Contents

[1. Project Team 3](#_Toc330197377)

[2. Introduction 3](#_Toc330197378)

[3. Background Research and Investigations 3](#_Toc330197379)

[4. Project Plan 3](#_Toc330197380)

[5. Datasets Used in Project 4](#_Toc330197381)

[6. Software Development Methodology Employed 4](#_Toc330197382)

[7. Requirements Analysis 4](#_Toc330197383)

[8. Use Cases 4](#_Toc330197384)

[9. Generate Data Model 5](#_Toc330197385)

[10. Identify outputs for the user – describe use cases 5](#_Toc330197386)

[11. Typical User Queries 5](#_Toc330197387)

[12. Testing 7](#_Toc330197388)

[13. Data Model 9](#_Toc330197389)

[14. Populate the database with datasets 10](#_Toc330197390)

[15. Decide what data manipulation is required – (C#) 11](#_Toc330197391)

[16. Design interface to the database – (ASP MVC) 11](#_Toc330197392)

[17. Frontend design (ASP MVC) 12](#_Toc330197393)

[18. Architecture 12](#_Toc330197394)

[19. Architecture/Design Approach 13](#_Toc330197395)

[20. Models 13](#_Toc330197396)

[21. Implementation of Particular OOP constructs 13](#_Toc330197397)

[22. Design Patterns and Architectural Patterns Implemented 13](#_Toc330197398)

[23. How cross cutting concerns have been handled 13](#_Toc330197399)

[24. Security of Application 13](#_Toc330197400)

[25. Configuration of Application 13](#_Toc330197401)

[26. Scalability of Application 13](#_Toc330197402)

[27. Testing Approach 13](#_Toc330197403)

## Project Team

Fintan Costello – X11106441

Darragh Breathnach – X11106417

Alan McCormack – X11102616

Eoin O’Loideain – X11109513

## Introduction

The application will allow users to log into a website, and build custom reports based on data pertaining to campaign contributions during the 2006 and 2010 local elections in Toronto, Canada. Users will be able to filter reports by a number of criteria, and their reports will be automatically saved on their user account. Generated reports will be presented in the user’s chosen fashion. This may include output to screen, export to files, or specific user generated request.

Some sample reports will be made available to all users demonstrating the possibilities of the application. These reports will be outputted to screen.

The application is hosted in Microsoft Azure.

## Background Research and Investigations

Our initial aim was to find a comprehensive dataset to work on. We were looking for quality open data, possibly from government organisations. From previous project experience, Dublinked was a good place to start. Further investigations led us to Toronto.ca. Here we found what we needed in terms of a large dataset, with content that could be parsed, manipulated and presented in a user specified format.

We utilised [www.w3schools.com](http://www.w3schools.com) for a basic understanding of C# MVC projects. Sample projects can be created here touching on a number of relevant areas to the application.

In class sample project demonstrations were followed to assist with our programming competency.

## Project Plan

Create 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

## Datasets Used in Project

The dataset 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 excel format.

## Software Development Methodology Employed

Agile...............

## 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

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

## Generate Data Model

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.

## 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

## 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

## 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 )

## Implementation of Particular OOP constructs

## Design Patterns and Architectural Patterns Implemented

Domain Model

## How cross cutting concerns have been handled

## Security of Application

User Registration

User login

Azure.......

## Configuration of Application

## Scalability of Application

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

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

## Testing Approach

A test project has been created and testing will be carried out concurrently with the development of the application.

The project will follow a Test-Driven Development approach where unit tests are developed at an early stage and are used as the basis for the further development of the functionality and business logic. ASP.NET MVC provides a substantial set of in-built testing tools that provide a framework for writing unit tests.

Here are a few boundary cases that you should have thought through and have test cases for.

1.Basic field. ,foo,

2. Basic quoted field. ,"foo",

3.Quoted field with embedded newline. ,"foo\nbar"

4.Quoted field with embedded comma. ,"foo,bar"

5.Quoted field with embedded quote. ,"foo""bar"

6.Do you distinguish between empty strings and nulls? If you do then ,, should be a null and ,"", should give an empty string.

7.Do you try to detect data types and do the right thing? CSV is often used for numerical data. Add whatever tests you think appropriate for that.

8.If you write data, you should cover all of the above cases.

9.What do you do with lines with different numbers of fields? (Test it.)

10.What do you do with trailing blank lines? (Test it.)

11.How is performance on a large file? (Test it. I've seen too many homegrown CSV parsers that use strings inefficiently and as a result take quadratic time, leading to simple stuff becoming painfully slow.)

Definition: Unit Testing

A unit test is an automated piece of code that invokes the method or class being tested and then checks some assumptions about the logical behavior of that method or class. A unit test is almost always written using a unit-testing framework. It can be written easily and runs quickly. It’s fully automated, trustworthy, readable, and maintainable.

The technique of test-driven development is quite simple:

1 Write a failing test to prove code or functionality is missing from the end product.  
The test is written as if the production code were already working, so the test failing means there’s a bug in the production code. For example, if I wanted to add a new feature to a calculator class that remembers the LastSum value, I would write a test that verifies that LastSum is indeed a number. The test will fail because we haven’t implemented that functionality yet.

2 Make the test pass by writing production code that meets the expectations of your test.  
It should be written as simply as possible.

3 Refactor your code.

When the test passes, you’re free to move on to the next unit test or to refactor your code to make it more readable, to remove code duplication, and so on.

<http://artofunittesting.com/storage/chapters/SampleChapter1.htm>