Terrier Trivia

Boston University

- Pooja Chainani - Vanessa Giron - Esperanza Hernandez -
- Se’Lina Lasher - Alexandra O’Connor -

<u>Audience</u>

The title of our project is ‘Terrier Trivia’ and thus, our project’s audience is intended to be our fellow Terriers, either new or returning alumni. The first part of our project consists of an Android Application in the form of a trivia app. When started the application opens to a loading page that displays our logo and slogan while the questions and categories are being generated. Once the categories and questions are ready to be used the loading page exits and becomes the category select page. We chose to user 5 categories to make the app seem more gamelike. Adding 5 levels of difficulty (or in our case school years) we were able to appeal to the user by allowing them to have input on the types of questions they would like to receive. Depending on the level the user selected their points per question will be changed. For an easier category like Freshmen, the user only gets a hundred points per correct answer, however for a more difficult category such as Senior, the user will receive four hundred points per question. The user can then answer questions until they receive three strikes or until they get to the end of the questions in their chosen category (the user never receives repeat questions). Once they’ve completed a round of trivia a results page will be displayed that shows their current score and their overall high score on the app. The user can then choose to either return to the main menu, or to post their current score to our leadership board on the website.

The app’s website includes a Home, About Us, and Leaderboard page. The user is first led to the Home page that displays a welcome message. They can then navigate from the website’s aside to either the About Us page or the Leaderboard. If they choose to view the other sites, they can click on one of the navigation links to learn more about the team members on the About page or to catch up on the current game leaders on theLeaderboard page.

<u>Android Application</u>

<u>*Trivia Questions Implementation*</u>

In the Main Activity of the code an array was created that would hold all 100 questions that the app would cycle through for the user if they choose the all category (if the user choose one of the other four the question pool is limited to 25 questions but the process is the same). The template for the array was: {“Question”,”Right Answer”,”Choice 2”,”Choice 3”,”Choice 4”}. By using this template we were able to index into each row of the array to isolate the question from the choices. This left the choices that were than randomized in the order that they appeared on the buttons in the app. This made it so the right answer was not always the first answer that appeared. The question was put in a text box above all the answer choices. Once the question has been used it is then removed from the array so it can never be repeated in a game cycle. In the background of the app as the user is answering the questions a checkAnswer function is being used. After each question the checkAnswer function checks if the button pressed held the right answer, if not the user is given a strike, else the user's right answer count is incremented. Regardless of if the user answered the question correct or not a background counter is being incremented to keep track of the amount of questions the user answers. These counters will be used to calculate the high score and total score. Once the user receives three strikes or has answered all the questions in the array, the results page is displayed.

Total and High Score Implementation

The total score is calculated by taking the right answer count and multiplying the count by a number that is determined by the category. To determine which number to multiply the right answer count by a case statement was used. If the category the user choose is All, the total score is multiplied by 250, 100 for Freshmen, 200 for Sophomore, 300 for Junior and 400 for Senior. The higher the difficulty the more points the user will receive per question. The total score is then compared to the user's high score. If this the user's first time playing the game then their stored high score is automatically defaulted to zero, and thus whatever score they receive will automatically become their high score. This is then committed to the apps memory. Meaning that even if the user exits the app and reloads it, their high score will still be saved. The next time they play a category their total score will be compared to the saved high score. If the total score is higher than the saved high score, the high score is overridden with the total and it becomes the high score. And the process is repeated every time the user chooses to play.

Sound Effects Implementation

Two sound files were downloaded from a website called ‘Freesound’. It is a large database that collects audio snippets, sounds, samples, recordings, and bleeps released under a Creative Commons license that allows people to reuse. These two sound files are then stored in the file named ‘raw’ in the android program. In the ‘checkAnswer’ function of the class Main Activity, the MediaPlayer class is used to control the playback of these two audio samples (correctsound.mp3 and wrongsound.mp3). Within the ‘checkAnswer’ function, there is an ‘if’ statement that compares if the choice made by player equals the correct answer to the question given. If the choice made the player is correct, the ‘correctsound.mp3’ audio is accessed from the ‘raw’ file and stored in a new variable called ‘correctsoundMP’. If the choice made the player is incorrect, the ‘incorrectsound.mp3’ audio is accessed from the ‘raw’ file and stored in a new variable called ‘incorrectsoundMP’. In order to play the sound samples, ‘correctsoundMP’ and ‘incorrectsoundMP’ call the MediaPlayer built-in start() function. The purpose of the two sound files is to indicate the user if they got the question correct or incorrect, and these sound effects improve the overall design of the application.

Randomized Commentary Implementation

In Main Activity, two string arrays called ‘correctResponses’ and ‘incorrectResponses’ are created with several comments that appear after a player chooses an answer. In the function ‘checkAnswer’, there is an ‘if’ statement that compares if the choice made by player equals the correct answer to the question given. If the choice made by the player is correct, a random number within the range of 0-11, which represents the twelve comments stored in the ‘correctResponses’ string, is generated. If the choice made by the player is incorrect, a random number within the range of 0-8, which represents the nine comments stored in the ‘incorrectResponses’ string, is generated. The randomly generated number is then used as the index number of the string and the comment is then stored to the string (‘correctResponse’ or ‘incorrectResponse’) called ‘alertTitle’. If the answer the player chooses is correct, a positive message is shown that states that they are correct. However, if the answer is incorrect, a negative message is shown that states that they are incorrect.

Leaderboard Activity Implementation:

The Leaderboard Activity is used by asking the player for their username and posting their scores to the leadership board on the website. When a player inserts their username ('userInput') in the provided box, the getText() function takes the text of the name, converts it to a string by using the toString() function, and stores it to string variable called 'username'. The variable 'username' is then passed as an argument to the function showToast(). Once the player enters their username, they have to click on the 'Post' button that pushes their username, id number, and current total score to the Firebase database. Once in the database the User's score will be pulled and then added to the website's leaderboard. Once the scores are posted, the player clicks on the 'Return' button and it will take them back to the main page to restart the trivia game.

GUI Implementation

The graphical user interface was designed in a manner to match the Boston University official colors and create a fun and classy look for users, who would most probably be Boston University students, to interact with. The application logo was designed in Adobe XD and the image was downloaded from www.stockphoto.com. The logo was then replaced with the default Android Studio logo as a new .xml file and the icon_launcher_background was used to replace the app icon and the logo was displayed in several screens throughout the app. The background of the screens were made as a new .xml file called bgapp.xml were coded by using a startColor as BU's official red color and an endColor as white to make it a gradient. The background for the buttons were created in a new .xml file called bg_white.xml and the corners were curved to create visually appealing buttons. Additionally, the layouts and sizes of the text and buttons were adjusted and coded in the LinearLayout code so the layout would stay uniform and visually appealing no matter if the question and answers were short or long and to ensure a presentable view of the app in all mobile types. Lastly, the Splash Screen was implemented at the start of the program so that when the user initially opens the app, the app logo and slogan would appear on the screen for 3 seconds and then fade into the home page of the app. The Splash Screen was made by creating a new activity called Splash Activity and was run using a private static int variable to store the variable for how long the screen should appear for. After three seconds the Splash Screen transitions to the Start Activity screen.

Website Application

The website was designed using basic web design tools such as HTML and CSS and youtube tutorials. To start off, the index page was first developed along with the corresponding main.css file. The index.html file structures the website according to nav, aside, ul elements and a main. It introduces the user to the app and what to expect to find on the website. The navigation bar, which is present on every page, includes 'Home,' 'About Us,' and 'Leaderboard.' The about.html file includes a brief overview of the team members and a thank you to TF Miguel Mark for helping the team get the leaderboard working on the website and app. The leaderboard.html file required some JavaScript to get the Firebase database connected. The css files for the leaderboard and about html files followed a similar format for the main.css file in incorporating the navigation and clickable links. The links on the navigation aside required a feature in the css files called 'hover' that made the background black when the user hovered over the link. The retrieved data from Firebase was then formatted to match the font and set to an appropriate font size. The website pages are all linked within each other by including the html links on the navigation element. It should be noted that the download button -- cta class in index.html -- is not yet linked to the google play website since the team has not yet submitted the app for review to Google.

Firebase Application

Usernames and scores are stored on Firebase, a NoSQL database provided by Google. When a user finishes a quiz, they are given the option to post their score. If the user selects "Post", then they are prompted to enter a username. The username and score are then pushed to the database as a document. The website includes a javascript file, app.js, that queries the database for usernames and passwords. This is included in the leaderboard.html file. It accesses the database using the API key, the domain, URL, and storage bucket. A reference object is then created using the database name. The database can then be queried using firebase query functions such as ".on". When this function is called, it calls our function onDatabaseChange(snap). We wrote this function in order to print the results of the query which are assigned to snap. onDatabaseChange loops through the query result object, assigning values to an html variable. This variable is then displayed on the website using .innerHTML. The javascript file is called in leaderboard.html.

Final Thoughts:

Challenges:

One of the main challenges was integrating the app and the website with a database, mainly due to our lack of experience with servers and database and because of the many options available. A number of options were explored. The first option was using MySQL with PHP and also with python. MySQL required a lot of downloads and didn't work well with the Android base. We then tried a NoSQL method, MongoDB. This was easier to work with and could be used entirely online. In researching how to integrate this with the app, we discoverer MongoDB stitch, a version intended for integration with the Android SDK. The documentation for this platform was not very detailed however and there seemed to be a debate online as to whether or not Mongo could work with Android. It was then suggested that we use FireBase because it worked similarly to Mongo but worked better with Android on account of also being a Google product.

Achievements:

Learning how to integrate a database that connects the app to the website forced us to learn how databases worked and forced us to dive deeper into learning HTML and Javascript. Going forward we can further develop these skills for the workforce and for our own personal benefit.

The further we coded the Android App the more we noticed how we were implementing OOP in something other than C++. For example, to post to the Firebase database we needed to create a User class that included a default constructor, a setter and a getter, which is very similar to the previous Programming Assignments.

Links needed for this project:

GitHub: https://github.com/Van526g/Terrier-Trivia-
Freesound: https://freesound.org
Website: https://oconnoran17.github.io/TerrierTrivia/index.html