

Project 3

System Prototype and Evaluation Plan

Team B-3: The Rule of Two

Wade King and Carolina McGlaulin

Usability Specifications and Evaluation Plans

Quantitative Benchmarks

- How long are users willing to interact with the stream with and without a chat functionality?
 - While users are testing the system, we will time how long users remain interested in interacting. Some users will be given a version with a chat and others without. Users without the chat will have capabilities to interact with the question-and-answer mechanic and view the live stream video. We will determine if there is a significant difference in time, which will determine if the chat is useful.
- How often do individual users interact with the chat functionality versus the question submission functionality?
 - For users who end up being able to use the chat functionality, we will measure how long they use the question-and-answer functionality. We will also measure how long they dedicate to the chat functionality, and compare it to the Q&A use time.
- What percent of users make use of the question-and-answer functionality?
 - Among all users, we will count how many use the Q&A system, including those who succeed and fail to submit and vote on questions.
- For how long do users pay attention to the streamed content?
 - We will monitor all users and time their total interaction time. We will then subtract the amount of time spent using doing an action other than looking directly at the stream (chatting, submitting questions, navigation/explorations, etc.) in order to determine what keeps people most interested.
- How long does it take for users to find important functionality in the website?
 - At the start of each test, we will measure the amount of time it takes a user to create an account, log in, chat, submit questions, and find streams to watch.

In general, the main goal of our quantitative benchmarks will be used to determine how easy or difficult individual system components are to use and whether users are willing to use them. Through this, we can discover if certain components are important or useful to the entire system.

Qualitative Benchmarks

- Are users more interested in the internet-based public/government interactions or in-person public/government interactions?
 - At the end of each test with a user, we will ask the user whether they are interested in involvement with government in-person. We will then ask if they would prefer an online alternative such as the one they just used.
- Are users encouraged to be more willing to provide login credentials to a public service, such as Twitch Does Town Hall?
 - Assuming this Twitch Does Town Hall would be used as a public service, users may have to provide login credentials. We will see if our system encourages users to be more willing to provide login credentials (perhaps containing some personal

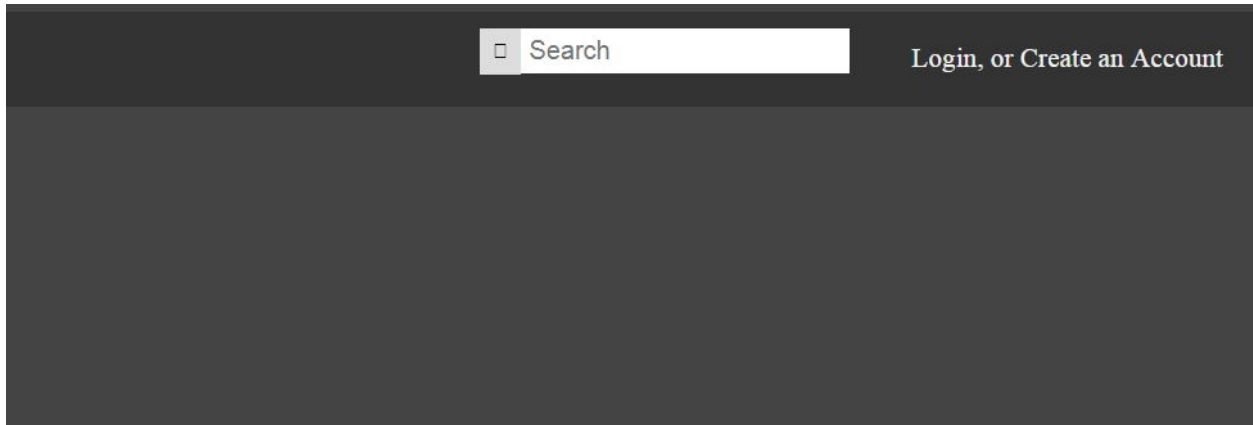
information) to a government service. We will gather initial data before each use and compare it to answers to the same question after system use.

- Do users prefer to focus on international, national, state, or local government or public affairs?
 - We will provide user with access to streams and accounts which feature different levels of government. We will monitor user behavior to determine which levels they enjoy viewing the most and which they spend the most time with. We will also ask users which they personally enjoyed the most.
- How do users react to having popular or well-known public figures (president, governors, prominent community leaders, etc.) involved or participating in the streams?
 - We will include streams that contain familiar or popular figures in government. We will observe user behavior for reactions to interesting figures. Afterward, we will also ask users who they recognized and if they were drawn in by popular figures versus whether they preferred to find unfamiliar content.
- Are users likely to notice blatant bias from moderators or participants of the stream?
 - In some streams, we will attempt to simulate obvious bias in streams and observe how users react or fail to react to this. At the end of each use, each user will be asked about this bias.
- Are users likely to perceive bias from moderators or participants of the streams?
 - We will determine how much user bias plays in perceiving bias (whether it exists or not) in how content is displayed or suppressed by moderators. When we ask users about this perception, we will also ask how it affected their opinions on the system.
- How do users perceive public figures or officials after interaction?
 - Before each use, users will be asked about their perception of certain figures. If users happen to come across these figures during a test, we will ask whether any perception changed about them.

Our qualitative benchmarks will contribute to determining how our system affects user perception of government and officials, and if their willingness to be involved changes in any measurable way. We will also measure how users are interacting with the system in order to determine how it differs from in-person interaction and other analog systems for government interaction.

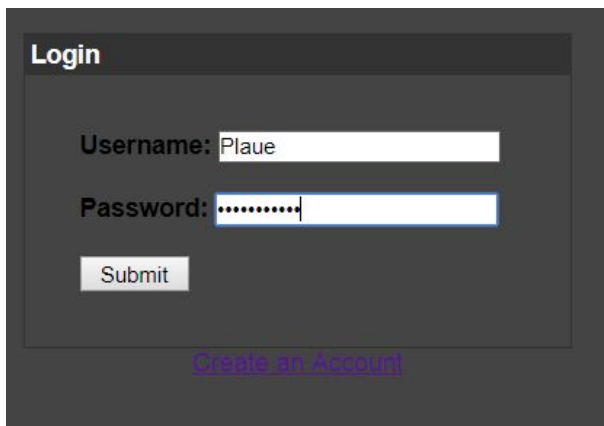
Prototype

A useable version of our prototype can found at <http://webapp.cs.clemson.edu/~jwk/>. The initial page is known as the homepage. It contains a navigation bar at the top with (inactive) links and a search bar. Next to the search bar is a link that takes the user to the login page, where they may login with a current account or create a new account.



□ Search Login, or Create an Account

After clicking the “Login, or Create an Account” link, the user will be taken to the login page. They will be given the option to login with a current username and password or create a new account. A current account for the instructor (Username: Plaue, Password: watchstream) is provided for testing the login functionality. The user may also click the “Create and Account” link below the login form. This will take them to the create account page.



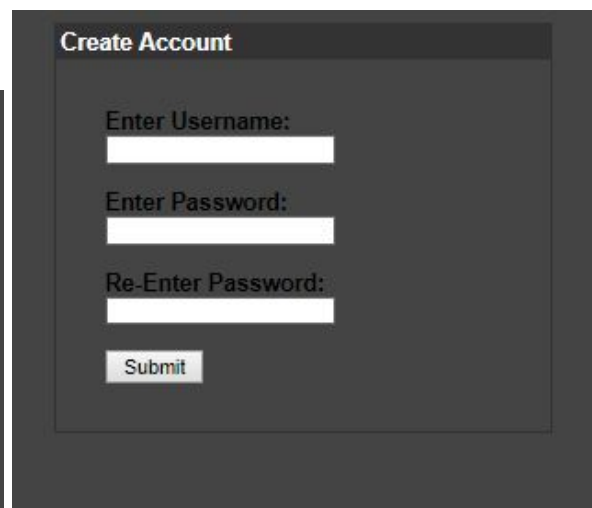
Login

Username: Plaue

Password:

Submit

[Create an Account](#)



Create Account

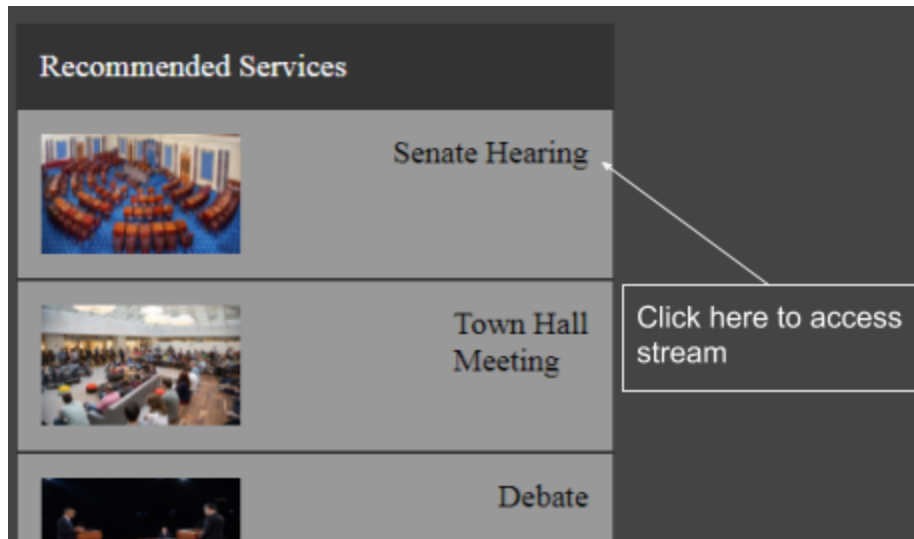
Enter Username:

Enter Password:

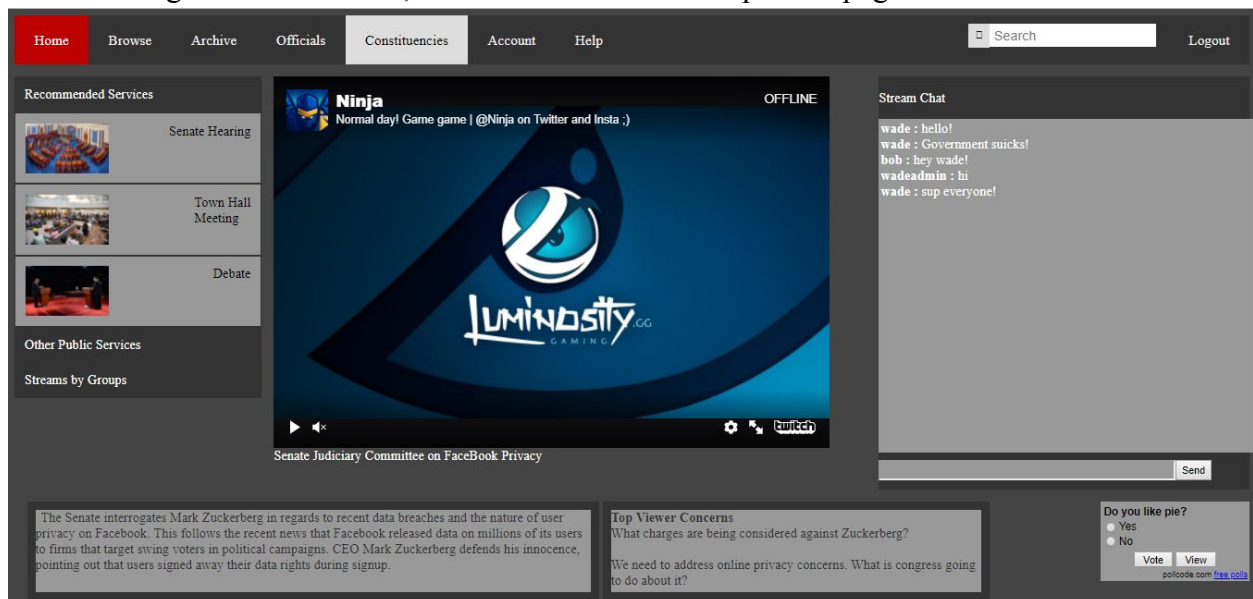
Re-Enter Password:

Submit

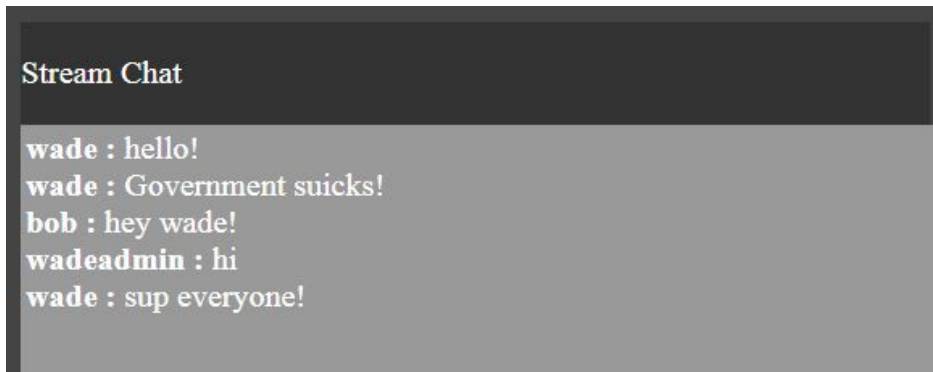
If the user decides to create an account, the user will be brought back to the login page once the account has been created. From here, the user may attempt to login. Once the user logs in, they will be brought back to the homepage. The user can then start to browse and watch streams. An example stream can be accessed by clicking the “Senate Hearing” option. This option will take the user to a stream individually associated with the link. Under the Senate Hearing link, there are links to two other streams which can also be viewed.



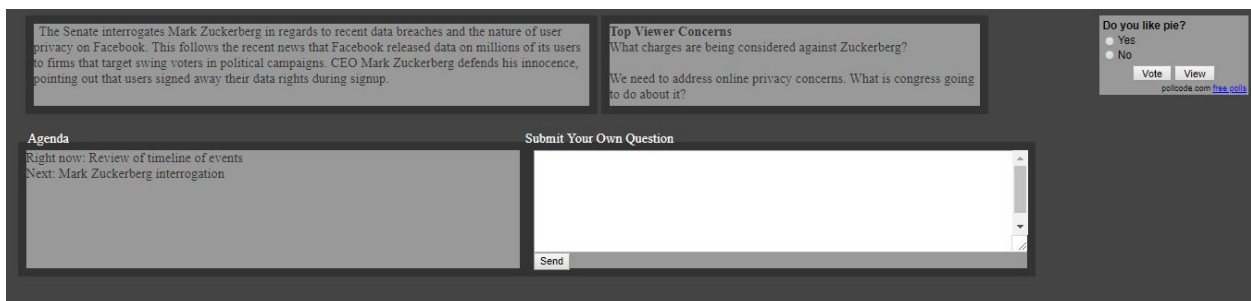
After clicking the stream's link, the user is taken to its respective page.



Much like on the homepage, the user may browse streams in the same side menu. Beside the side menu is the video stream for the webpage. Currently, this displays a stream from the website Twitch.tv as a placeholder. On the other side of the stream, is the stream's chat. The logged in user may type into the text box at the bottom and press send in order to send a message to the chat. Each message is associated with a user.



This chat provides an outlet for those who wish to discuss or argue with other viewers on the topics of the stream. Since the focus of the system is on communication between viewers and their representatives, this chat is of minimal importance. A profanity filter and moderators assigned by the broadcaster administrators would be all that filters this chat.



Below the stream, the user will find a description of the stream's contents and the current popular questions other viewers have asked toward the stream so far. Directly under the chat, a poll for the user is available for further feedback. The polling and viewer submitted questions are the primary functions of the system. These allow the viewer to voice their opinion on topics to those in the broadcasted event in a more formal and, hopefully, more involved way. The user is also able to submit their own questions for the stream that may be answered in the stream itself.

The central focus of our prototype is to facilitate two-way communication between constituents and their representatives during publicly broadcasted meetings. We aimed to accomplish this by adding polling and question submission functions to public meeting streams. In retrospect, our UI could have better focused on these functionalities by putting them above the fold, and closer to where the stream was displayed. The chat should have been lower down as that functionality takes lower priority.

Development Challenges

Our greatest challenge when developing the prototype for Twitch Does Town Hall was the lack of experience within the team in developing front-end web applications, especially when using HTML and CSS. Designing the structural and visual elements of the prototype were most difficult because HTML and CSS are the sole technologies for doing these on websites.

Implementing back end functionality was not extremely difficult, as the team had experience with back-end functionality and scripting. However, because HTML and CSS are not meant for scripting and only service front-end content, both team members had a difficult time learning the paradigms involved in each. Because back end functionality could not be fully tested without first creating the front end visuals, development became a slow process of adding increments of HTML and CSS until it worked, then adding back end functionality later. Additionally, back end functionality requirements will change depending on the structure of the website defined by HTML, so many changes to the front end were met with refactoring the back end.

As far as implementation of individual functionality goes, the most difficult portion was the chat functionality. This part of the site required integration of all technologies involved in the project. HTML and CSS created the structure of the chat box section of a single web page, and the MySQL database held all of the messages sent within the chat for an individual stream. PHP was then used to send and retrieve chat messages from the database, but JavaScript was also required to work with both the HTML tags and PHP scripting to prevent web pages from refreshing entirely when a message was sent. If any of these implementations were to change, especially the HTML, then the other implementations of these technologies would also have to be adjusted to work with the other new changes. Because of our team's inexperience in HTML and CSS, this was a common occurrence in developing the prototype.

Because of these development issues and time constraints, we have yet to implement certain parts of the functionality, and perhaps will never have time to do so. By far the most intensive functionality to implement would be live streaming of video through the website. There are multiple third-party solutions for adding live streaming to websites, but testing such an implementation will be difficult and expensive. Testing would involve having a high internet speed and quality, powerful video and computer hardware, and high-quality video software for encoding and uploading video.

Prototype Design Justifications

We chose this design primarily due to the popular reception of the idea in our second studio. People loved the idea of a digital service for live two-way interaction with governing bodies. Our design allows government meetings to reach more people who otherwise may not have participated. Our system is different from government broadcast channels in that television as a medium is losing popularity, particularly among young people. And even more, television does not allow for constituent voices to be heard, as users have to manually call in and hope they get heard. By implementing functions such as streaming, polling, and voting on user submitted questions, we allow for organized communication in both directions between constituents and representatives. Given the fact that nearly all internet-connected devices can access streams, we lower the work burden put on people to participate in government. Instead of having to drive and show up at a meeting to both listen and have their voice heard, they could participate from the palm of their hand.

Studio Feedback and Response

We received excellent constructive criticism, focusing mainly on the UI, and polling/voting functionalities. People found the color scheme to be dull, and lacking contrast. Font size was a little low, so people with less than decent eyesight may have trouble reading text. The placement of our elements is also a big concern. Given our goal is communication between viewers and those in the meeting, we will later place the viewer chat beneath the fold of the page, and place viewer polls and viewer submitted questions next to the stream above the fold.

We will expand on the functionalities of the viewer poll. Anytime a new poll became available, we would make the interface blink colors slightly, to notify the user of the new poll, we would also put a timer on the poll to indicate how much time left.

Feedback also indicated that we should have increased functionality for the submission, voting, and viewing of user submitted questions. Given the time, we would display questions in more of a vertical queue to the user. As new questions get submitted, they slide in at the top pushing older questions down. Users can “bump” questions to keep them at the top longer for representatives to see. Due to the very polarized nature of politics, we would not include any downvote/disapproval button, and we would focus on moderation techniques to remove inappropriate questions.

Technology Justification

We chose to use a wide set of web programming technologies in developing our Twitch Does Town Hall application. Hosting this on the internet would be essential for making it easier for people to be more involved in government. All together, our prototype used HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets), JavaScript, PHP (Hypertext Preprocessor), Apache, and MySQL (My Structured Query Language).

Our front-end development required a combination of HTML, CSS, and JavaScript. HTML acted as the basis for our website and provided its structure. HTML is essentially required for any website on the internet, as it is the only web programming language which provides objects and structure for a website. CSS provided extra functionality for the visual style for Twitch Does Town Hall. For any website that wishes to have more complex visuals, compared to a website which uses only HTML, CSS provides a way to create sets of HTML style tags which can be easily and repeatedly integrated into another HTML page. JavaScript was also implemented into our prototype as a way to make our site more dynamic on the front end, such as allowing users to interact with a chat or question submission. Bare-bones HTML pages are mostly static and don't allow for much user interaction. JavaScript was also used to deploy jQuery into the site. JavaScript can also react to server-side events, allowing it to also act on the back end of the website.

The back end for Twitch Does Town Hall relied on PHP, an Apache web server, and a MySQL database. PHP provided scripting for our back end server for extra functionality in our website's behavior, such as determining situations under which additional content should be displayed. PHP also provided a library for our website to query a MySQL database for storing and retrieving information dynamically. Essentially, PHP provided the connection between our front

end and back end. Our back end was hosted on the School of Computing Webapp Apache web server. As Clemson SoC students, Webapp provided a free and reliable host for the website for both development and user access. The Apache web server on Webapp allowed us to host our HTML and CSS files and integrate PHP and CSS with them, as Apache provides functionality for interpreting PHP and CSS. Apache is also free, open-source, and works on many platforms, so our site can easily be ported over to other servers and work with other technologies. The final portion of our back end was a MySQL database held on Clemson's MySQL1 server using their Buffet service. Like Webapp, Buffet is a free service for Clemson students, so it was the easiest and cheapest solution for our site. We chose to use a MySQL database on Buffet because it was the most familiar database system within our team, the easiest to access, and the easiest to integrate with PHP.