

# **Project Management Plan**

## **Electronic Voting System**

**Group C**

**Pádraig Doheny,**

**Adrian Duffy,**

**David Mills,**

**Stephen Morrissey,**

**Brian Twomey.**

## **Contents:**

### **1 INTRODUCTION**

- 1.1 Document Identification
- 1.2 Project Overview
  - 1.2.1 Customer Requirements
  - 1.2.2 Learning Aspect
- 1.3 Project Deliverables
- 1.4 Definitions, Acronyms and Abbreviations
- 1.5 Evolution of the PMP

### **2 OVERALL PLANNING**

- 2.1 List of Deliverables
  - 2.1.1 End-User Deliverables
  - 2.1.2 System Deliverables
- 2.2 Phase Plan
- 2.3 Organisation
  - 2.3.1 Organisation of People
- 2.4 Organisational Boundaries and Interfaces
- 2.5 Risk Management
  - 2.5.1 The Risks
  - 2.5.2 The Planned Solutions

### **3 BASELINE INITIATION PHASE**

- 3.1 List of Deliverables
- 3.2 Activity List
- 3.3 Skills
- 3.4 Non-Human Resources Needed
- 3.5 Quality
- 3.6 Information

### **4 DETAILED PLANNING DEFINITION PHASE**

- 4.1 List of Deliverables
- 4.2 Activity List
- 4.3 Skills
- 4.4 Non-Human Resources Needed
- 4.5 Quality
- 4.6 Information

### **5 DETAILED PLANNING DESIGN PHASE**

- 5.1 List of Deliverables
- 5.2 Activity List
- 5.3 Skills
- 5.4 Non-Human Resources Needed
- 5.5 Quality
- 5.6 Information

### **6 GLOBAL PLANNING REALISATION PHASE**

- 6.1 List of Deliverables
- 6.2 Activity List

- 6.3 Skills
- 6.4 Non-Human Resources Needed
- 6.5 Quality
- 6.6 Information

## **7 SOFTWARE QUALITY ASSURANCE PLAN**

- 7.1 Purpose
- 7.2 Scope
- 7.3 Management
- 7.4 Risk Management
  - 7.4.1 The Risks
  - 7.4.2 The Planned Solutions
- 7.5 Documentation
  - 7.5.1 System Documentation
  - 7.5.2 User Documentation
  - 7.5.3 Project Documentation
  - 7.5.4 Tools and Procedures
  - 7.5.5 Page Layout
  - 7.5.6 Document Layout
  - 7.5.7 Users of Documentation
- 7.6 Software Standards
- 7.7 Reviews and Audits
- 7.8 Software Verification and Validation Plan
  - 7.8.1 General Test Approach
- 7.9 Configuration Management
  - 7.9.1 Introduction
  - 7.9.2 SCM Management
  - 7.9.3 SCM Activities
  - 7.9.4 Tools, Techniques and Methodologies

## **APPENDICES**

- A. Survey of peoples thoughts on electronic voting
- B. Risk Management – System Vulnerabilities
- C. Requirements
- D. System Design
  - 1 Basic Design
  - 2 Detailed Design
- E. Future Developments
- F. Learning Aspect

# 1 Introduction

The Project Management Plan (PMP) is a plan which guides a software project from the start until a successful finish. Some chapters of the PMP will be final while others will be updated or added during the development cycle. The planning of the PMP is also part of the PMP. In the final document, this planning becomes section 4 in the PMP. The PMP is intended to help the group throughout every aspect of the project. It will aid decision making and help in allotting time to particular tasks which need to be done as well as helping us to prioritise the more crucial aspects (just in case we need to drop or downgrade some of our initial plans).

## 1.1 Document Identification

The reference name of this document is “*Project Management Plan for an Electronic Voting System*”.

Document date: 05/11/2003

Authors:

1. Pádraig Doheny,
2. Adrian Duffy,
3. Stephen Morrissey,
4. David Mills,
5. Brian Twomey.

## 1.2 Project Overview

This PMP has been written for the following project:

### **An Electronic Voting System**

The intended audience is:

- Students of CS3000.
- Course leaders.

This PMP serves as a study project of the CS3000 course. The main goal of the group in this course is to learn the skills to develop a large scale software project.

### **1.2.1 Customer Requirements:**

The client wishes to implement an Electronic Voting System suitable for use in national elections. This system will need to adhere to all the current electoral laws. It will need to be able:

- To identify registered voters and conversely exclude ineligible ones.
- To cater for referendum as well as general elections.
- To calculate the results of the vote quickly and accurately.
- To be used comfortably by the entire electorate.
- To allow flexibility in the location from where the vote is cast.
- To guarantee voters privacy and anonymity.
- To be secure from corruption.

- To be verifiable by an independent third party.
- Ideally the system should allow the voter to cast his/her vote with a minimum of difficulty and confusion regardless of his/her computer literacy.

### 1.2.2 Learning Aspect

The learning aspect of this project concerns gaining experience in working in a group. It involves researching and incorporating new technology into a project, as well as communicating with others in the group in order to allow everyone to keep up-to-date with the progress of the project. This project requires considerable documentation. It is necessary to learn to combine everyone's work in a coherent manner. The project is designed in an Object Orientated style and time and resources have to be managed efficiently to ensure the task is completed successfully.

### 1.3 Project Deliverables

The list of deliverables is as follows:

- Project Management Plan (PMP)
- Group Meeting Minutes
- Software Quality Assurance Plan
- Testing Documents
- User Interface Manual
- Source Code
- Executable

Documents used for input for this PMP were:

- PMP templates provided by Dr. Morrison.

### 1.4 Definitions, Acronyms and Abbreviations

The following acronyms and abbreviations are used throughout the project documentation:

<b>AADD</b>	Application Area Design Description
<b>AAS</b>	Application Area Specification
<b>ATR</b>	Acceptance Test Reports
<b>ATS</b>	Acceptance Test Specification
<b>CA</b>	Configuration Auditing
<b>CS</b>	Component Source Code
<b>CTR</b>	Component Test Reports
<b>CTS</b>	Component Test Specifications
<b>FSR</b>	Feasibility Study Report
<b>IG</b>	Installation Guide
<b>IM</b>	Introductory Manual
<b>ISM</b>	Information Structure Model

<b>ISM-IE</b>	ISM of the information engineer
<b>ISM-SE</b>	ISM of the software engineer
<b>ITR</b>	Integration Test Reports
<b>ITS</b>	Integration Test Specification
<b>PMI</b>	Product Manufacturing Instructions
<b>PMP</b>	Project Management Plan
<b>PDI</b>	Product Distribution Instructions
<b>PII</b>	Product Installation Instructions
<b>PPI</b>	Product Packaging Instructions
<b>QRC</b>	Quick Reference Card
<b>RM</b>	Reference Manual
<b>SCI</b>	Software Configuration Item
<b>SCM</b>	Software Configuration Management
<b>SCMP</b>	Software Configuration Management Plan
<b>SDD</b>	Complete Software Design Description
<b>SE</b>	System and Environment
<b>SEC</b>	Software Engineering Course
<b>SMG</b>	System Manager's Guide
<b>SQA</b>	Software Quality Assurance
<b>SQAP</b>	Software Quality Assurance Plan
<b>SRS</b>	Software Requirements Specification
<b>STR</b>	System Test Report
<b>STS</b>	System Test Specification
<b>UIM</b>	User Introductory Manual
<b>UM</b>	User Manual

### **1.5 Evolution of the PMP**

In order to ensure that our PMP evolves with the progress of the project we set out at each phase to update it accordingly. This is also done to ensure that each phase of the project is done efficiently and within the time specified. Also the PMP specifies the actual time spent on each section. We do this by setting deadlines for each section and splitting up the total work evenly between the members of the group.

## **2 Overall Planning**

### **2.1 List of Deliverables**

#### **2.1.1 End-User Deliverables**

- A Voters Quick Reference Card – explaining how to vote using the system.
- A Setup Guide – explaining how to configure and setup the system for each election.

### 2.1.2 System Deliverables

- PMP (Initiation Phase).
- PMP (Updates per Phase).
- Meeting Minutes.
- Survey Results – A survey carried out to obtain voters views on electronic voting of which the results can be seen in Appendix A.

The following documentation will not be delivered:

- Product Manufacturing Instructions.
- Product Packaging Instructions.
- Product Distribution Instructions.

### 2.2 Phase Plan

Scheduling of Phases:

Phase	Start Date	End Date	#Scheduled Hours	#Man Hours
Initiation	17/10/2003	03/11/2003	20	22
Definition	03/11/2003	20/11/2003	25	40
Design	20/11/2003	05/01/2004	60	
Realisation	05/01/2004	30/01/2004	100	

Functional Dependencies in Deliverables:

- **Definition Phase**
  - Application Area Specification --->|
    - | Software Requirements Specification --->|
      - | ---> User Introductory Manual
- **Design Phase**
  - Application Area Design Description ----->|
    - Software Design Description ----->|
      - | ---> System Test Specification
        - Component Test Specifications ---> Integration Test Specifications --->|
- **Realisation Phase**
  - | ---> System and Environment
    - Component Source Code ----->|
      - Component Test Results ---> Integration Test Results --->|
        - | ---> System Test Results ---> Acceptance Test Reports

Scheduling of Deliverables:

Phase	Deliverable	Start Date	End Date
Initiation	Project Management Plan	17/10/2003	03/11/2003
Definition	Application Area Specification	03/11/2003	14/11/2003
Definition	Software Requirements Specification	17/11/2003	19/11/2003
Definition	User Introductory Manual	19/11/2003	24/11/2003

Definition	Acceptance Test Specification	Given	Given
Design	Application Area Design Description	24/11/2003	05/01/2004
Design	Complete Software Design Description	01/12/2003	05/01/2004
Design	Component Test Specifications	08/12/2003	05/01/2004
Design	Integration Test Specifications	15/12/2003	05/01/2004
Design	System Test Specification	15/12/2003	05/01/2004
Realisation	Component Source Code	05/01/2004	30/01/2004
Realisation	Component Test Reports	05/01/2004	30/01/2004
Realisation	Integration Test Reports	12/01/2004	30/01/2004
Realisation	System Test Report	19/01/2004	30/01/2004
Realisation	System and Environment	26/01/2004	30/01/2004
Realisation	Acceptance Test Reports	30/01/2004	30/01/2004

## 2.3 Organisation

Number of available people per phase:

Phase	#Available People
Initiation	5
Definition	5
Design	5
Realisation	5

### 2.3.1 Organisation of People

Due to the small size of our group, we chose to adopt a flat organisational structure, i.e., having no group hierarchy. Everyone will provide equal input in terms of ideas and tasks completed. The entire project team will meet once or twice a week in a formal 1-2 hour meeting to examine the progress of the project, re-appraise progress and timing, plan for future developments, discuss problems encountered, and allocate or reallocate tasks as necessary. Apart from this group members may meet to discuss issues or work on particular tasks. These will be documented and presented at the next formal meeting. A record of all meeting minutes is being recorded and will be presented with the final documentation.

Specific roles and functions will be assigned to each group member throughout the development of the project and reviewed at each meeting.

## 2.4 Organisational Boundaries and Interfaces

Due to no member of the project group having administrative access, we will have to work with the system administrators when we require certain software to be installed on the server.

## 2.5 Risk Management

### 2.5.1 The Risks

The risks that may occur are:

- Allowing too much time to research required elements of the project, security, etc.
- Not having the required software at our disposal on the server.
- Missing deadlines.
- Bugs in final code.
- Insufficient knowledge of languages, databases, etc to complete the project.
- Customer dissatisfaction with certain aspects of the final product.
- Vulnerabilities in the system as documented in Appendix B.

### 2.5.2 The Planned Solutions

The solutions to the above risks are:

- Putting time limits on research time by setting deadlines.
- Interacting with the system administrators to get software installed on the server.
- We will try to meet all deadlines. When this does not occur we will discuss as a group reallocating deadlines and refining our schedule.
- We will complete rigorous testing to try and ensure our software is of a high quality.
- We will try to study material as we require it for the project.
- All possible measures will be taken to ensure customer satisfaction.

## 3 Baseline Initiation Phase

### 3.1 List of Deliverables

The deliverable of this phase is:

- The PMP (Project Management Plan).

### 3.2 Activity List

Act#	Activity	# of People	Start Date	End Date
3.1	Detailed Planning of PMP	5	05/11/2003	06/11/2003
3.2	Write Introduction	5	07/11/2003	07/11/2003
3.3	Review Introduction	5	07/11/2003	07/11/2003
3.4	Write Overall Planning	5	12/11/2003	12/11/2003
3.5	Review Overall Planning	5	12/11/2003	12/11/2003
3.6	Write Initiation Phase	5	13/11/2003	13/11/2003
3.7	Review Initiation Phase	5	13/11/2003	13/11/2003
3.8	Write Planning Definition Phase	5	14/11/2003	14/11/2003
3.9	Review Planning Definition Phase	5	14/11/2003	14/11/2003
3.10	Write Planning Design Phase	5	18/11/2003	18/11/2003

3.11	Plan Realisation Phase	5	27/11/2003	27/11/2003
3.12	Plan & Begin Software Quality Phase	5	27/11/2003	*

\* The Software Quality Assurance Plan (Section 7) will continue to be written through the proceeding phases.

Each section of the PMP was written and reviewed during our formal meetings. All members had an input into the contents of the document.

### 3.3 Skills

Knowledge of the following tools is required for producing the documents required in this phase:

- Familiarity with Microsoft Word.
- Familiarity with Microsoft Excel.

All group members are expected to have similar ability here.

### 3.4 Non-Human Resources Needed

The non-human resources for this phase are:

- P.C.'s
- Windows 2000 complete with MS Excel and MS Word.
- A printer.
- A laboratory with desks and chairs.
- A desk and chairs at the Multi-Functional Hall in the Student Centre.

### 3.5 Quality

The quality plan is laid out in detail in the Software Quality Assurance Plan (Section7).

### 3.6 Information

The general Configuration Management for the documents is described in the Software Quality Assurance Plan (Section7).

## 4 Detailed Planning Definition Phase

### 4.1 List of Deliverables

In this phase the following deliverables will be produced:

- Update of the PMP.

## 4.2 Activity List

Act#	Activity	# of People	Start Date	End Date
4.1	Update of PMP	5	19/11/2003	19/11/2003
4.2	List user requirements	5	20/11/2003	20/11/2003
4.3	Analyse requirements	5	20/11/2003	20/11/2003
4.4	Discuss & determine the user interface	5	26/11/2003	26/11/2003
4.5	Discuss & determine the functionality of the system	5	26/11/2003	26/11/2003
4.6	Decide on the system structure	5	26/11/2003	26/11/2003

All five members of the group took part in Act# 4.3 – Act# 4.6 but certain members had varying amounts of knowledge on the tasks due to the fact that Adrian, Pádraig and Stephen are studying a Database module while David and Brian are not.

## 4.3 Skills

Knowledge of the following items is essential for producing the documents required in this phase:

- MS Word and MS Excel.

## 4.4 Non-Human Resources Needed

The non-human resources required for this phase are:

- P.C.'s
- Windows 2000 complete with MS Excel and MS Word.
- A printer.
- A laboratory with desks and chairs.

## 4.5 Quality

The quality plan is described in the Software Quality Assurance Plan (Section 7).

## 4.6 Information

The general Configuration Management for the documents is described in the Software Quality Assurance Plan (Section 7).

# 5 Detailed Planning Design Phase

## 5.1 List of Deliverables

In this phase the following deliverables will be produced:

- PMP update.

The timing of these activities and the functional dependencies, which exist between them are represented in Overall Planning (Section 2).

## 5.2 Activity List

Act#	Activity	# of People	Start Date	End Date
5.1	PMP update	5	27/11/2003	27/11/2003
5.2	Database design	5	03/12/2003	03/12/2003
5.3	Fundamental networking design	5	03/12/2003	03/12/2003
5.4	User interface design	5	03/12/2003	03/12/2003
5.5	Meeting & presenting designs to course leaders	5	12/12/2003	12/12/2003
5.6	Modify database design	5	17/12/2003	17/12/2003
5.7	Modify networking design	5	17/12/2003	17/12/2003
5.8	Modify user interface design	5	17/12/2003	17/12/2003
5.9	Activity list update	5	17/12/2003	17/12/2003
5.10	PMP update	5	18/12/2003	18/12/2003

Following the original design our group had to make a presentation to course leaders on Friday 12<sup>th</sup> of December. Taking into account their input time was set aside for a lengthy meeting the following week to discuss their thoughts and suggestions and make any necessary revisions to the plans for the system. At this point the activity list and the PMP are both updated and the project is ready to enter the Realisation Phase after the Christmas break.

## 5.3 Skills

Knowledge of the following items is essential for producing the documents required in this phase:

- Planning.
- Object Oriented Approach.
- Databases.
- PHP (or suitable alternative).
- MS Word and MS Excel.

## 5.4 Non-Human Resources Needed

The non-human resources required for this phase are:

- P.C.'s
- Windows 2000 complete with MS Excel and MS Word.
- A printer.
- A laboratory with desks and chairs.
- A desk in the Multifunctional Hall in the Student Centre.

## 5.5 Quality

The quality plan is described in the Software Quality Assurance Plan (Section 7).

## 5.6 Information

The general Configuration Management for the documents is described in the Software Quality Assurance Plan (Section 7).

## 6 Global Planning Realisation Phase

### 6.1 List of Deliverables

In this phase the following deliverables will be produced:

- Update of PMP.
- Component Source Code.
- System and Environment.
- Component Test Reports.
- Integration Test Reports.
- System Test Reports.
- Acceptance Test Reports.

### 6.2 Activity List

Act#	Activity	# of People	Start Date	End Date
6.1	Type & compile code for each component	5	05/01/2004	21/01/2004
6.2	Test each component	5	05/01/2004	21/01/2004
6.3	Write Component Test Reports	5	05/01/2004	21/01/2004
6.4	Integrate components	5	05/01/2004	21/01/2004
6.5	Test integration	5	05/01/2004	21/01/2004
6.6	Write Integration Test Reports	5	05/01/2004	21/01/2004
6.7	Test system	5	21/01/2004	21/01/2004
6.8	Write System Test Report	5	23/01/2004	23/01/2004
6.9	Perform Acceptance Test	5	26/01/2004	28/01/2004
6.10	Write Acceptance Test Report	5	28/01/2004	28/01/2004
6.11	PMP update	5	30/01/2004	30/01/2004
6.12	Final Presentation	5	Feb/Mar	Feb/Mar

The above dates are planned dates which we would hope to achieve. We have however left the month of February free to cover running over time due to unforeseen problems which we may encounter.

### 6.3 Skills

Knowledge of the following tools is essential for producing the documents required in this phase:

- Java.

- HTML.
- PHP (or alternative).
- Databases.
- MS Word and MS Excel.

#### **6.4 Non-Human Resources Needed**

The non-human resources required for this phase are:

- P.C.'s
- Java Development Kit
- Windows 2000 complete with MS Excel, MS Word and a web browser.
- A printer.
- A laboratory with desks and chairs.
- A desk in the Multifunctional Hall in the Student Centre.

#### **6.5 Quality**

The general quality plan is described in the Software Quality Assurance Plan (Section 7).

#### **6.6 Information**

The general Configuration Management for the documents is described in the Software Quality Assurance Plan (Section 7).

## **7 Software Quality Assurance Plan**

### **7.1 Purpose**

The purpose of our Software Quality Assurance Plan (SQAP) is to assure quality of the Software Engineering Process as a whole and of the result of the process in particular. This concerns documentation and code standards as well as the Software Verification and Validation Plan.

### **7.2 Scope**

This document is intended for all the phases of the project development.

### **7.3 Management**

We discriminate between several managing functions during the course of this project. Below we mention each function with the corresponding tasks for which the persons assigned are responsible.

SQA-manager (Software Quality Assurance Manager)

This manager is responsible for:

- Creating and updating the SQAP (chapter 7 in the PMP).
- Maintaining available tools and utilities used by the group during the course.
- Creating document and code standards and see to it that they are applied.

SCM-manager (Software Configuration Management Manager)

This manager is responsible for:

- Creating the environment for archiving documentation and code required for this course.
- Administering the document and source code archive.

PLAN-manager (Planning-manager)

This manager is responsible for:

- Writing down and making copies of the weekly planning for the relevant persons.
- Maintaining the overall planning.

During the course of our project to keep in line with our flat organisational structure these positions were not held by any particular group member. We made sure as a group at each meeting that the tasks that would be carried out by such a manager were done so within our group structure.

## **7.4 Risk Management**

### **7.4.1 The Risks**

The risks that may occur are:

- Allowing too much time to research required elements of the project, security, etc.
- Not having the required software at our disposal on the server.
- Missing deadlines.
- Bugs in final code.
- Insufficient knowledge of languages, databases, etc to complete the project.
- Customer dissatisfaction with certain aspects of the final product.
- Vulnerabilities in the system as documented in Appendix B.

### **7.4.2 The Planned Solutions**

The solutions to the above risks are:

- Putting time limits on research time by setting deadlines.
- Interacting with the system administrators to get software installed on the server.
- We will try to meet all deadlines. When this does not occur we will discuss as a group reallocating deadlines and refining our schedule.
- We will complete rigorous testing to try and ensure our software is of a high quality.
- We will try to study material as we require it for the project.
- All possible measures will be taken to ensure customer satisfaction.

## 7.5 Documentation

### 7.5.1 System Documentation

General applicable documentation standards can be found in sections 7.5.4, 7.5.5 and 7.5.6. The CS3000 lecture guidelines will be followed as much as possible with respect to the documentation.

### 7.5.2 User Documentation

General applicable documentation standards can be found in sections 7.5.4, 7.5.5 and 7.5.6.

### 7.5.3 Project Documentation

General applicable documentation standards can be found in sections 7.5.4, 7.5.5 and 7.5.6.

### 7.5.4 Tools and Procedures

MS Word and MS Excel will be used as the document processing tools for the documentation accompanying this project. Procedures with respect to subdivision of the documents are given in section 7.9. The language used in all documents is English.

### 7.5.5 Page Layout

No special page format will be used in the documentation. The default MS Word style will be used.

### 7.5.6 Document Layout

The document layout is defined by using the default MS Word style. Furthermore, each document should have a front page (on which should be the title and the authors of the document) and a table of contents.

### 7.5.7 Users of Documentation

The following table specifies, for each kind of document, the possible users of those documents.

Kind of Documentation	User
System documentation	Group members
Product reproduction and installation instructions	Election Administrators
Tools and Procedures	Group members
User documentation	Product users (voters)
Project documentation	Group members
Page layout	Group members
Document layout	Group members

## **7.6 Software Standards**

The programming languages used in the project are Java, SQL, HTML, PHP and JavaScript.

## **7.7 Reviews and Audits**

- A review consists of three phases: preparation (reading), discussion and updating.
- Usually two persons do a review: a writer and reviewer.
- A distinction is made between document and code review. A document review concerns the reviewing of possible informal, baseline documents.
- A reviewer must allocate time to read a document (i.e., preparation time).
- The persons in the review group must solve points of issue, if not, a meeting is convened.
- Comments generated during the review must be implemented in the ultimate result.
- The writer is responsible for requesting review-time during the weekly planning meetings.
- A review will be finished after acceptance.
- In case of more than one writer of a piece, only one of the writers will act as the writer during the review session.

## **7.8 Software Verification and Validation Plan**

### **7.8.1 General Test Approach**

- The writer himself must test every piece of code.
- Another group member is then assigned to review and test the code further.
- A tester must allocate time in order to test particular code.
- The basic view is that a tester tests every piece of code only if the planning allows for this.
- In case of a time shortage the writer edits the composition of the material to be tested by the tester.
- The reviewer of the code is automatically assigned as tester of the code.
- The 'black-box' and 'glass-box' testing techniques are to be adopted, along with equivalence partitioning.
- Also integration testing, structure testing, stress testing and performance testing are to be used.

## **7.9 Configuration Management**

This section deals with simple configuration management only, because of the group organisation. In it, we define the activities and the used tools, techniques and methods.

### **7.9.2 SCM Management**

See section 7.3.

### **7.9.3 SCM Activities**

In the SQAP we can distinguish between the four following activities:

#### **Configuration Identification**

The configuration tree is mapped on a directory structure. Each individual's code makes up the elements of the configuration itself. Everything we produce as part of this project has to be part of the configuration. This comprises test reports, source code, the PMP, etc. The current configuration we use is outlined in section 7.9.4.

#### **Configuration Control**

All changes have to be reported to the SCM-manager (in our case, the other members of the group). This includes the need for change and the actual change.

#### **Configuration Auditing**

CA is done on the fly. No special activity is planned to monitor this activity. It is part of the activities of the planning managers.

#### **Configuration Status Accounting**

This should be done by the developer (of either documentation or source code) by adding comments at the top of the file.

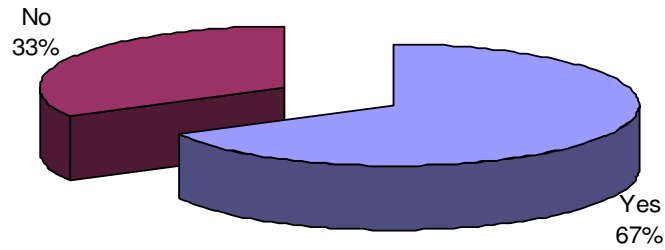
### **7.9.4 Tools, Techniques and Methodologies**

The structure of the central archive is as follows:

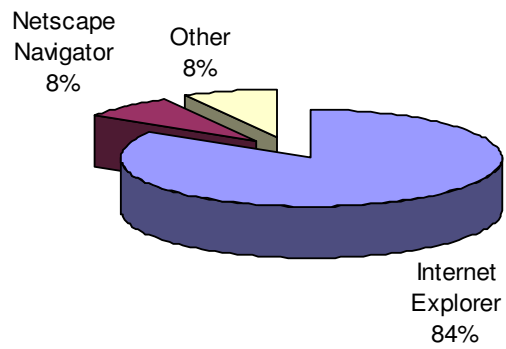
- Latest build: This contains a backup copy of the project code so far. It is updated after any changes to the code have been made and tested to work correctly.
- Application: This directory contains all the code relevant for the project.
- Documentation: This directory contains all the documentation we produce.

## Appendix A – Survey Results

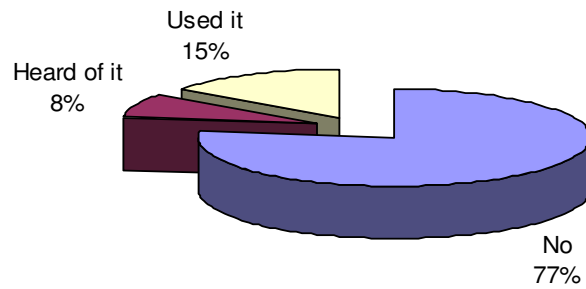
Do you consider yourself to be computer literate?



Which web browser do you use?

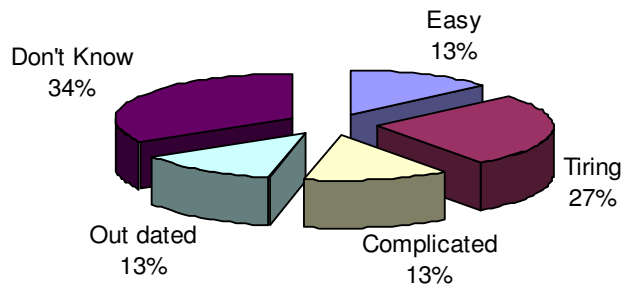


Have you heard of or used electronic voting before?

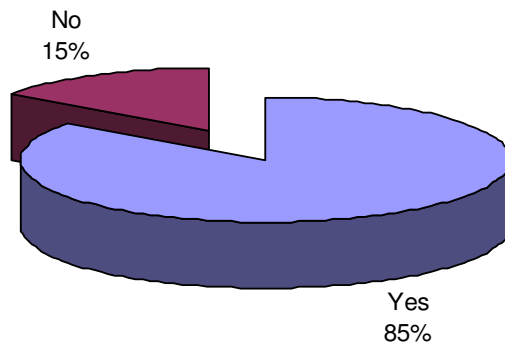


Group C – Electronic Voting System

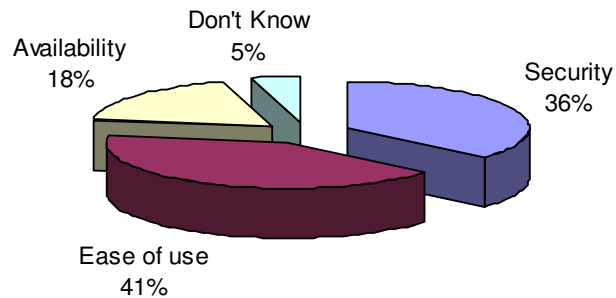
How do you judge the current General Election voting process?



Would participation in elections increase if people had the option of voting from home?

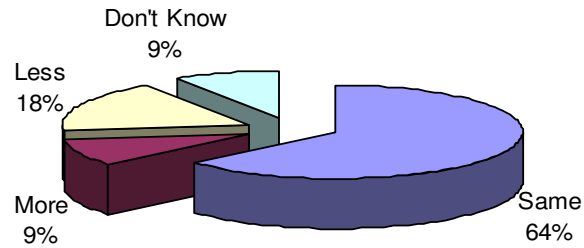


What do you consider the most important issues with regard to electronic voting?

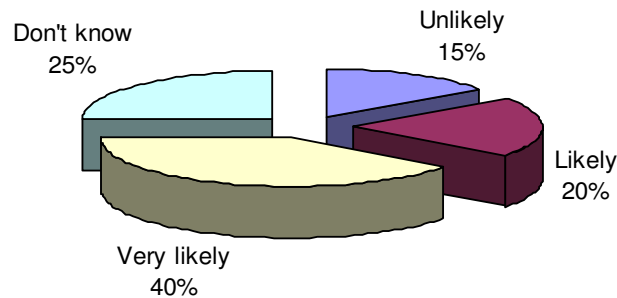


*Group C – Electronic Voting System*

What level of trust would you have in an electronic voting system compared with the current system?



How likely is it that Internet voting will become a part of our lives in the future?



**Conclusions:**

- Two thirds of people consider themselves computer literate.
- The majority of people have not heard of electronic voting before.
- Most believe electronic voting is the way of the future.
- It is the belief that election participation would increase if people could vote from home.
- Main concerns are ease of use and security.

These results convinced us that voting from home and hence over the Internet was the way to go. We need to keep in mind the concerns of the public however and make the system as user friendly as possible while trying to maintain security.

This survey aided us in deciding on necessary requirements when we began planning the system.

## **Appendix B – Vulnerabilities of the System:**

### **Client System Compromise:**

A Trojan, virus or operating system hole can allow an attacker to view how a user has voted, manipulate the outgoing vote or prevent it from being sent - yet give the user the appearance of a successful vote (quite a nice attack - with the right data you could prevent the right number of a certain type of voters from succeeding, and 'encourage' a win for your party. And the data would look perfectly valid).

### **Client Software Compromise:**

Many proposed systems rely on the web browser as their client interface. This is a terrible idea when one considers the number of plug-ins and features lurking in the code of the average browser, none of which are checked and could hide any number of malicious Trojans. Custom-made client software is likely to be more secure but distribution is a huge problem - a cunning attack could involve issuing fake client voting software.

### **Reliability of Infrastructure:**

There are a huge number of sub-systems involved in a successful electronic vote. Even if they work reliably with no problems there are opportunities for staff in many organisations to disrupt or fully stop an election, by attacking the communications infrastructure in telephone exchanges, service providers and hosting centres. Electricity is also a mission-critical resource across the nation in an electronic vote.

### **Incompetence:**

Setting up electronic voting systems is likely to be a complex task which few understand, and an even smaller number will get totally right. The smallest error can compromise the security and anonymity of a vote.

### **Trusting the Administrators:**

There are a large number of volunteers and paid staff in any election. They are trusted parts of the system, but we do have some checks in place such as allowing candidates to watch the count to keep the admin staff honest. The opaque nature of electronic systems makes it significantly more difficult for us to watch the admin staff. Thus the trust we put in them increases and the potential damage they can cause also grows.

### **Closed Source:**

All the systems currently being evaluated or used by the US and UK governments are closed source. No third party has been allowed to assess the design and implementation of the systems. Not only could malicious backdoors be used to undermine an election result, but innocent errors could also be lurking. Ideally the source should be available for any citizen to examine but this still presents huge problems. How do we know that the published source is actually that used in the live system? Every time there is a bug-fix or update do we re-audit all the code? We should. One fix can introduce three more bugs, or could hide a cunning exploit for future use by a disgruntled employee. How do we check that each server or kiosk is indeed using the code that has been audited?

## Appendix C – Requirements

This is a list of the requirements that we discussed at our meetings prior to constructing the Requirements Specification document.

### 1. Registered Voters

- Only eligible voters can vote and can do so only once.
- Registered voting number and pin number to ensure identity.
- Database of registered voters

### 2. Privacy

- Each vote must be completely anonymous and can't be linked to a voter.

### 3. Integrity

- Each vote must be unchangeable.

### 4. Accuracy

- The result must be unfalsifiable.
- An independent observer must be able to verify the result if necessary.

### 5. Verifiability

- Database is openly verifiable.
- Source code is freely available, excluding security issues.
- The votes should be able to be independently tallied.

### 6. Convenience

- Allow voters to vote quickly and easily.
- Possibly from home e.g. a web page.

### 7. Flexibility

- Should allow a variety of ballot formats (referendum/election).
- A setup program should allow input of choices (candidates).

### 8. Mobility

- Voters can vote from a variety of locations.
- On line voting – from home, local libraries, Internet cafés, etc.

### 9. Efficiency

- An efficient algorithm should be used to calculate results quickly.

### 10. Contingency Plan

- Polling stations could be a backup for network failure.
- Regular backups of the database should be made to cater for database failure.

### 11. Security

- Viruses.
- Database firewall compromise, etc.

## **Appendix D – System Design**

### **Contents**

#### **1 BASIC DESIGN**

#### **2 DETAILED DESIGN**

##### 2.1 Database

2.1.1 Why we Required a Database

2.1.2 Resources Required

2.1.3 Time Allocation

2.1.4 Detailed Design

2.1.4.1 Data Model

2.1.4.2 Physical Data and Its Purpose

2.1.5 A Real World Database

2.1.6 Population and Testing

##### 2.2 Setup Program

2.2.1 Why we Required a Setup Program

2.2.2 Resources Required

2.2.3 Time Allocation

2.2.4 Detailed Design

2.2.5 Research

##### 2.3 Web Page (HTML and JavaScript)

2.3.1 Why we Required a Web Page

2.3.2 Resources Required

2.3.3 Time Allocation

2.3.4 General Design

2.3.5 Detailed Design

2.3.5.1 HTML Design

2.3.5.2 JavaScript Design

2.3.6 Browser Compatibility

##### 2.4 Web Page (PHP)

2.4.1 Why we Required PHP

2.4.2 Resources Required

2.4.3 Time Allocation

2.4.4 Detailed Design

2.4.5 Research

##### 2.5 Results Program

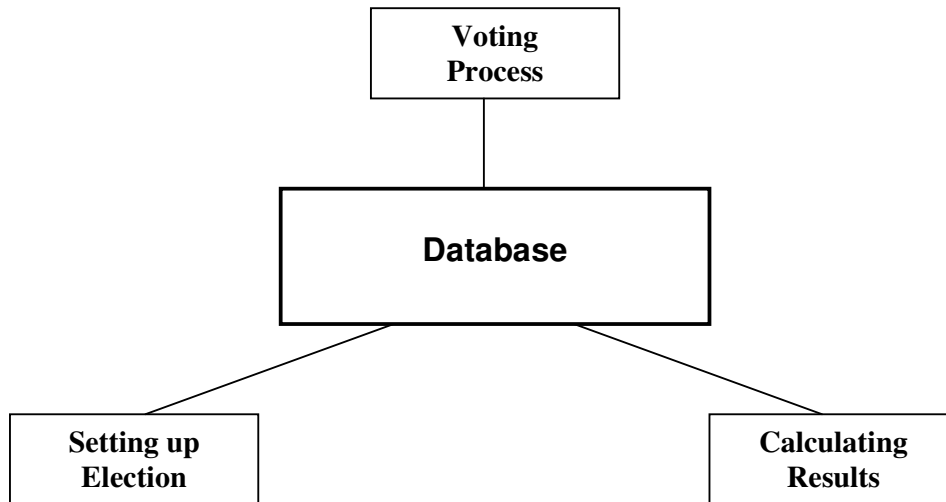
2.5.1 Why we Required Java

2.5.2 Resources Required

2.5.3 Time Allocation

2.5.4 Detailed Design

## 1 Basic Design



- The *Voting Process* is a web implementation which is carried out on line at remote locations e.g. from home.
- *Setting up the Election* is carried out by an Election administrator through a java program run on location.
- *Calculating Results* is carried out by a java program run by an Election administrator.
- Each of the above three processes access the database, which is stored at a central government location.

## 2 Detailed Design

### 2.1 Database Design

#### 2.1.1 Why we required a Database

Our system required a database to store vital information about Voters, Constituencies, Candidates and most importantly, votes. Using a database, we could do the following things:

- Store information about the voters, for example, their Constituency. We could tell if they had voted or not by checking specific attributes.
- Store all the votes. We needed to store each vote in order to calculate the result as accurately as possible.
- Flexibly add constituencies for a given election, add candidates to constituencies, add referendum questions. The user could then have the freedom of specifying the layout and design of his/her election.

#### 2.1.2 Resources Required

- A computer with internet access
- A database on the server

- Java version 1.4.2
- MySql
- A Driver to access the database
- Together 6
- MS Word
- MySql.com

### **2.1.3 Time Allocation**

In our PMP we had specified that our design phase would run throughout the month of December. By the first week of December we had initiated our design phase, and the first thing we tried to do was design a database that suited the needs of everyone. Although we had long discussions about the design, we still managed to finish the phase before December. Our aim was to have a perfect database created and populated by the last week of January – with this we succeeded.

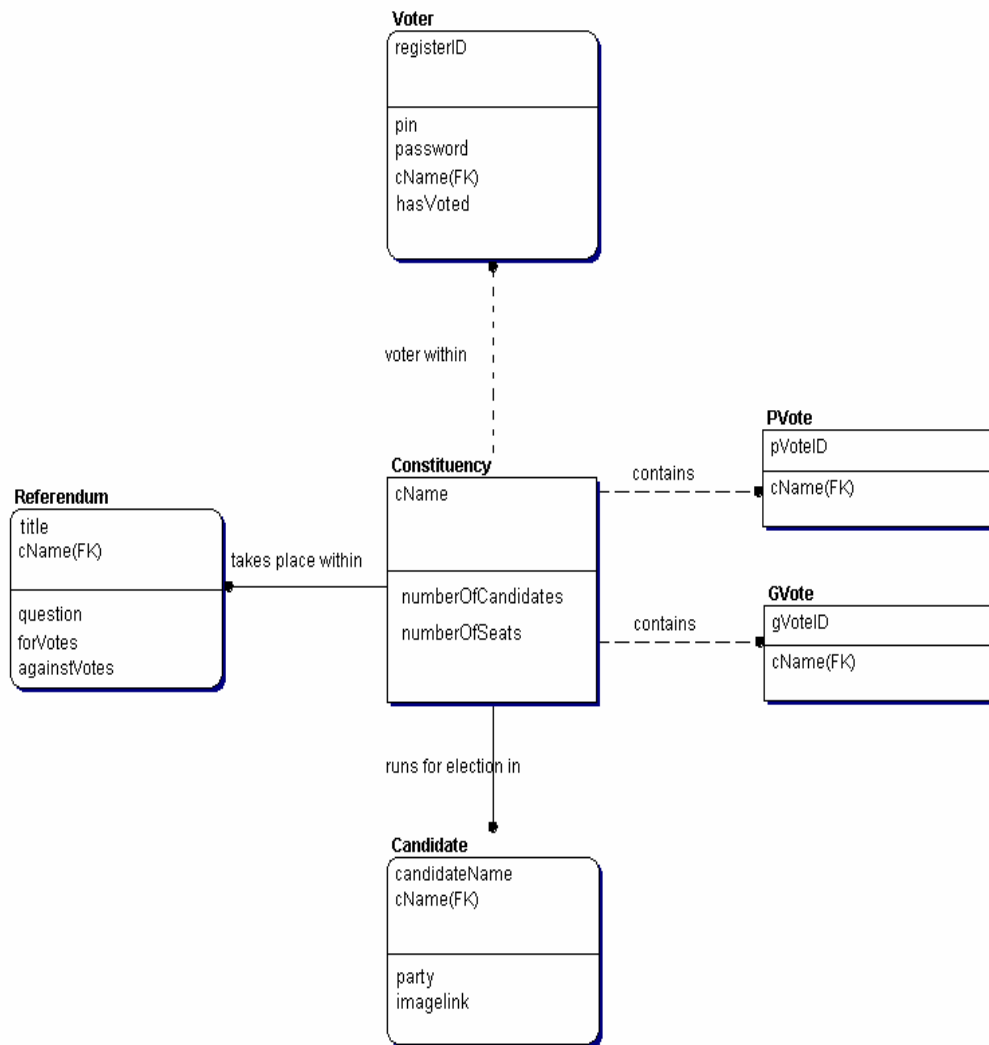
### **2.1.4 Detailed Design**

The functions of the database were discussed and agreed upon by the entire group on numerous occasions at our project meetings. It was most important that everyone discussed this topic because as the database would act as a connecting medium between the website and the setup utility we had to ensure that the design had no ambiguities. Hence the Entity – Relationship Diagrams (ERDs) were drawn when every member was present.

During our design phase, we came up with two database designs. The first was really efficient but hard to implement, while the second was easy to implement but far less efficient. As we could not reach an agreement, we decided to seek outside help and ultimately came to the conclusion that we would use the second option. This would mean that the results program would be far more difficult to write, and that the database would require much more storage capacity. However the database programming would be far simpler and we would hopefully save some time in this way. Pádraig was assigned to draw the ERD using the Together 6 program, which would automatically generate the required DDL (Data Definition Language).

The database was designed and redesigned on many occasions for the simple reason that someone always thought up of something that needed to be added or something that had been overlooked. Then the ERD would have to be remodelled, which Pádraig undertook on each occasion very patiently! The final and perfect model was eventually achieved:

### 2.1.4.1 Data Model



### 2.1.4.2 Physical data and its purpose

Separate entities were created in order to store vital information. Each table and attribute has a specific purpose.

#### The “Voter” Table

We created a table to store information on the registered voters as follows:

Table name:	Voter	
Primary Keys	Data Type	Purpose
regeisterID	integer	A unique id for each voter.
Non-Key Attributes		
pin	varchar(25)	A pin number for a voter.
password	varchar(25)	A voter’s login password.
cName	varchar(30)	The constituency in which the voter resides.

hasVoted	varchar(3)	Tells if a voter has voted or not. Prevents a person casting more than one vote.
----------	------------	--

The each voter is identified by its “registerID”. For security purposes we would have a pin number and a password for each voter. These attributes would both be necessary when logging onto the system (along with the registerID). There were two reasons for this. Firstly we wanted to have a password which a user could change, in the event that someone maliciously found it out. Ultimately we wanted to setup a system where voters could change this password, given their old password. Secondly the fact each voter would have two separate attributes to remember would decrease the chance that they would be impersonated, as it is much more difficult to guess two numbers as opposed to guessing one. We also had a “cName” for each Voter. This attribute represents the constituency in which the voter would finally cast his or her vote. The value of this attribute was vital for building the required web page for that voter. The attribute “hasVoted” is used to allow people to log in as often as they wish, until they cast their vote. Its default value is the string “NO”. This value is checked before a voter is allowed to log in. Once a voter casts their vote the value is set to “YES”, and hence they are disqualified from voting again.

**The “Constituency” Table**

<b>Table name:</b>	Constituency	
<b>Primary Keys</b>	<b>Data type</b>	<b>Purpose</b>
cName	varchar(30)	The name of the constituency
<b>Non-Key Attributes</b>		
numberOfCandidates	integer	The number of candidates running for election in that constituency
numberOfSeats	integer	The number of seats available in that constituency

The constituency table is used to store data about each constituency. Each constituency is identified by its “cName” (constituency name, for example “Kilkenny” or “Cork South Central”). The “numberOfCandidates” attributes specifies the number of candidates running for election in that constituency. The “numberOfSeats” attribute represents the number of seats available in that constituency.

**The “Candidate” Table**

<b>Table name:</b>	Candidate	
<b>Primary Keys</b>	<b>Data Type</b>	<b>Purpose</b>
candidateName	varchar(50)	The candidates name
cName	varchar(30)	The constituency in which this candidate runs for election
<b>Non-Key Attributes</b>		
party	varchar(30)	The candidates political

		party
imagelink	varchar(200)	A path to an image of that candidate

This table represents each candidate. Each candidate is identified by its “candidateName” and its “cName”. The attribute “cName” references the Constituency table, so that this attribute can tell us which in which constituency this candidate runs for election. We also include attributes to tell us a candidate’s party (if any), and the location of an image of that candidate, which would finally be used on the ballot sheet.

### The Referendum Table

<b>Table name:</b>	Referendum	
<b>Primary Keys</b>	<b>Data Type</b>	<b>Purpose</b>
title	varchar(120)	The referendums title
cName	varchar(30)	The constituency in which this referendum is running.
<b>Non-Key Attributes</b>		
question	varchar(220)	The question in debate
forVotes	integer	The number of votes agreeing
againstVotes	integer	The number of votes against

We decided that we would need to find out what constituencies voted for or against a given motive. To do this we looked at the problem for the point of view of holding the same referendum in each constituency (represented by “cName”). This way we could tell what each Constituency decided. By tallying up the votes over all constituencies, we could then find a result for the entire country. Each Referendum tuple has a “title” and a “question” to tell us what the debate is about. The “forVotes” and “againstVotes” attributes are integers that record the number of votes for and against the debate respectively. These values are may be updated (increased by one) once a voter casts his or her vote depending on whether a voter votes with or against the motive.

### The PVote Table

<b>Table name:</b>	PVote	
<b>Primary Keys</b>	<b>Data Type</b>	<b>Purpose</b>
pVoteID	integer	The id of each vote
<b>Non-Key Attributes</b>		
cName	varchar(30)	The constituency in which this vote was cast

The PVote table is used to record each vote in a presidential election. Each vote is uniquely identified by its “pVoteId”. The “cName” represents the constituency of the person who cast the vote. This table grows *dynamically* depending on the number of Presidential Candidates that are running for election. In the event that a candidate decides to run for election a column is added named e.g., “pref1”. The number of columns added ultimately determines the number of Presidential candidates running for election. This gives the system the flexibility to allow an infinitely large number (in theory) of candidates to run for election.

**The GVote Table**

<b>Table name:</b>	GVote	
<b>Primary Keys</b>	<b>Data Type</b>	<b>Purpose</b>
gVoteID	integer	The id of each vote
<b>Non-Key Attributes</b>		
cName	varchar(30)	The constituency in which this vote was cast

The GVote table is used to record each vote in a general election. Each vote is uniquely identified by its “gVoteId”. The “cName” represents the constituency of the person who cast the vote. This table grows *dynamically* depending on the number of candidates running for general election in the constituency which has the most candidates. In the event that this maximum is changed, a column is added named e.g., “pref1”. This gives the system the flexibility to allow an infinitely large number (in theory) of candidates to run for election in any constituency.

**2.1.5 A real world database**

In order to allow government officials to interact with the database e.g. say what constituencies exist, what Candidates are running for election, what referendums are in debate, our system contains a setup utility which allows government officials to input their own information. To do this successfully, a comprehensive java class was required, which contained the necessary functionality (through the incorporation of java methods) to carry out the required tasks. The following pages are taken from the generated java documentation, created using UNIX commands. They describe in detail the methods that are implemented in this class.

Method Summary	
void	<a href="#"><b>addConstituency</b></a> (java.lang.String constituencyname, int numberofseats) Creates a new constituency in the database.
void	<a href="#"><b>addGenElecCandidateToConstituency</b></a> (java.lang.String name, java.lang.String party, java.lang.String imagelocation, java.lang.String constituency) Add a general election Candidate to a given constituency.
void	<a href="#"><b>addPresidentialCandidate</b></a> (java.lang.String name, java.lang.String party, java.lang.String imagelocation) Add a presidential Candidate - the candidate is added for all existing constituencies For all presidential candidates the constituency is automatically called "IRELAND"
void	<a href="#"><b>addReferendum</b></a> (java.lang.String title, java.lang.String question) Add a referendum for all constituencies.
void	<a href="#"><b>addVoter</b></a> (int id, java.lang.String pin, java.lang.String password, java.lang.String constituency) Creates a new voter in the database.

void	<a href="#"><u>addVotersFromFile</u></a> (java.lang.String filename) Takes voters form a file and puts them into the database. The file must be a text file. The entries on each line MUST have the following format: id_number:pin_number:password:constuency: eg 12345:ab34nj:cat:Kilkenny:
void	<a href="#"><u>commit</u></a> () This method saves all changes to the database since the last commit was made.
java.util.Vector	<a href="#"><u>generateResultsForConstituency</u></a> (java.lang.String constituency, java.lang.String resulttype) This method generates the results and returns them in a specifically formatted vector.
java.util.Vector	<a href="#"><u>getConstituencies</u></a> () A method to return the list of current constituencies.
java.util.Vector	<a href="#"><u>getGenElecCandidatesForConstituency</u></a> (java.lang.String constituency) A method to return the list of existing general election candidates for a given constituency.
java.lang.String	<a href="#"><u>getImageURL</u></a> (java.lang.String candidate, java.lang.String constituency) A new method to find out the URL of a candidates image.
int	<a href="#"><u>getNumberOfSeats</u></a> (java.lang.String constituency) A method to return the number of seats in a given constituency.
java.lang.String	<a href="#"><u>getPartyName</u></a> (java.lang.String candidate, java.lang.String constituency) A new method to get a candidate's party.
java.util.Vector	<a href="#"><u>getPresidentialCandidates</u></a> () A method to return the list of existing presidential candidates.
int	<a href="#"><u>getQuota</u></a> (java.lang.String constituencyName) A method to return the quota (i.e. the number of votes a candidate must have before they are deemed elected) for a general election in a given constituency
java.util.Vector	<a href="#"><u>getReferendums</u></a> () A method to return the list of existing referendum titles and questions.
int	<a href="#"><u>getTotalElectors</u></a> (java.lang.String constituency) A method to return the number of Electors (all those registered to vote) for a given constituency.
int	<a href="#"><u>getTotalGVotesCast</u></a> (java.lang.String constituencyName) A method to return the total votes cast for an general election in a given constituency.
int	<a href="#"><u>getTotalPVotesCast</u></a> () A method to return the total votes cast throughout Ireland in a presidential election.
static void	<a href="#"><u>main</u></a> (java.lang.String[] args)

void	<a href="#"><u>removeConstituency</u></a> (java.lang.String constituencyname) Remove a constituency from the database.
void	<a href="#"><u>removeGenElecCandidateFromConstituency</u></a> (java.lang.String candidatename, java.lang.String constituency) Remove a general election Candidate from a given constituency.
void	<a href="#"><u>removePresidentialCandidate</u></a> (java.lang.String candidatename) Remove a presidential Candidate - the candidate is removed for all existing constituencies.
void	<a href="#"><u>removeReferendum</u></a> (java.lang.String title) Remove a referendum for all constituencies.
void	<a href="#"><u>rollback</u></a> () This method “rolls back” the database to the last commit. Changes made since the last commit are not saved.
void	<a href="#"><u>setNumberOfSeats</u></a> (java.lang.String constituency, int numOfSeats) A method to return the number of seats in a given constituency
void	<a href="#"><u>startNewElection</u></a> () A method to set up the required database tables on disk
void	<a href="#"><u>writeVotersToFile</u></a> (java.lang.String filename) Writes all the voters in the database to a file Specifications: The file must be a text file. The entries on each line MUST have the following format: id_number:pin_number:password:constuency: e.g. 12345:ab34nj:cat:Kilkenny:

### 2.1.6 Population and Testing

To test the above methods a tester class was written. This class consisted of a main method, and many exception handling statements. The database was populated and depopulated using these methods, plus we used these methods to see if the correct results were being generated. Each method was tested with valid information, and then with false information to ensure that the methods recognised incorrect parameters. We also ran various SQL queries to verify the information that was added or removed against the answers that the methods gave us.

### 2.1.7 Research

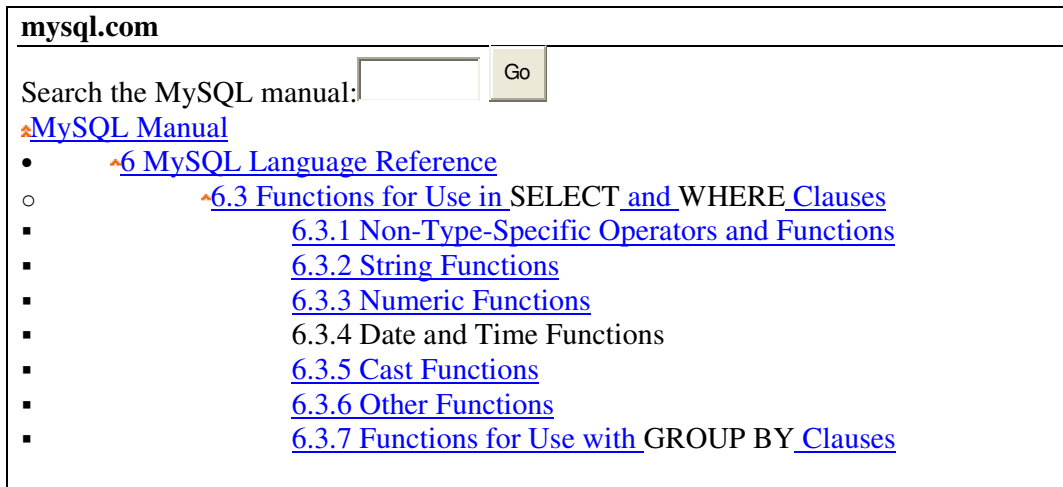
Not a lot of initial research was needed because four out of five members of the group were doing the CS3040 course and we had all created simple databases before. There was certainly a lot of trial and error though!

We encountered problems when using the Together 6 program for drawing the ER diagrams. The program is slow and often unreliable. It is also an incredibly complex program to use, which led to a lot of frustration as a lot of information was lost and had to be redrawn.

The fact that we were supplied with a MySql database server proved problematic at the beginning of the year. Those of us involved in the CS3040 course were used to using Oracle

based servers. We found that Sequences (invented by Oracle and now a SQL standard used to generate numbers) were not supported, and this led to many design changes as our code was orientated to use them. We also found that the “ON UPDATE CASCADE” function was also not supported, and hence this led to extra work in the debugging of our programs.

To improve our knowledge of the MySQL server we made use of the MySQL web site at [www.mysql.com](http://www.mysql.com). The tools on the site proved most useful, and the site as a whole was clear, easily navigated, and to the point. Here is an example of what the site provides:



## 2.2 Setup Program

### 2.2.1 Why we Required a Setup Program

The setup program allows the adding of constituencies and their candidates. It allows adding of Presidential candidates and referendum. We felt it was necessary for our product to have this functionality and this is the reason the setup program is required.

### 2.2.2 Resources Required

- A computer with JDK
- A text editor
- Java API Documentation

### 2.2.3 Time Allocation

We felt this was a major component of the project and so dedicated a suitable amount of time to it. We discussed the functionality it should have in group meetings. We then allocated the detailed design and implementation of this to David. He designed it during December and we discussed the design at meetings. David and Pádraig worked closely on this as Pádraig was designing the database and it was necessary for the setup program to connect to the database.

### 2.2.4 Detailed Design

David was given responsibility for the design of the setup program. He tried a number of designs. These were then discussed at group meetings. Additional functionality which was deemed necessary by other group members had to be taken into account and then the program was re-designed. When everyone thought that the design had the required functionality David constructed the final design. On the next page is a basic sequence diagram Figure 2.2.4.1, which was used to construct the program. David constructed detailed class diagrams from this. He then concentrated on writing the code for these classes after Christmas.

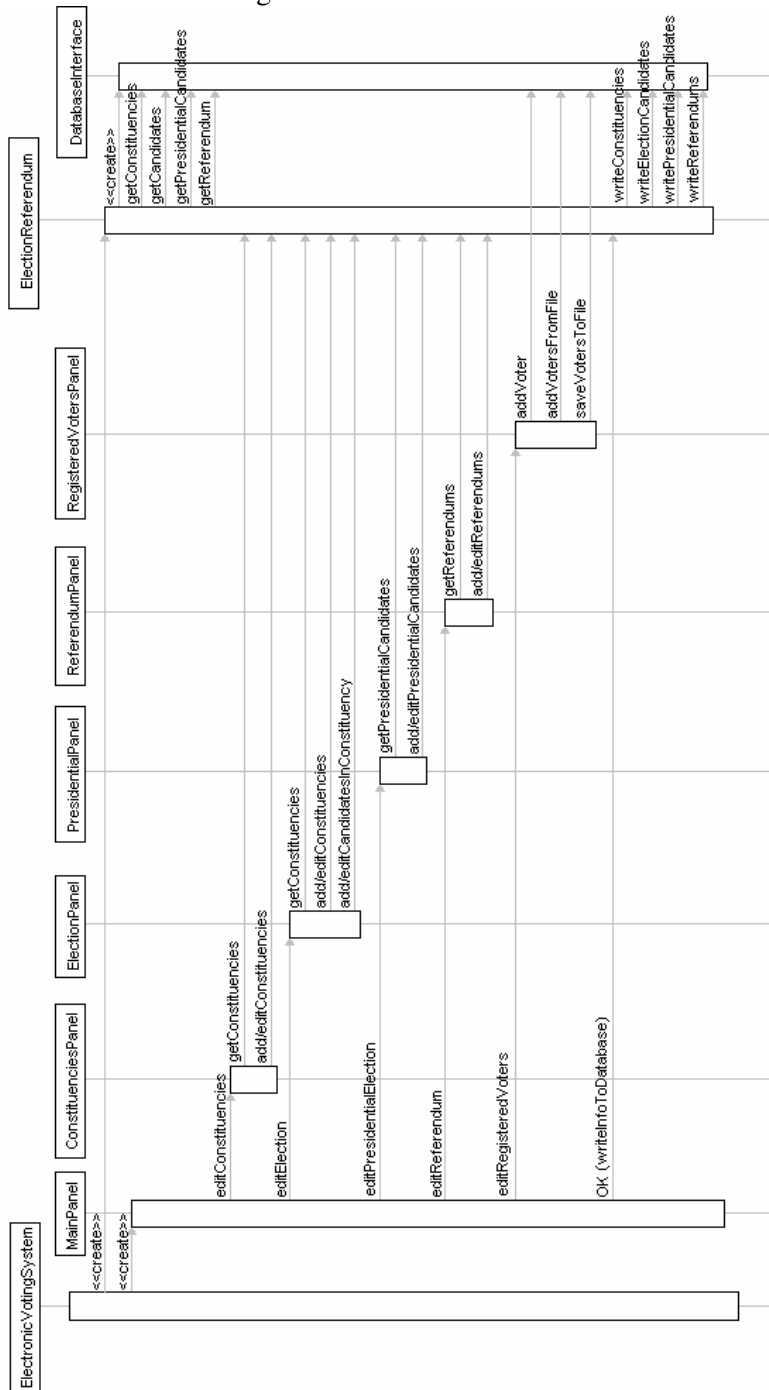


Figure 2.2.4.1

## **2.2.5 Research**

The Java API on-line documentation was used for all the research necessary while designing the setup program.

## **2.3 Web Page (HTML and JavaScript)**

This section was designed and implemented by Adrian.

### **2.3.1 Why we Required a Web Page**

Firstly after thinking about the general design of the project as a whole, we decided that we would be using a web based voting system, that would enable anyone with a pc and Internet connection to vote, once they have registered and are valid voters. This would be available through voting booths for those who don't have access to a pc and internet connection.

### **2.3.2 Resources Required**

- A computer with internet access
- A domain name
- Macromedia Dream Weaver
- Macromedia Fireworks
- Paint shop pro
- Tutorials on Dynamic HTML, JavaScript, PHP

### **2.3.3 Time Allocation**

As we have worked with websites beforehand we had some idea on the length of time needed, for both learning new software and coding. The new aspects I had to learn were Paint shop pro 7, Dream Weaver MX 2004, Dynamic HTML, and also to go into a lot more detail in JavaScript. After Christmas I started making prototypes of the webpage and experimenting with colours images etc. It took approximately three weeks to decide on the general design. I worked on the JavaScript with static information firstly, and after roughly a week I got the functionality working. The biggest problem I came across was the linking and integration of the PHP and the JavaScript. That took roughly two weeks to get working properly. In future designs I probably would have either used an external JavaScript file, or completely do the project in PHP.

### **2.3.4 General Design**

The immediate look of the website was very important and not to be rushed into, as many factors on a webpage influence the user, and as many users might not have a lot of knowledge of working with computers. The initial page must be welcoming, clear, and easy to use, but also professional, making the user feel confident in the online voting system. Many factors

influence the user, colours, layout and wording, therefore we realised the importance of the planning stage.

**Login Page:**

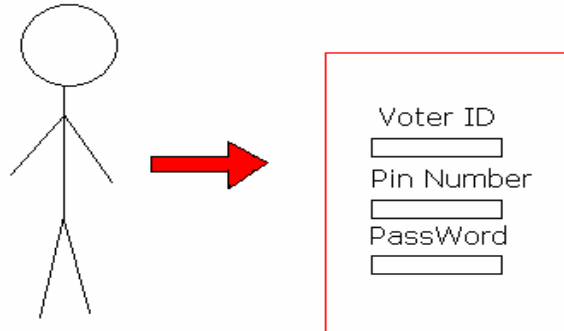


Figure 2.3.4.1

After a lot of discussion and research we concluded that the simpler the better when it comes to the layout, avoiding confusing the voter and making it clear what to do. Firstly the voter needs three bits of information in order to vote, a Voter ID, Pin Number and Password. Therefore the initial page will consist of a form, in which the voter can enter the three bits of information.

The voting system we are designing caters for a General Election, a Presidential Election and a Referendum. The next choice we had to make was how to present the information to the voter, whether we display it on three separate pages or one page. If we were to have a separate page for each election/referendum it would somewhat confuse the voter, jumping forward and back between pages could complicate the process, but if a single page was to be used all the election information could be displayed to the voter. When setting up a candidate in the setup program, the candidate name, party and picture address are entered, and when setting up a referendum the question is entered so the task at hand was to figure out how to display this information clearly for each candidate, and also for each referendum question.

**Voting Page:**

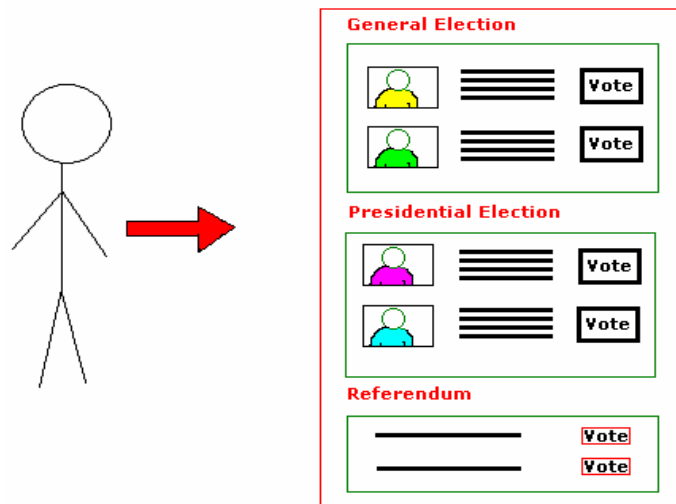


Figure 2.3.4.2

One thing we had to take into account is that with the amount of choices the voter has the probability of making a mistake would be high. After researching the internet, it seemed that a confirmation page would be quite functional. In many online shops confirm pages are used, adding many benefits. In our voting system it would allow the voter to see the votes he/she has cast, giving the option of returning to the voting page and changing the vote, votes he/she has cast, or to confirm and finalise the vote.

**Confirmation Page:**

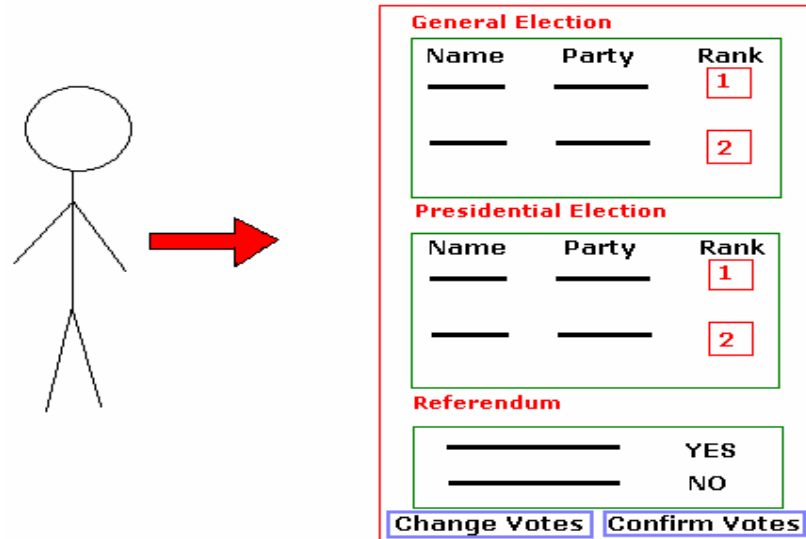


Figure 2.3.4.3

After confirming the vote, the voter would be presented with a page that informs the voter whether the vote has been cast successfully.

**2.3.5 Detailed Design**

We planned to use dynamic html JavaScript and PHP to get full functionality from our website.

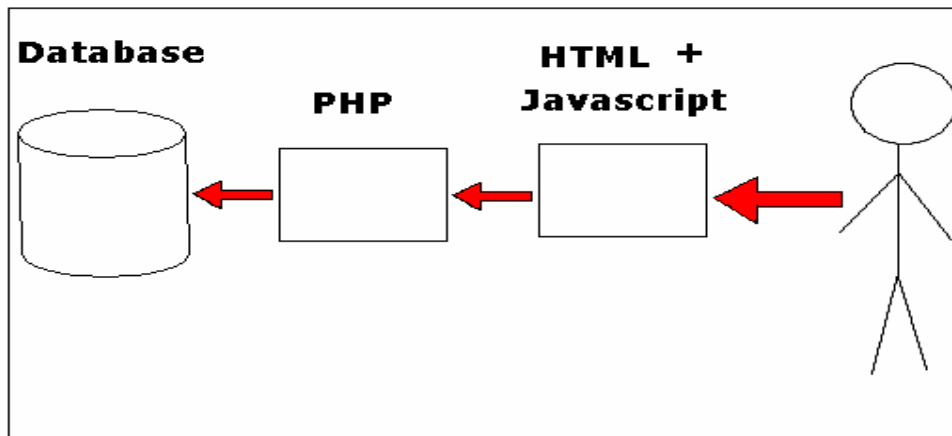





Figure 2.3.5.1

### 2.3.5.1 HTML Design

In this section I will work through the design of the html discussing the look of the website, including colours fonts etc.

Firstly with regards to colours, as this project is for the Irish voting System, I thought it would be appropriate to take the three colours of the flag and use them throughout. I carried out quite a lot of research about colour theory and how colours can influence users, and change users' moods. Taking this into account I was quite confident what mood I wanted to create and had an idea which colours would work. When the voter sees the webpage for the first time I want him/her to feel confident in our system, making it look professional, and to use neutral calm colours not to influence the voter.

The basic colours used throughout are , ,  including a wide variety of greens. I would have liked to have more time to analyse the colours and the response of users to various colours used.

The main goal in creating this page is to have it simple and straightforward to use, which generally means to have the instructions clear and to keep from adding non vital information. Thus keeping away from menus, background images etc. I created three images for the header of the website which will remain throughout the voting process. I initially copied these images from the Irish government website, but then personalised them so suit our needs.



Figure 2.3.5.1.1

I created a wide variety of buttons, for use throughout the site.



Figure 2.3.5.1.2

Finally I created two logos to personalise the site.



Figure 2.3.5.1.3

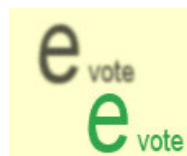


Figure 2.3.5.1.4

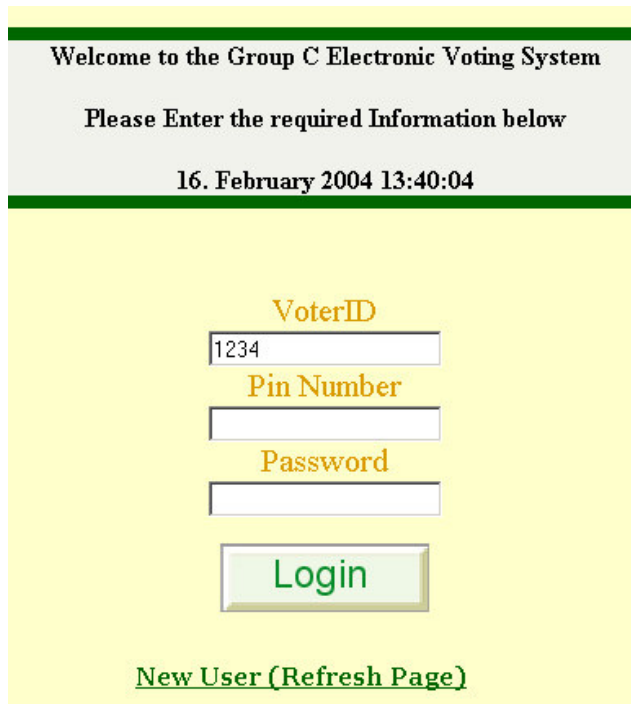
### 2.3.5.2 JavaScript Design

In this section I will work through the design of the JavaScript discussing the functionality of the website.

#### Login Page:

The introduction page and the login page do not contain any JavaScript code with respect to voting. The voter enters the three required fields before he/she can vote. The information is sent in through a basic form. We disabled auto complete in the voter id field to ensure that the voter's voter number cannot be seen by the next user of the webpage.

```
<input name="idnumber" type="text" id="idnumber" size="20" autocomplete="off">
```



Welcome to the Group C Electronic Voting System

Please Enter the required Information below

16. February 2004 13:40:04

VoterID  
1234

Pin Number

Password

Login

[New User \(Refresh Page\)](#)

Figure 2.3.5.2.1

#### Voting Page:

This is the page in which most of the JavaScript is used and therefore the page that took most of the planning and coding time. Firstly I was faced with the problem of showing the relevant information for each voter. When the voter enters the required information successfully an array is passed to the page from PHP with the relevant candidates and referendum questions. Therefore I could not use static html, as the number of candidates/referendum questions changes according to the constituency. Loading the required information dynamically was the best option I came across. So the static html page consists of three tables with <div> division tags with ids, allowing me to create tables dynamically in these division tags. Firstly checking whether the array from PHP was null, indicating that there are no Presidential elections, general elections or referendums running and showing a suitable message.

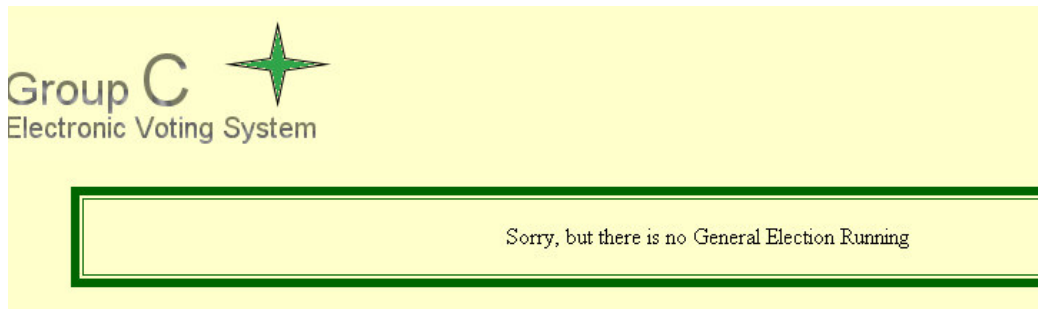


Figure 2.3.5.2.2

When the body of the html loads my function “addTables()” is called which calls each one of these methods ”addPresTable(), addReferendumTable(), addFinalButton()” which create the table for presidential elections, general elections and referendum questions. The “addFinalButton” method checks whether there are any elections/referendums running before displaying the vote button, to avoid confusion.



Figure 2.3.5.2.3

The methods above are very similar to one another so I will discuss one of them. Using the document.createElement() method I created a table dynamically, entering all the data from the PHP array into it. Firstly the Picture of the candidate, next the name and party of the candidate, and finally the “click here to Vote” area. Using a span tag with an id allowed me to reference the exact candidate table, and to add an onclick event to it.. This onclick event would call the corresponding vote function (i.e. Genvote(can) for the General Election) which would replace the “Click here to vote” with the corresponding vote preference.

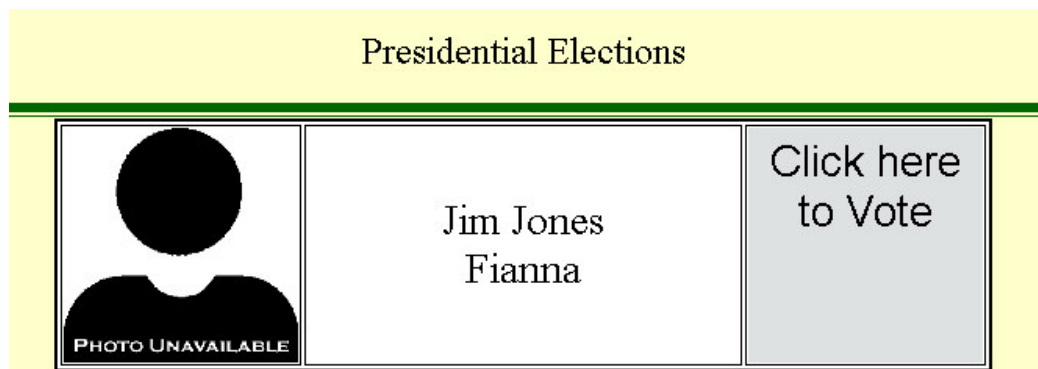


Figure 2.3.5.2.4

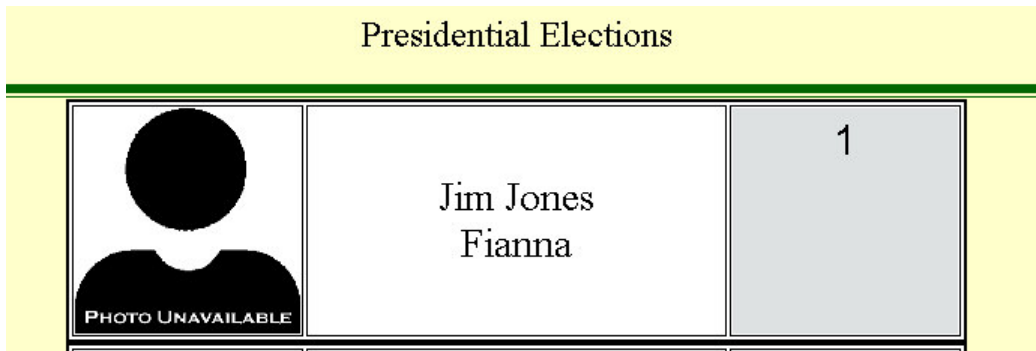


Figure 2.3.5.2.5

Similarly for the Referendum, changing the buttons with onclick events

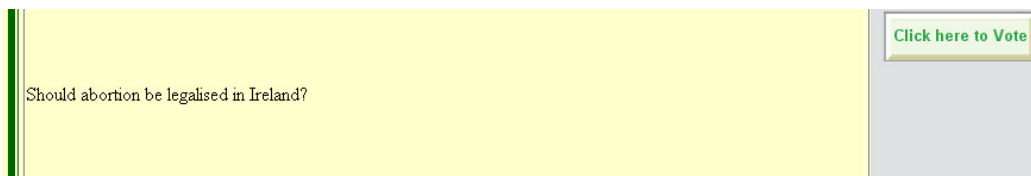


Figure 2.3.5.2.6

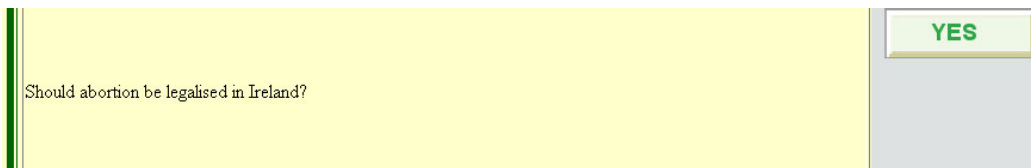


Figure 2.3.5.2.7

For each of the general elections, Presidential elections and referendums, I added buttons to increase the usability and functionality of the system, these buttons being “Reset all Votes” and “Reset last Vote”.

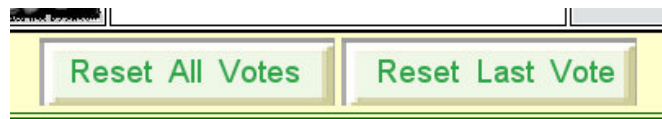


Figure 2.3.5.2.8

Each of these buttons being linked to a corresponding function. When a voter has made a decision on his/her votes, he/she clicks the “Confirm Votes” button. At this point two major things occur. The first being “SortVotes()” is called, which puts all the votes cast into arrays ready to be sent on to the next page. The second being the calling of the method “ConfirmVotes()”. This method dynamically creates a Confirm page inside the existing one using document.write() method. It uses the votes the voter has cast to create table showing all the votes made by the voter.

Please Confirm Your Votes

Name	Party	Vote Preference
B J Mc Guinness	Ind	4
Paide O Toole	Fianna Fail	3
Sean O Reilly	Labour	1
Seymour Crawford	FG	2

Name	Party	Vote Preference
Jim Jones	Fianna	1
Mary Robinson	IND	2
Shane Mc Gowan	IND	3

Referendum Question	Vote Preference
Is he worth it	YES
Should abortion be legalised in Ireland?	No Vote Cast
Should Ireland agree to the Nice Treaty?	YES

Recast Votes

Confirm Votes

Figure 2.3.5.2.9

The two buttons give the voter two options, either to Recast the votes made previously or to confirm the votes made previously. In side this page I created a hidden form with the information regarding the user’s votes, so that if the voter chooses “Confirm Votes”, these hidden form variables are sent to the final voting page, where the votes are send to the database using PHP. To ensure user privacy I included JavaScript which removes the last item from the history, which basically Stops any user from viewing what the last voter has voted.

### 2.3.6 Browser Compatibility

In order that we have a legitimate Voting System, it is essential that our voting system will work on any browser. Firstly we would check for the browser type and then display the corresponding code that will work

For example:

```
if (!isIE3Mac && is.nav5up) {
    {
        // document.write() statements to create markup for
        Navigator 5 and later using HTML 4.0
    }
    else if (!isIE3Mac && is.nav4) {
        {
            // document.write() statements to create markup for
            Navigator 4 and later versions
        }
        else if (!isIE3Mac && is.ie4up). . . . .
```

Browser	Problems?	Future Solutions
Internet Explorer	none	
Netscape Navigator		
Mozilla		document.all.element must be replaced by document.getElementById
Konqueror	none	

## 2.4 Web Page (PHP)

Stephen designed and implemented this section.

### 2.4.1 Why we Required PHP

We required the use of PHP because we needed our html document to be able to interact with the database. PHP would also allow us to create a more secure site by allowing us to encrypt certain data i.e. passwords of the users. It allowed us to do easy error-checking of incorrect information, because of PHP we could easily return error messages stating where the user went wrong in filling out the ballot or entering wrong information at the login screen. Due to the fact that information transferred should not be able to be accessed easily, PHP naturally provided good security as anybody wishing to view the source code of the site would only see the HTML with all the sensitive pieces of code hidden away by the PHP.

### 2.4.2 Resources Required

- A computer with internet access
- Access to ocean
- Windows/Linux editor (TextPad)
- Php.net
- Tutorial on PHP (PHP book from the library and various online documentation)

### 2.4.3 Time Allocation

PHP was a new language to us all so it was decided that Stephen would take this side of the project. I decided to start learning this language before Christmas 2003 because I wasn't sure how difficult it would be. I gained some sort of backbone from one of my courses on databases which covered JDBC which is the java side of PHP. Both of these languages have similar aspects by the way they interact with mySql and Oracle. I started practising this when we started back in January 2004. Because there was a lot of ideas that could be added to our project to increase its performance PHP was continually used in the first week of February. One of the biggest problems which occurred was passing values between PHP and JavaScript, since PHP runs on the server and JavaScript runs at runtime. This problem nearly took up a week to overcome and we eventually solved this problem by passing values using JavaScript with a form and receiving from this form on a new page using PHP.

## 2.4.4 Detailed Design

The first part of the coding that was done was the login page.

There is three fields on the this page

VoterID

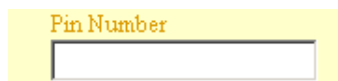
This is where the user types his ID



A screenshot of a web form with a yellow background. At the top, the text "VoterID" is written in orange. Below it is a white rectangular input field with a thin black border.

Pin Number

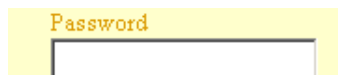
This is where the user types in his pin number which is given to the person before the elections occur.



A screenshot of a web form with a yellow background. At the top, the text "Pin Number" is written in orange. Below it is a white rectangular input field with a thin black border.

Password

The user is also given a password because this decreases the chance of an unauthorised user from entering the system by two fold. Also to increase the security more we also encrypted the password using Md5 encryption.



A screenshot of a web form with a yellow background. At the top, the text "Password" is written in orange. Below it is a white rectangular input field with a thin black border.

These values were then passed using a form on the login.php

(<http://student.cs.ucc.ie/~smm6/php/index.php>) page that passed these three values to the checker.php file that checked:

1) If all the fields were filled in and if there was a problem gave the following error message.



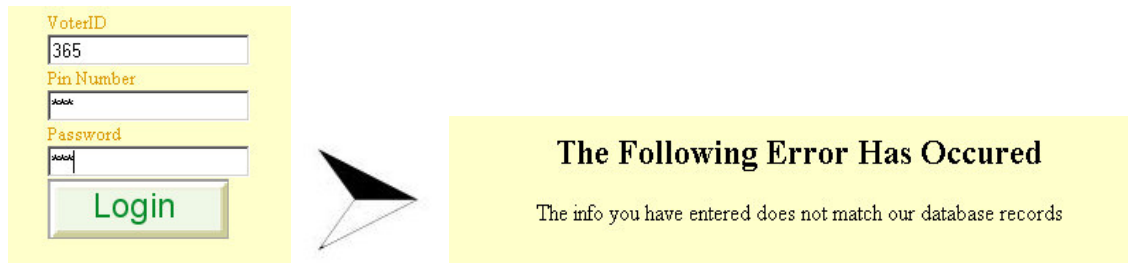
A screenshot of a web form with a yellow background. It contains three input fields labeled "VoterID", "Pin Number", and "Password" in orange. The "VoterID" field contains the text "1213". Below the fields is a green "Login" button. To the right of the form, a black arrow points to a yellow error message box. The error message box contains the text: "The Following Error Has Occurred", "You have not entered in all the required fields", and "Please login in above and Try again".

2) If the database records show that the person has voted already then the following message will occur



A screenshot of a web form with a yellow background. It contains three input fields labeled "VoterID", "Pin Number", and "Password" in orange. The "VoterID" field contains the text "20", the "Pin Number" field contains "1234", and the "Password" field contains "1234". Below the fields is a green "Login" button. To the right of the form, a black arrow points to a yellow error message box. The error message box contains the text: "The Following Error Has Occurred", "You have already cast your vote and cannot do so again".

3) If the database returns no match of the data entered then the following message will be displayed



The login.php file also contained code that displayed the current server time which would be used to record the current time and so if a user logged in wrong then he would have to wait a certain time before he would be allowed to try and log in to the voting page again.

15. February 2004 19:02:02

Because of time constraint I got the clock working but did not get a chance to implement the attempted login script.

If the user is successfully logged in then,

A session is started and some variables are passed in from db.php which contains details about the database connection and a session variable which states whether there is a session open already. If a session is already open then that session is destroyed so the session variables can be cleared. The password is encrypted using Md5 encryption.

Some session variables are declared such as:

Voter Id.

Pin Number.

Password.

Constituency Name.

Variables that will be used to show if there is certain types of elections on e.g. If there is a referendum running.

If this succeeds then the user is directed to the voting page.

The User enters the Voting Page,

A session is started on this page so this script can gain access to the session variables. The page welcomes the user and displays their constituency and their id by accessing the session variables that were set earlier on. The PHP code then checks the database to see if there is a general election on for this constituency and if:

There is no general election it will put null into the array and declare the session variable to state that there is no general election on at the moment.

If there is a general election on then it will fill arrays with candidate names, parties and the relevant images of the candidates.

It will do the following for presidential elections, referendums and general elections so you will have :

- a. general election candidate names, parties and candidate image arrays.
- b. presidential election candidate names, parties and candidate image arrays.
- c. referendum name arrays.
- d. 3 session variables stating whether there is any of the proceeding on at the moment.

These are then passed to the JavaScript code which handles the rest of the setup of this page.

The next section where PHP is used is the Confirm.php page where the following happens: A session is started on this page so this script can gain access to the session variables. Then PHP takes in variables from a form that has been passed to this page and collects this information and stores this in arrays.

Then these arrays are accessed and the data is transferred to the database using a series of mySql statements.

If this is done successfully then the user's session is cancelled and they are informed that their vote has been successfully cast.

### **2.4.5 Research**

Since we had never had any experience before with PHP and had not covered it in our course so far, Stephen decided that he would take that side of the project and learn the main techniques and methods needed so that the needs of our project could be fulfilled. He learned PHP through a library book and various other Internet locations on the web.

It took some time to work out the various ways PHP has of interacting with a database using mySql. Stephen looked at various other PHP scripts to get some idea of the best way to build the system.

## **2.5 Results Program**

Brian designed and implemented this section.

### **2.5.1 Why we Required Java**

We chose Java as the language to create/setup elections as it provided us with means to create functional graphical user interfaces for system users. It has allowed convenient access to the database for creating elections/referenda and accessing the results of these.

### **2.5.2 Resources Required**

- A computer with internet access
- Access to ocean
- Windows / Linux editor
- A computer with JDK 1.3.1 compiler and Java Runtime Environment
- Java 1.3.1 API documentation – <http://java.sun.com/j2se/1.3/docs/api/>

### 2.5.3 Time Allocation

Java was not exactly new us but we needed some time to research some of the more difficult aspects of developing java Graphical User Interfaces.

We spent the first week of January completing the design of the GUI, and by then we believed we had a reasonable understanding of what we wanted to do.

### 2.5.4 Detailed Design

**GENERAL ELECTION — 2002**  
**CONSTITUENCY OF GALWAY EAST**

<b>TOTAL ELECTORATE</b>	73,659
<b>INVALID BALLOT PAPERS</b>	452
<b>VALID POLL</b>	49,422
<b>NUMBER OF SEATS</b>	4
<b>QUOTA</b>	9,885

**NAMES OF CANDIDATES ELECTED**

- PAUL CONNAUGHTON (F.G.)
- JOE CALLANAN (F.F.)
- PADDY McHUGH (Non-Party)
- NOEL TREACY (F.F.)

NAMES OF CANDIDATES	First Count	Second Count	Third Count	Fourth Count	Fifth Count
	Number of Votes	Transfer of MacMeanmain's and Ní Bhroin's Votes and Result	Transfer of Mac an Bhaird's Votes and Result	Transfer of Burke's Votes and Result	Transfer of Connaughton's Surplus and Result
*BURKE, Ulick (F.G.)	6,941	+211 7,152	-293 7,445	-7,445	—
CALLANAN, Joe (F.F.)	7,898	+65 7,963	-352 8,315	+1,099 9,414	+952 10,366
*CONNAUGHTON, Paul (F.G.)	8,635	+131 8,766	-218 8,984	+4,987 13,971	-4,086 9,885
*KITT, Michael P. (F.F.)	7,454	+59 7,513	+198 7,711	+239 7,950	+293 8,243
MAC AN BHAIRD, Daithí (S.F.)	1,828	+240 2,068	-2,068	—	—
MacMEANMAIN, Manus (C.S.P.)	93	-93	—	—	—
McHUGH, Paddy (Non-Party)	7,786	+256 8,042	-539 8,581	+244 8,825	+1,056 9,881
NÍ BHRÓIN, Úna (G.P.)	1,022	-1,022	—	—	—
*TREACY, Noel (F.F.)	7,765	+82 7,847	-244 8,091	+635 8,726	+527 9,253
Non-transferrable papers not effective	—	+71 71	+224 295	+241 536	+1,258 1,794
<b>TOTAL</b>	<b>49,422</b>	<b>49,422</b>	<b>49,422</b>	<b>49,422</b>	<b>49,422</b>

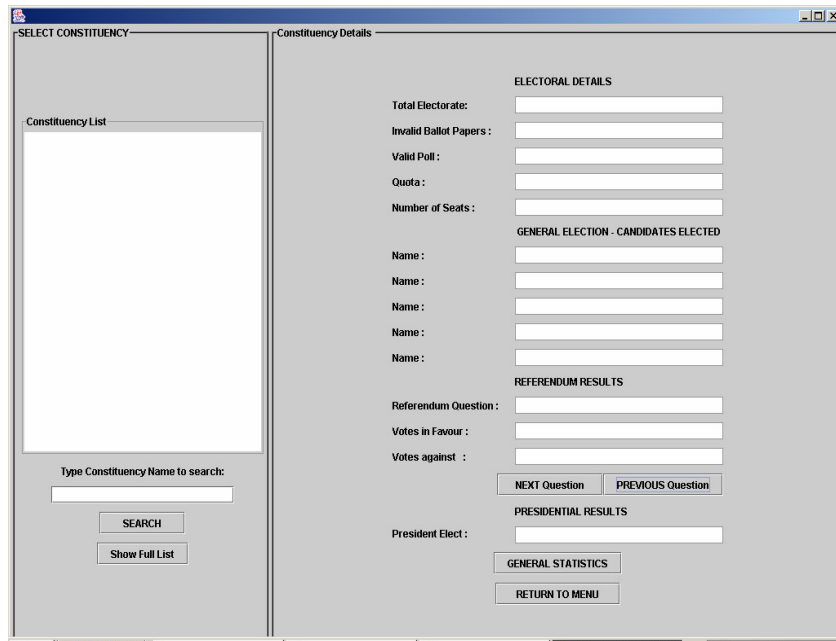
Ref: [www.irlgov.ie/oireachtas/press/GeneralElections2002.htm](http://www.irlgov.ie/oireachtas/press/GeneralElections2002.htm)

In designing the GUI that would represent the Election/Referendum results we hoped to achieve something similar to the above, if not in the same format, at least in that it should display the same essential information.

In making the Java GUI it was decided that to incorporate all the desired features would take an inordinate time in coding by hand.

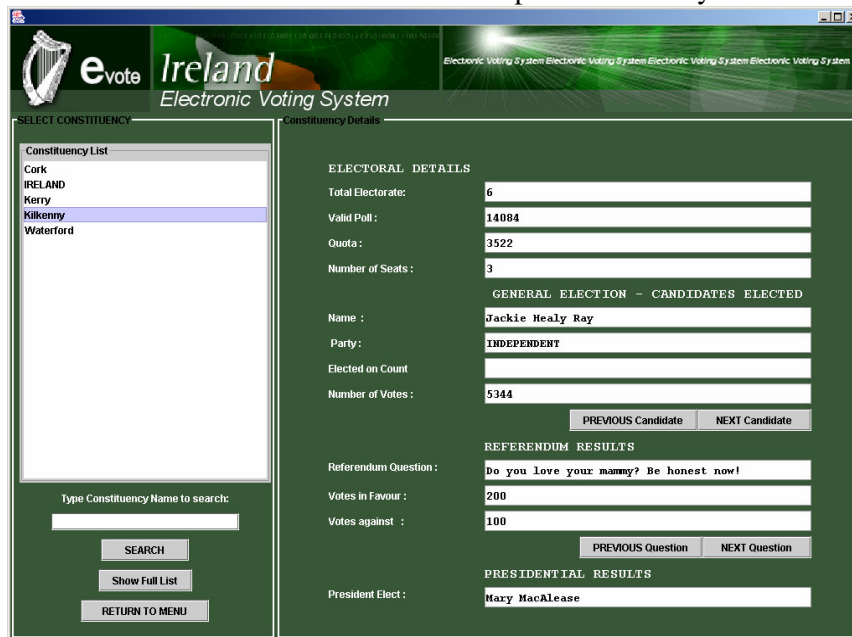
For this reason we hoped to use an industry standard product i.e. J-Builder to generate the java GUI code but we found that these types of products were largely unavailable unless we were willing to purchase the fully licensed software. Even if we had managed to avail of such software we would still have encountered difficulties in mastering what is a quite a complicated application which would certainly have taken a lot of time away from the other necessary coding.

Failing to find any alternative we began and continued to manually code the java GUI.



We decided to break down results by constituency and so we have represented results in this format. A list of constituencies will appear on the User interface in a scroll-pane so the user can select a constituency. Electoral details, General Election results, Referendum Results and Presidential Results are then displayed per constituency.

The screenshot above shows an earlier version of the main Results Graphical User Interface which had a more basic 'General Election' layout where all (five) of the elected candidates are displayed on screen simultaneously. The design was altered in order to accommodate a situation where more than five candidates are elected per constituency.



The completed GUI including alterations

## **Appendix E – Future Developments**

At present we realise our system doesn't contain all the elements we would desire. We have a number of additions and updates we would like to make in the future.

- The system should be implemented in both English and Irish. This can be done relatively easily with a little more time.
- System documentation in particular the Voter's Guide should be made available in Irish. This can also be done simply with a little time.
- The Setup and Results program should be protected by a login system. This is to prevent access to unauthorised personnel and will be implemented in the next version of the system.
- The passwords, pin numbers used both in the web login and the setup login mentioned above will be encrypted. They have not been in the prototype because of time constraints.
- A web based utility could be set up to allow voters to change their own pin numbers and passwords.
- Improved functionality of the start and end election buttons should be implemented. For example the connection from the web to the database could be disabled while elections are not running and only enabled when the start button is pressed.
- The above process could also be automated where elections are programmed to occur on set dates between set times by the Election Administrator.
- The Results facility could be improved to give a better break down of results for all candidates and parties constituency by constituency.
- A facility to print results and write results to file should be added to the system.
- The current system allows voters to login a number of times simultaneously on different browser windows. It still only allows the voter to cast their vote once. The system should be changed to only allow one login however.

## Appendix F – Learning Aspects

This was the first real Software Development project that all the group members experienced. It was also the first time we had to produce extensive documentation for a project.

During the completion of the project we learned through our experience of the following:

### **Project Management**

We learned that Project Management incorporates many different aspects. These aspects include project organisation, planning, good design, man management, time and resource management as well as a knowledgeable understanding of the project area.

### **Working as a Group**

One of the main things we learned was how to function as a group. This required organising and assigning tasks to members, taking into account each person's abilities and skills. Throughout the project we made sure we communicated any developments and issues to all members. This was achieved by having regular formal meetings where minutes were taken. Communication was also done via e-mail and when working together on coding etc. Learning to assign tasks in a fair way so as to maximise progress of the project was the most vital thing that we learned from working as a group.

### **Documentation**

This was the one aspect of the project which was new to all members. We had never documented our work before. From the outset we had to document all ideas and developments regarding the project. It was essential that the PMP and all other documents be updated as the project progressed.

### **Meeting deadlines**

We found that the setting of deadlines was hard due in part to the fact that this was our first group project and we didn't know how time consuming many areas were going to be. As time went by the deadlines for some parts of the project were pushed back. This led to some of the functions we desired for the project such as those mentioned in Appendix E being omitted from the current version. We learned that all deadlines should be definite and not approximate. We found that we should always concentrate on the more important and functional tasks first and afterwards time permitting, pursue further developments and enhancements.

### **Programming and Testing**

To ensure that our project was fully functional with as few bugs as possible we had to complete a great deal of testing. This meant that many bugs were detected during development, thus preventing them from becoming defects at the final delivery stage. Testing was to a large extent a new experience also as we would never previously have tested our code rigorously. Each group member's programming knowledge was greatly improved. Brian and David learned a great deal more about Java and in particular GUIs. Stephen learned PHP from scratch. Adrian learned much more JavaScript than he previously knew and Pádraig's database knowledge was improved by designing the one used in the project.