* **Requirements Summarization**

Overall, I need to create a polling/voting website that allows the user to create a poll (prompts the user for a poll question and up to six options), displays the available polls (all the polls that can receive votes), displays the results of a poll (the options that the user chose for a poll), display the results of a poll using a bar graph (each bar calculates the percentage of votes each option received in relation to the overall poll), and a backend that records all the functionality of the website (polls, and poll results). When a user votes, he/she can only select one option before submitting a vote; if there is an issue with them submitting a vote, the user will be redirected to a 404 page.

* **Project Analysis and Design**

My project was designed using Bootstrap, Handlebars, Font Awesome, Node JS (and its dependencies such as Express, and NPM), JavaScript, jQuery, SQL, and CSS. Although it was not a part of the requirements, I make it so that the user of my website must create an account before they have access to any functionalities. To elaborate, the user cannot do anything unless they create an account with my website called Ur Voice. After creating an account with Ur Voice, the user will have the ability to create a poll, answer polls, keep tracks of their poll, and see the results of any poll.

The backend of my website, Ur Voice, is based on Node JS and its dependencies such as Express and NPM – Node JS replaces the traditional PHP functionalities. After a user makes a request to the server (which is set up with Node JS) through either a POST or GET request, the server responds to the request accordingly. If a query is needed, my server will grab all the variables associated with the request (POST or GET) and perform a SQL query based on the request. None of my web pages are coded in HTML but instead, are generated with Handlebars through Node JS. Node JS has predefined paths to my handlebars page which is populated with the requested information before it is returned to the user. Since every page that is associated with my website requires the same navbar, CSS, and JavaScript contents, I decided to make a parent Handlebars script which is used by each individual page as a template; in detail, the navbar, CSS, and JavaScript contents do not have to be recreated for each page (the templet handlebar file is stored as “main.handlebars”).

After a user makes a request, my server generates the frontend webpage using handlebars and returns it to the user. The Handlebars pages are converted into HTML before it is sent to the end user (keep in mind I did not implement any HTML code for my frontend but the code in Handlebars is converted into HTML by my server before it is returned to the user). My frontend content relies on Bootstrap (functionality and style), CSS (style), JavaScript (functionality), jQuery and Font Awesome (small icons used throughout my webpage). Bootstrap, jQuery, and Font Awesome are linked to my webpage through a CDN. This means that my server does not have to host these contents (the utilization of my server will be less) and the webpage can load faster for the end user if they have visited/cached other web pages that use the same CDNs. On the contrary, I do host the custom CSS and JavaScript code associated with my frontend in my server. Lastly, the percentages used in the bar charts that display the results for each poll question is calculated at the frontend (the backend only passes the counts for each question option and the frontend JavaScript calculates the percentage); I chose to do this so that the resources for my server will not be used for calculating percentages (saves my server resource time).

The databased that I use for my project is hosted on GearHost. Through GearHost, I set up a SQL server. Although MongoDB is mainly used with Node JS, I decided to use SQL because I am more familiar with a relational database structure when compared to object-oriented data structures like MongoDB. The database design section will summarize the design behind the SQL server.

* **Database Design**

Below are the SQL statement used to create the customer, poll, and question option table:

#Create customer table for login

CREATE TABLE users(userID INT AUTO\_INCREMENT, firstName VARCHAR(55), lastName VARCHAR(55), email VARCHAR(255), password VARCHAR(60), KEY(userID, email));

#Create poll table

CREATE TABLE poll(questionID INT AUTO\_INCREMENT, pollQuestion VARCHAR(60), userID int, KEY(questionID, pollQuestion, userID), FOREIGN KEY (userID) REFERENCES users(userID));

#Create question option table

CREATE TABLE question\_options(optionID INT AUTO\_INCREMENT,questionID INT, optionValue VARCHAR(60) NOT NULL, voteCount INT, PRIMARY KEY(optionID), FOREIGN KEY (questionID) REFERENCES poll(questionID));

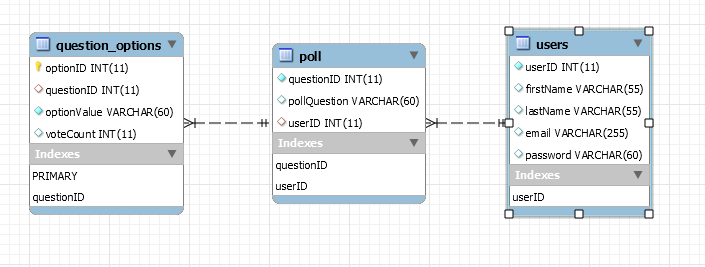
  
**Figure 1: ER Diagram**

Figure 1 shows the ER diagram. Based on Figure 1 and the SQL codes shows each attribute, foreign key, composite keys, and primary key for each of the tables. Table 1 lists all the attributes for the table and their description.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute** | **Description** |
| users | userID | The userID is used in conjunction with the email as a composite key used to identify each unique user. |
| firstName | Stores the user’s first name. |
| lastName | Stores the user’s last name. |
| email | email is used in conjunction with userID used as a composite key used to identify each unique user. This is used in the sessions as the userID (keeping the user logged in). |
| password | Holds the user’s password so they can have access to their account. |
| poll | questionID | A unique id used in conjunction with the userID as a composite key to make each question unique. |
| pollQuestion | The question that the user wants their poll to prompt. |
| userID | Used in conjunction with the questionID as the composite key. It stores the user who created the poll. When the user navigates to their poll page, this is used to only show the polls belonging to a user. This is a foreign key that references the userID in the users table |
| question\_options | optionID | This is a primary key that uniquely identifies each option for each poll. |
| questionID | This is a foreign key that references the questionID in the poll table. This is used to retrieve all the options associated with a poll question. |
| optionValue | This holds the option for a poll question that the user can select. |
| voteCount | This holds the vote count for each option. The voteCount is incremented each time a user submits a vote. |

* User manual

Before the user can use any functionalities associated with my website, UrVoice, they must create an account through the *SignUp* page that can be found on the navbar of any page while not logged in.

After the user signs up, they will be redirected to the home page with a message that congratulates the user for signing up. From here the user will be able to log into Ur Voice with their credentials.

Once the user is logged into their account, they will be given the option to *Ur Voice*, *Create Poll*, *All Polls, Ur Polls,* and *Sign Out* in the navbar. Below are the navbar descriptions and their functionalities.

***Ur Voice*** – directs the user to the home page of Ur Voice website and displays a big green button labeled, “Discover Poll” that redirects the user to the *All Poll* webpage (The button is only visible if the user is logged in).

***Create Poll*** – allows the user to create a poll that consists of one poll question and up to six options per poll question (with a minimum of two poll question). Using the green “+” button, the user will be able to add more poll question (this button will be hidden if the total poll question option count is 6). If the user decides to delete the last option, they can hit the red “-” button to remove that option (the red button will be hidden if the total question option count is 2). Once the user has successfully completed the form, they can submit it using the green “Post Poll” button (if the form is incomplete, a prompt will be displayed to the user).

***All Polls*** – this page displays all the poll available for any user to vote on. Each card element arranged in the staggering brick pattern displays the poll question, the option to vote, and the option to view the results of the poll. By clicking the green button labeled, “Vote” the user can place a vote on that poll question. On the other hand, by clicking the green button labeled, “Results” the user will be able to see the results associated with that poll question.

***Ur Polls*** – has the same concept behind the *All Polls* page but only displays the logged in user’s polls.

***Sign Out*** – destroys the user’s session and returns the user to the homepage.

Two pages not displayed on the navbar is *Vote* and Results. These pages can be accessed through the *All Polls* or *Ur Polls* tab.

***Vote*** – allow the user to pick a question option for a poll question (based on the poll question they select in either *All Polls* or *Ur Polls* page).

***Results*** – allow the user to view the results for a poll question (based on the poll question they select in either *All Polls* or *Ur Polls* page).

* **Known Problems, Bugs, Limitations, Unimplemented Features**

***Problems/Bugs***

After testing my webpage, I have not found any problems that are associated with the requirements. The requirements of this project have been fulfilled.

***Limitations/Unimplemented Features***

Since it was not a requirement, I did not create any hashing functions for both frontend and backend to securely handle the user password. In addition, I did not implement the features that allow each user one and only one vote per poll. Moreover, since my website does not keep track of who has voted, it does not redirect the user to the vote page if they have not voted and wanted to go to the results page.

* **References, Acknowledgements, And Outside Sources**
* The Bootstrap framework was used in association with my project to help with its scalability.

< https://getbootstrap.com/>

* The project was coded using Sublime Text 3 Trial version.

<https://www.sublimetext.com/>

* Knowledge about JQuery, Node, JavaScript, and CSS was gathering from w3schools.

<https://www.w3schools.com/

* Handlebars tutorial for generating frontend webpages.

https://www.youtube.com/watch?v=4HuAnM6b2d8

* Font Awesome iconic fonts

https://fontawesome.com/