Smart Voting Group LLC

Smart Voting

Project Closure Report

Version 1.0

2022 April 3

Project Name & Title

Smart Voting

Project Domains

Main Website	API Explorer	GitHub Organisation
smartvoting.cc	api.smartvoting.cc	github.com/smartvoting

Project Team

Stephen K. Davis	101294116	CRN ID: 49741-202102	stephen.davis@georgebrown.ca
Matthew Campbell	101289518		matthew.campbell@georgebrown.ca
Satabdi Sangma	101287632		satabdi.sangma@georgebrown.ca
Michael Sirna	101278670		michael.sirna@georgebrown.ca

Company Name & Contact

Not Applicable - Not partnered with a stakeholder or industry organisation.

Reason For Project Closure

End of academic term at George Brown College.

Project Start Date

Monday, September 20, 2021

Project End Date

Sunday, April 3, 2022

Client Acceptance Date

Not Applicable - Not partnered with a stakeholder or industry organisation.

Project Description

The Smart Voting platform is a secure web application that allows for elections to be held digitally, all while ensuring the integrity of the ballots. Smart Voting utilises a ledger database to securely and transparently keep track of votes as they are cast. As with all elections, ballots must be anonymous, but valid. This means checks must be done to make sure the voter is legally authorised to cast a ballot. Additionally, election officials and parties want to know as much data as possible about how and where people voted. There are various unique tools built into the system to assist with interpreting this data. Some of these tools will be publicly available and others will only be accessible by election officials and parties.

Closure Activity Confirmation

Project Considered A Success: Yes. The team considers this project a success.
Comments: We will not be celebrating the success of this project due to the ongoing global COVID-19 pandemic. Even though all team members are fully vaccinated, we wish to reduce the likelihood of contracting the virus. We have decided to celebrate at a later time when we deem it is safer to do so.

Release of Hardware & Software

React Application

As of 2022 April 3, our React front end application has been deployed to the Amazon Web Services App Runner environment. It can be viewed at smartvoting.cc.

.NET 6 Web API

As of 2022 April 3, our .NET 6 Web API has been deployed to the Amazon Web Services App Runner environment. It can be explored using Swagger UI at api.smartvoting.cc.

PostgreSQL

As of 2022 April 3, our PostgreSQL database has been deployed using Amazon Web Services RDS platform. It has successfully been linked to the .NET 6 Web API.

Amazon DynamoDB

As of 2022 April 3, our DynamoDB NoSQL database has been deployed on Amazon Web Services. It has successfully been linked to the .NET 6 Web API.

Amazon Quantum Ledger Database

As of 2022 April 3, our QLDB NoSQL database has been deployed on Amazon Web Services. It has successfully been linked to the .NET 6 Web API.

Important Note

All of the above resources will be deployed live until 2022 August 31. On this date, everything will be removed from Amazon Web Services to end the ongoing running costs. Additionally, on this date, the domain will be set up to redirect to the GitHub organisation where the source code, datasets and documentation will be available for viewing.

Project Archival List

All Smart Voting information will be stored on the Smart Voting GitHub Organisation.

Project Resource	GitHub Link
Smart Voting React Application	View Repository
Smart Voting .NET 6 Web API	View Repository
Smart Voting NodeJS API (deprecated)	View Repository
Smart Voting PostgreSQL Datasets	View Repository
Smart Voting Project Sprint Documents	View Repository

Reusable Components & Tools Developed

Component #1 – Base Controller

In the web api, we utilised a base controller that allowed us to store common methods that needed to be accessible throughout the api. This reduced the amount of repeated code we had and improved efficiency. An example of a method stored here is our method to send emails through Amazon Simple Email Service using the SMTP protocol.

Project Value & Benefits

Value & Benefit #1 – Future Possible Applications

As the world is becoming more connected and dependent on technology, voting in governmental elections is bound to eventually modernise. Blockchain and ledger databases will play a key role in this new era of voting. These technologies will aid in election security and integrity. The Smart Voting project is one example on how ledger databases can be implemented in an online election.

Value & Benefit #2 – More Accessible to People

COVID-19 showed the world how a global event can impact every person on Earth. In most democratic countries, elections must be held every four to six years. Having a global event such as a pandemic can make this voting process less accessible, especially to people who are at higher risk. Having an electronic voting system that a person can complete anywhere makes voting more accessible to people. Statistically, this accessibility encourages more people to participate in elections.

Lessons Learned

Lesson #1 – Utilise Familiar Frameworks Over Popular Frameworks

In the early development stages of this capstone project, we opted to use a NodeJS and Express API backend. However, as we were all still new to this technology, there were many struggles and difficulties. About half way through our development timeline, we decided to switch to the .NET 6 Web API framework with C#. Although half the team was not very strong with C#, they focused on the front end. This left those who were strong in C# and .NET to implement a Web API for our back end. We managed to accomplish our core tasks and some nice-to-have tasks with C# and .NET in about 28 days.

Lesson #2 – Know Your Goalposts

With any project such as Capstone, a major bottleneck and determinant of success is communication within the team. While we found this generally was not too much of an issue, some of our goals and ideas were easily misunderstood and should have been elaborated further way earlier on in the project. Because of this, many parts of the project had to be omitted not only due to time constraints but also because they were outside the realm of possibility for the project.

Lesson #3 – Learn When to Quit

Tying into both lessons #1 and #2, stubbornness to drop something that was not working was a large obstacle in the development process. While it is usually depressing to drop something worked on for so long, to continue the project and not waste time it is sometimes necessary to know when to stop working on something, and try it a different way. While we were generally comfortable with dropping certain aspects of our plans because of time constraints, this was by the end of the project where we were dropping because of conflicts with APIs and lack of time. In hindsight, we should have known to stop much earlier and only if and when we had time should we have gone back.

Best Practices

Item #1 – Utilising Data Transfer Objects

While each table has a mapped object in the API, some contents of those mapped objects contain sensitive records, such as entry ID keys or password hash values. As some of this information should not be publicly available, we implemented the usage of data transfer objects that would allow the API to function and return the data requested by the user or front end application without exposing any sensitive information. DTOs also allowed us to control the exact route input and return value patterns.

Item #2 – Utilising a Base Controller

As our web API utilised controllers instead of a minimal API format to better organise our routes and actions, we implemented a BaseController. By using a BaseController, we stored various methods and variables that could be accessed by any controller. This allowed us to reduce the amount of repeated code and the possibility of errors.

Item #3 – Database Relations and Auto Generating Keys

One good practice when designing and managing a database is to use a UUID or Integer as a primary key. A table should never use a user-entered string as a primary key, as it prevents issues in the long term. Most of our primary keys are of the UUID format but some are Integers. We used UUID where we did not know the number of possible rows and Integers where we could reasonably foresee the possible number of rows.

Item #4 – Dynamicity and Maintainability

A main goal with the front-end was to try and make the code as dynamic and maintainable as possible, so that future members (or other members working) would not need to edit the same information in multiple files. Most files on the system use side components and objects to try and mitigate the amount of code we were writing and to make changes in the code as easy as possible.

Item #5 – Simplicity is Key

Tying into Item #4, simplicity should be a bigger part of the front-end development process. Hypothetically, front-end developers should be striving to have as little work as possible. The more work a front-end developer is putting into one page, the more complex that page is going to be for both them and the user. This is not to say that a lot of work should not be done to make a page look nice or perform a certain way, but if there are too many things going on at once it can be confusing for the users. Less is more in this case.

Prepared By

Stephen K. Davis	*stephenkdavis@outlook.com*	2022 April 3
Matthew Campbell	*matthew.campbell@georgebrown.ca*	2022 April 3
Satabdi Sangma	*satabdi.sangma@georgebrown.ca*	2022 April 3
Michael Sirna	*michael.sirna@georgebrown.ca*	2022 April 3

Handover Approvals

Not Applicable - Not partnered with a stakeholder or industry organisation.

Project Closure Approval

Not Applicable - Not partnered with a stakeholder or industry organisation.