I Wanna Vote

by

Trevor Cromwell
Bradley Imsicke
Elijah Klopfstein

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology/Cybersecurity

DocuSigned by:
853A07E88B5A432...
Trevor Cromwell — 4/19/2022
Date

DocuSigned by:
CBD2A0451F68403...
Bradley Imsicke — 4/19/2022
Date

DocuSigned by:
FED46EE771A14FD...
Elijah Klopfstein — 4/18/2022
Date

DocuSigned by:
E6E18DB28163495...
Tyler Hopperton, Faculty Advisor — 4/18/2022
Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2022

Table of Contents

List of Illustrations

Abstract

Using online voting tools in America is a very controversial and hot topic and has been for about the past 20 years. In our project we tackle this issue by looking into the viability of using online tools to conduct voting in secure and accessible ways. Over the course of the class, we show who would need to use this system, what help they might need to use it, and how it can be conducted as secure or more secure than current voting systems – both public and private. Through work and research, we proved that an online voting system can have the security of modern voting systems with high accessibility while costing a fraction of the current system. This means that with an upgrade of current voting systems the state can both save money and increase accessibility all in one move.

Introduction

Project Name:

I Wanna Vote

Project Summary:

A voting tool with enhanced security and accessibility that can be used by private entities to run polls/mock elections and a road map for upscaling the project for federal elections in the future.

Problem Statement:

According to a study done by Pew Research (B), in 2016 only 56% of citizens eligible to vote in the United States casted a ballot. This is trailing behind most other developed nations, and many cite lack of ballot access as their reason for not voting. But many Americans fear using alternative voting systems due to security flaws and little transparency.

Solution:

We aim to create a voting/polling platform that focuses on security and accessibility first. The system will use an array of technologies to guarantee the integrity of a poll/vote. Authentication and auditability will be the driving force behind the "security" the software is aiming for. We can also look at the failures of other tools past and present to inform our design. In summary, we aim to create a system as close to Survey Monkey level security for no price and with an end goal of a sister system being used in federal elections.

Project Source:

Due to the COVID-19 pandemic, the latest U.S. presidential election was very controversial with how the different states conducted their polling. While the main subject of scrutiny was the mail-in ballots, there are states that have conducted electronic voting for decades and are always being decried for the lack of public auditability that method provides. We were inspired by this to create an electronic voting system that solved all the current problems with electronic voting, but we had to quickly scale back the scope of the project. A group of 3 college students are not going to create a system in 7 months that is practical for federal use, it is just not possible. We looked to the internet for inspiration and found that most online polling systems can be easily manipulated or outright broken by anyone who can conduct a google search. Elijah and Trevor already knew each other, but we found Bradley through the Senior Design Teams group.

Discussion

Project Objectives/Goals:

This year our team hopes to build a web app that is accessible from all mobile devices and web browsers while being easy to use and understand for people of all ages and disabilities. We aim to allow anyone to create an account and create polls to be easily distributed without fear of tampering or bad results. By the end of the year, we will have a road map made for creating a sister system made for elections in the state of Ohio that will feature state of the art security and secure identity verification.

Project Scope:

Our team will develop a functional web application that enables private groups to run polls/mock elections easily and securely over the internet and create a practical plan to create a sister system secure enough for federal use by the state of Ohio.

Quick Project Timeline:

Table 1 Project Timeline

Task #	Task Name	Duration	Start Date	End Date
1	Report sources	3 days	9/20/21	9/23/21
2	Website Design	2 days	9/21/21	9/22/21
3	Report draft 1	4 days	9/23/21	9/27/21
4	Design Document & repo	7 days	9/20/21	9/27/21
5	Report draft 2	4 days	9/27/21	9/30/21
6	Infrastructure setup	14 days	9/20/21	10/4/21
7	Basic website	7 days	9/27/21	10/4/21
8	Report draft 3	4 days	10/4/21	10/8/21
9	User creation	5 days	10/5/21	10/10/21
10	Report draft 4	4 days	10/11/21	10/14/21
11	Poll creation	10 days	10/10/21	10/20/21
12	Poll security	15 days	10/10/21	10/25/21
13	Report draft 5	4 days	11/1/21	11/5/21
14	Report final	4 days	11/8/21	11/12/21
15	Presentation	15 days	10/25/21	11/9/21
16	Mobile friendly	7 days	10/25/21	11/1/21
17	Disability friendly	7 days	11/1/21	11/8/21
18	Language support	7 days	11/8/21	11/15/21
19	Sample Ballot	31 days	1/1/22	1/31/22
20	User Acceptance Testing	56 days	2/1/22	3/25/22
21	Code Review 1	41 days	2/15/22	3/25/22
22	Federal system road map	16 days	3/16/22	3/31/22
23	Expo presentation prep	10 days	4/1/22	4/11/22

Technologies Used:

- AWS Cloud Services
- Serverless Microsoft SQL Server
- Kali Linux for pen testing
- Smartphone (IOS and Android)
- Registered domain
- Git
- Microsoft .NET Core

Technical Architecture Diagram:

Below are two diagrams of the architecture of the application. The first one shows how data is stored in the database from polls to users. The second details how the website and AWS interact, how the website is hosted, and how it oversees changing traffic sizes and where data is store in the code itself.

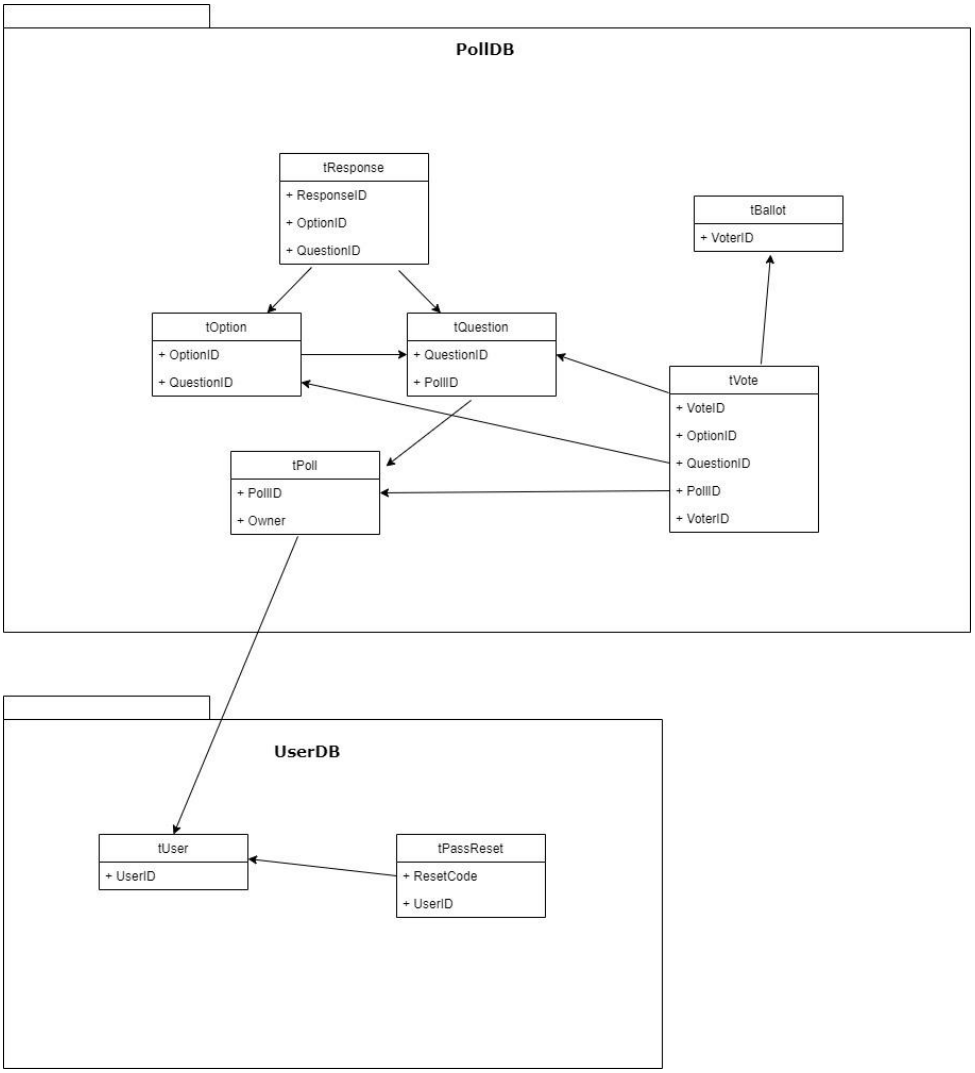


Figure 1 Diagram of database design

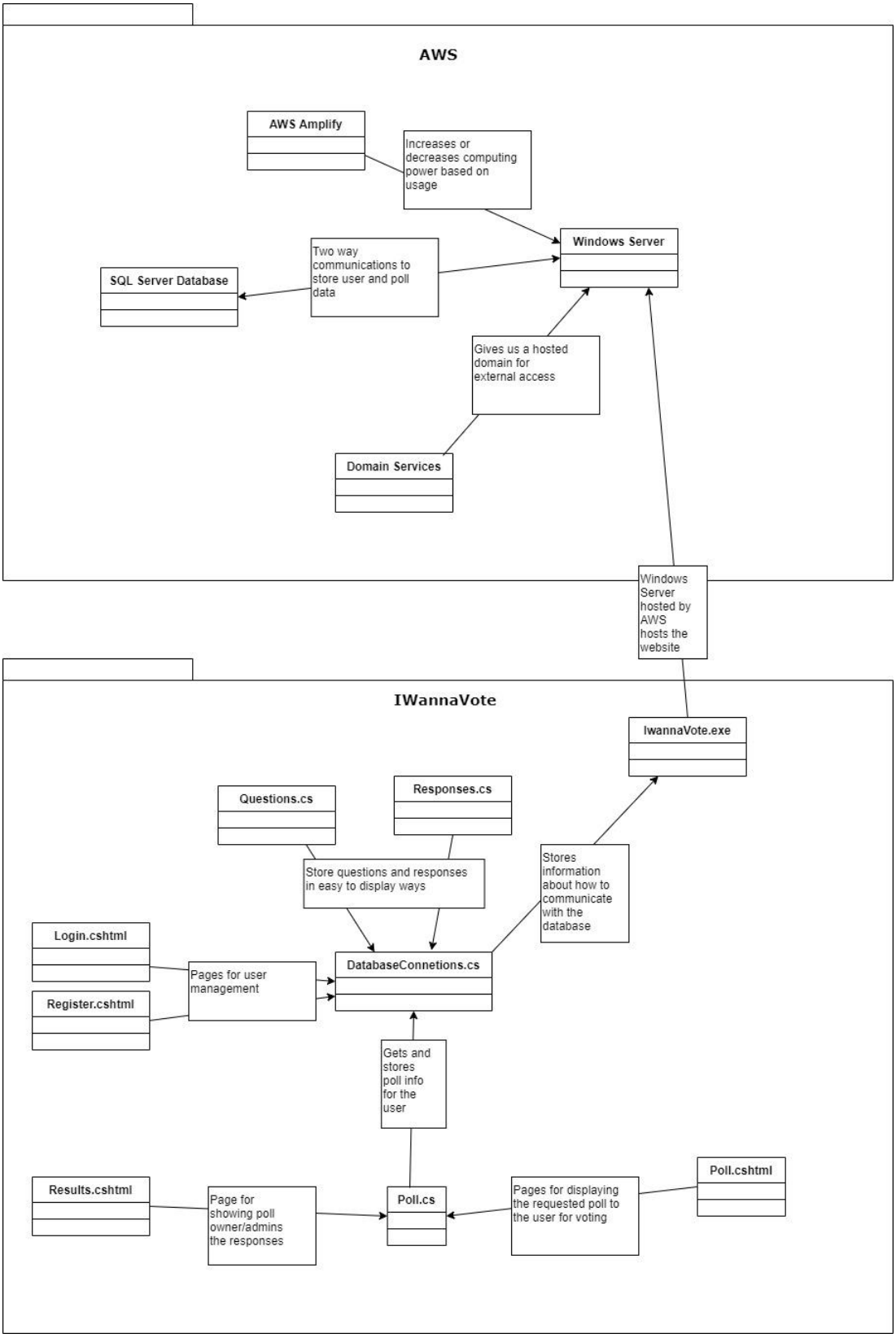
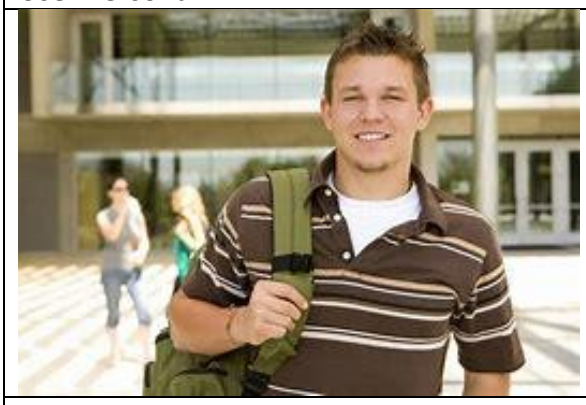


Figure 2 Diagram of AWS and Code design

User Personas:
In this section we will go over some user personas of people that would use the application either for running polls or voting in them. All persons depicted in this section are completely fictional and designed to resemble a member from the group they are assigned.

Table 2 User personas

User Persona 1:	
	College student
	Tom Smith
	22
	Male
Behavior	Herd mentality, reliant on tech, always busy
Pain	Uninformed voting and low ballot access
Needs & Goals	Wants voting to be easy to use and accessible to everyone and wants voting to work with their life schedule

User Persona 2:	
	Disabled person
	Brett King
	35
	Male
Behavior	Low mobility, needs an aid who is not always available
Pain	Low number of handicap parking spots and friendly voting locations
Needs & Goals	Wants to be able to easily vote without much hassle and vote when it works for their aid

User Persona 3:	
	Construction worker Sally Ran 42 Female
Behavior	Herd mentality, low tech use, and family first values
Pain	Irregular and busy schedule and sometimes out of town for work
Needs & Goals	Wants an easy option to vote from afar that is easy for her to use

User Persona 4:	
	CEO Chole Wesser 65 Female
Behavior	Profit values, highly connected, and a leader
Pain	Busy schedule and large tasks
Needs & Goals	Wants to be able to easily get opinions from employees to keep them with her

User Persona 5:	
	High School Principal
	Tim Burr
	35
	Male
Behavior	Community first, a leader, and cares for the youth
Pain	Trying to keep a diverse group of students and parents happy
Needs & Goals	Wants to easily get opinions from the community to make educated decisions

Use Cases:

In this section we will go over the different use cases that the user might encounter and problems they might run into while doing so. We will also go over the steps the user can go through in order to fix the problem they encountered.

Table 3 Use Case Tables

Use Case ID	Use Case 1
Use Case Name	Create a user
End Objective	Create a user to run polls with
User/Actor	Someone who wants to run a poll
Trigger	Signing up for an account
Frequency of Use	Often
Preconditions	None
Basic Flow	1. I sign up for an account 2. My account is validated and made
Alternate Flow	2a. My account is taken 2ba. I use “forgot my password” to reset it 2bb. I use a different email
Postconditions	I now have an account to create polls with

Use Case ID	Use case 2
Use Case Name	Create a poll
End Objective	Have a poll created
User/Actor	Someone who wants to run polls
Trigger	A user goes to create a poll
Frequency of Use	Once per poll
Preconditions	A user account has been created
Basic Flow	1. I go to the create a poll page 2. I add all questions I want on the poll 3. I create the poll and get a sharable link 4. I share the link and wait for results 5. I view the results of my poll
Alternate Flow	3a. I try to create a poll and get an error 3b. I wait 24 hours and try again 3c. repeat until systems are back up 4a. A voter says the link does not work 4b. I resend the link ensuring that it is correct 5a. I have no results to view 5b. I reshare the link to get people to vote
Postconditions	I have created a poll and have results to view

Use Case ID	Use case 3
Use Case Name	IWannaVote
End Objective	Cast a vote on a poll
User/Actor	Anyone who wants/needs to vote on a poll
Trigger	A user follows a poll's shareable link
Frequency of Use	Several times per poll
Preconditions	A user has received a link to a poll
Basic Flow	1. I follow the link to the poll 2. I pick an option for each question 3. I submit my poll
Alternate Flow	1a. The link leads to an error page 1b. I contact the poll runner and ask for a new link 1c. I use the new link to reach the poll
Postconditions	I have submitted my poll results

Use Case Diagram:

Below is a simple that illustrates how the different types of users will interact with the application with a path showing the paid tier network would look like as well.

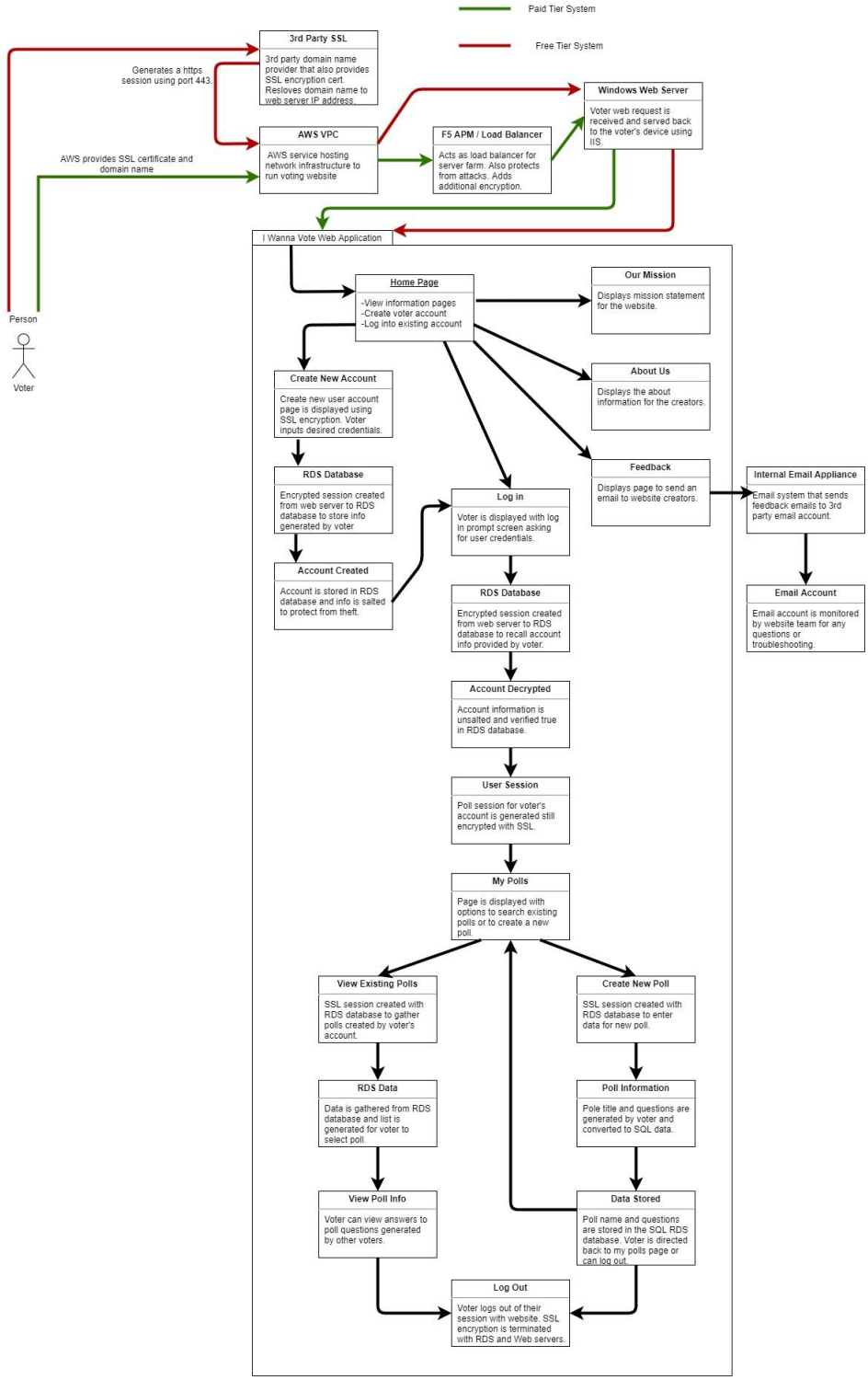


Figure 3 display of user interaction with web server

Testing Plan:

Overview

This section we will go over the testing that was done to ensure functionality and ease of use for users from all expected groups. We will also go over the results of the tests and changes made due to their results.

Methodology

During development we used test driven development (TDD) to ensure the code we wrote fully completed a task that was needed for some use case or expected user acceptance. We evaluated our back and front-end code with this method. Later we tested the application with emulators to ensure the website was fully functional for a broad range of users. Top ensure we had a strong website we ran penetration, backup plans, and failover plans for the website and other servers. And lastly, we tested the website by allowing users to run polls and vote on them to enable feedback from potential users of our platform.

Scope

The unit testing is mostly focused on back-end work and ensuring that it is secure and fast. It will test poll creating, voting, and database security. It will also see a bit of testing the website's ability to handle substantial amounts of visitors at once. The emulators will allow us to test our front-end work ensuring it is usable for most browsers. Lastly the user acceptance testing will be used to ensure users find the final product as smooth and easily to use as possible and make any changes we feel needed from their feedback.

Objectives

TDD

1. Ensure all new features are functional and secure
2. Ensure all new features are speedy
3. Ensure functions and database work together properly
4. Ensure website can handle substantial amounts of traffic

Emulators

1. Ensure website works on all major browsers
2. Ensure polls can be created on all major browsers
3. Ensure votes can be casted on all major browsers

Security

1. Ensure backups are kept and easy to deploy
2. Ensure plans for the event of AWS outage
3. Ensure our website works with AWS security

User Acceptance

1. Final check for missing or broken features
2. Ensure users find the website easy to use
3. Test unexpected edge cases by having user with no prior experience use the site
4. Resolve all bugs and missing features before IT expo

   Define what your testing strategy is and what the end goal is.

Test Logs and Procedures

Each week we tried to add a new feature to either the server or the client-side interactions while continuing development on old features. Below is the end of week results of that functionality. These test and results are the product of test-driven development uses test classes to emulate real user interaction. These tests were ran and tracked by Elijah Klopfstein.

Table 4 TDD results

Test Case #	Area	Expected	Result	Pass/Fail	Reason	Date
1	Server	database content retrieved	Connection error	Fail	Data storage not communicating	10/8/21
2	Server	database content retrieved	database content retrieved	Pass	Data storage is communicating	10/15/21
3	Server	User created	IAM user error	Fail	User storage not communicating	10/22/21
4	Server	User created	IAM user error	Fail	User storage not communicating	10/29/21
5	Server	database content manipulated	database content manipulated	Pass	Server and database are working together	11/5/21
6	Client	Poll opened	Poll opened	Pass	Poll successfully opened	11/12/21
7	Client	Vote counted	Counted	Pass	Voted counted	11/19/21
8	Client	Question edited	Edited	Pass	Question edited	1/31/22
8	Client	Poll status changed	Changed	Pass	Status edited	2/15/22
8	Client	Ballot casted	Ballot casted	Pass	Ballot casted	3/7/22
8	Server	All votes shown correctly	No data errors found	Pass	All data is good	3/30/22

Next will be the test ran by opening the website on different devices and browsers to ensure compatibility and ease of use. We test the sites functions on all popular web browsers and mobile devices each week. These tests were ran and tracked by Trevor Cromwell.

Table 5 Compatibility testing results

Test Case #	Expected	Result	Pass/Fail	Reason	Date
1	Site loads on all platforms	Site loads on all platforms	Pass	Encountered no issue loading pages	10/29/21
2	Site loads on all platforms	Site loads on all platforms	Pass	Encountered no issue loading pages	11/5/21
3	Site loads on all platforms	Site loads on all platforms	Pass	Encountered no issue loading pages	11/12/21
4	Site loads on all platforms	Site loads on all platforms	Pass	Encountered no issue loading pages	11/19/21

5	Site loads on all platforms	Loading issue on mobile	Fail	Nav bar broken, footer hidden	1/15/22
6	Site loads on all platforms	Site loads on all platforms	Pass	Encountered no issue loading pages	1/31/22
7	Site loads on all platforms	Site loads on all platforms	Pass	Encountered no issue loading pages	2/31/22
7	Site loads on all platforms	Site loads on all platforms	Pass	Encountered no issue loading pages	3/31/22

Prior to the user acceptance testing we ran tests on our website and servers to ensure security is present and that we have plans for the event of an outage. These tests where ran and tracked by Bradley Imsicke.

Table 6 Security results

Test Case #	Subject	Result	Pass/Fail	Reason	Date
1	Backup/restore	Site and database restored quickly	Pass	Everything restored	1/30/22
2	Failover	Site kept live during AWS downtime	Pass	Website is redundant in AWS east zones	1/30/22
3	Penetration	Tool found no hanging connections	Pass	Website gives wanted security at free tier	3/1/22

In February we ran our user acceptance testing to ensure users do not encountered unexpected errors and fixed all problems or inconveniences that our users may encounter during early testing.

Table 7 User Acceptance Testing Changes

Test Case #	Subject	Result	Changes	Reason	Date
1	Concern with public feedback	Discussed other options	None	Using best free option	2/15/22
2	Session data error	Site stored session variables in shared location	Session data storage rework	Was breaking activity when more than one user is on at a time	3/17/22

Testing Review

In review, our use of testing and test-driven development has helped us keep track of our progress and know when a major issue was encountered. By have pre planned goals that we could test and see a pass/fail result for is one of the main things that helped on group ensure we stayed in tack this semester. Also, the user testing allowed us to find issues that we overlooked during development.

Change Management Plan:

Changes can be requested by any member of the group or group advisor/sponsor or a user acceptance tester. Changes of major effects can be discussed the week of and added to the current week's schedule if approved by the group. Minor changes can be discussed in next week's meeting and be added to the following week's plan if approved. This method will benefit us as it will prevent us from wasting time with meetings on minor changes and create a back log for minor changes. However, this might cause some of the minor changes that are approved to not be in the final product at the IT expo. This is an acceptable cost as minor changes are things we want but do not need.

To get an idea approved it can be brought up in our Teams chat or in a meeting. When brought up if the change is related to a major functionality and/or putting it off would lead to large technical dept it will be addressed immediately and, if approved, will be added to the current week's to do list. If the item is a minor change where it only adds noncritical functionality, then it will be set up for a week in the future or be set on a TBD list.

All changes suggested will be mentioned in meetings with the project advisor and sponsor where we will discuss what prompted the request, what it request would fix and how, and whether or not it was accepted. If accepted, we will also mention when the change is expected to be implemented into production.

Budget:

Our project budget for a private use system accounts for the team's labor as 20 hours per week. This is due to the expectation that the project would continue to be worked on while working a full-time job. We then have a yearly cost of $2,500 for AWS services and Visual Studio Professional version.

Table 8 Estimated cost of project continuation

Estimated Cost Rough Order of Magnitude:							
				Ongoing Annual			Comments:
	Rate Per/Hr	Work Effort (Hours)	1 X Costs	Rate Per/Hr	Work Effort (Hours)	1 X Support Cost	
Labor - IT	20	0	$ -	20	1040	$ 20,800.00	
Labor - External	0	0	$ -	0	0	$ -	
Software - External			$ -			$ 2,500.00	
Hardware - External			$ -			$ -	
Misc.			$ -			$ -	
TOTAL			$ -			$ 23,300.00	

If we started looking at the price of hosting all Ohio elections in our system, the price would change. It would be hard to estimate due the unknown factors of when people would vote changing how many votes would be processed at once and how many people would vote given the new easy method of voting. So, we high ball it saying it will take about 3 months of full-time effort from the team to maintain and half a million dollars to run during election periods. This guess is extremely high but is a safe "will not exceed" budget. Compared to current voting cost this is a substantial decrease in costs. In 2005 and 2006 we replaced most of our voting systems in Ohio costing us nearly 115 million dollars (A). We replaced them again in 2018 costing us

104.5 million dollars (C). That means we spent about 9.5 million dollars per year between 2006 and 2018 when distributing the cost of the machine amongst the years they were in use.

With these numbers we see that virtualizing voting and allowing people to vote from any device will significantly decrease the cost of maintaining our voting system while still enforcing high security by having people register to vote in person and then confirm their identity before voting.

Table 9 Estimated cost of hosting State voting

Estimated Cost Rough Order of Magnitude:							
	Rate Per/Hr	Work Effort (Hours)	1 X Costs	Ongoing Annual			Comments:
				Rate Per/Hr	Work Effort (Hours)	1 X Support Cost	
Labor - IT	20	0	$ -	20	2000	$ 40,000.00	
Labor - External	0	0	$ -	0	0	$ -	
Software - External			$ -			$ 500,000.00	
Hardware - External			$ -			$ -	
Misc.			$ -			$ -	
TOTAL			$ -			$ 540,000.00	
5-Year ROI Analysis							

Problems Encountered and Analysis of Problems Solved:

Support

When we first started work on our project, we went in trying to build it as an ASP.NET Web App like the one that we built in class. We quickly found out that the framework we were using is no longer supported by Microsoft and thus is not support by most companies like Amazon who would be hosting our web site. We solved this problem by upgrading to creating an MVC web app with modern ASP.NET Core which is supported by AWS.

Knowledge

When we had to switch to a different framework and style, we put ourselves at a difficult learning curve. With limited time we needed to learn this new system and implement in. To get through this major struggle we referenced the Microsoft documents countless times and used videos we found on YouTube to guide us through some of our bigger problems. This problem was frequent and, as the project is still in progress, we expect it to come up again.

Money

Part of our reason for going with AWS was for their free tier, specifically made for start-ups and college students. As we started working, we realized some of the things we wanted to do were not fully supported under the free tier and would cost us upwards of a hundred dollars a month. Without funding we had to find ways to get around that. So, we did just that, we found different services that they offered that tend towards a bit slower or a bit less security for free. We still ended up spending about fifty dollars a month to ensure it was still secure and fast but that was within a price we could afford without external funding.

Time

Another important thing we encountered was a lack of time. With such little time for development, we had to ensure we watched our scope creep and keep our goals achievable. To ensure this never caused significant issues, we had to toss out some of our wanted features to ensure we would have enough time to build and support the needed features and a few of our wanted ones.

Conclusion

Over the year we were faced with a large number of challenges. Our developers had to learn a new framework, our network expert had to get crafty with how he worked to minimize cost without compromising security, and overall, we had to make our first fully deployed application. Through this we all learned a little bit about each other's areas of expertise and got to see a side of IT that we had little to no experience in. And lastly, as we got to know each other more we developed as a team and started working as one solid unit instead of three separate developers.

Despite all the challenges faced we managed to build a fully functioned web hosted polling system with modern security which many current federal polling systems lack. This framework can very easily be a ready to use polling platform if the needed high security budget is met. As the leaders of the free world it should be our number one priority to bring modern security and accessibility to our polling system. With IWannaVote Ohio could have the best polling system out of all 50 states.

References

A. Chapin, Doug. "Ohio Debates How to Share Costs of New Voting Machines – Election Academy." *Election Academy*, 27 Feb. 2018, electionacademy.lib.umn.edu/2018/02/27/ohio-debates-how-to-share-costs-of-new-voting-machines/. Accessed 28 Nov. 2021.

B. DeSilver, Drew. "In Past Elections, U.S. TRAILED Most Developed Countries in Voter Turnout." Pew Research Center, www.pewresearch.org/fact-tank/2020/11/03/in-past-elections-u-s-trailed-most-developedcountries-in-voter-turnout/. 28 May 2021

C. Underhill, Wendy. "Funding Elections Technology." *National Conference of State Legislatures*, www.ncsl.org/research/elections-and-campaigns/funding-election-technology.aspx 28 Feb. 2020