**ELECTION SIMULATION**

**TECHNOLOGY MANIFESTO**

1. **TECHNOLOGIES**

*Web hosting*

**Windows Azure**

Free for to use within the scope of class.

Has great industrial capacity.

Strong community

*Web framework*

**Django web framework**

Client requires that the system will be able to runs on multiple platforms. Django is written in Python which can run on almost all popular operating system (Windows and UNIX like systems). Also Great Bear decided to make a website, so users can access easily from any operating system without special technical support.

*Database*

**Comma separated values (CSV)**

**SQLite 3**

Great Bear decided to convert all files to a CVS and then to SQLite for future immigration to a different database. Because SQLite is compact lightweight, portable database so it has a lot of benefit during early stage of development, and learning. However, because of being lightweight, later on, SQLite will have a lot of performance issues, so SQLite is used to be replaced.

Using a CSV as a buffer will simplify the future immigration procedures. Future development will have to only care about converting from a standardized CSV format to a new DB format.

*Repository*

**Github**

An industrial tool

*IDE*

**Pycharms community edition**

**Eclipse with PyDev plugin**

These tools have good reputations and community support.

These tools have good version control features

1. **FILE STRUCTURE**
   * ElectionSimulation : All the code for the website
     + ElectionSimulation : Main code for the website
       - CampaignMaster : The views file and url patterns at campaign master level
       - Models : The all models for the sites
         * models.py : Have import statement to include all the models for the site
         * <Model names> : Model files
       - static : All the static files for the website
         * bootstrap : All the js, css, image files for Twitter Bootstrap
         * css : All extra css files for the sites
         * js : All extra css files for the sites
       - templates
         * CampaignMaster : All the layout and templates for Campaign master section

layout.html : The layout for campaign master’s pages

list.html : The content for campaign master’s listing page

* + - * + layout.html : Main layout for the entire site
      * models.py

1. **MODELS DESIGN**

*Note: Remember to run “****python manage.py syncdb****” to create the DB file before running the server*

* + **Campaign**

ID : the id of a campaign – *This is made by the framework*

name : the name of the campaign

create\_date : The date of creation

candidates : List of all candidates and their party separated by new line character

voting\_system : The name of the voting system

formula : The foreign key to the formula

* + **Formula**

ID : the id of a formula– *This is made by the framework*

name : the name of the formula

* + **PoliticalParty**

ID : the id of a political party– *This is made by the framework*

name : the name of the political party

* + **Riding**

ID : the id of a riding– *This is made by the framework*

name : the name of the riding

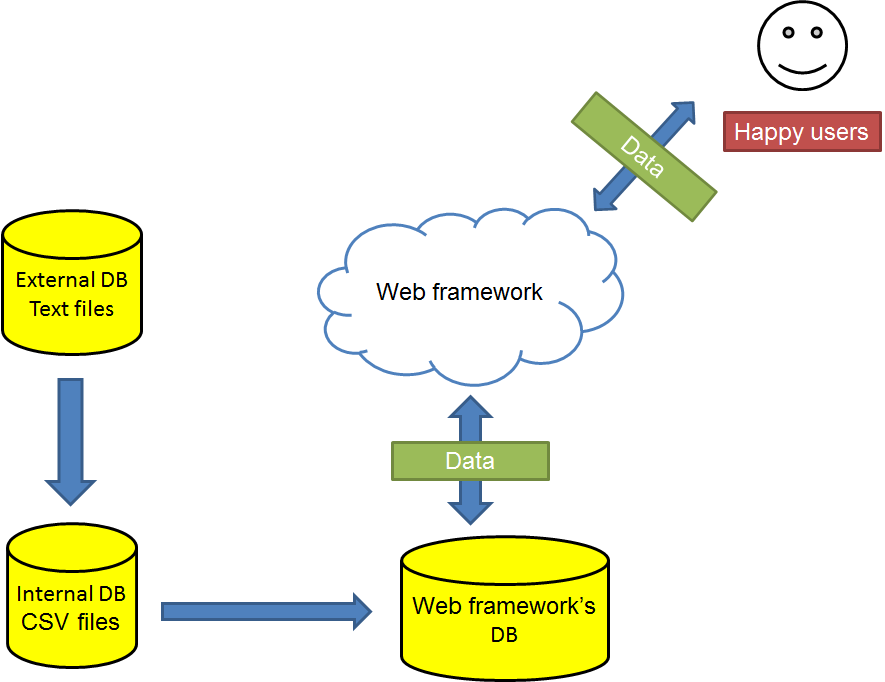
1. **FLOW OF EXTERNAL DATA**

Election data will be collect from external sources. All the external data will be store in

Then all the data will be converted into an internal format in CSV files.

Then all the data will be converted into web framework’s database.

The web framework will use those data for running simulations accordingly to user needs.



*Figure 1: Overview of the flow of data*

1. **URL LISTING:**

**/campaign\_master/**

**GET**: Show the list of campaigns created

**/campaign\_master/add\_campaign/**

**GET**: Show the form to create a new campaign

**POST**: Where to submit data to create a new campaign