# Object Oriented Programming Assignment 1

Voting System by Zack Jagger

[**Object Oriented Programming Assignment 1**](#_n5tf9imj6m03) **1**

[Overview](#_7xfcg88amx7) 2

[The User & Their Requirements](#_nin6ka7ik186) 2

[Proposed Technologies](#_o3vym2b22c58) 2

[Initial Class Diagram](#_53lukptzrt7y) 3

[Final Class Diagram](#_blyj80q89lz0) 4

[Class Diagram Updates & Justifications](#_qw6x3g2dpfhh) 5

[Sequence Diagrams](#_96y3bo38a6iu) 6

[Login](#_eo557qnvi04p) 6

[Create Election](#_dtpw55siwnvg) 7

[Submit Vote](#_nljr0uk85wlb) 7

## 

## Overview

SBMM is looking to develop a system that supports electronic voting in person and remotely, whilst keeping up security and privacy standards that are currently in place with traditional voting systems. I will be creating a prototype for this software solution as a part of the first stage of bidding in order to get the contract to develop this application for SBMM. Throughout this document, I will be analysing the different technologies available to create this application, explaining who the target audience for this application is, as well as their requirements for the application, and disclosing why I have made certain decisions whilst developing the application.

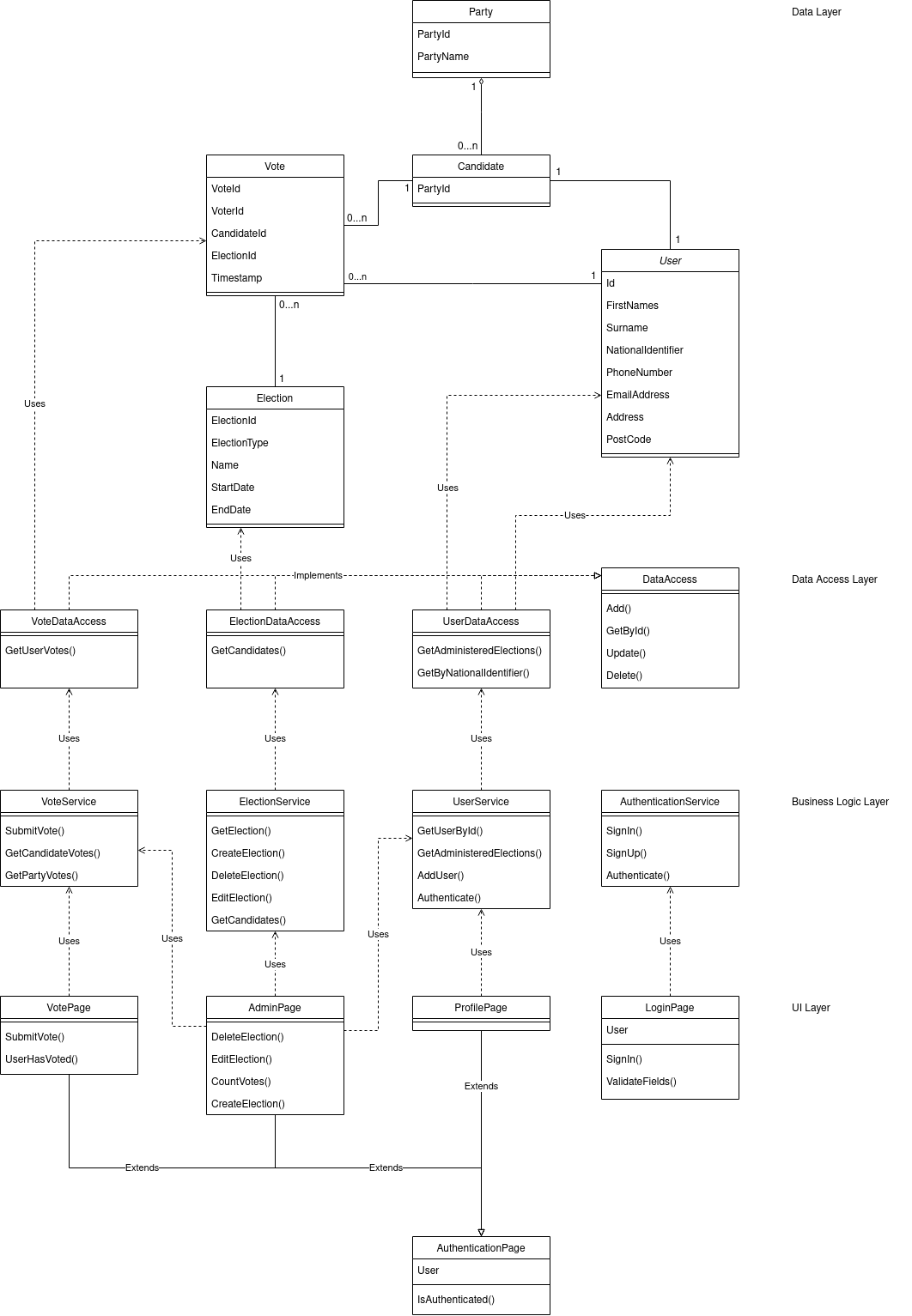
## The User & Their Requirements

There are two main stakeholders to keep in mind whilst designing and developing this application: voters & government officials. Voters are the main users of this application, meaning that the overall user interface and user experience should be designed to give the voters the best experience possible. Therefore, the design should be familiar, meaning that it should represent existing ballot cards, in order to avoid confusion. Additionally, it should be simple to create an account and ensure that their votes are registered correctly. On the other hand, this application is going to be presented to government officials, in order to persuade them to switch over their current systems to this more modern system. This means that the application needs to look professional, but more importantly it needs to offer functionality that will make the counting of votes easier, as well as ensuring that the votes are legitimate, private, and secure.

## Proposed Technologies

This application needs to be available to as many people as possible across all devices, including Apple, Android, & Windows devices. Therefore, I will be developing a web application, as these do not need to be compiled into executables, which are unique to each operating system. The framework that I will be using for the web application will be Blazor Server in C#. I have chosen to do this in C# as the team at Redev are very experienced with this language, meaning that there are many people that would be able to update and maintain the application. Additionally, I have chosen Blazor Server over Blazor Web Assembly and regular ASP.NET websites because all the C# code will be executed from a server, as opposed to executing it directly from the user’s web browser, meaning that the application will be more secure.

## Initial Class Diagram



## Final Class Diagram

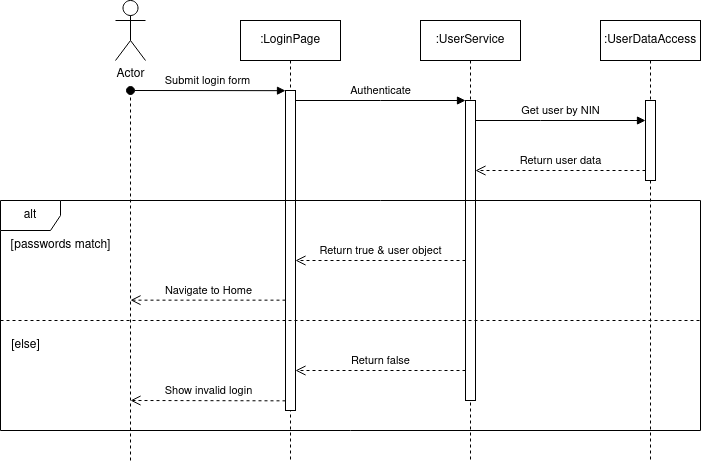
## Class Diagram Updates & Justifications

Throughout the development process, I realised that there were some aspects that I did not consider whilst designing the application. Therefore, the final class diagram of the application differs from the diagram created before development. I have composed a list of the changes in the class diagram along with the justifications for each one.

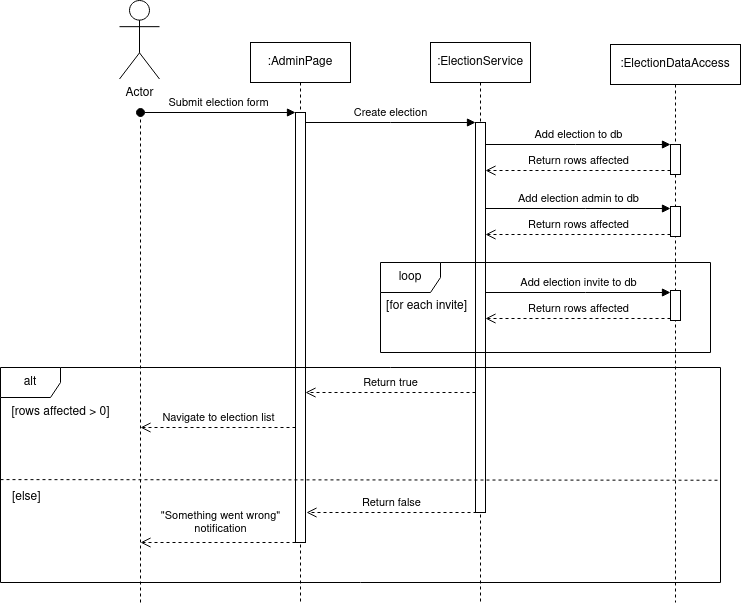
| Change | Justification |
| --- | --- |
| Removed Candidate Class | Unable to add functionality for parties due to time constraints and therefore candidate child class was unnecessary. |
| Removed Authentication Service Class | All functions within the Authentication were going to interact with the UserDataAccess. Therefore, I felt it was better to move these functions to the UserService so that all user functions are in one place. |
| Login Page - ValidateFields split into two methods to validate fields | This means that the user's inputs are validated on input as opposed to when the form is submitted. |
| Admin Page - Panel functions to change how the UI is displayed | This was mainly for aesthetic and UX reasons, as I felt it was better to display to the user different panels as opposed to all the admin options at once. |
| Admin Page - Invite functions added | To invite candidates to an election, instead of using private data like their National Insurance Number, they use an email. This requires new invite functionality on the admin page. |
| Moved UserHasVoted from Vote Page to Vote Service | Moving this logic to the vote service makes it more reusable. |
| Authentication Page - Moved storing user object to storing just the id | Storing just the id is much safer as it means that private user data isn’t always on the device whilst logged in. |
| Vote Service - Count votes for the entire election function. | Easier than calling the get candidate vote count function multiple times. |
| Election service - Added GetByNation | Required to show the users what elections are available to them. |
| Election Service - Get Invites | Required for the new candidate invite logic. |
| Election Service - Add & Delete Candidate | Required due to the ability to update elections, adding or removing candidates. |
| User Service - GetByEmail | Checks if a user exists with a given email. This is used so that no two accounts can be created with the same email address. |
| UserService - Invite functions | Required for the new candidate invite logic. |
| ElectionInvite - New model used to store details of who has been invited to be a candidate in an election. | Required for the new candidate invite logic. |
| ElectionInviteUpdate - Tracks changes to the invite table. | Required for the new candidate invite logic. |

## Sequence Diagrams

### Login



### Create Election



### Submit Vote

### 