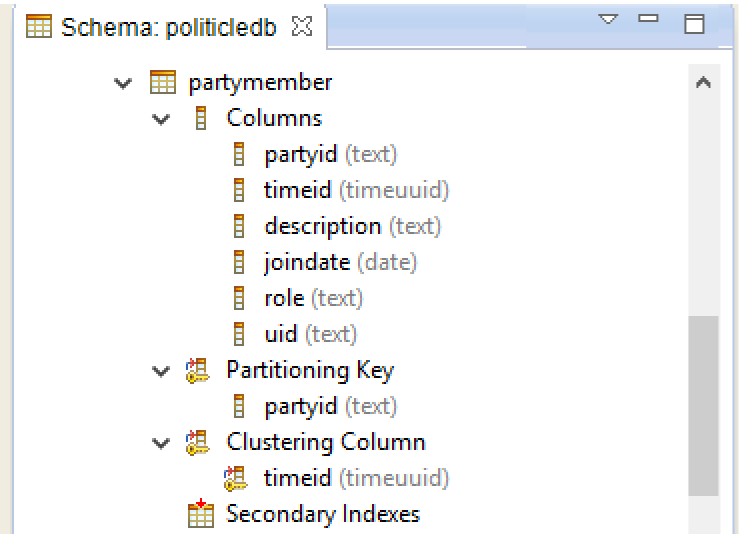
Party Follower column family consists of party id, a time id, user id, username.



```
Schema: politicledb
    partyfol
        Columns
            partyid (text)
            timeid (timeuuid)
            uid (text)
            uname (text)
        Partitioning Key
            partyid (text)
        Clustering Column
            timeid (timeuuid)
        Secondary Indexes
            userid uid (text)
```

### 2.4 Party Member Column Family:

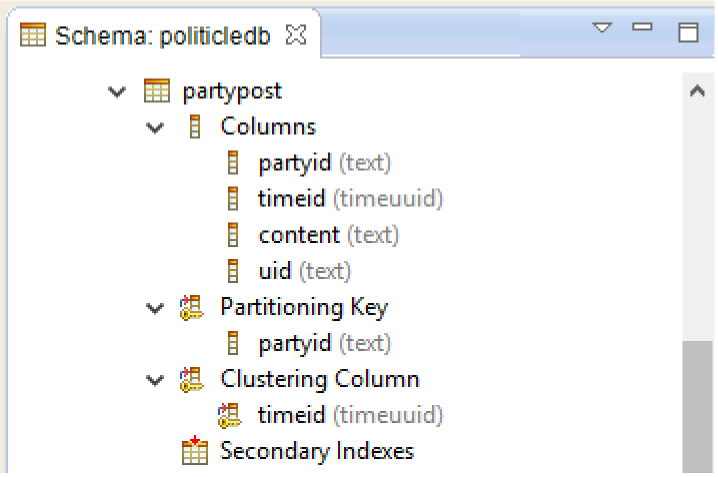
Party member column family consists of party id, time id, description, join date, role, user id.



```
Schema: politicledb
    partymember
        Columns
            partyid (text)
            timeid (timeuuid)
            description (text)
            joindate (date)
            role (text)
            uid (text)
        Partitioning Key
            partyid (text)
        Clustering Column
            timeid (timeuuid)
        Secondary Indexes
```

### Party Post Column Family:

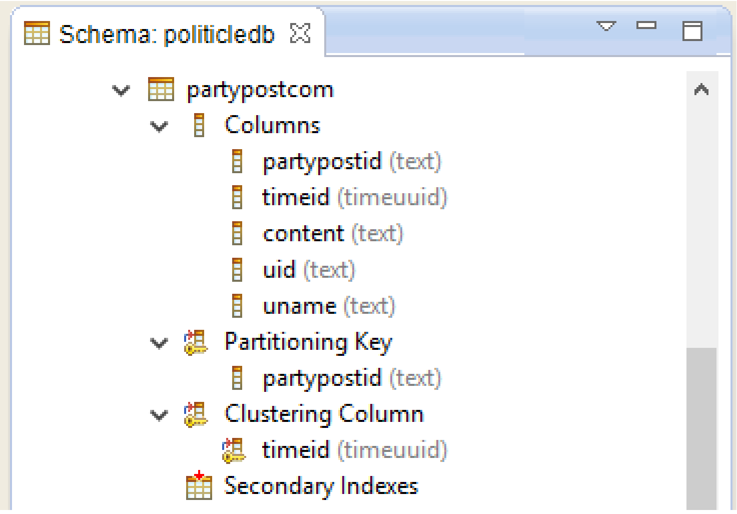
Party post column family consists of party id, time id, content, user id.



```
Schema: politicledb
  partypost
    Columns
      partyid (text)
      timeid (timeuuid)
      content (text)
      uid (text)
    Partitioning Key
      partyid (text)
    Clustering Column
      timeid (timeuuid)
    Secondary Indexes
```

### 2.4.1 Party Post Comment Column Family:

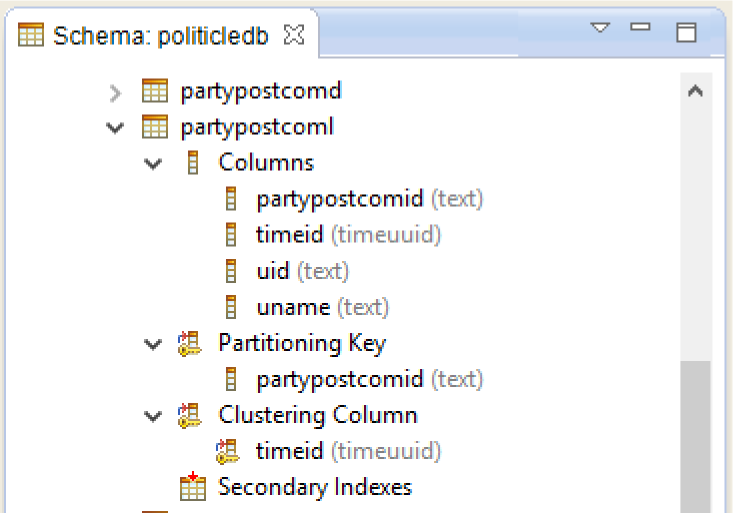
Party post comment column family has the following attributes.



```
Schema: politicledb
  partypostcom
    Columns
      partypostid (text)
      timeid (timeuuid)
      content (text)
      uid (text)
      uname (text)
    Partitioning Key
      partypostid (text)
    Clustering Column
      timeid (timeuuid)
    Secondary Indexes
```

### 2.4.2 Party Post Comment Like Column Family:

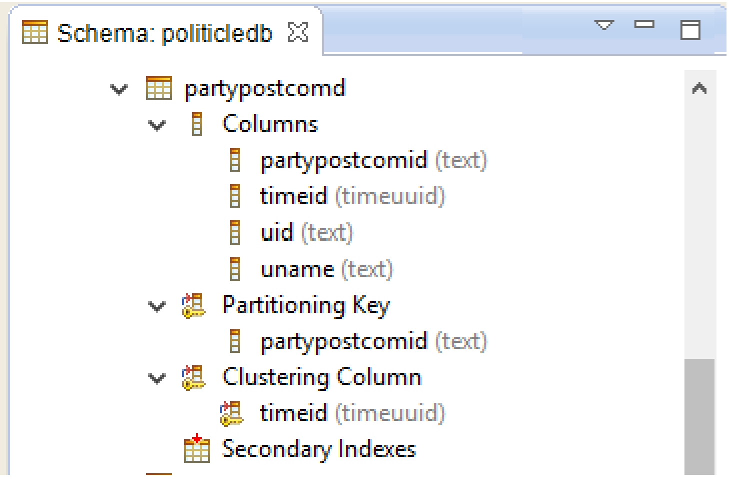
Schema of party post comments like column family is as follows.



```
Schema: politicledb
    > partypostcomd
    v partypostcoml
        v Columns
              partypostcomid (text)
              timeid (timeuuid)
              uid (text)
              uname (text)
        v Partitioning Key
              partypostcomid (text)
        v Clustering Column
              timeid (timeuuid)
          Secondary Indexes
```

### 2.4.3 Party Post Comment DisLike Column Family:

Schema of party post comments dislike column family is as follows.

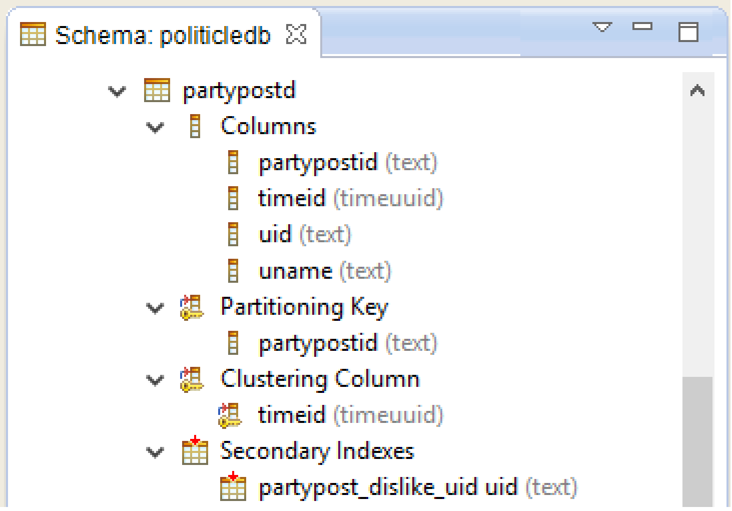


```
Schema: politicledb
    v partypostcomd
        v Columns
              partypostcomid (text)
              timeid (timeuuid)
              uid (text)
              uname (text)
        v Partitioning Key
              partypostcomid (text)
        v Clustering Column
              timeid (timeuuid)
          Secondary Indexes
```

## 3.

### 2.1.9 Party Post Dislike Column Family:

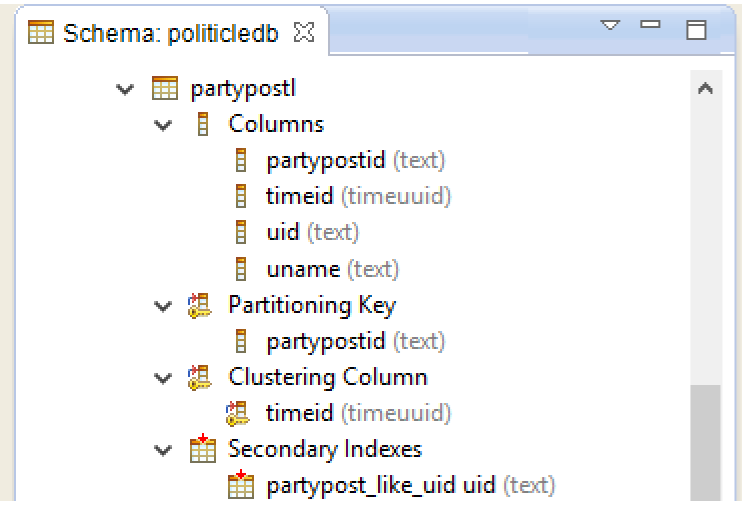
Party post dislike column family has the following schema.



- partypostd
  - Columns
    - partypostid (text)
    - timeid (timeuuid)
    - uid (text)
    - uname (text)
  - Partitioning Key
    - partypostid (text)
  - Clustering Column
    - timeid (timeuuid)
  - Secondary Indexes
    - partypost_dislike_uid uid (text)

### 2.1.10 Party Post Like Column Family:

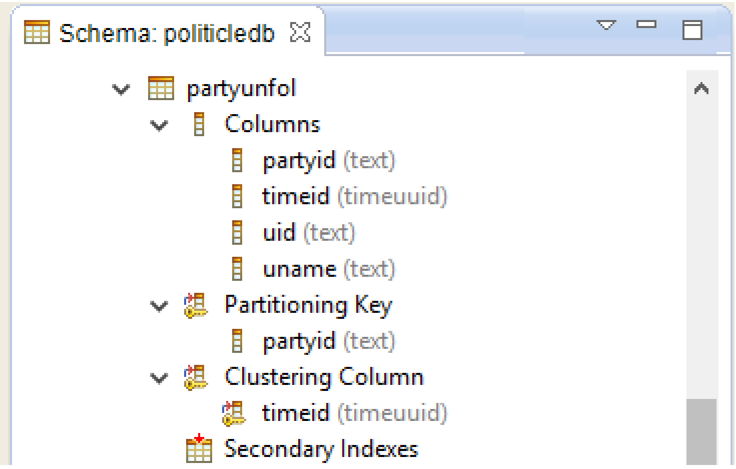
Party post like column family has the following schema.



- partypostl
  - Columns
    - partypostid (text)
    - timeid (timeuuid)
    - uid (text)
    - uname (text)
  - Partitioning Key
    - partypostid (text)
  - Clustering Column
    - timeid (timeuuid)
  - Secondary Indexes
    - partypost_like_uid uid (text)

### 2.1.11 Party Unfollower Column Family:

Party unfollower column family to maintain statistics and records has the following schema.

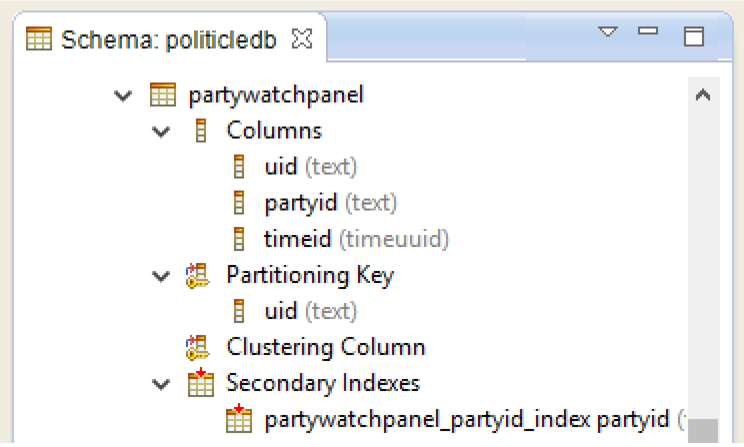


Schema: politicledb

- partyunfol
  - Columns
    - partyid (text)
    - timeid (timeuuid)
    - uid (text)
    - uname (text)
  - Partitioning Key
    - partyid (text)
  - Clustering Column
    - timeid (timeuuid)
  - Secondary Indexes

### 2.1.12 Party Watchpanel Column Family:

Party watchpanel column family to maintain watchpanel and show statistics to users, has the following schema.

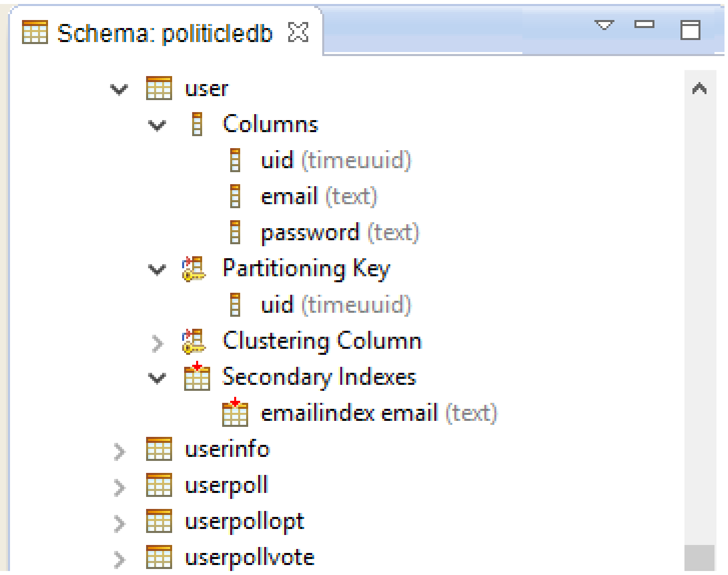


Schema: politicledb

- partywatchpanel
  - Columns
    - uid (text)
    - partyid (text)
    - timeid (timeuuid)
  - Partitioning Key
    - uid (text)
  - Clustering Column
  - Secondary Indexes
    - partywatchpanel_partyid_index partyid (

### 2.1.13 User Column Family:

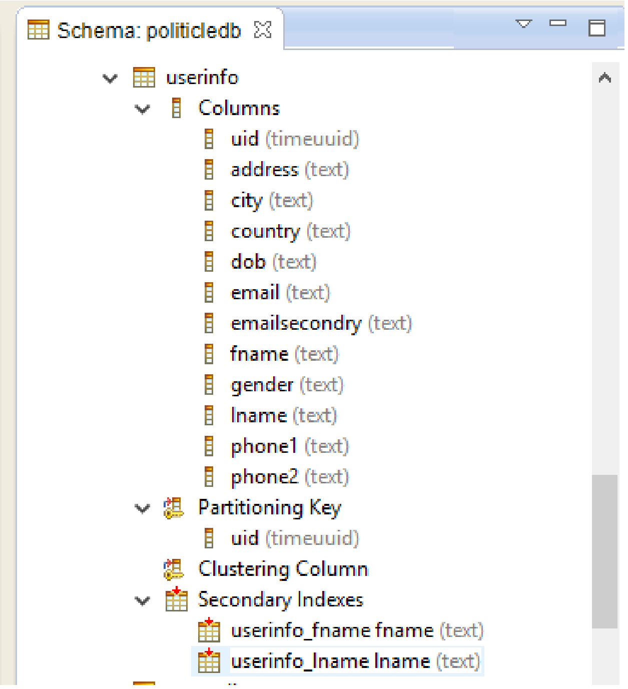
Column Family for user login has the following schema.



```
Schema: politicledb
  user
    Columns
      uid (timeuuid)
      email (text)
      password (text)
    Partitioning Key
      uid (timeuuid)
    Clustering Column
    Secondary Indexes
      emailindex email (text)
  userinfo
  userpoll
  userpollopt
  userpollvote
```

### 2.1.14 User Information Column Family:

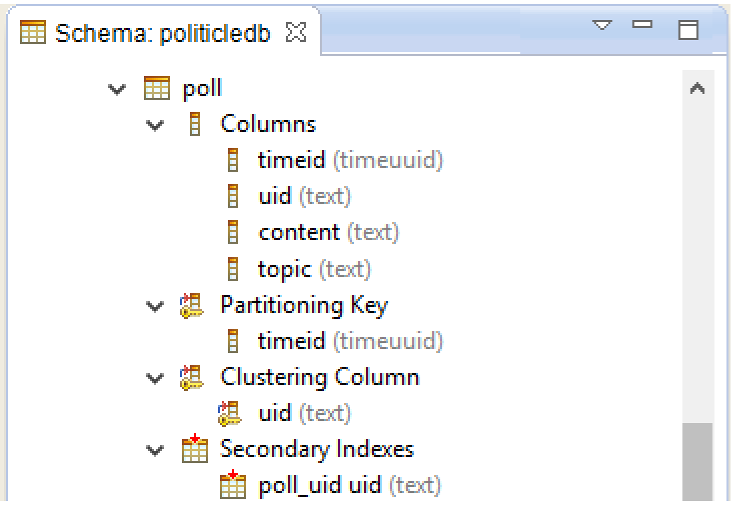
Column Family for user detailed information has the following schema.



```
Schema: politicledb
  userinfo
    Columns
      uid (timeuuid)
      address (text)
      city (text)
      country (text)
      dob (text)
      email (text)
      emailsecondry (text)
      fname (text)
      gender (text)
      lname (text)
      phone1 (text)
      phone2 (text)
    Partitioning Key
      uid (timeuuid)
    Clustering Column
    Secondary Indexes
      userinfo_fname fname (text)
      userinfo_lname lname (text)
```

### 2.1.15 Poll Column family:

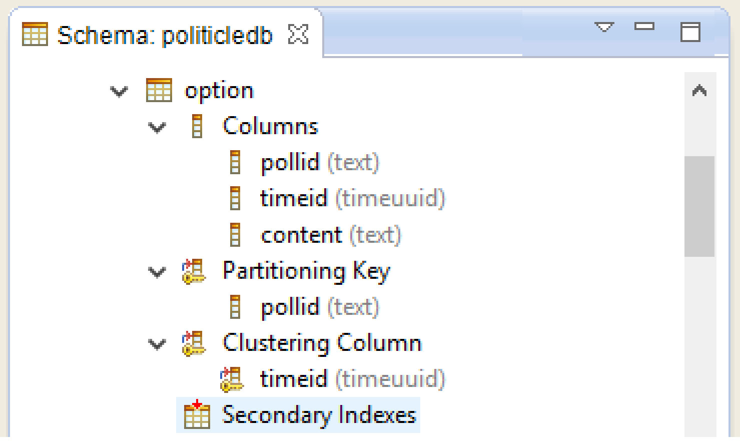
Poll column family has the following schema.



```
Schema: politicledb
  poll
    Columns
      timeid (timeuuid)
      uid (text)
      content (text)
      topic (text)
    Partitioning Key
      timeid (timeuuid)
    Clustering Column
      uid (text)
    Secondary Indexes
      poll_uid uid (text)
```

### 2.1.16 Poll Options Column Family:

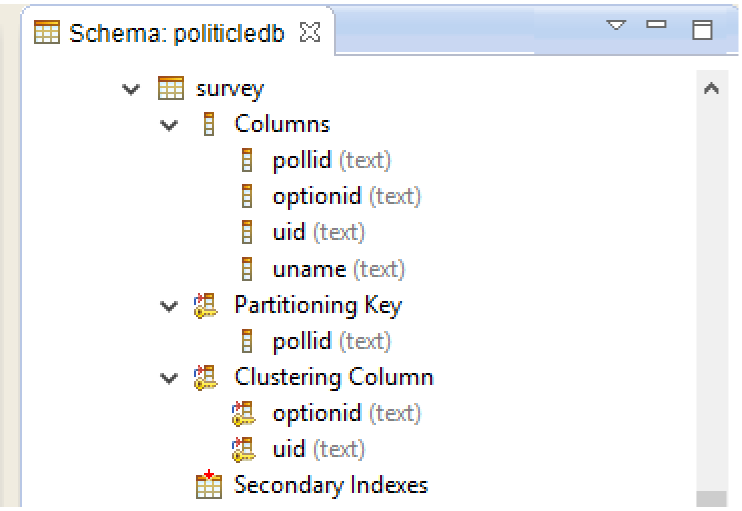
Option column family for poll has the following attributes.



```
Schema: politicledb
  option
    Columns
      pollid (text)
      timeid (timeuuid)
      content (text)
    Partitioning Key
      pollid (text)
    Clustering Column
      timeid (timeuuid)
    Secondary Indexes
```

### 2.1.17 Survey Column Family:

Schema of survey column family is as follows.

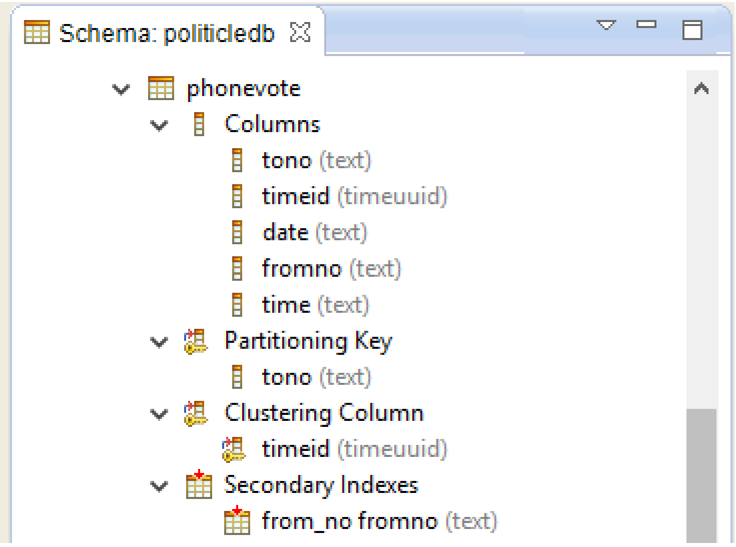


Schema: politicledb

```
survey
  Columns
    pollid (text)
    optionid (text)
    uid (text)
    uname (text)
  Partitioning Key
    pollid (text)
  Clustering Column
    optionid (text)
    uid (text)
  Secondary Indexes
```

### 2.1.18 Phone Vote Column Family:

Column family for casting votes through android app has the following structure.

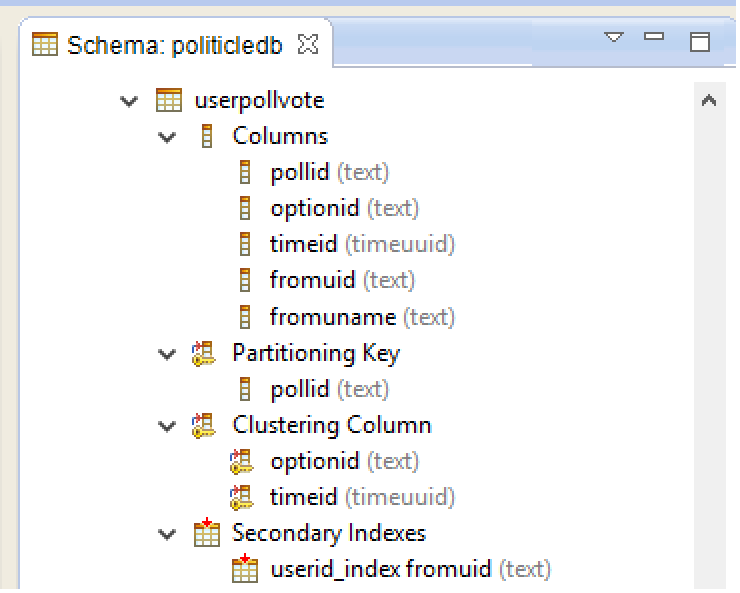


Schema: politicledb

```
phonevote
  Columns
    tono (text)
    timeid (timeuuid)
    date (text)
    fromno (text)
    time (text)
  Partitioning Key
    tono (text)
  Clustering Column
    timeid (timeuuid)
  Secondary Indexes
    from_no fromno (text)
```

### 2.1.19 Poll Vote Column Family:

Column family for votes for polls created by users has the following structure.



Schema: politicledb

```
userpollvote
  Columns
    pollid (text)
    optionid (text)
    timeid (timeuuid)
    fromuid (text)
    fromuname (text)
  Partitioning Key
    pollid (text)
  Clustering Column
    optionid (text)
    timeid (timeuuid)
  Secondary Indexes
    userid_index fromuid (text)
```

Schema for pages is also similar to the parties.

## How we are using it in our application?

Our database layer is completely separated from our front end and backend. Cassandra provides the mapping feature. We have created all models for mapping the whole object on the table. We have created the Data Access class which handles connecting application with the database by using the node address.

We have created the different classes for handling different types of works. Below are the classes and their description.

- UserOperation: This class handles all the queries related to user as entity.
- PartyOperation: This class handles all the queries relayed to party as entity.
- PageOperation:  This class handles all the queries related to Page as entity.
- PollOperation: This class handles all the queries related to Poll as entity.
- PhoneVoteOperation: This class handles all the queries related to phone voting module of our application.

We have functions in these classes for handling the specific query.

## How we query from database?

To query from the Database, we are using the powerful mapping technique. For using this we first have to model the table over the object like this.

Suppose we have friend table in database we want to model this table over the object so we can easily perform some CRUD operation with little limitation.

Each line written with the [] have specific meaning.

```
[Table("politicledb.friends")]
    public class Friend
    {
        [PartitionKey]
        [Column("uid")]
        public string uid { get; set; }
        [ClusteringKey]
        [Column("timeid")]
        public Guid id { get; set; }
```

```
 [Column("friendid")]
 public string friendid { get; set; }
 [Column("friendname")]
 public string friendname { get; set; }
}
```

After that we can get all the friend table rows by just using the Imapper object like this.

```
mapper.Fetch<Page>().ToList();
```

```
here mapper is Imapper object.
```

We are using this technique in our whole application.

## 3.1 ***Front End:***

⇨ We are using the hybrid approach for front end of our application.

⇨ We are using the AngularJS and .net C# razor syntax for front end of our application.

⇨ All the request related to API's calling handles using the AngularJS services.

⇨ All the pages first rendered by .net controller approach. Then the control is given to the angular.

⇨ Here we have completely separated AngularJS code from the HTML by making controller and services.

⇨ We have created JS file for each HTML view. So that we can completely handle AngularJS code in another file. Below are some of the files name.

- UserLogin.js
- Userfeed.js
- Config.js
- Party.js
- Page.js
- Profile.js
- FollowePages.js

- FollowParties.js

There are various more JS files in which each one handles its own functionality.

### 3.1.1 UserLogin.js

- Here we have our ng-app module namely **homepage.** This module have following controllers
  - **LoginCtrl**: This controller is bind to the front end UI of login. When user click on the login button the event listener in this controller namely CheckUser() called which handles the service calling.
  - **SignUpCtrl**: This controller is bind to the front end UI of signup form. When user click on the login button the event listener in this controller namely SignUpUser () called which handles the service calling.
- We also have one Directive
  - **Username**: This directive is related to checking the email as user type email in the email box of the front end.

### 3.1.2 UserFeed.js

- Here we have our ng-app module namely **feed.** This module have following controllers
  - **AboutCtrl:** This controller is responsible to get the user detail which is currently login either by using the cookies or by hitting the service.
  - **FeedPostCtrl:** All the posts related to parties, follower and pages and their likes, deslikes and comments are fetch here by using the services.
  - **LikesCtrl:** Handling the like button and posting the like using service requirement are met here.
  - **PostStatusCtrl:** Status posting handles by this controller by using service.

    - **PostCommentCtrl:** Comment posting controller handles simple status posting using service.
    - **DoughnutChartCtrl:** Getting the DoughnutChart related data handles by this controller by using the service.
    - **LineChartCtrl:** Getting the LineChart related data handles by this controller by using the service.
    - **BarChartCtrl:** Getting the BarChart related data handles by this controller by using the service.
    - **PostVoteCtrl:** Posting the Vote of the Poll using service call handles by this controller.
    - **VoteCtrl:** Election voting data fetch by this controller.

### 3.1.3 Profile.js

- Here we have our ng-app module namely **Profile.** This module have following controllers
    - **AboutCtrl:** This controller is responsible to get the user detail which is currently login either by using the cookies or by hitting the service.
    - **ProfilePostCtrl:** All the posts of the user, their likes, deslikes and comments are fetch here by using the services.
    - **LikesCtrl:** Handling the like button and posting the like using service requirement are met here.
    - **PostStatusCtrl:** Status posting handles by this controller by using service.
    - **PostCommentCtrl:** Comment posting controller handles simple status posting using service.
    - **DoughnutChartCtrl:** Getting the DoughnutChart related data handles by this controller by using the service.
    - **LineChartCtrl:** Getting the LineChart related data handles by this controller by using the service.
    - **BarChartCtrl:** Getting the BarChart related data handles by this controller by using the service.
    - **PostVoteCtrl:** Posting the Vote of the Poll using service call handles by this controller.

### 3.1.4 Party.js

- Here we have our ng-app module namely **Party.** This module have following controllers
    - **PartyCreationCtrl:** This handles the Party signup and posting it API's using service call.
    - **AboutCtrl:** This controller is responsible to get the user detail which is currently login either by using the cookies or by hitting the service.
    - **PartyPostCtrl:** All the posts of the Party, their likes, deslikes and comments are fetch here by using the services.
    - **LikesCtrl:** Handling the like button and posting the like using service requirement are met here.
    - **PostStatusCtrl:** Status posting handles by this controller by using service.
    - **PostCommentCtrl:** Comment posting controller handles simple status posting using service.
    - **PostVoteCtrl:** Posting the Vote of the Poll using service call handles by this controller

### 3.1.5 Page.js

- Here we have our ng-app module namely **page.** This module have following controllers
    - **PageCreationCtrl:** This handles the page signup and posting it API's using service call.
    - **AboutCtrl:** This controller is responsible to get the user detail which is currently login either by using the cookies or by hitting the service.
    - **PartyPostCtrl:** All the posts of the Party, their likes, deslikes and comments are fetch here by using the services.
    - **LikesCtrl:** Handling the like button and posting the like using service requirement are met here.
    - **PostStatusCtrl:** Status posting handles by this controller by using service.
    - **PostCommentCtrl:** Comment posting controller handles simple status posting using service.

    - **PostVoteCtrl:** Posting the Vote of the Poll using service call handles by this controller.

### 3.1.6 Search.js

- Here we have our ng-app module namely **search.** This module is using the UI of the Header.cshtml file which is common in most pages.This module have following controllers
    - **SearchCtrl:** Handling the searches all over application is handled by this controller. Whether you are searching about user or party or page you just have to type name or keyword for search.

### 3.1.7 ManageFollowers.js

- Here we have our ng-app module namely **ManageFollowers.** This module have following controllers

- **AboutCtrl:** This controller is responsible to get the user detail which is currently login either by using the cookies or by hitting the service.

- **findUserCtrl:** This controller is responsible to get current user which is followed by current user. This controller also handles search request type in search box by name box. It populate the below list with query search related users. This controller also have event listeners related to follow or unfollow events.

### 3.1.8 FollowPages.js

Here we have our ng-app module namely **selectpage.** This module have following controllers

- **UserInfoCtrl:** This controller is responsible to get the user detail which is currently login either by using the cookies or by hitting the service.

- **SelectPageCtrl:** This controller is responsible to get all the pages details and its follower count. This controller also handles the Follow and unfollow events.

### 3.1.9 FollowParties.js

Here we have our ng-app module namely **selectparty.** This module have following controllers

- **UserInfoCtrl:** This controller is responsible to get the user detail which is currently login either by using the cookies or by hitting the service.

- **SelectPartiesCtrl:** This controller is responsible to get all the Parties details and its follower count. This controller also handles the Follow and unfollow events.

### 3.1.10 Shared.js

Here we have our factories and services written which is common across application

**Factory:**

- **$localstorage:** This factory is responsible to handle all local storage of our application

**Services:**

- **UserInfo:** User related service are written here.
- **Likes:** Like and dislike related services are written here.
- **PageManager:** Basically all the service related to pages, parties and users are written here which is common across all.

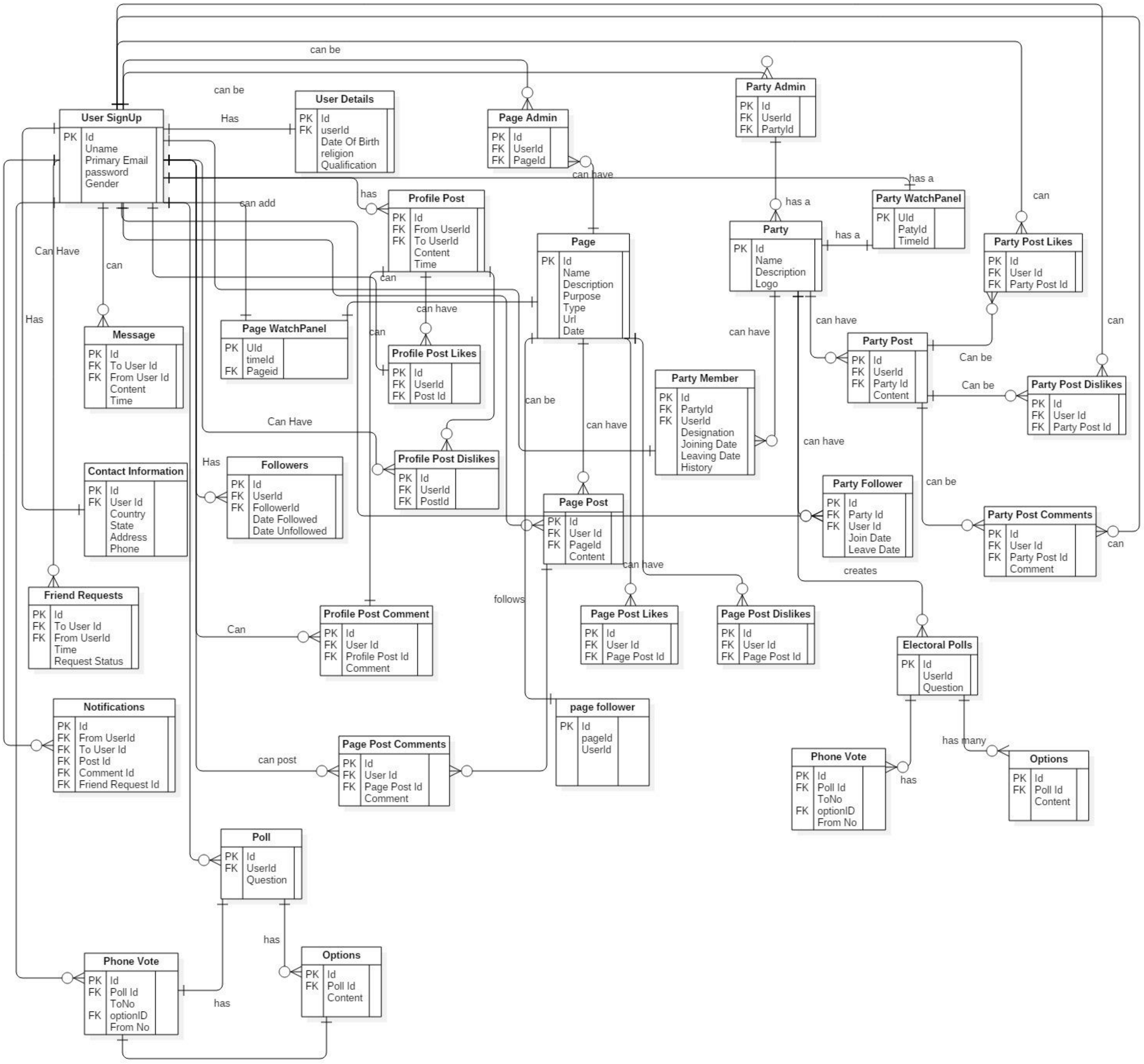
**Directives:**

- **ngReallyClick:** This handles the popup of the click event if application show message that is because of this.
- **errSrc:** This handle the images if because of some reason image URL responded with 404 error then this directive shows default instead of broken image.

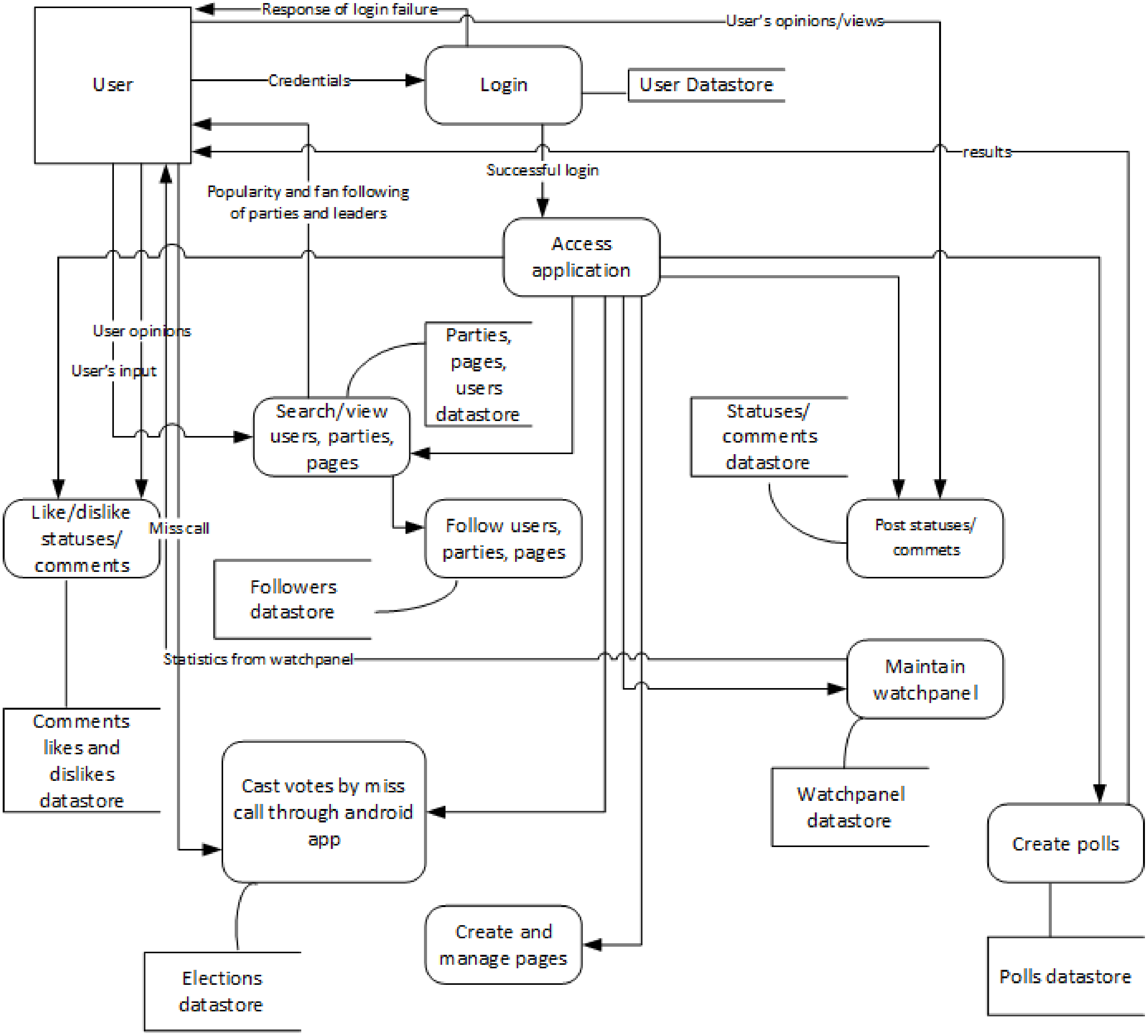
### 3.1.11 Config.js

Here we have written constant that are common across all application.

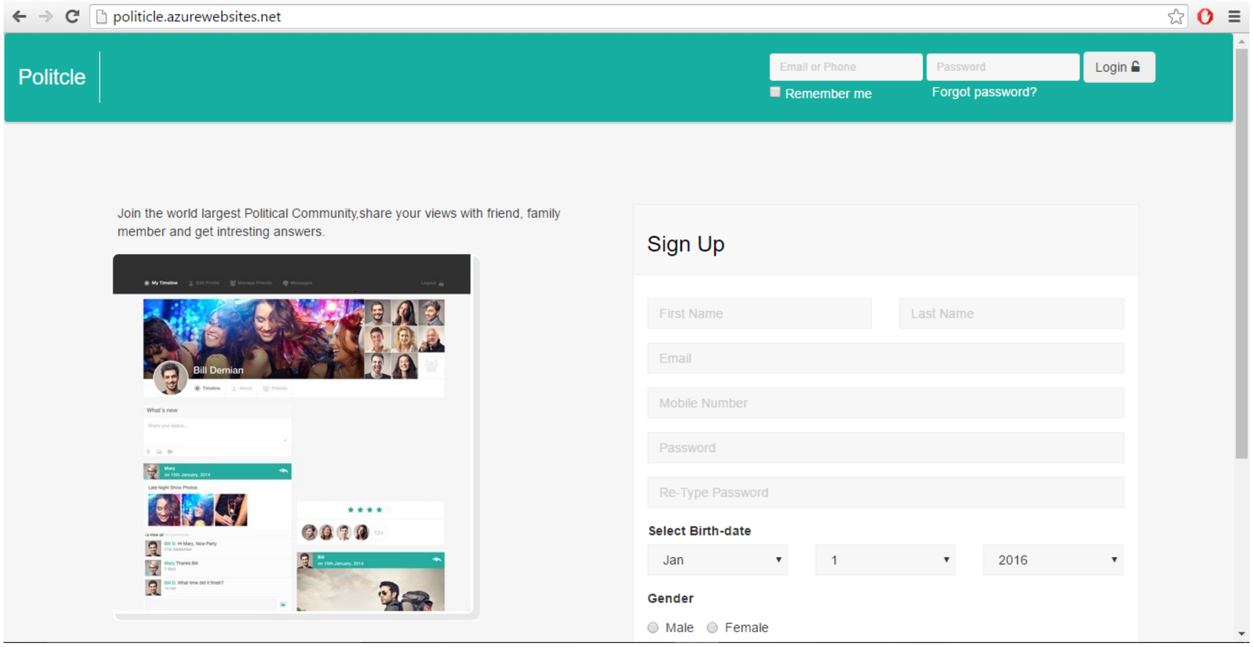
# 4. Entity Relationship Diagram (ERD)



## 5. Dataflow diagram (DFD)


User
Response of login failure
User's opinions/views
Credentials
Login
User Datastore
results
Successful login
Popularity and fan following of parties and leaders
Access application
User opinions
User's input
Parties, pages, users datastore
Search/view users, parties, pages
Statuses/ comments datastore
Like/dislike statuses/ comments
Miss call
Follow users, parties, pages
Post statuses/ commets
Followers datastore
Statistics from watchpanel
Maintain watchpanel
Comments likes and dislikes datastore
Watchpanel datastore
Cast votes by miss call through android app
Create polls
Create and manage pages
Polls datastore
Elections datastore

# 6. Web Views Screenshot



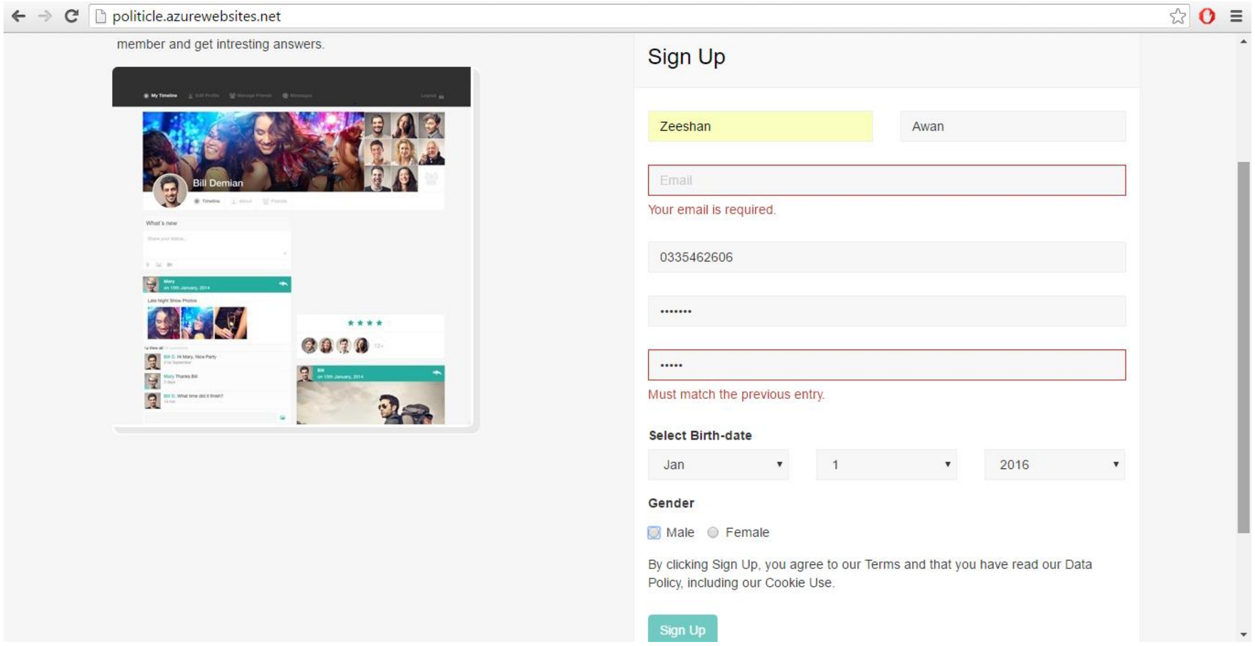
politicle.azurewebsites.net

Join the world largest Political Community,share your views with friend, family member and get intresting answers.

Select Birth-date: Jan ▾ 1 ▾ 2016 ▾

Gender

○ Male ○ Female



politicle.azurewebsites.net

member and get intresting answers.

Zeeshan

Awan

Your email is required.

0335462606

•••••••

•••••

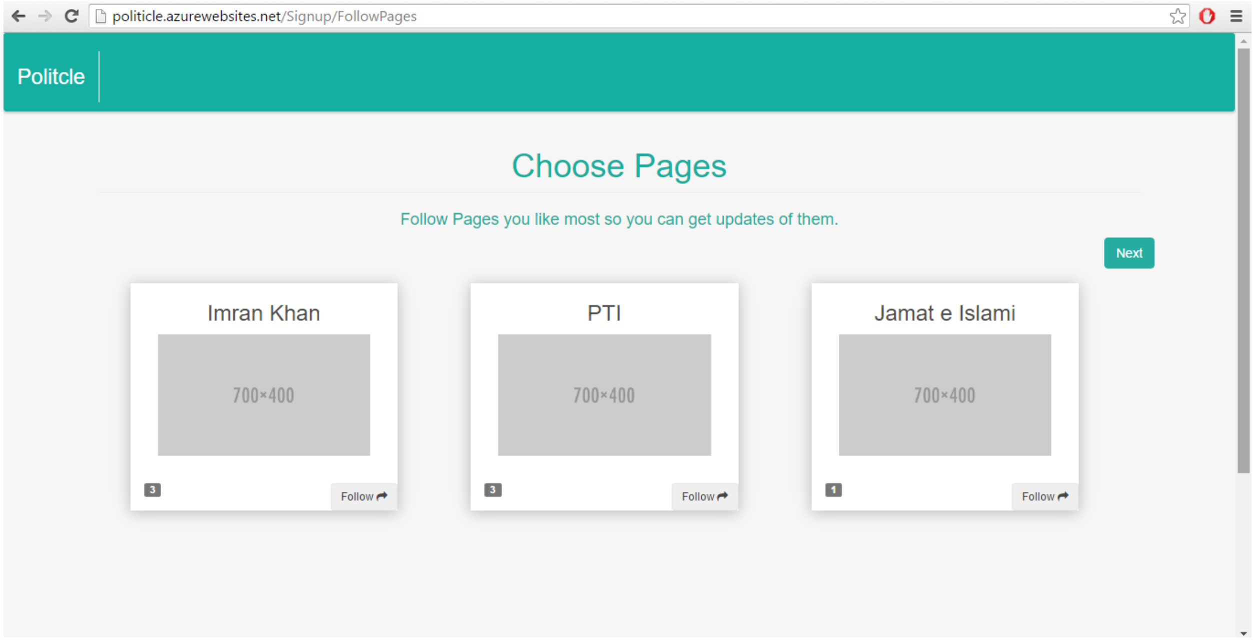
Must match the previous entry.

Select Birth-date: Jan ▾ 1 ▾ 2016 ▾

Gender

☑ Male ○ Female

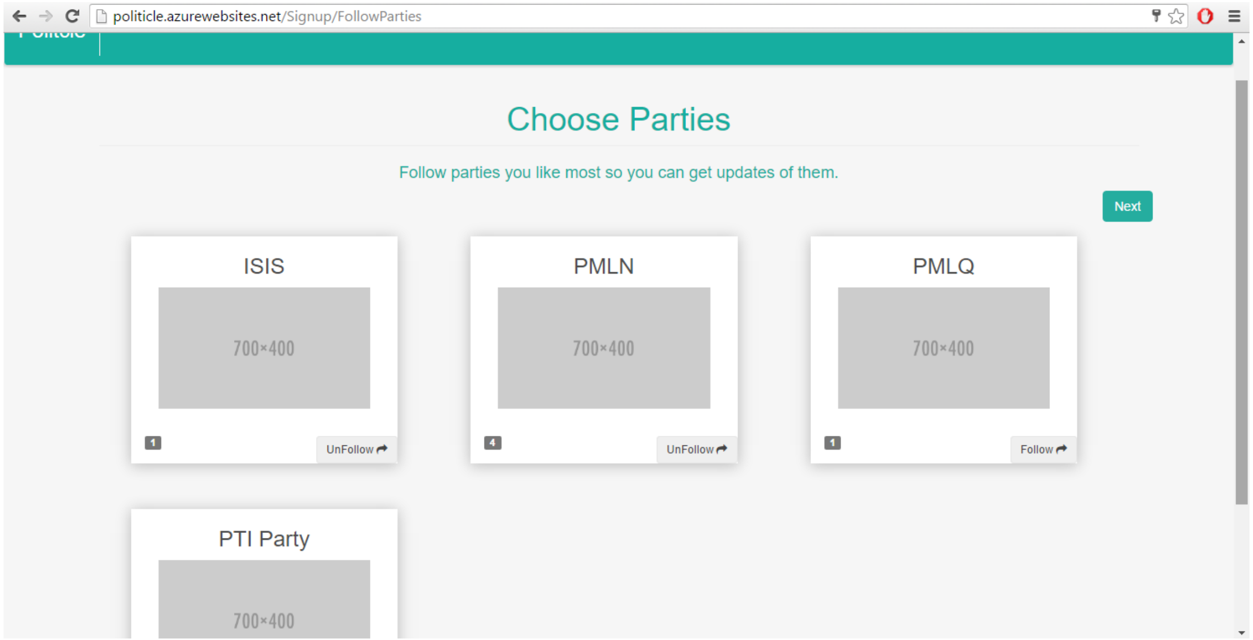
By clicking Sign Up, you agree to our Terms and that you have read our Data Policy, including our Cookie Use.



politicle.azurewebsites.net/Signup/FollowPages

Choose Pages

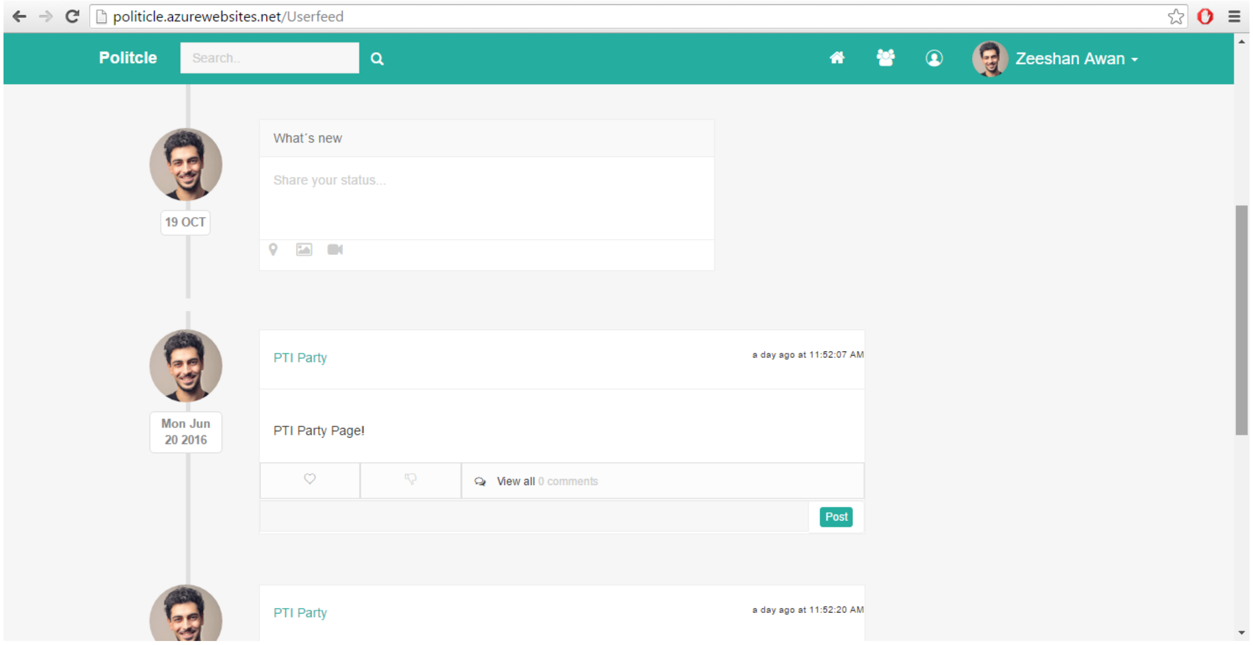
Follow Pages you like most so you can get updates of them.



politicle.azurewebsites.net/Signup/FollowParties

Choose Parties

Follow parties you like most so you can get updates of them.



politicle.azurewebsites.net/Userfeed

What's new

Share your status...

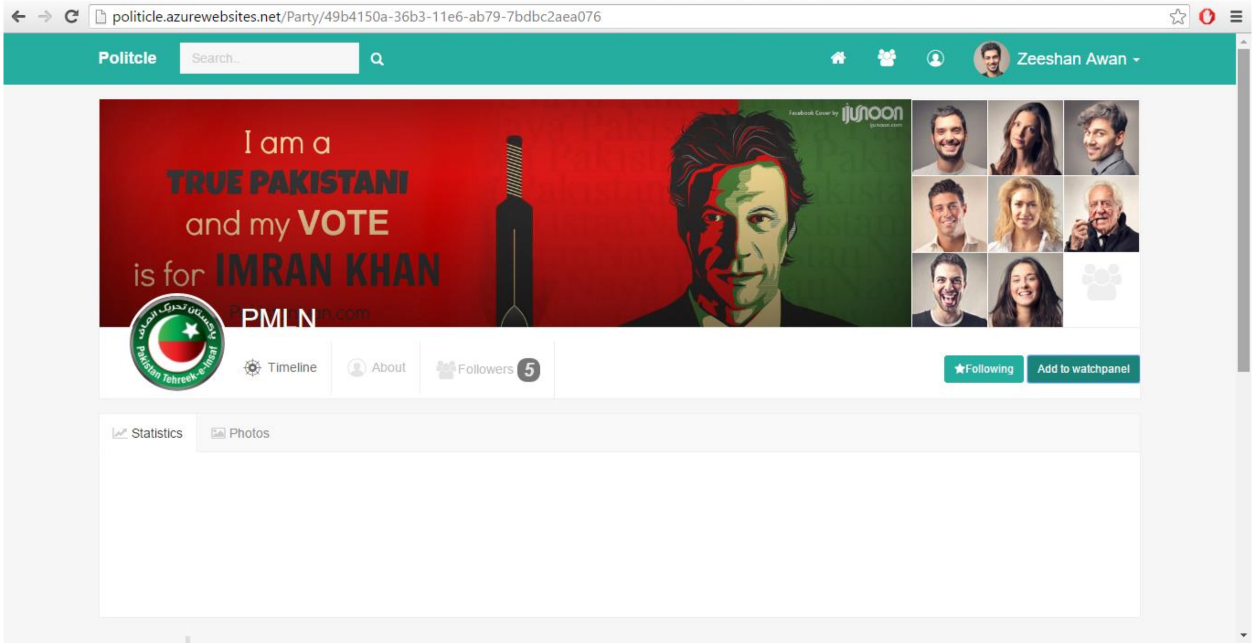
PTI Party

a day ago at 11:52:07 AM

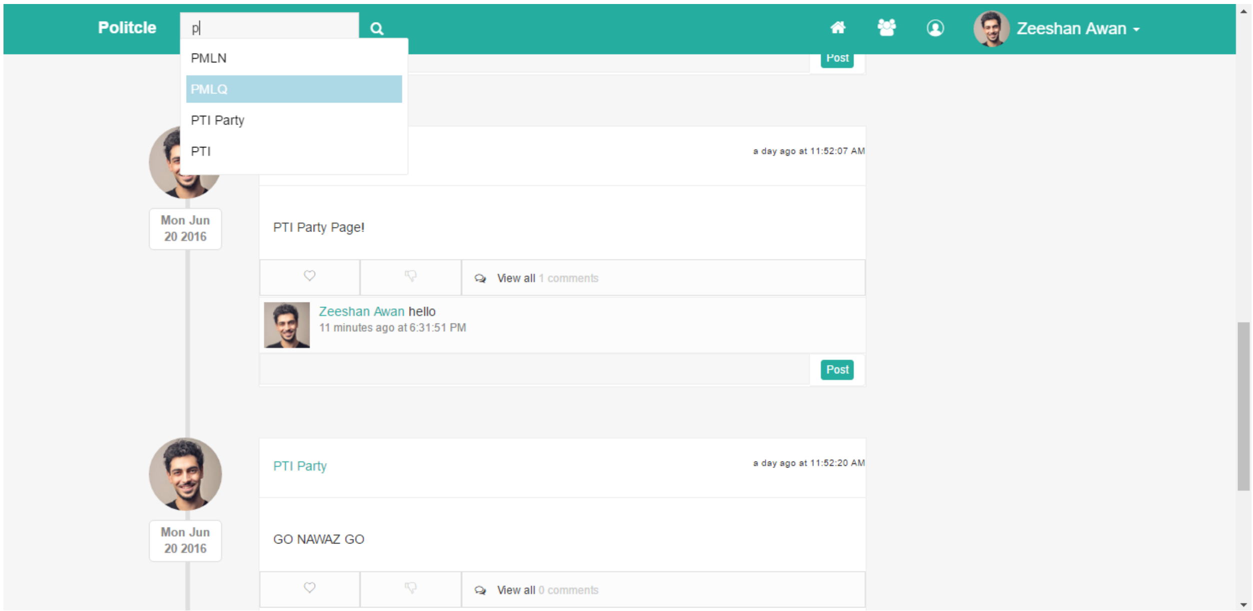
PTI Party Page!

PTI Party

a day ago at 11:52:20 AM



politicle.azurewebsites.net/Party/49b4150a-36b3-11e6-ab79-7bdbc2aea076



a day ago at 11:52:07 AM

Mon Jun 20 2016

PTI Party Page!

View all 1 comments

Zeeshan Awan hello
11 minutes ago at 6:31:51 PM

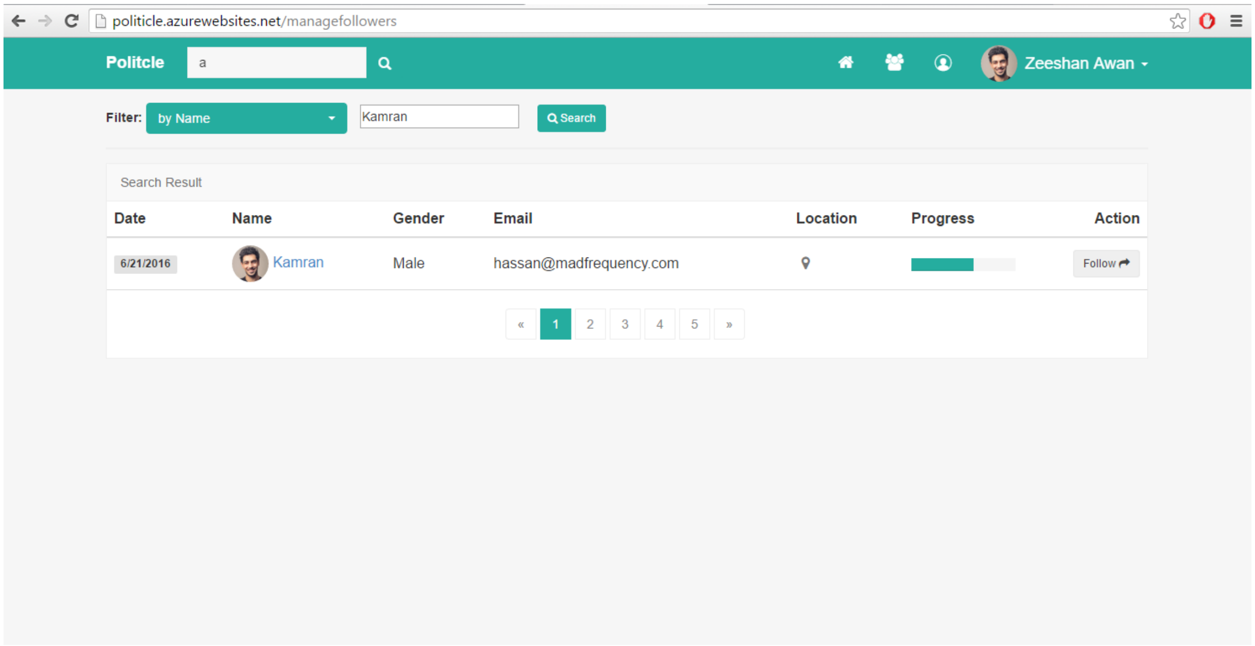
PTI Party

a day ago at 11:52:20 AM

Mon Jun 20 2016

GO NAWAZ GO

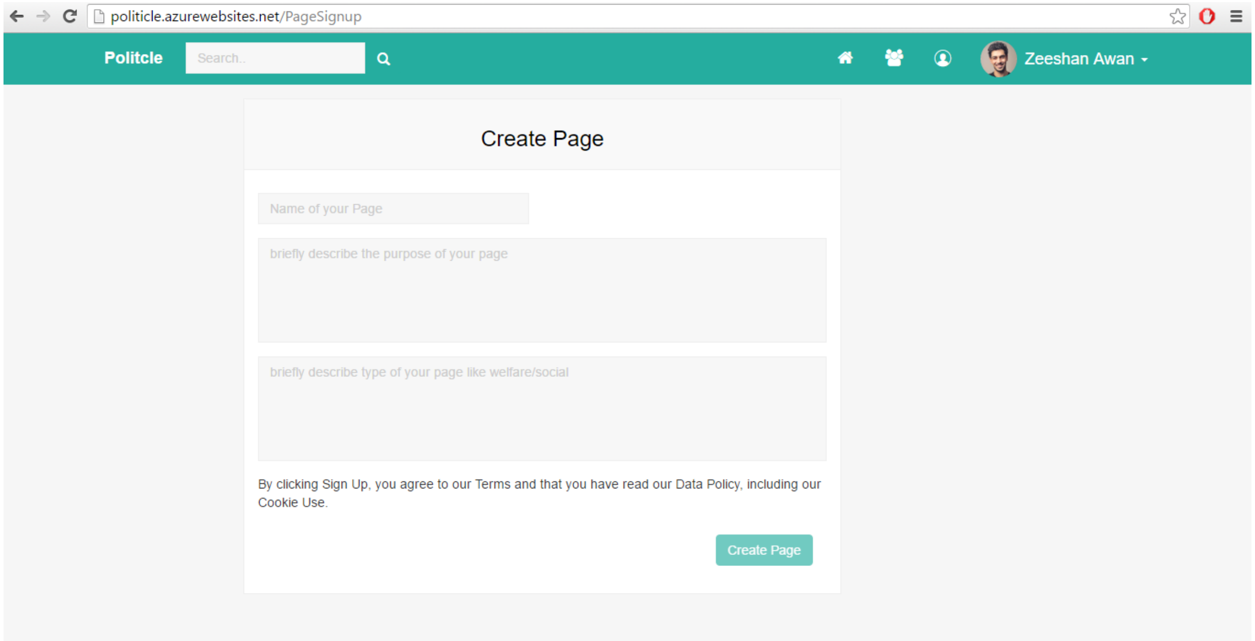
View all 0 comments



Filter: by Name Kamran Search

Search Result

| Date | Name | Gender | Email | Location | Progress | Action |
|---|---|---|---|---|---|---|
| 6/21/2016 | Kamran | Male | hassan@madfrequency.com |  |  | Follow |



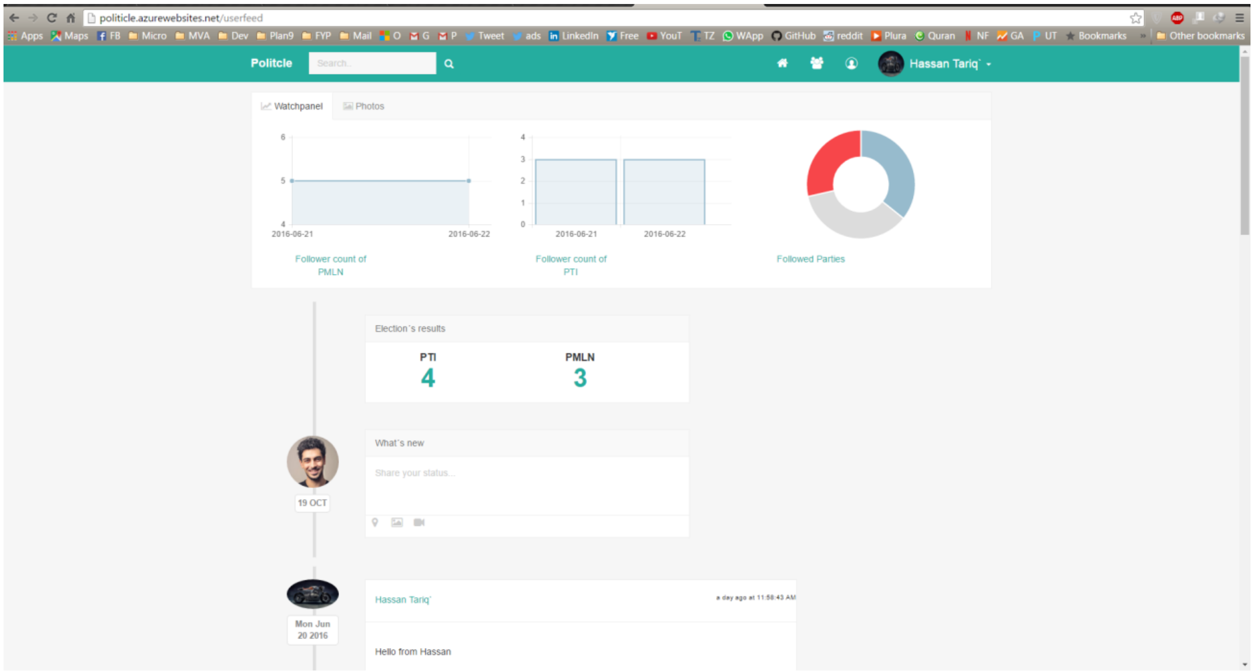
Create Page

Name of your Page

briefly describe the purpose of your page

briefly describe type of your page like welfare/social

By clicking Sign Up, you agree to our Terms and that you have read our Data Policy, including our Cookie Use.



Election´s results

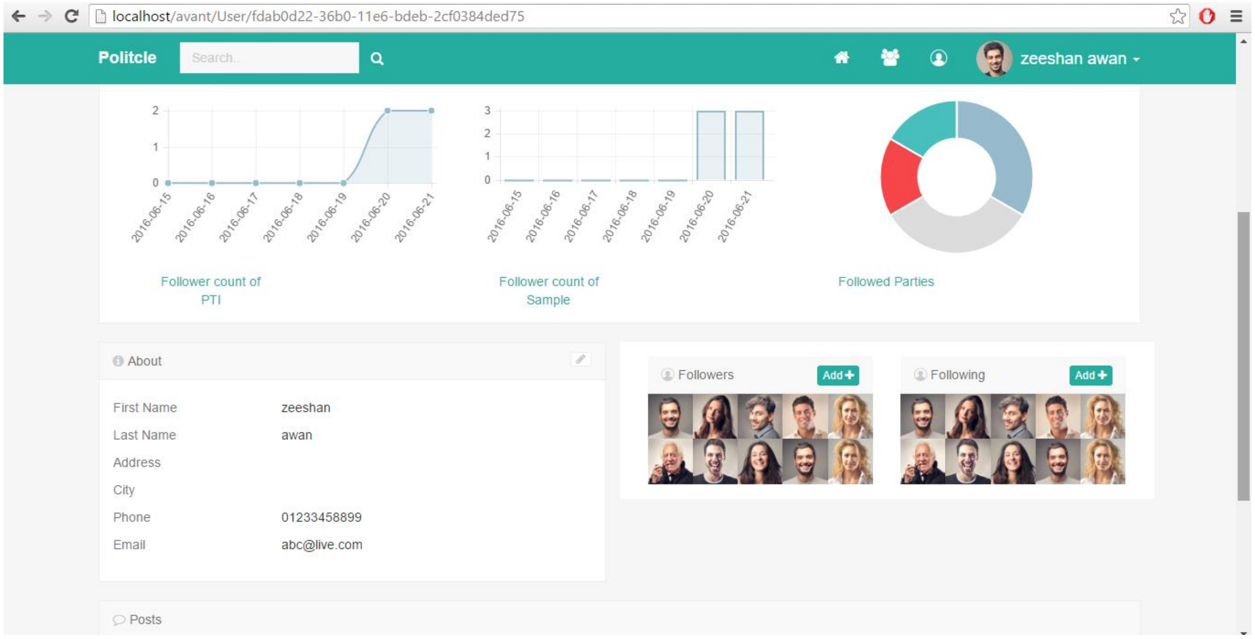
| PTI | PMLN |
|---|---|
| 4 | 3 |

What´s new

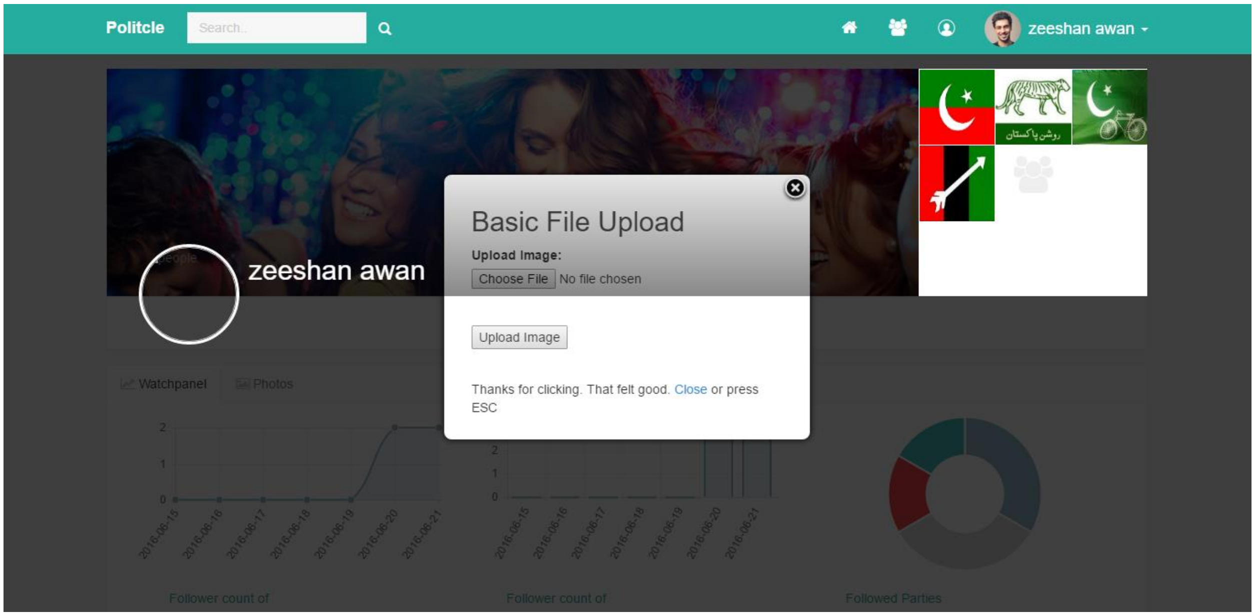
Share your status...

Hassan Tariq` a day ago at 11:58:43 AM

Hello from Hassan



| | |
|---|---|
| First Name | zeeshan |
| Last Name | awan |
| Address | |
| City | |
| Phone | 01233458899 |
| Email | abc@live.com |



Basic File Upload

Upload Image:

Choose File No file chosen

Upload Image

Thanks for clicking. That felt good. Close or press ESC

# 7. Android APP implementation

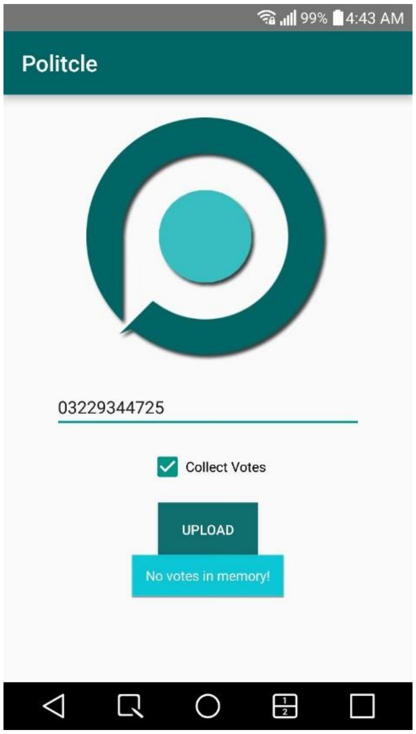
Android APP has been made for voting system through miss calls system.

## Use Case:

A user let say Ali wants to collects public vote for his cause. He will publically advertise his mobile number and install our app on his android phone.
When someone call his number, App will automatically reject call and save the incoming mobile number in memory.
Later on, Ali can upload all the votes to server along with their timings and details and can easily views status from our website.

- Android App is using phone memory to save incoming call numbers and their timing.
- It can be used with WIFI or Mobile Data connection to upload votes to server.
- User can view votes through web page with all details.
- App is supported with most of android phones.
- App can also be used for Election system to collect votes against many users.

# 8. Android APP views

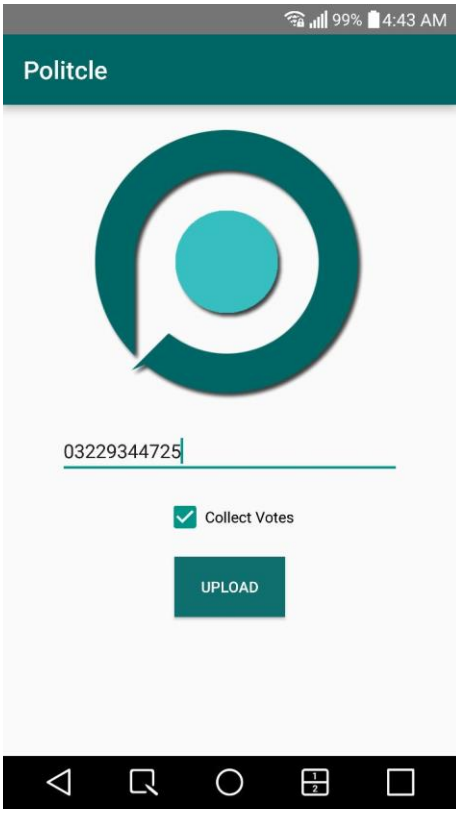


Politcle

03229344725

☑ Collect Votes

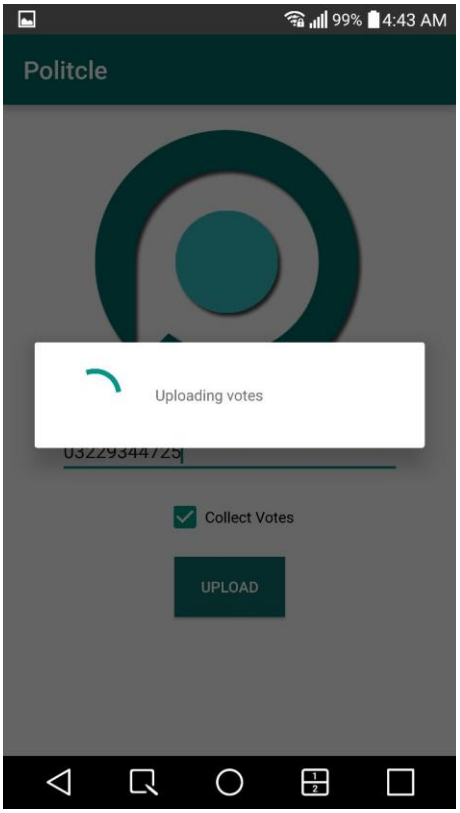
No votes in memory!



Politcle

03229344725

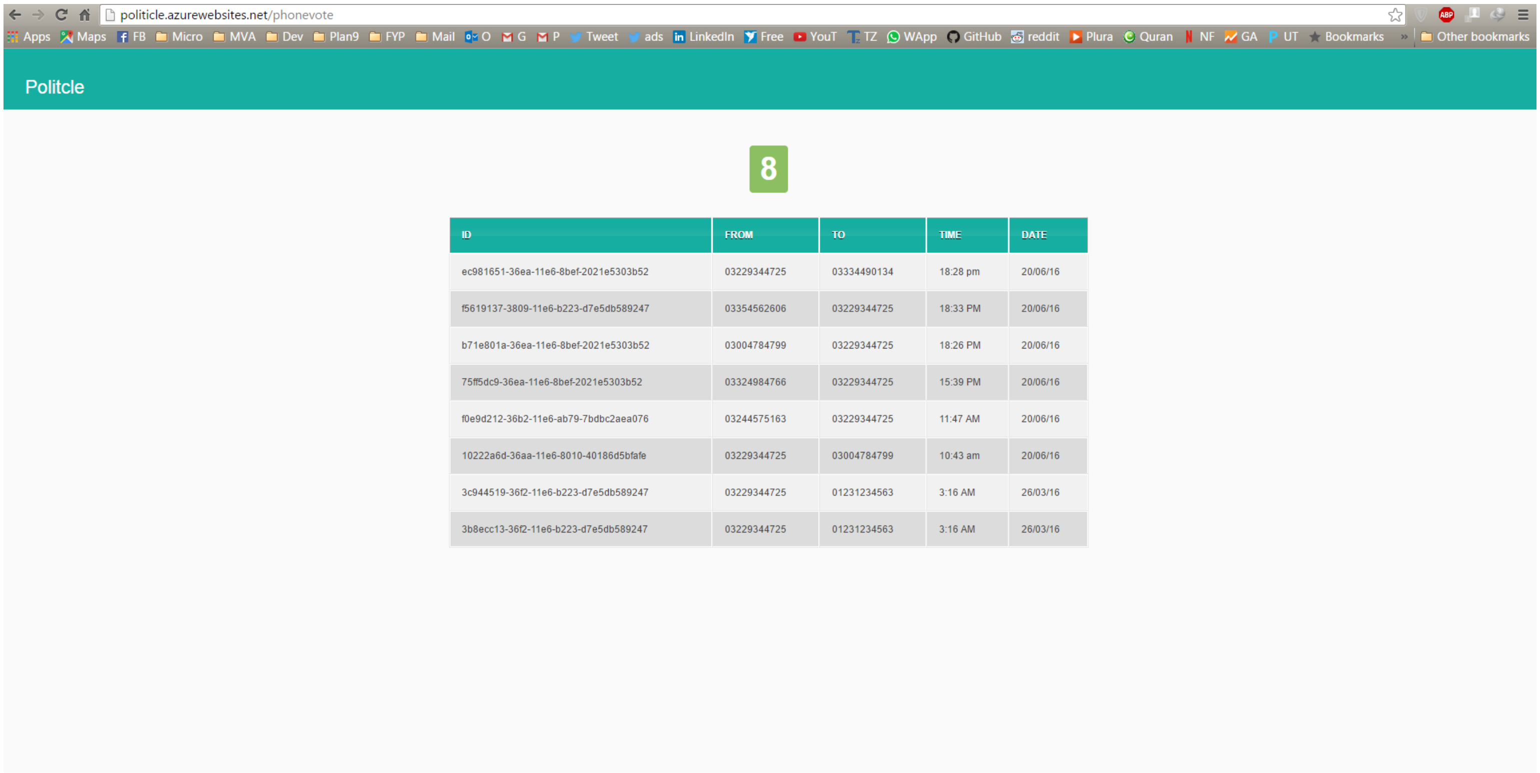
☑ Collect Votes



Uploading votes

03229344725

☑ Collect Votes



8

| ID | FROM | TO | TIME | DATE |
|---|---|---|---|---|
| ec981651-36ea-11e6-8bef-2021e5303b52 | 03229344725 | 03334490134 | 18:28 pm | 20/06/16 |
| f5619137-3809-11e6-b223-d7e5db589247 | 03354562606 | 03229344725 | 18:33 PM | 20/06/16 |
| b71e801a-36ea-11e6-8bef-2021e5303b52 | 03004784799 | 03229344725 | 18:26 PM | 20/06/16 |
| 75ff5dc9-36ea-11e6-8bef-2021e5303b52 | 03324984766 | 03229344725 | 15:39 PM | 20/06/16 |
| f0e9d212-36b2-11e6-ab79-7bdbc2aea076 | 03244575163 | 03229344725 | 11:47 AM | 20/06/16 |
| 10222a6d-36aa-11e6-8010-40186d5bfafe | 03229344725 | 03004784799 | 10:43 am | 20/06/16 |
| 3c944519-36f2-11e6-b223-d7e5db589247 | 03229344725 | 01231234563 | 3:16 AM | 26/03/16 |
| 3b8ecc13-36f2-11e6-b223-d7e5db589247 | 03229344725 | 01231234563 | 3:16 AM | 26/03/16 |

# 9. System testing

## 9.1 Introduction:

System testing and implementation is final phase of any system development life cycle. As a step in system development cycle, testing and implementation consists of final steps that put the new system into operation.

## 9.2 Forms and program testing

First the system deals with large number of state, complex logic activities. So some error might occur in the system. Errors may be in software, which is known as “software errors”, i.e. the software does not do what the requirement says. So an exhaustive and through testing must be conducted to ascertain whether the system produce right result.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of a system are correct, the goal of system design will be successfully achieved. Another reason for system testing is to check its utility as a user oriented vehicle before implementation. A program represents the logical elements of a system. For a program to run satisfactory, it must compile and test data correctly and tie in properly with other programs. The best program is worthless if it does not meet user need.

This system can be designed according to the requirements of the system and needs of the user. Yet complete accuracy cannot be claimed. So, an exhaustive and through testing of the system was conducted to a certain whether the produced the correct results.

This system can be designed according to the requirements of the system and needs of the user. Yet complete accuracy cannot always be claimed. So, an exhaustive and through testing of the system produced the correct results.

The testing has been done in several stages. First each program module was tested as a single program, which also known as module testing or unit testing. In unit testing a set of data as input was given to the module and observed what the output is. In addition, the logic and boundary condition for input and output data has also been checked. The interface between this module and others was also check for correctness. While collecting the input data for testing the module it is kept in mind that data should be from all the classes, so that entire condition of the program could be checked. In testing the roll of the user is also important, since they are right person can understand the full range of data and condition that might occur in the system. So a wide range of data was collected from the user to test the program thoroughly.

## 9.3 Integration and system testing

When individual program modules work properly, we combine the module into a working system. This integration is planned and coordinated so that an error occurs; we have an idea of what caused the former. Integration testing is a process of verifying that the component of a system work together as described in the program design and system specification.
For testing the entire system was viewed as a hierarchy of modules we began with the module at the highest level of design and worked down. Next modules to be tested are those that call previously tested modules.

In each of the individual program and forms were found to be working properly they were combined into a working module and then tested. The integration is planned and coordinated in such a way that when an error occurs it would give an idea of where and why it occurred. Integration testing is the process of verifying whether the components of a system work together as described in program design and system design specifications. The system developed was tested as a whole and found to be performing properly.

Our bitbucket repository helped us a lot in unit testing and then merging changes with the original version.

## 9.4 Functional testing

Once we are sure that information is passed among the module according to the design prescription we tested to ensure whether the function described the requirement specification are performed be the integrated system. System testing checks the readiness and accuracy of the system to access update and retrieve data from new files. Once the program becomes available, test data are read into the computer and process against the files for testing. If successful, program is run with live data. Else a diagnostic procedure is adopted to locate and correct the errors in the program. In most cases a "parallel run" is conducted where the new system runs simultaneously with old system.

## 9.5 Acceptance testing

When all the tests on the system were over the users were involved to make sure that the system was working according to the users' exceptions. Thus finally the user did acceptance.

## 9.6 Installation testing

When acceptance test was completed, the acceptance system was installed in the environment in which it will be used and a final installation test was performed to make sure, that the system would function, as it should. i.e its deployment on azure server.