|  |  |  |
| --- | --- | --- |
| Team Canada | 2012 | |
| Group project for Enterprise Frameworks , pgCloud | |  |

Contents

[1. Project Team 3](#_Toc330563344)

[2. Introduction 4](#_Toc330563345)

[3. Background Research and Investigations 5](#_Toc330563346)

[4. Project Plan 6](#_Toc330563347)

[5. Datasets Used in Project 7](#_Toc330563348)

[6. Software Development Methodology Employed 8](#_Toc330563349)

[7. Requirements Analysis 9](#_Toc330563350)

[a. Functional Requirements 9](#_Toc330563351)

[b. Non Functional Requirements 9](#_Toc330563352)

[8. Use Cases 10](#_Toc330563353)

[9. Generate Data Model 14](#_Toc330563354)

[10. Identify outputs for the user – describe use cases Generate Data Model 15](#_Toc330563355)

[11. Typical User Queries 16](#_Toc330563356)

[12. Data Model 17](#_Toc330563357)

[13. Populate the database with datasets 18](#_Toc330563358)

[14. Decide what data manipulation is required – (C#) 19](#_Toc330563359)

[15. Design interface to the database – (ASP MVC) 20](#_Toc330563360)

[16. Frontend design (ASP MVC) 21](#_Toc330563361)

[17. Architecture/Design Approach 22](#_Toc330563362)

[18. Models 23](#_Toc330563363)

[19. Implementation of Particular OOP constructs 25](#_Toc330563364)

[20. Design Patterns and Architectural Patterns Implemented 26](#_Toc330563365)

[21. How cross cutting concerns have been handled 27](#_Toc330563366)

[22. Security of Application 28](#_Toc330563367)

[23. Configuration of Application 29](#_Toc330563368)

[24. Scalability of Application 30](#_Toc330563369)

[25. Testing Approach 31](#_Toc330563370)

[26. References 32](#_Toc330563371)

[Appendix A – Project Management 33](#_Toc330563372)

## Project Team

Fintan Costello – X11106441

Darragh Breathnach – X11106417

Alan McCormack – X11102616

Eoin Ó Loideáin – X11109513

## Introduction

The aim application is to allow users to log into a website, and build custom reports based on data pertaining to campaign contributions and votes received during the 2006 and 2010 local elections in Toronto, Canada.

The two main functions of our web application are as follows:

1. Users will be able to filter reports by a number of criteria, and their reports can be automatically saved on their user account, downloaded immediately as a CSV file, PDF document, MS Excel file or simply outputted to the screen
2. Users can run an analysis on the election results and contributions data to determine the relationship between the two variables.

The application is hosted in Microsoft Azure. ??

## Background Research and Investigations

To start the project the acquisition of a comprehensive dataset was required. We were looking for a good quality open dataset that required a minimum amount of ‘cleaning’. With the growing trend towards open data, one obvious possibility was datasets published by government organisations. Exploration began with the Dublin open datasets published at www.dublinked.ie , however initial investigation revealed that much of the datasets here are very unstructured and the data itself is ‘noisy’ and would require substantial effort to clean up for use as the basis of our web application.

Further investigations led us to a Canadian government website; www.toronto.ca where we acquired several large datasets, containing relatively well formatted data that could be parsed and manipulated with a minimum need for cleaning. Several datasets were explored and the contributions and the election results were finally selected as there was a convenient overlap in the two datasets that would allow us to join the two datasets together.

w3schools and the .net website were utilised to get a basic understanding of C# and MVC projects. Sample projects were created here touching on a number of relevant are as to the application. The class tutorial for ingesting the CSV was completed and then modified for use in the team Canada context.

DATA

1. Acquire a dataset
2. Parse & filter the data, top clean up irregular/noisy data
3. Explore data, to see what is available/ what use cases it affords

PROGRAMMING

1. Lecture notes/ lessons learnt
2. Online tutorials
3. Existing web apis
4. TDD

## Project Plan

The plan was to create a web application that would allow easy access to the cleaned up datasets to allow for customised reports, and to also provide a basic data analysis tool.

As there was a tight deadline and a steep learning curve, the project was immediately split into tasks and assigned to team members as shown in Appendix A, using a simple MS Excel Todo list and also Github.

Github reports

## Datasets Used in Project

The datasets used to be used is a breakdown of campaign contributions and election results for the following elections in Toronto, Canada:

* Mayoral Race 2006, 2010
* Council Elections, 2006, 2010

All data is taken from the Open Data Initiative in Toronto, located at <http://www.toronto.ca/open>. The data is in CSV format.

* Scale of the dataset – size
* Final CSV structure & size

Size of original files

1 MB

1.5 MB approx

1.8

2.3

## Software Development Methodology Employed

Test-driven development is an agile development technique used by team Canada. It relies on a simple, iterative process that ensures that all the code in the finished project has been tested and it instils confidence that all the code in the completed application should behave as it was intended it should behave.



*Fig.1 Test-driven development cycle (source: Wikipedia)*

The TDD process is illustrated in Fig.1 above and adheres to the following simple procedure:

1. Write a failing test to prove that a functionality or code is missing from the completed code.
2. Write code or correct it to allow the test to pass
3. When it passes, the code can be cleaned up or refactored to increase readability and to avoid duplication.
4. Start process again and write a test for the next feature requirement.

## Requirements Analysis

### Functional Requirements

* Parse data
* Read data into database
* User account registration
* User login
* User report selection
* Generate/Display report

The user interaction with the application will be as follows:

* User will log into to application; if it is their first access then they will be prompted to register
* Sample reports are available for all users
* User is given a brief explanation of the datasets.
* User builds custom reports by selecting from dropdown menus or similar, filtering on the election type, candidate name, ward number, and so on
* The report is built for the user and displayed on screen. Options are given to export the report as a PDF or spreadsheet.

All user-generated reports will be stored in the database and will be accessible on subsequent logins.

### Non Functional Requirements

## Use Cases

Defining Use cases was our starting point for creating our application; they acted as a guide for generating tests for initiating our test-driven development process. They act to identify and to clarify our requirements. By creating specific use cases for our requirements, it makes them easier to read and to track. The purpose is not to design a UI, it is to literally describe how our application should work.

**Use case #1**

Name: Creating a Report  
Actors: User and API

Summary: The user can select from one of a number of preset reports

Pre-conditions: Data in the system & user has chosen a report

Flow of events: 1. Presentation layer serves form for user to make report selection  
 2. Data inputted by user is caught and sent to

Error conditions: No data in DB, no selections made by user

Post-conditions: No change? or Log of client’s IP address & time stamp created.

**Use case #2**

Name: Data analysis – Votes vs. Contributions  
Actors: User & API

Summary:

Pre-conditions:

Flow of events:

Error conditions:

Post-conditions:

<http://aspalliance.com/765_Use_Cases_and_Their_Importance.all>

The typical report building sequence will be:

1. Select a specific election (i.e. 2010 city council election)
2. Filter by one of the following:
   1. Candidate name
   2. Contributor name
   3. Post Code or Ward Name
   4. Amount of Donation
   5. Contribution type (personal, services, etc.)
3. Create report and display on screen
4. Export report in alternative format

E.g.:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Create a Report** | | | |  |  |  |
|  |  |  |  |  |  |  |
| **1.** | Select an election: | | | |  | | --- | |  | |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **2.** | Filter by: | |  | |  | | --- | |  | |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **3.** | Export as: | |  | |  | | --- | |  | |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  | submit button | |  |
|  |  |  |  |  |  |  |

When filtering by contributor, the following conditions will apply:

* If the contributor is an individual, no postal address will be returned.
* If the contributor is an organisation, a postal address will be returned.

**Business Logic Description**

Report #1:

* sum up the contributions & calculate the Largest total contribution
* sum up the contributions & calculate the Smallest total contribution
* sum up the contributions & calculate the Average total contribution
* sort candidates returned, Alphabetically
* sort candidates returned, from Largest to Smallest

Report #2:

* sum up the contributions & sort
* sum up the contributions & sort by Ward No., use lookup file for Postcode V Ward No.
* calculate average per Ward

Report #3:

* sum up the votes & sort
* sum up the votes & sort by Ward No., use lookup file for Postcode V Ward No.
* calculate average per Ward

**Report #4 – Election Results analysis**

Using a “Least Squares Regression method”, the total number of votes and contributions per candidate will be analysed to determine approximately how much in contributions it takes to achieve a given number of votes. The analysis can also be reversed to determine the number of votes to expect for a given level of contributions. We will allow the user to analyse and compare data from the two different year’s results to ascertain what the difference was and to help make predictions on the next election results.

|  |  |
| --- | --- |
| Total No. Votes |  |

Total Contributions ($)  
  
*Chart 1. – Analysis of Total Votes vs. Total Contributions, for Toronto Mayoral Elections 2006*

Chart 1., is a scatter plot depicting how many votes each candidate achieved along with the corresponding total value of contributions received by that candidate. By using the least squares regression analysis, a ‘best fit’ line can be plotted through our given data points. It allows us to predict what number of votes can be expected if a certain level of contributions is achieved and vice versa.

The Least Squares Regression model is based on the following model:

|  |  |  |  |
| --- | --- | --- | --- |
| slope = m | = | |  | | --- | | n(Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/SIG.GIFxy) - (Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/SIG.GIFx)(Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/SIG.GIFy) Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/FR.GIF n(Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/SIG.GIFx2) - (Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/SIG.GIFx)2 | |
|  |  |  |

And ;

|  |  |  |  |
| --- | --- | --- | --- |
| intercept = b | = | |  | | --- | | Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/SIG.GIFy - m(Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/SIG.GIFx) Description: http://people.hofstra.edu/stefan_waner/calctopic1/SYMB/FR.GIF n | |

Therefore for any given x-coordinate a corresponding ‘best fit’ y-coordinate can be calculated using the simple formula:

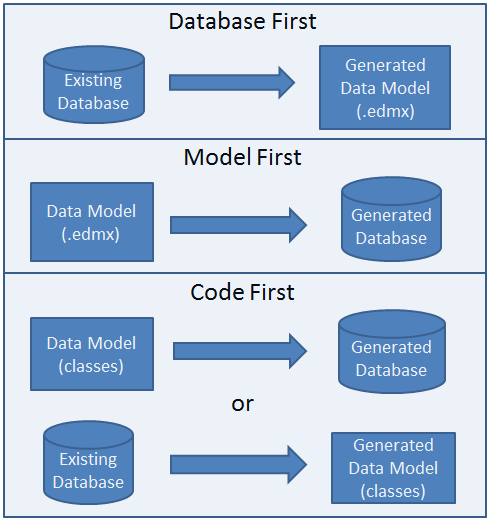
y = mx + b

Practically what this means with from our Business Logic model perspective is that when a user asks for instance what approximate value of contributions is required to achieve a certain amount of votes:

1. The view model will need to pass the query to the DAL,
2. The DAL will query the database to return all the votes received by all the candidates, and all the contributions received by all the candidates,
3. The DAL will pass the response to the BLL for processing , where it will sum up all the votes per candidate and associate this list of values with the sum of all the contributions per candidate.
4. A calculation will them be carried out as per our least squares model above and the result will be returned to the user via the presentation layer.

## Generate Data Model

<http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application>

`

**Plain Old** [**CLR**](http://en.wikipedia.org/wiki/Common_Language_Runtime) **Object POCO**

Using Linq Builder, we created a list of data models based on the data contained in our datasets and generated the relationship between them. Models created;

* Candidate: Candidates name, number of votes and total contributions received.
* Contribution: Contribution amount, type of contribution,(cash or goods/services), type of contributor, (private/corporate), candidate’s ID and contributor’s ID.
* Contributor: Contributor’s name, post code and relationship to candidate.
* Election: Election type (mayoral/council), the year, total number of votes and total number of candidates.
* Ward: Ward number and name, the number of candidates in that ward, the number of votes cast in that ward, and the total contributions in the ward.

## Identify outputs for the user – describe use cases Generate Data Model

Drop this – same as section 8.??????

## Typical User Queries

Is this frontend design => not really required

## Data Model

Drop this - Same as section 9. Generate data

## Populate the database with datasets

As part of the business logic of the application, the data, contained in csv files, will be read into the database. This data is currently clean, although prolific. Some data normalisation will take place to integrate postal code areas and wards. This is necessary in eliminate further problems down the line.

C# parses are required to read the csv files into the database. The parsers from the sample project created in class had been modified to meet our requirements.

Code snippet:

[insert code here]

## Decide what data manipulation is required – (C#)

Data manipulation may be limited, due to the nature of the existing data, i.e. clean.

The following manipulations may be required:

* Calculations might be made once the data is extracted e.g. get total sum of contributions made to a specific candidate
* Calculate what the winning candidate had to spend in 2006 compared to 2010, & adjust both figures for inflation.
* Calculate the sum totals by contributors or by postcode area – can we cross reference this against any other dataset/register – see if we can identify possible lobbyists.
* Calculate total contributions for a given set of post codes by summing, & then calculate the maximum value & the minimum value, then calculate the appropriate radius for a circle to be placed on a map at the coordinates of the postcode, the radius being proportional to the value of the contribution – the results could be used for a data-visualisation
* Maybe look at adding data from Canadian census to get a demographic profile for a given ward, e.g. ethnic/age/household/income/education/poverty rates – there might be something here: <http://datalib.chass.utoronto.ca/codebooks/cstdli/pccf_how.htm> and also here:  
  <http://www.toronto.ca/wards2000/ward1.htm>
* This whole section is carrying out arithmetical type calculations on the returned data before releasing it for output – the business logic

Is this not Business logic ???????

## Design interface to the database – (ASP MVC)

* Interface required allowing the website Admin to load the raw initial dataset into the database, similar to attachment to an email. This will be a rare/once off function
* Admin will require a login but not the API users, perhaps they may need an API key for access & logging usage – this might complicate things though?
* Simple CRUD functionality required for Admin

Is this backend UI???

## Frontend design (ASP MVC)

(I think we don’t need to get too complicated here, to start with at least this is going to demonstrate that we can extract specific chunks of data from the DB using the API – so simply a list of report choices, e.g. “Print table of what candidate received the most contributions” & tick box for format; CSV, PDF or HTML table.)

* Login screen – do we need login?
* New user registration screen
* Home screen with list of stored reports
* Report query screen
* Report output screen

## Architecture/Design Approach

* 1. C#
  2. MVC4
  3. ASPX

## Models

Toronto is divided into 44 electoral wards. Each ward is has 42 subdivisions, numbered 1-34 and then 93-99.

***Mayoral Election 2006 / 2010***

* Each candidate runs in all 44 city wards and an individual vote count for each ward and each subdivision are given. Totals for each subdivision and an overall total are given
* Campaign contributions are not listed with a ward number as the candidate runs in all wards
* Campaign contributors are listed by postcode, amount donated, contribution type (i.e. cash) and candidate donated to

***Council Elections 2006 / 2010***

* Each candidate runs in 1 ward and an individual vote count for that ward and each subdivision are given. Totals for each subdivision and an overall total for that ward are given
* Campaign contributors are listed by postcode, ward number, amount donated, contribution type (i.e. cash) and candidate donated to

***Sample Reports to be created***

1. A report which maps the campaign contributions to the Mayoral and Council races using the postcode as key identifier – i.e. which areas contributed most money during the election campaign? (raw data and heat map)
2. A report which shows the correlation between the total amount of contributions made and the number of votes cast for a specific candidate – i.e. does a higher level of campaign contributions translate into more votes for that candidate? (raw data)
3. A report which shows the increase or decrease of campaign contributions per ward from 2006 to 2010 – i.e. how has the amount of money donated to candidates change from one election to the next? (raw data, bar chart)

(For this data modeling section I think we need to also look at data types and relationships in this section e.g. Zip code is a string, long/lat is a float, candidates has one to many relationship to contributors )

Should this be before DATA MODEL section ?????

## Implementation of Particular OOP constructs

???????????

## Design Patterns and Architectural Patterns Implemented

Domain Model

Write shite on the domain model architectural pattern. Take it from the notes.

???????????

## How cross cutting concerns have been handled

Information in notes...........

???????????

## Security of Application

User Registration

User login

Azure.......

Built in ???????

## Configuration of Application

## Scalability of Application

If we go to azure..........

Ability to scale csv files........

## Testing Approach

Test driven development (TDD) was applied to this project. TDD encourages simplicity and assures that any completed code behaves as it expected to behave and also helps to avoid duplication and a long drawn out debugging exercise at the end of the project. A test project was created at the beginning of the project development and as much as possible, testing was carried out, before any code was written. As coding progressed, the initial test fails were resolved. With each new addition or functionality that was added to the code, a unit test was first written and then run, with the associated test failures being resolved. In some cases coding had preceded the unit test but this approach avoided as much as possible.

ASP.NET MVC provides a substantial set of in-built testing tools that provide a framework for writing unit tests. Two broad sets of tests were written, one for the CSV parser that ingested our raw data and one for the business logic layer. Testing was carried out on all associated classes and methods.

For the parser tests the following cases were tested:

1. Test for null values
2. Test for empty strings i.e. “”
3. Test for strings and for integers and floats
4. Test for different numbers of fields in a given CSV line
5. Test for performance & profiling?????

For the Business Logic the following cases were tested:

## References

1. 25.Testing approach:
2. <http://artofunittesting.com/storage/chapters/SampleChapter1.htm>
3. Use case 4:
4. <http://people.hofstra.edu/stefan_waner/calctopic1/regression.html>):

## Appendix A – Project Management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Team Canada | | | |  |
| To-do List | | | |  |
|  |  |  |  |  |
| **Task No.** | **Task** | **By** | **% Complete** | **Date** |
| **A** | **Documentation** |  |  |  |
| 1 | Proposal | all | 85% |  |
| 2 | Design Use cases/sample reports layout | EÓL | 100% | 02/07/2012 |
| 3 | Build lookup file-postcode V ward # | EÓL | 100% | 02/07/2012 |
| 4 | Update main document with use cases | EÓL | 100% | 02/07/2012 |
| 5 | Write formal intro and content | DB | 10% | 27/07/2012 |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| **B** | **General** |  |  |  |
| 1 | Install MVC 4 | all | 100% | 09/06/2012 |
| ~~2~~ | ~~Install Github plugin for Visual Studio~~ | ~~all~~ |  | ~~n/a~~ |
| 3 | Create MVC4 Project and upload to Git | FC | 100% | 14/06/2012 |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| **C** | **Data Modelling** |  |  |  |
| 1 | Create Data model in RISE | AMC | 100% | 13/06/2012 |
| 2 | Generate DB code from RISE | FC | 100% | 14/06/2012 |
| 3 | Use LINQ builder to create schema | FC | 100% | 14/06/2012 |
| 4 | Identify Excel parser | FC | 100% | 04/07/2012 |
| 5 | Write parsers for elec. and contrib. | FC | 100% | 04/07/2012 |
| 6 | Tidy up data formatting in Excel | FC | 100% | 04/07/2012 |
| 7 | update election data | EÓL/DB | 100% | 05/07/2012 |
| 8 |  |  |  |  |
| **D** | **Web API** |  |  |  |
| 1 | Get simple Web API working | EÓL | 100% | 09/06/2012 |
| ~~2~~ | ~~Adding CRUD functionality~~ | ~~EÓL~~ | ~~0%~~ | ~~n/a~~ |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| **E** | **MVC** |  |  |  |
| 1 | Design controllers | AMC |  | 07/07/2012 |
| 2 | Design views | FC |  |  |
| 3 | Design routes |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| **E** | **Test Driven Development** |  |  |  |
| 1 | Test Driven Development Demo | AMC |  | 07/07/2012 |
| 2 | Update test section of document | AMC |  | 07/07/2012 |
| 3 | Unit Test Suites | EÓL |  | 14/07/2012 |
| 4 |  |  |  |  |
| **E** | **Server** |  |  |  |
| 1 | Setup Azure account | FC | 100% | 13/06/2012 |
| 2 | Install Azure SDK | all | 100% |  |
| 3 | Add new Azure website | FC | 100% | 14/06/2012 |
| 4 | Investigate data storage on Azure | DB |  | 07/07/2012 |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |

**Github Report?????**

Github repository – Darragh

Create new project application in visual studio – Fintan

Review datasets – Eoin, Fintan, Alan

Data normalisation – Eoin, Darragh, Fintan

Create data model – Alan, Fintan

Use cases – Alan

Test cases – Darragh, Alan

Project Report – All

Presentation – All