**Presidential Election Campaign Management**

Table of Contents

[Presidential Election Campaign Management 4](#_Toc437713276)

[1. Requirements 4](#_Toc437713277)

[2. Compliance Requirements: 5](#_Toc437713278)

[3. Reporting requirements: 5](#_Toc437713279)

[4. Use Cases 6](#_Toc437713280)

[4.1 Store candidate information 6](#_Toc437713281)

[4.2 Track candidate propositions (Expanded Use-Case) 7](#_Toc437713282)

[4.3 Display campaign events 8](#_Toc437713283)

[4.4 Store voter information 9](#_Toc437713284)

[4.5 Store voter's proposition 10](#_Toc437713285)

[4.6 Store voter identification information 11](#_Toc437713286)

[4.7 Track vendor payments 12](#_Toc437713287)

[4.8 Get popular propositions report 13](#_Toc437713289)

[4.9 Get new propositions report 14](#_Toc437713290)

[4.10 Get the donations received by the campaign during the last month. 15](#_Toc437713291)

[4.11 Track the payments made to vendors 16](#_Toc437713293)

[5. Entity Relation Diagram 17](#_Toc437713294)

[5.1 Entities 18](#_Toc437713295)

[6. Architectural Diagram 19](#_Toc437713296)

[7. Sequence Diagrams 20](#_Toc437713297)

[7.1 Enter Candidate Information 20](#_Toc437713298)

[7.2 Track Candidate Propositions 20](#_Toc437713299)

[7.3 Display Campaign Events 21](#_Toc437713300)

[7.4 Register Voter 21](#_Toc437713301)

[7.5 Track Voter Propositions 22](#_Toc437713302)

[7.6 Donations 22](#_Toc437713303)

[7.7 Vendor Payments 23](#_Toc437713304)

[8. Class Diagram 23](#_Toc437713305)

[9. Component Diagram 24](#_Toc437713306)

[10. Collaboration Diagram 25](#_Toc437713307)

[11. State Chart Diagram 26](#_Toc437713308)

[12. Code Design and Documentation 27](#_Toc437713309)

# Presidential Election Campaign Management

The website will allow the campaign organizers to keep track of the activities.

The website will support the following:

* Candidate Proposition management
* Voter registrations
* Voter Surveys
* Campaign Event Management
* Store rival candidate information
* Campaign Donor tracking
* Campaign Financial management

# Requirements

We want to develop an application that helps manage a Presidential Election Campaign for our candidate

The requirements for this application are as follows:

* 1. The system should be able to track information about other candidates.
  2. The system should allow to track propositions for the candidates. New propositions can be added anytime.
  3. The application will be used by the Campaign staff. The Campaign staff is divided into the following roles:
     1. The Campaign manager
     2. IT staff
     3. Finances
     4. Legal and Compliance
  4. The application should display upcoming campaign events.
  5. The system will be used to store voter survey information. This survey data will be used by the campaign staff.
  6. A survey will contain the Voter responses to candidate propositions.
  7. The system should store voter identification information as well as provide a way to update it.
  8. Voters will be divided into categories deemed important for the campaign.
  9. The campaign will be funded using donations from the voters. These donations will be accepted only from registered voters.
  10. Campaign operations will be supported by external vendors. The system should keep records of vendor information and payments.

# Compliance Requirements:

* 1. For legal and compliance, the system must require donor's tax identification number.
  2. Financial reporting as described in the reporting requirements below.

# Reporting requirements:

* 1. Monthly report on popular propositions in voter’s surveys.
  2. Monthly report on voter's new suggested propositions.
  3. Monthly report on donations.
  4. Monthly report on payments.

# Use Cases

The application requirements are broken down into the following use cases:

## 4.1 Store candidate information

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* Campaign Management Staff:

The staff should be able to enter, update and read the competing candidate's information.

* The system:

Stores the information as provided and makes it available.

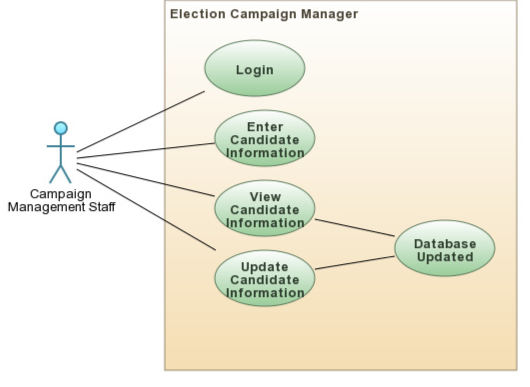
**Pre-Conditions:**

* Staff member has logged in with his credentials

**Post-conditions:**

* Information is persisted.

**Diagram:**



## 4.2 Track candidate propositions (Expanded Use-Case)

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* Campaign Management Staff:

The staff wants to see what propositions are currently supported by the candidate.

Based on the feedback from surveys, staff can enter new proposition to the list of propositions supported by the candidate.

* The system:

Stores the information as provided.

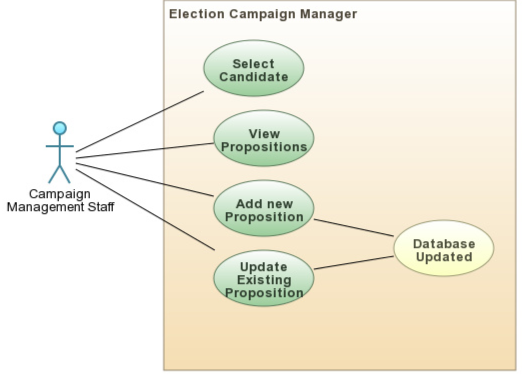
**Pre-Conditions:**

* Staff member has logged in with his credentials

**Post-conditions:**

* Information is persisted.

**Diagram:**



**Typical course of events:**

* 1. [Actor: Staff] The campaign manager staff logs on to the website.
  2. [System Response] The system displays the home page.
  3. [Actor: Staff] Views the candidate’s propositions.
  4. [System Response] The system displays the current propositions for selected candidate.
  5. [Actor: Staff] Adds a new proposition.
  6. [System Response] Form to enter new proposition is displayed.
  7. [Actor: Staff] Enters the new proposition description and submits
  8. [System Response] The new proposition is entered in the database and is displayed in list of propositions.
  9. [Actor: Staff] Associates this new proposition to the candidate by adding it to the candidate’s propositions.

## ­4.3 Display campaign events

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Voter

**Stakeholders and interested parties:**

* Voter

The voter wants to see the upcoming campaign events.

* The system:

Wants to provide a list of upcoming campaign events to any user.

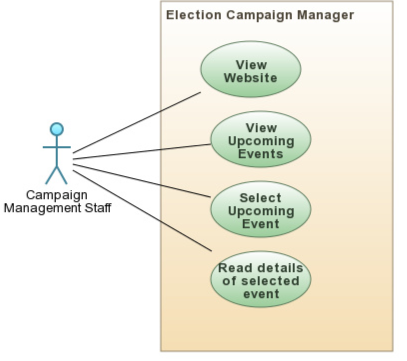
**Pre-Conditions:**

* The voter visits the web application.

**Post-conditions:**

* None

**Diagram:**



## 4.4 Store voter information

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Voter

**Stakeholders and interested parties:**

* The Voter:

Give survey and register the information.

* Campaign Management Staff:

The staff should be able to enter, update and read the voter's information.

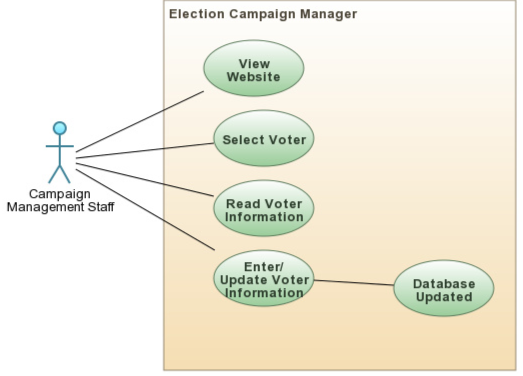
**Pre-Conditions:**

* Staff member has logged in with his credentials

**Post-conditions:**

* Information is persisted.

**Diagram:**



## 4.5 Store voter's proposition

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Voter

**Stakeholders and interested parties:**

* The Voter:

Wants to submit responses to the candidate proposition survey.

* The system:

Stores the survey responses.

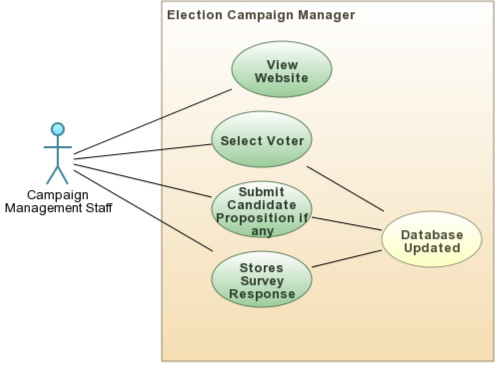
**Pre-Conditions:**

* The voter is registered.

**Post-conditions:**

* The voter survey responses are recorded.

**Diagram:**



## 4.6 Store voter identification information

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* The Voter:

The registered voter has provided his identification information to the management staff.

The registered voter has provided his information to campaign management staff to update the record

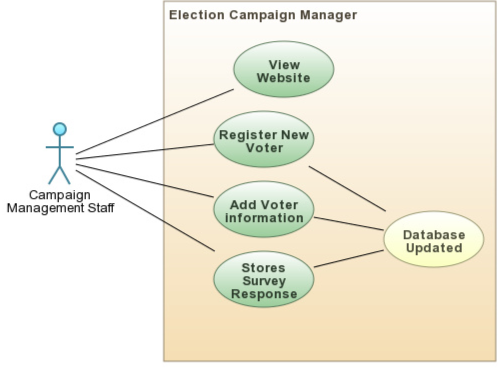
**Pre-Conditions:**

* Voter has provided information to the management staff

**Post-conditions:**

* Campaign management staff has recorded the voter's information or updated it

**Diagram:**



## 4.7 Track vendor payments

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* Campaign management staff

The staff wants to enter the payment information for vendor services.

The staff wants to retrieve the information for reporting purposes.

* The system:

Persists the information in a format which satisfies the reporting requirements.

**Pre-Conditions:**

* Staff member has logged in with his credentials.

**Post-conditions:**

* Information is persisted.
* Information is retrieved accurately.

**Diagram:**

## 

## 4.8 Get popular propositions report

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* Campaign management staff

Wants to get a report on the popular propositions in the surveys conducted during the previous month.

* The system:

Gets the popular propositions in the order of decreasing popularity.

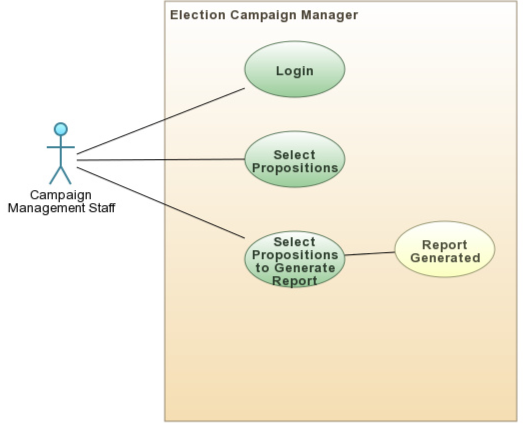
**Pre-Conditions:**

* Staff member has logged in with his credentials.
* The survey responses are persisted accurately.

**Post-conditions:**

* The required report is provided successfully.

**Diagram:**



## 4.9 Get new propositions report

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* Campaign management staff

Wants to get a report on the new propositions suggested in the surveys from the last month.

* The system:

Gets the list of new propositions.

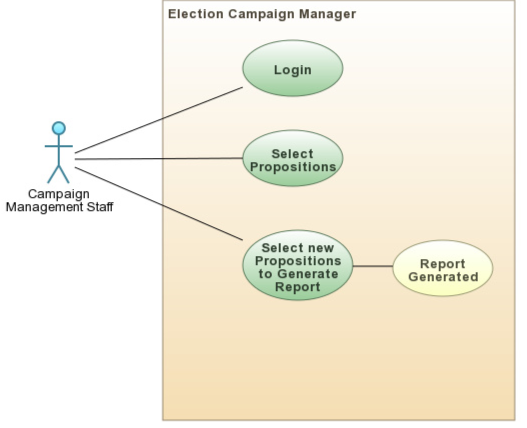
**Pre-Conditions:**

* Staff member has logged in with his credentials.
* The survey responses are persisted accurately.

**Post-conditions:**

* The required report is provided successfully.

**Diagram:**



## 4.10 Get the donations received by the campaign during the last month.

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* Campaign management staff

Wants to get a report on the donations made to the campaign during the last month.

* The system:

Gets the list of donations and provides a report.

**Pre-Conditions:**

* Staff member has logged in with his credentials.
* The donation entries are persisted in the system accurately.

**Post-conditions:**

* The required report is provided successfully.

**Diagram:**

## 

## 4.11 Track the payments made to vendors

**Scope:** The system under design

**Level:** User-goal level

**Primary Actor:** Campaign management staff

**Stakeholders and interested parties:**

* Campaign management staff

Wants to get a report on the donations made to the campaign during the last month.

* The system:

Gets the list of donations and provides a report.

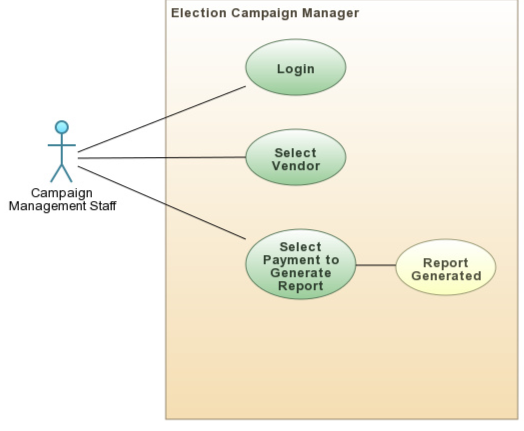
**Pre-Conditions:**

* Staff member has logged in with his credentials.
* The donation entries are persisted in the system accurately.

**Post-conditions:**

* The required report is provided successfully.

**Diagram:**



# Entity Relation Diagram

## 5.1 Entities

* **Voter**

Voter entity stores the voter’s information from survey.

* **Candidate**

Candidate entity stores the information of the candidate, his propositions.

* **Donor**

Donor entity stores the donor's name and tax ID number.

* **Category**

Category entity stores the various categories that classify the voters.

* **Proposition**

Proposition entity will describe the Candidate Propositions, Competitors Propositions, Propositions suggested by the voters.

* **Donation**

Donation stores the Donation amount and Donation Date.

* **Vendor**

Vendor Entity will store vendor’s information. Their name, service and Tax ID number.

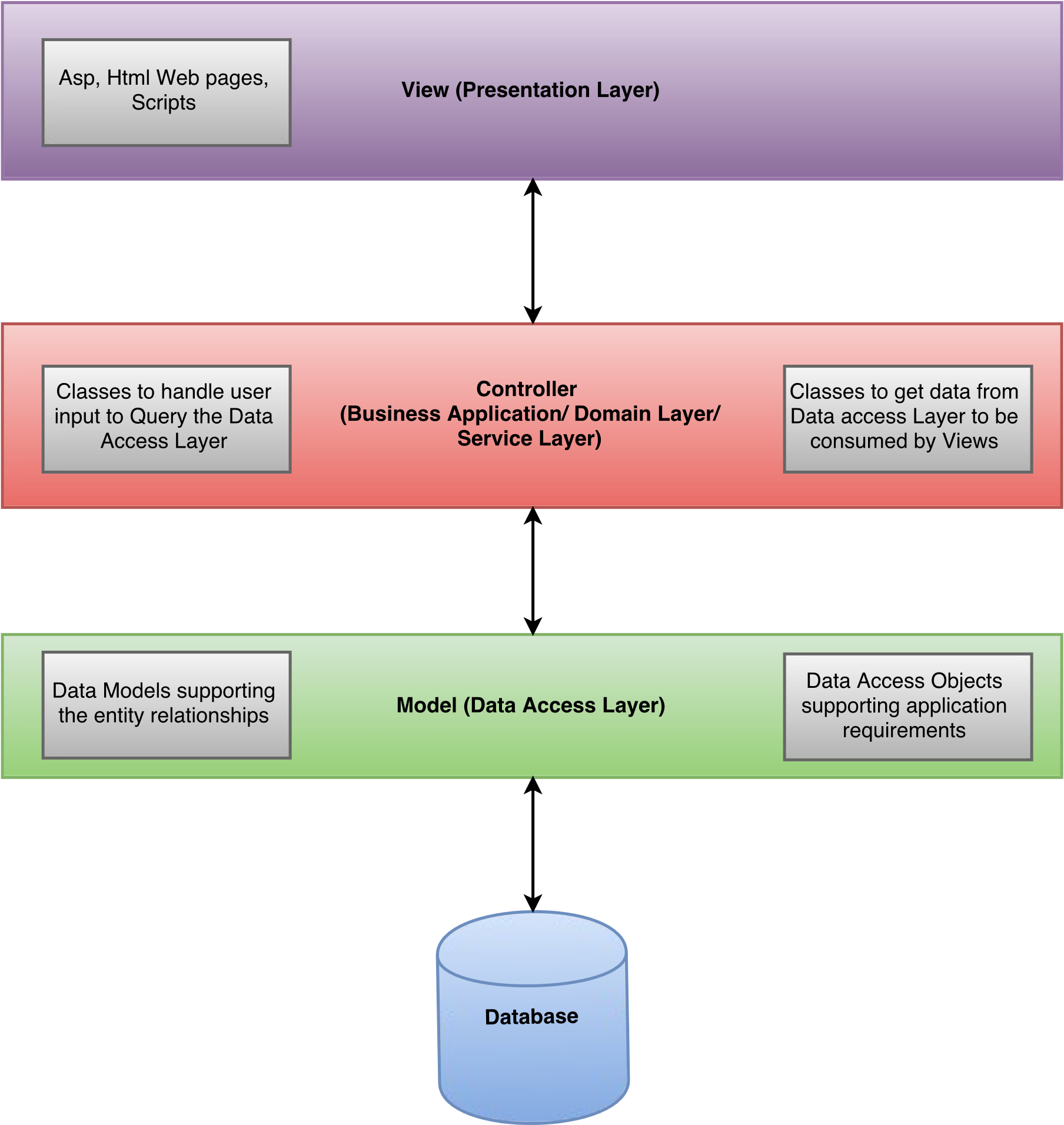
* **Campaign Event**

Campaign Event entity stores the campaign event schedule.

# Architectural Diagram

The architecture is based on the MVC pattern

The web application will be implemented in ASP.Net

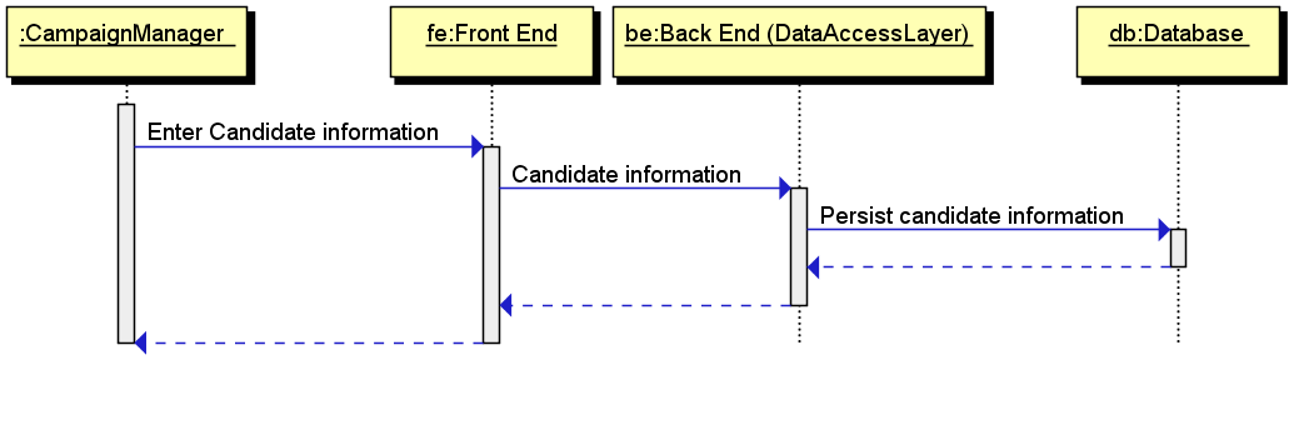


The architecture is a classic three-tiered architecture.

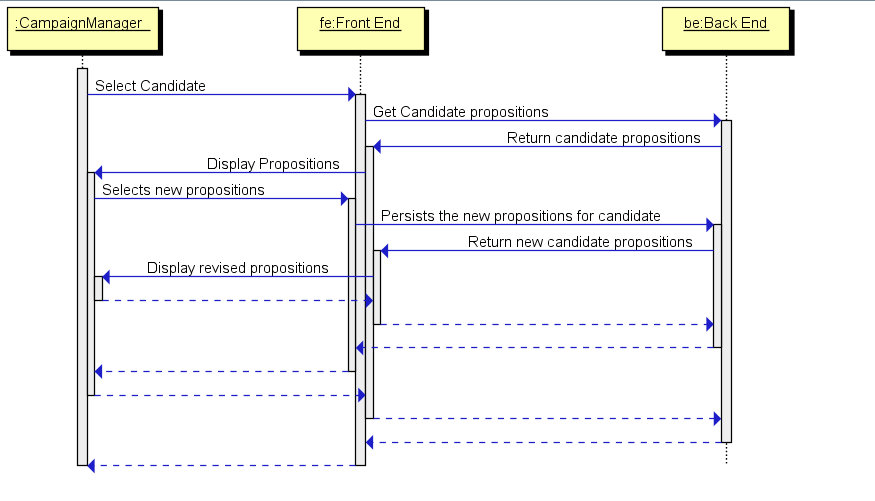
# Sequence Diagrams

Following are the sequence diagrams for the system requirements:

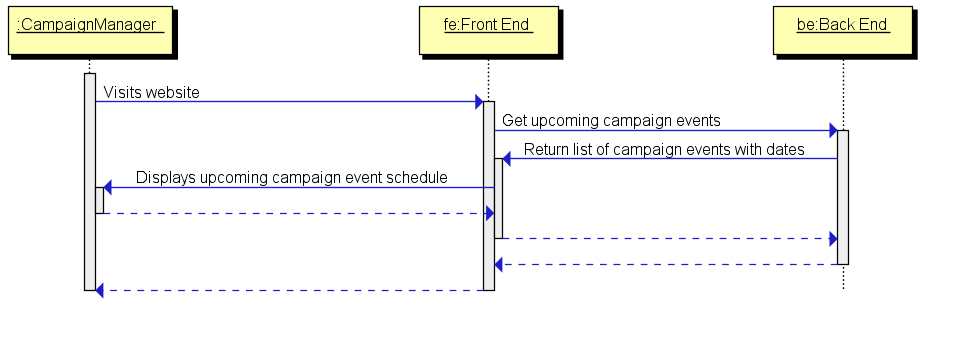
## 7.1 Enter Candidate Information



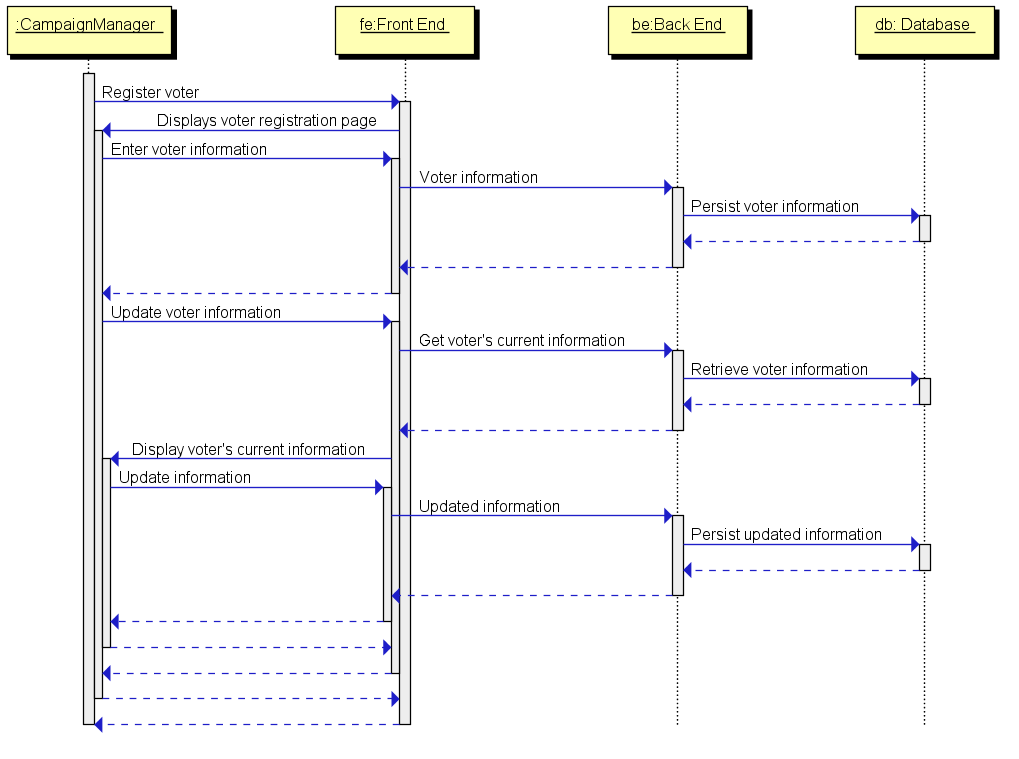
## 7.2 Track Candidate Propositions



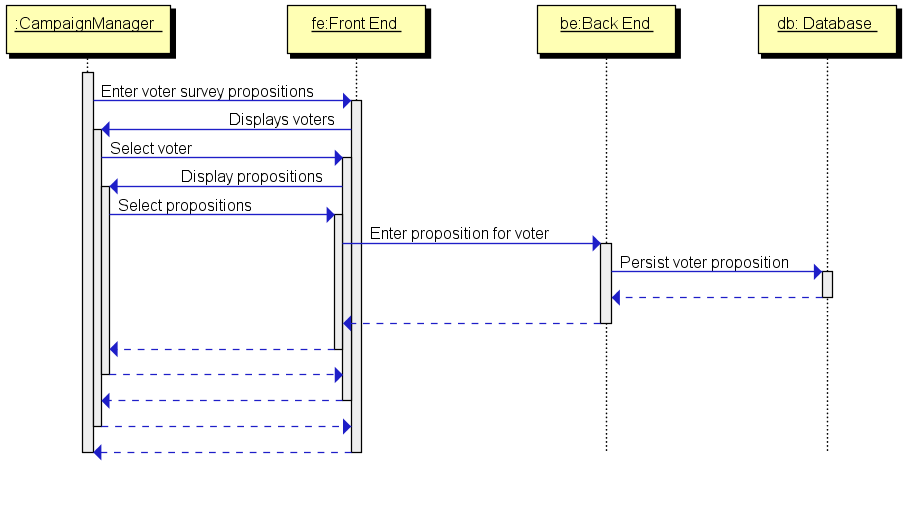
## 7.3 Display Campaign Events



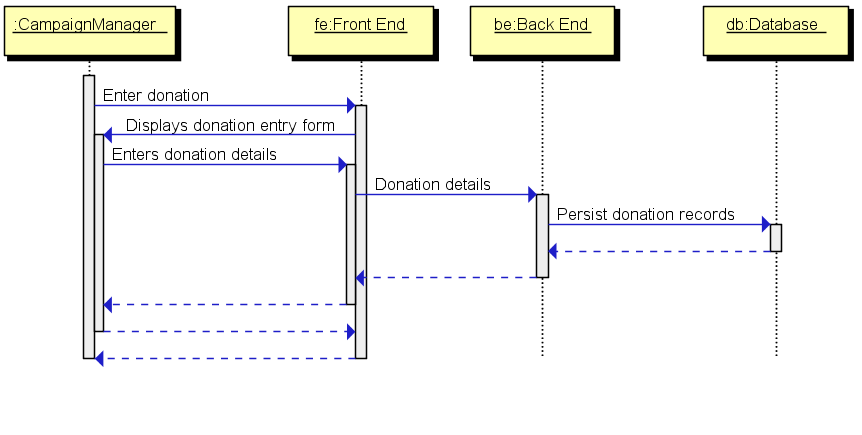
## 7.4 Register Voter



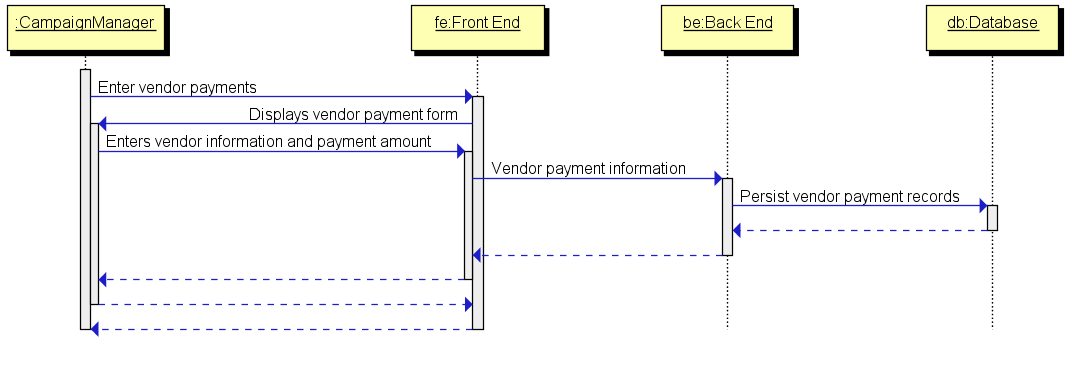
## 7.5 Track Voter Propositions



## 7.6 Donations



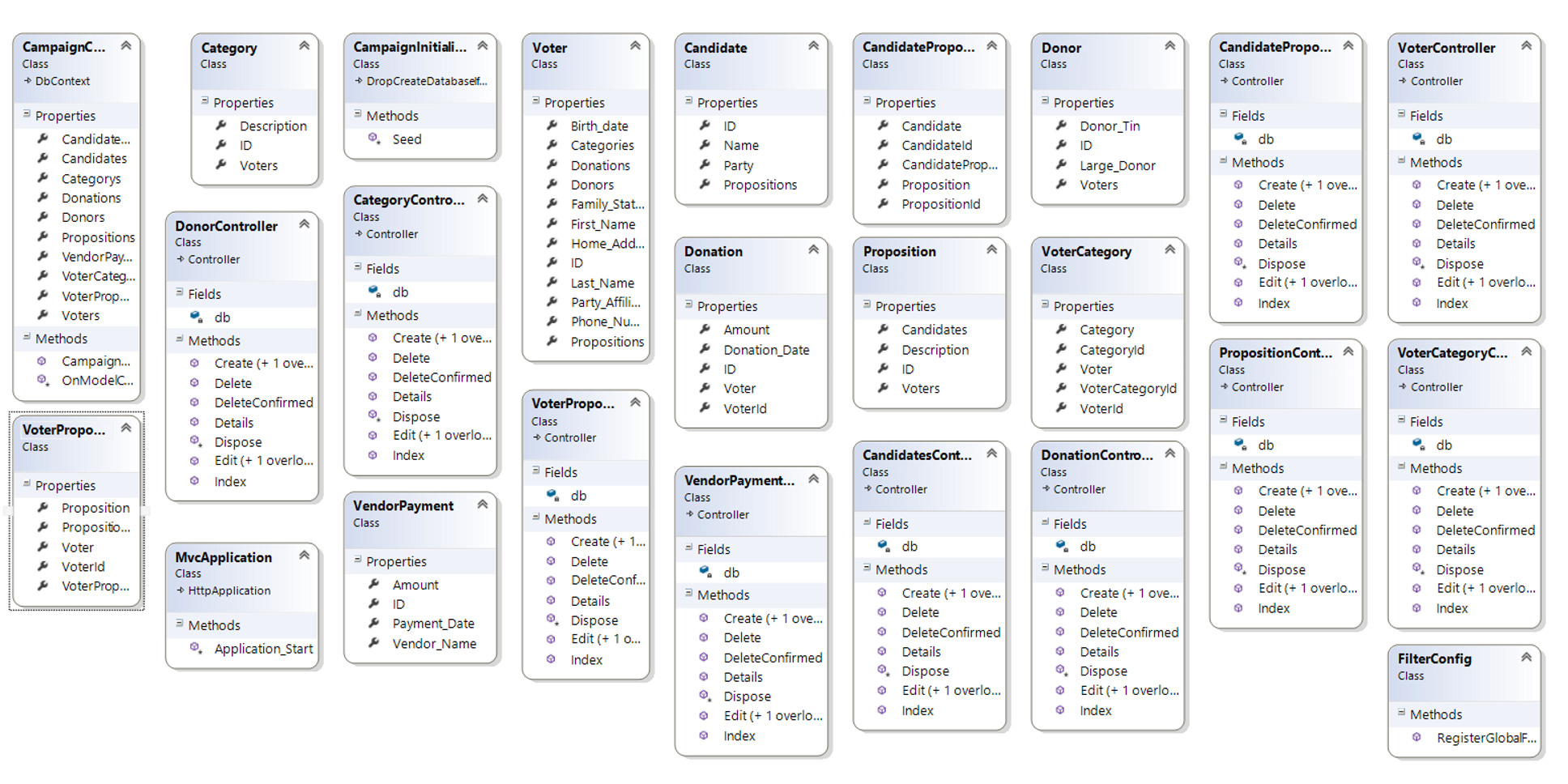
## 7.7 Vendor Payments



# Class Diagram

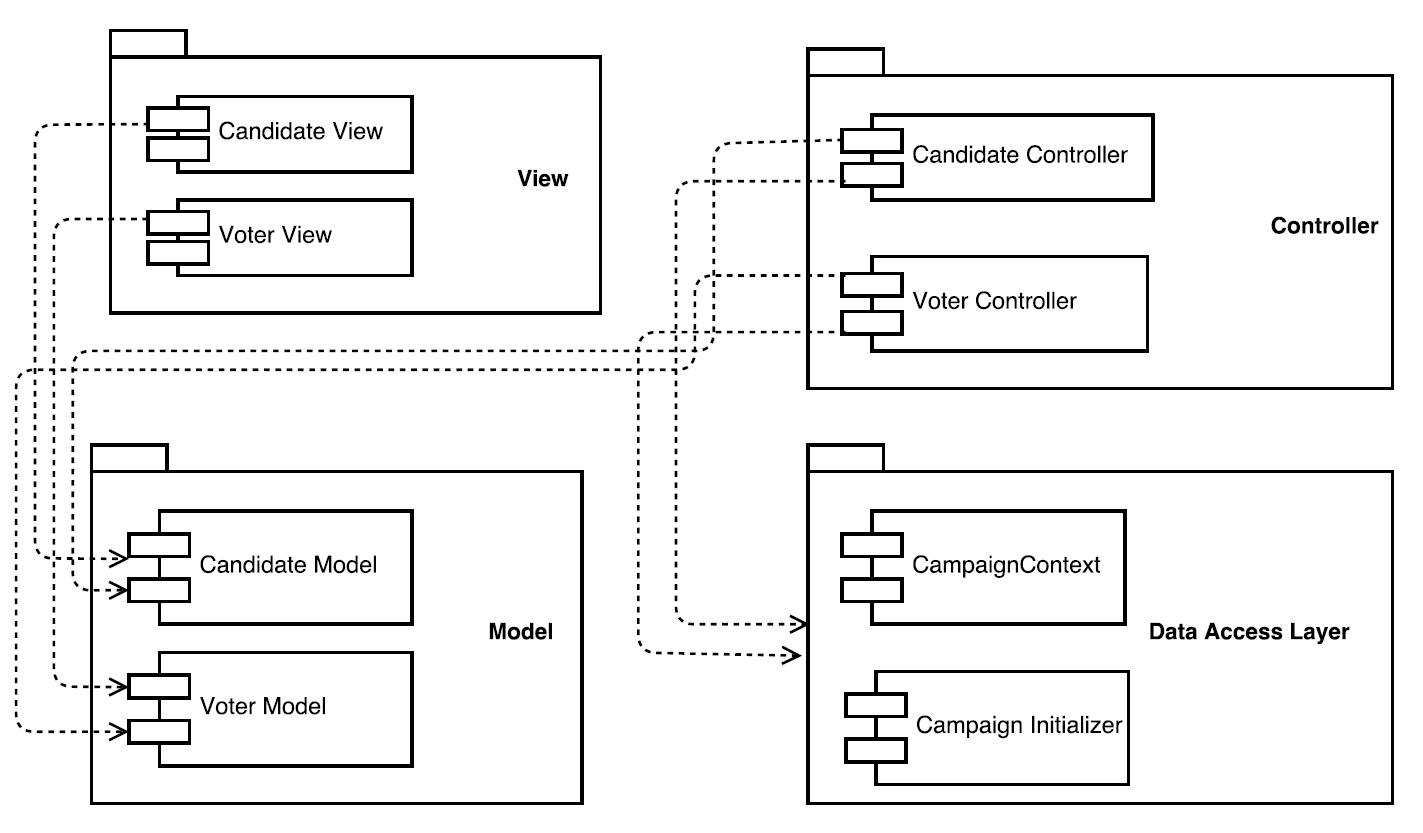
From the entity relationship diagram, the model classes are built that results in the class diagram below. The relationship between the model classes are visible in the entity-relation diagram. The diagram below also shows the controller, data access layer and application driver classes.

For each class has its attributes and methods.



# Component Diagram

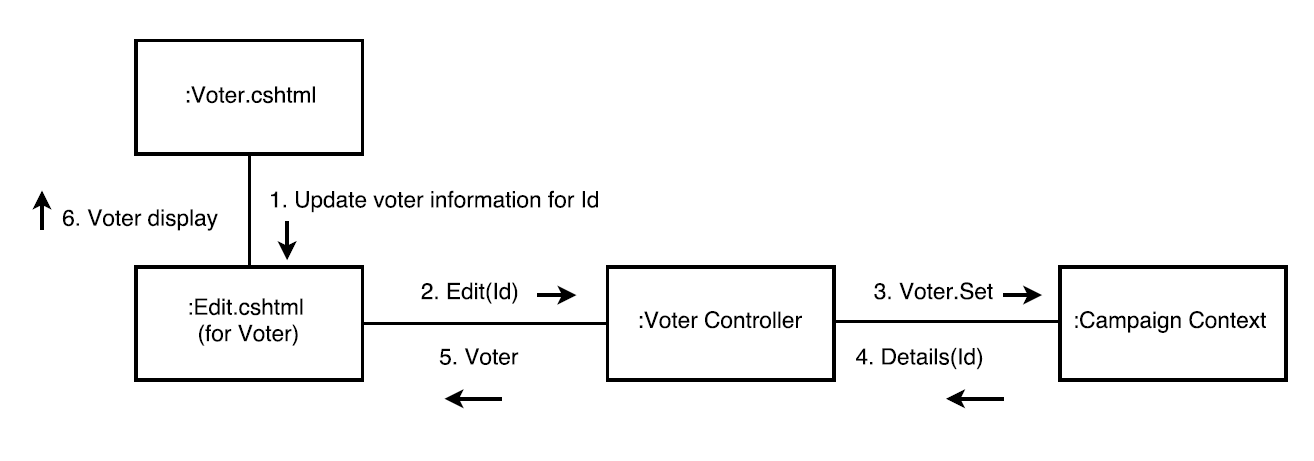
The figure below shows a component diagram. The MVC architecture is broken down in system components. This diagram below shows these components and some sample entities to illustrate the relationship between the classes and components.



Using the above diagram, all the classes in the class diagram can be assigned to the respective components.

# Collaboration Diagram

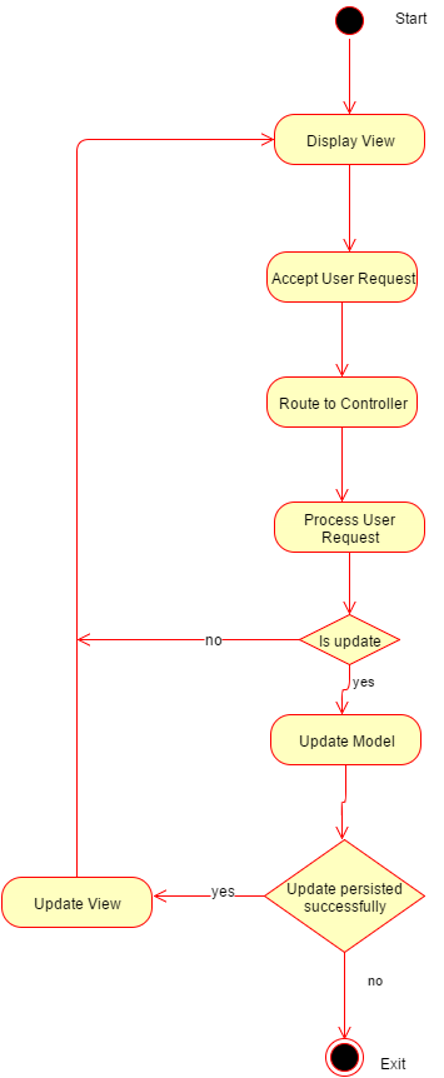
The collaboration diagram below illustrates how the classes collaborate for the ‘Update Voter Information’ scenario.



Similarly, other scenarios can be illustrated using collaborations diagrams.

# State Chart Diagram

The web application architecture is MVC (Model-View-Controller). The state transitions between these components and a database backed application are captured in the diagram below. This is a generic state transition diagram.



# Code Design and Documentation

The code contains comments and description for operations and entities.

Entities are designed with the GRASP principles in mind to create highly cohesive models and low coupling.

System events are handled by controllers that are assigned a specific responsibilities.

The controllers are experts in handling the events on the views for each model. Thus the low coupling between classes enables easy enhancements.