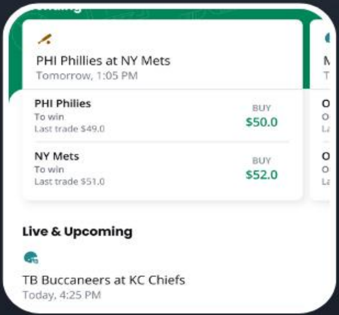
GAMBLING SCREEN PROPOSAL

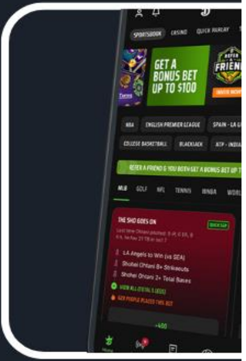
Prepared for :

University of Notre Dame

Prepared by :

Joe Huebner,
Tom Maier,
John Noonan,
Martin Rybertt,
Justin Share





+1-800-426-2537
GamblingScreen.com
South Bend, Indiana

Contents

1. Introduction

Sports gambling is everywhere in the world around us whether we like it or not, however, the increasing use of sports betting has gotten rid of some of the best things about sports. These things include: meeting new people who either are for or against the team you are rooting for, celebrating a big play in a social setting, or even just enjoying every second of the game. However, as sports betting and gambling gets more popular people are starting to lose touch with what made sports so alluring as a child because they are micromanaging every play and looking down at their phone throughout the whole game. The problem in question is that we spend too much time looking down at our phones checking out bets that we sometimes miss the actual game going on. With this dire problem impacting our viewing experience, we came up with a solution of an overlay for your screen that displays your bets in real time. Using the API from SportRadar to track stats, the device will read that data and check to see if you have any bets placed regarding the play that happened. For example, if you took action of “Derrick Henry over 70.5 yards rushing” and he rips off a 7-yard run, a small popup will appear in the corner of the screen showing a small description of what happened and a bar filling up. This bar will be relative to the placed bet value. There will be feedback given for legs of parlays hit or for bets being placed. This will alleviate the need to look down at one’s phone every single second trying to figure out how close their bets are to hitting and can bring back the social aspect of sports.

The solution that we came up with, as discussed earlier, is to create an auxiliary screen that will display bets from different users and will track them in real-time. When a bet hits or loses then the system will display lights and sounds that alert the user whether or not their bet hit. The design will need to include various different subsystems in order to achieve this goal that we have. This is first done through the use of a computer monitor that is connected to the board via a Video Graphics Array (VGA) cable. This monitor will then be powered through its wall power outlet to not require the board to be able to take 120 V at 60 Hz and rather can be allowed to just run based on a different power source. This monitor will be necessary to be able to update frequently in real-time to show the bets hitting and to be able to track what is happening to each bet as well. This monitor will need to be able to take in information through a microcontroller like an ESP32. The next thing that is needed for this design to work is a way to take in user input to display the correct bets. The solution we came up with for this problem is to create a website that users can input their bets into that will then be inputted into the screen. The website consists of a page that will allow the user to input their name to distinguish their own bets on the screen. Then through a drop down menu the user will pick their sport, what type of bet they want to place(game outcome, player stat, etc), and then finally will be able to input the odds and bet that they want to place. An example of this would be if the user's name was Shaun they would input “Shaun” and if they want to put a baseball bet for Yoshinobu Yamamoto over 7.5 strikeouts they would follow the prompts until they can upload it to the screen. This then will connect it to a database that the ESP32 can pull from to upload it to the screen. The next part of the system would be to have something that could go read the bets and get the stats from the game in progress. The team decided the best way to do this would be through the use of a sports API. The first problem was finding one that would be able to allow us to get the stats in real time which was when we found SportRadar API. This is the API that many actual sportsbooks use which made it a perfect candidate for our system. This API would give us any game in any sport we wanted as long as we had a membership or in one sport if we possessed a free trial. The API would then get a call and provide a block of text that would then be parsed through to see the progress of each of the bets. After we are able to read in the bets we need to be able to provide feedback to the user. The team decided that the best way to give feedback would be to add audio feedback and to provide additional visual feedback. The audio feedback was done through the use of a speaker that would be connected to the board and it would take in whether or not the bet would increase or decrease, or if it would win or lose. When the bet would win or lose the speaker had a collection of 16 bit wav files that were either audibly pleasing or unpleasant sounds. These sounds would then increase in intensity of good or bad sounds based on whether or not the bet just incrementally increased or if the entire bet hit. For example, if the bet incremented the sound of collecting a ring in the Sonic the Hedgehog games would then play, furthermore, if the whole bet hit the sound of the Boston Bruins goal horn would then play. This audio feedback helped the user be able to tell whether or not their bet is doing well based on audio cues alone. The next way that the team helped show that a bet hit would be through an auxiliary LED strip that would flash different colors based on the bets. This LED strip was connected to flash either green or red depending on the incrementation of the bet. The strip also would not flash at full brightness unless a bet fully hit. This flashing helps correspond with the audio feedback where the intensity would increase based on the severity of whether or not the bet hit or just incremented. Finally, the system needed a power system that would provide the correct amount of power to the individual subsystems without frying the board. This was done through the use of 3 LDOs that were wired in parallel that provided different voltages to the different subsystems. The power system was connected to a 4.5 V wall wart that connected through a barrel plug that was positioned on the board. However, the board didn't just need to be powered, it needed to be programmed as well. Since the board needed to be programmed the team decided to put a MOSFET switch on the board to protect the ESP32. This MOSFET Switch was constructed using a PMOS transistor and a Schottky diode to allow the USB to be the only thing that powered the ESP32 when it was connected. Through the collection of subsystems our team was able to create an auxiliary gambling screen that helped to solve our original problem that we wanted to solve.

The design that we created through the use of various subsystems was able to solve the problem in a way that was satisfactory. The design was able to provide the user with their bets hitting in real time and was able to create a user interface that was pleasing to the eye. The power system of the design was able to provide enough power to the necessary subsystems to allow them to work in conjunction with one another. Furthermore, the MOSFET Switch was able to provide safety to the microcontroller while also allowing the power to be plugged into the board through the barrel plug. The design was able to play sounds and display lights when the bets were incremented. In addition to the bets displaying sounds and displaying the lights when incremented, they also were able to provide when the bets hit or even lost. Through the use of the website and utilizing the database the correct bets were able to be made and then were displayed to the screen. Also, through the API calls these bets were able to be parsed through in real time and were able to be displayed as they were happening in real life. Nevertheless, the project is still just a prototype where there were a couple bugs that would need to be fixed if it were to become a product in the open market. The first of which is that the LEDs would require their own driver/microcontroller as they are unable to run without using an interrupt function due to the tight timing that is needed to run them correctly. The power on the board could be improved as the LDOs were unable to run with as much of a load as they were experiencing creating problems with the sound. Additionally, a product on the open market would need to be able to take in bets from other people and be only displaying the ones that an individual user wants and not all the bets that are in the system. Finally, the need for more API keys is necessary since SportRadar only provides one sport per free trial and limits the amount of calls one can make per day. This however was not in the budget since a membership costs around $15,000. Therefore, the design was able to meet our requirements and solve the problem that we were trying to fix as it definitely enhances the viewing experience while also not taking away from the game. It is deemed a success even though it has improvements that could be made. Nevertheless, given more time and resources this design would be able to go on the open market or be sold to a sports betting company if there was one willing.

The system that was created is able to do what we expected it to do and more. The system that was created by the team was able to fulfill all of the design requirements we aimed to provide. This design was able to provide visual and auditory stimulation due to bets hitting or losing which is ultimately what we wanted to be able to produce. Nevertheless, when we approached this project it was expected that real-time updates would be quite the challenge and we might've been blocked due to budget constraints. This is because when we first looked at APIs we realized that a lot of them were out of the budget or would not provide live games. This was an issue of concern as being able to update live games is a design requirement that we wanted to achieve. Nevertheless, through contacting sportsbooks directly and asking them what sports API that we used we were able to find SportRadar. This is the same API that many different sportsbooks use and they said that it was fine for us to use free trials as our use rate was so low and would not be for profit. This allowed us to be able to get real-time bets that actually would have the screen update before TV broadcasts due to the integrated delay in TV broadcasting. This created interesting interactions with viewing since the score in games would update before you could see it on screen which could either bring joy or despair in fans that are watching it. However, if this would end up going into the market a delay could be created that would prevent this from happening. However, the design did not do everything that we wanted it to do originally due to unforeseen problems with integration. As stated before the lights turn off the screen when they run which could be due to many different problems regarding: library incompatibility, power limitations, or even shielding problems. This was expected to be resolved through RTOS, but it was not, which led to many man hours trying to find a solution that was not resolved. Furthermore, the monitor that was used would not be able to stay on consistently and would seem to go into sleep mode at times which was frustrating as well since the sleep settings on the monitor were unable to be accessed leading to there to be confusion on what the problem was. Therefore, if a different monitor was used that would be able to provide the sleep settings then it would be able to stay on consistently throughout the whole process. While the bugs in the system are frustrating the system was still able to function and was functioning better than expected throughout testing and demonstration. Something that was also better than expected was the VGAs ability to update in real time without tearing since the screen was a lot lower of a resolution than was originally wanted due to limitations on the ESP32. However, despite these limitations on the ESP32 the display was able to look aesthetically pleasing and user friendly throughout the whole process.

Therefore, while the design might not have come out as lofty as allowing there to be an interactive display on one monitor that would display the game behind it. This is due to the need to partner with a broadcasting company, it was able to be created in a way that fulfilled the requirements of the system. This design also was created with room to grow which would help if someone in the future would want to create a product like this as it would need to undergo a few changes before it goes to market. All in all the auxiliary gambling screen can be described as a success because it is able to enhance the user experience of a sporting event, while also bringing back the social aspect of sports. This social aspect of sports being brought back is beneficial to keeping sports as intriguing to new fans as it was for fans of the past and allows betting to not take away from the love of the game that many people might have.

2. System Requirements

The system requirements of our gambling screen need to balance the reception of user data input and the extraction of data from within the SportRadar API. Then process the data from the API in our processor to show on our display. One of the main challenges in our current proposal is being able to optimize the data processing such that the ESP32 does not crash. We are not in need of an extremely fast system as current sports betting applications are not fast either, we need to update the display information every thirty seconds. We are also using our speakers and LED lights to react to changes in the states of our users' bets, stimulating the user by giving either positive or negative audiovisual feedback depending on how it affected the outcome of their bets. Our motivation is that by only causing users to react when an update on their bet occurs, they will return to paying attention to the game and socializing with people instead of looking at their phones.

In short, the user will acquire our sports gambling display, and either mount it to their own TV screen or place it on a stand right next to their screen, making it ready to use. The user will connect to our website, upload their sports betting data, then sit back and enjoy the game. Our system will have to be able to connect to the internet via WiFi to both scrape the sports data API, receive user input from a website to know which bets to track, and with which priority.

Therefore, the requirements that needs to be met for the system to work properly include:

- **User Input Parse:** For this requirement we need to be able to create an interface (most likely an app or a website) for users to input their current bets, place priorities, current money invested, and more. This requirement is important because it will guide what we extract from the SportRadar API to minimize total memory use and the times we access the SportRadar API which has a limited number of uses per month, thus minimizing cost and making our memory usage more efficient.

- **Reading sports data API:** This is the integral section of our project, we need to be able to access, read, and interpret information from a sports data API, all of which with a ESP32 microcontroller. There are several projects on Github where they have been able to achieve this, we will be using code from those projects and adapt it to our needs. Additionally, we need to create a parsing algorithm that is able to extract the information provided by the user from the data provided by the API.
- **Creating a real-time display screen reaction logic:** The user inputs need to be properly identified and matched with the data received from the API to send instructions to our peripheral systems to provide adequate and pleasing responses when there is a change in-game.
- **LED Display:** The LED display has to be able to connect to ESP32, we will either use USB or VGA protocols to connect our screen with our microcontroller. Both of these protocols have been seen to work. Furthermore, we are not concerned with constant instantaneous updates to our screen so speed is not our priority. We need to be able to create firmware with an aesthetically pleasing and simple GUI that can receive updates when their bet information is updated.
- **Audio Processing:** We need to create appropriate firmware that is able to process the selected audios we select for positive or negative reactions, and transform them into signals our speaker can understand.
- **Speaker Driving Circuitry:** If we want to connect a speaker to our board we will need to create a driving circuit that will protect the speaker from possible shorts and will also provide sufficient gain while maintaining signal fidelity. We need to select an appropriate amplifier along with creating a proper power supply for the speaker to function with a low signal-to-noise ratio.
- **Power Regulation:** Our board will need to handle different levels of current in our board. We will receive power from an outlet, however we need to transform that into currents for our board, our screen, and our speakers. Moreover, our power regulator needs to protect the ESP32 microcontroller when both USB and power from the outlet are connected to the board as these could be providing different grounds that could fry our PCB. The power in the board has to be enough for the function of an LED strip, a speaker, and the ESP32 and its dependents. Our auxiliary display will receive current on its own.

Mechanical Requirements: We need to create a casing that is able to protect our PCB board from hits, falls, etc., while at the same time providing an aesthetically pleasing design that is not distracting. Moreover, this casing needs to have space to connect our peripherals, USB, barrel jack connectors and our speakers, making sure that these are safe in place and will not move causing issues with connections. In the case of the speaker, it also has the additional consideration that it should be a good fit with his surroundings so when it is outputting audio, it does not cause noise when moving.

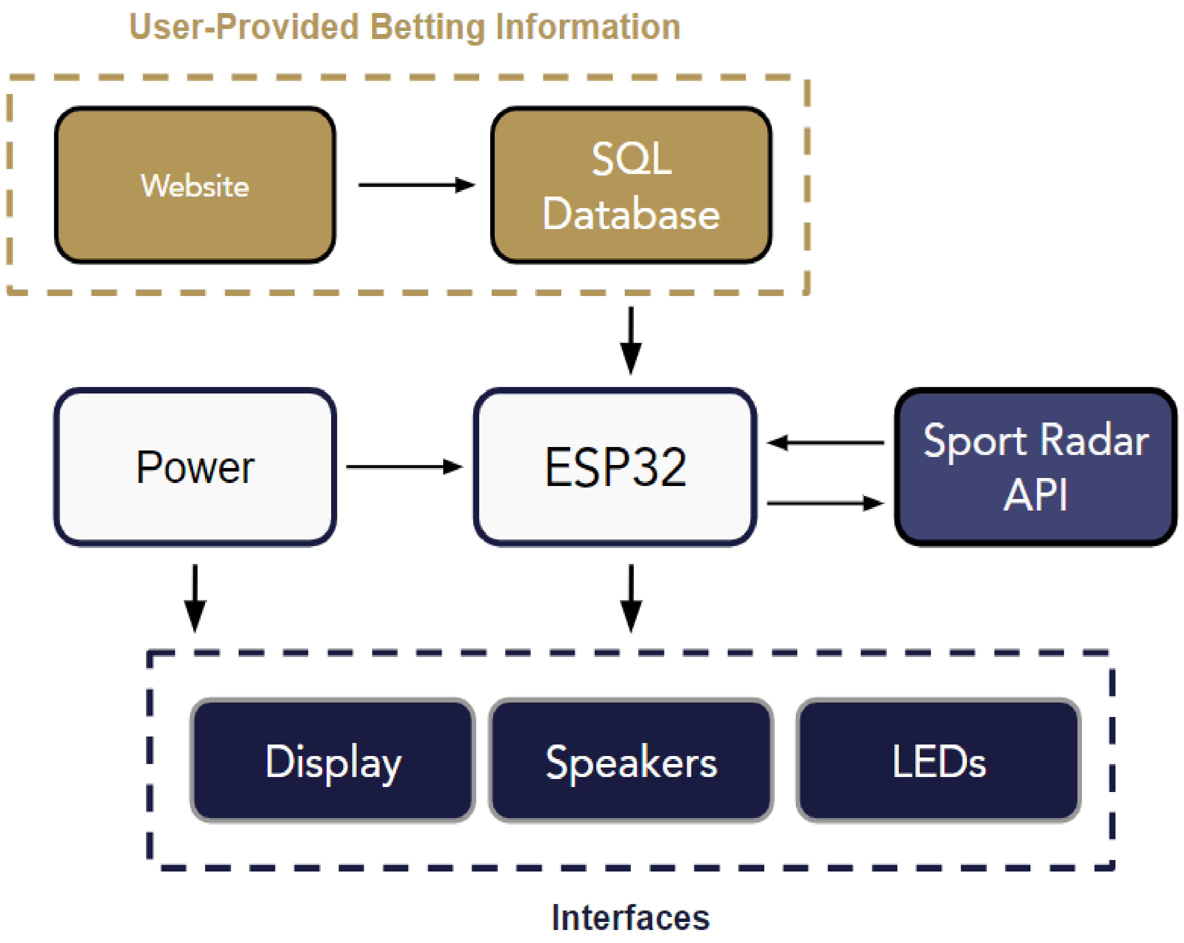
- **Website & user GUI design:** The main consideration in this section is to have a simple and easy-to-understand interface for the user. We require a website that allows the user to upload the full information of their bet, i.e. their name, the player or team they bet on, the type of bet, over or under. The website needs to be intuitive for new users to be able to quickly input their betting information and it needs to be able to convert the information from the user and send it to our microcontroller to use in the parsing of our data.

3. Project Description

3.1. Project Description

Users will input which statistics they would like to track via the website gamblingscreen.com. This will populate on the display that the device is connected to. Calling data from SportRadar provided API, the screen will update as necessary/appropriate. Based on the status of the desired statistics, different sound and light combinations will play over the course of the tracking period.

3.2. System Block Diagram



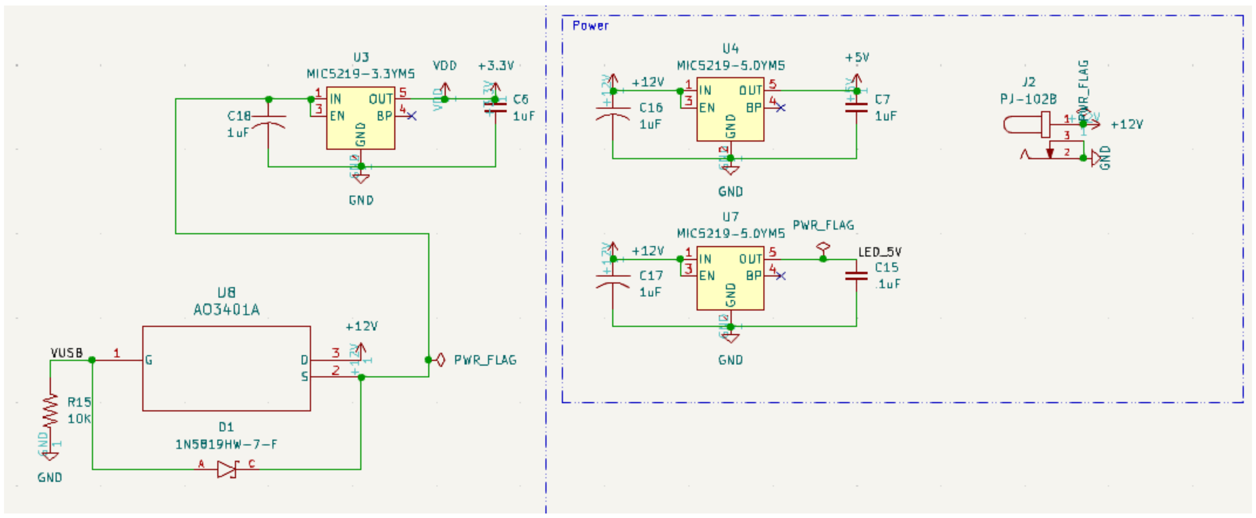
3.3. Operation of Subsystems

3.3.1. Power Subsystem

3.3.1.1. Power Requirements

- Deliver 3.3 V at 600 mA for the microcontroller
- Deliver 5 V at 500 mA for the speakers
- Deliver 5 V at 500 mA for the LEDs
- Provide a way that the board can run without a USB-C
- Provide a way that the board can run with both USB-C and Power connected
- Separate VGA power from the rest of the board to not overload the board

3.3.1.2. Power Schematic



3.3.1.3. Power Function and Design

The power requirements of the system is to deliver the correct power and current to all of the available subsystems. The power subsystem consists of 3 Low dropout voltage regulators wired in parallel to give the subsystem the same amount of volts across the whole system. The LDOs that were used were two Mic5219-5.0YM5-TR and one Mic5219-3.3YM5-TR. These LDOs were necessary to give the correct voltage and amps to the needed components as the LEDs and speakers both need 5 V and more than 500 mA. The Mic5219-3.3YM5-TR is the LDO that is going to be used for the ESP32 since it provides 3.3 V and 600 mA to the microcontroller. The monitor is powered by its own wall outlet that will be providing it 120V at 60 Hz throughout the whole time. The design team made this decision to alleviate the difficulty of providing such a large difference in voltage to the board. The power that is being provided to the board is through a barrel plug that is providing 4.5 V at 1 amp total. This was done through looking on the datasheet to see how many volts and amps were needed to get the LDOs to be giving 5 V at 500mA for the speakers and LEDs. Therefore, through the parallel circuit it would be providing the same amount of volts and enough amperage will be distributed throughout the system to make all of the sequences work.

The board needed to be able to provide power to the board and run it without the need for a USB. This is where a MOSFET Switch comes into play. This switch was constructed using the AO3401A PMOS transistor in conjunction with the 1N5819HW-7-F Schottky diode. This construction is able to safely protect the board in case of both the USB and the barrel plug being connected at the same time. This also allows the board to be safely protected when the power supply is the only thing connected and likewise with the USB being the only power source connected. This design works because when the USB is not present the drain-to-source is forward biased and allows current to flow through the MOSFET. When this happens the Vgs becomes more negative than $Vgs_{(on)}$ and the switch closes allowing the power to flow through. After the switch closes, the MOSFET has a really small resistance from the drain-to-source and allows a near 0 voltage drop at the LDO that is powering the ESP32. When the USB is the only thing connected the PMOS is open and does not allow any current to flow through, but the Schottky diode allows the USB to flow through allowing the USB to be programmed and powered without the rest of the system needing to be powered as well. Finally, when both are present it comes down to whether or not there is a big enough voltage gap between the two sources of power. Therefore, when the USB and the barrel plug are connected, the USB provides a voltage difference of 4.6 V. This is found because the USB provides 5 V and the diode creates a voltage drop of .4 V. Therefore, since the PMOS is seeing only a .4 V difference between the gate and source it is under the threshold voltage that is needed to turn on the PMOS transistor creating an open MOSFET not allowing the ESP32 to be powered by the barrel plug. This power path design also can help alleviate stress that is on the LDOs through consistent testing if there happens to be larger voltage drops than anticipated by protecting the ESP32 from higher voltages that may be coming in from the barrel plug.

3.3.1.4. Power Design Process and Testing

The power system design process began as a necessity to provide the correct power to the board for all the subsystems to work together and to protect the system from the surge of excess power. The first place that was inspected was through the power system that was utilized in the first semester of Senior Design class that was offered in the fall of 2023. This class provided a basis of the knowledge on how to use LDOs on a board with the use of the XC6220 LDO on the kit board that was provided in the first semester. Through the use of creating the original kit board from scratch, knowledge of LDOs and how they can function was found.

Nevertheless, the power design needed much more thought than just the kit board's design as it needed to provide power to more than just the ESP32 as it needed to provide power to all the other components on the board as well. This was done through research to see if it was better to use LDOs in series or parallel for these processes because that way we could figure out what was best for our individual project. It was decided that LDOs in parallel would be the best since then we would not need to worry about voltage drops from other components when they are connected in series. This also would prevent the team from being stuck with the highest voltage we started with and if a LDO would fry that the whole board would still be able to work. This use of compartmentalization of the power system helped us troubleshoot challenges we faced in integration. Furthermore, the use of a barrel plug was decided because of the ease and plethora of choices of different types of barrel plug AC-DC converters on Amazon made it the right choice to help debug problems that may occur with our LDOs.

The use of LDOs in parallel was not just the only design challenge that was different from the original senior design kit board, the team also needed to figure out how to connect just power, USB, or both at the same time and not destroy the board or the microcontroller. This was done through research into the Applied Embedded System Design course offered in the fall of 2023. This course introduced a board that needed to be able to be powered by a battery and a USB either alone or in combination with one another. The fix that was found was through the use of a MOSFET Switch where a PMOS transistor in combination with a Schottky diode would provide the correct architecture for this problem. This works because the diode will pick the higher voltage coming in and will open based on if the gate-to-source voltage is higher than a certain threshold. This threshold voltage makes it impossible to power the LDO with the power supply when the USB is plugged into the board. This power path design allows the board to be protected from the power supply when it is being programmed.

When testing for the power system was starting the first thing to test was if the ESP32 was able to be powered. The first thing that was done was to check if the ESP32 was able to run a “Hello World” looping program while being connected to the USB. Then the team needed to test if the program would still run if we connected the power supply to it as well. Finally, the last thing to test was if the power supply would allow the program to still run while the USB is not connected to the board and only the barrel plug was connected. This was tested by using the LEDs. The LEDs were on a loop and would flash red every second and after disconnecting the USB the program continued to run and this allowed the team to see that the board’s MOSFET Switch worked. This allowed us to know that the board was protected throughout the time that it would be connected to the USB or even just the barrel plug allowing us to move further on in integrating the whole system to the board. However, the board had some struggles with the integration of all of the subsystems.

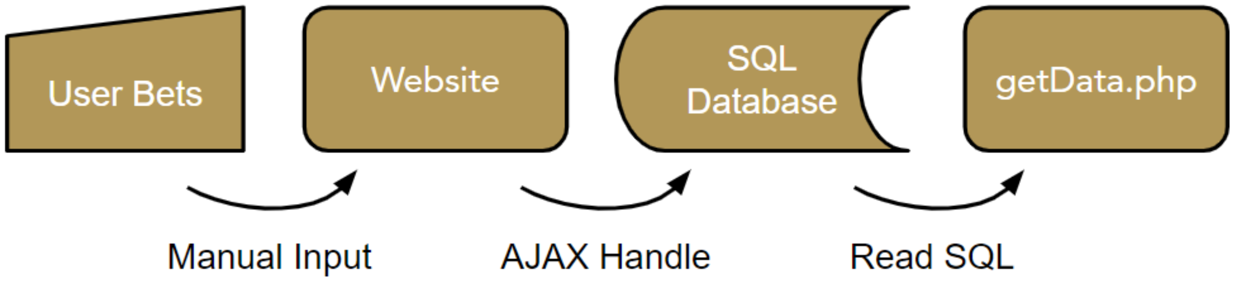
The integration of all of the subsystems was where the original power path design ran into some challenges. The original design called for 12 V because we did not want to experience a large amount of voltage drop within our components because that could ruin the quality of sound in the speakers or the brightness of the LEDs. However, this design choice provided the complete opposite effect because the LDOs were unable to dissipate the amount of heat that was provided by 12 V. The voltage drop that was experienced by the speakers at this voltage created them to fall under the threshold that was necessary for them to turn on leading to the speakers to cut out and play distorted noise. Throughout testing it was discovered that the board was increasing in heat and the voltage needed to be changed. The team discovered an adjustable power supply and proceeded to lower the voltage until the sound of the speaker would be high enough quality to provide adequate audio feedback to the user. The voltage amount that created this amount of audio feedback was 4.5 V. The LDOs at this voltage would still provide around 4.25 V and the voltage drop would only be around .25 V keeping the speakers in the correct range for use. This also provided enough voltage to the LEDs as well as they needed less amps than the speakers as they were only utilizing a third of their needed amps due to only using primary colors. Overall, the power system was able to provide the necessary amount of power and was able to integrate the subsystems together in a way that the screen could work.

3.3.2. Website Subsystem

3.3.2.1. Website Requirements

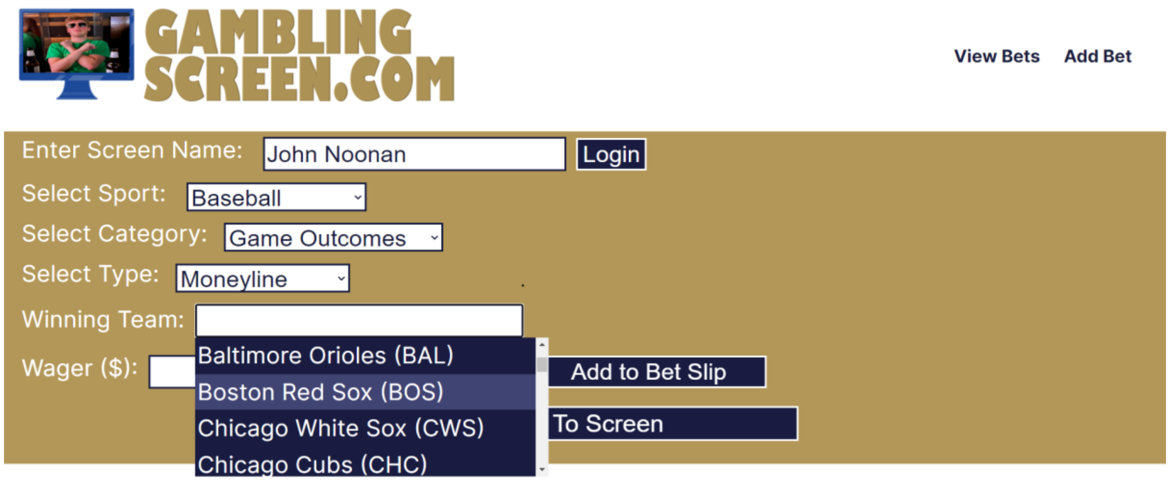
- Create a website that allows for users to input data about their bets
- Must be easily accessible and easy to use for users
- Must format user inputs in a way that is easy to process
- Must store user information somewhere that a ESP32 can access

3.3.2.2. Website Flowchart



3.3.2.3. Website Function and Design

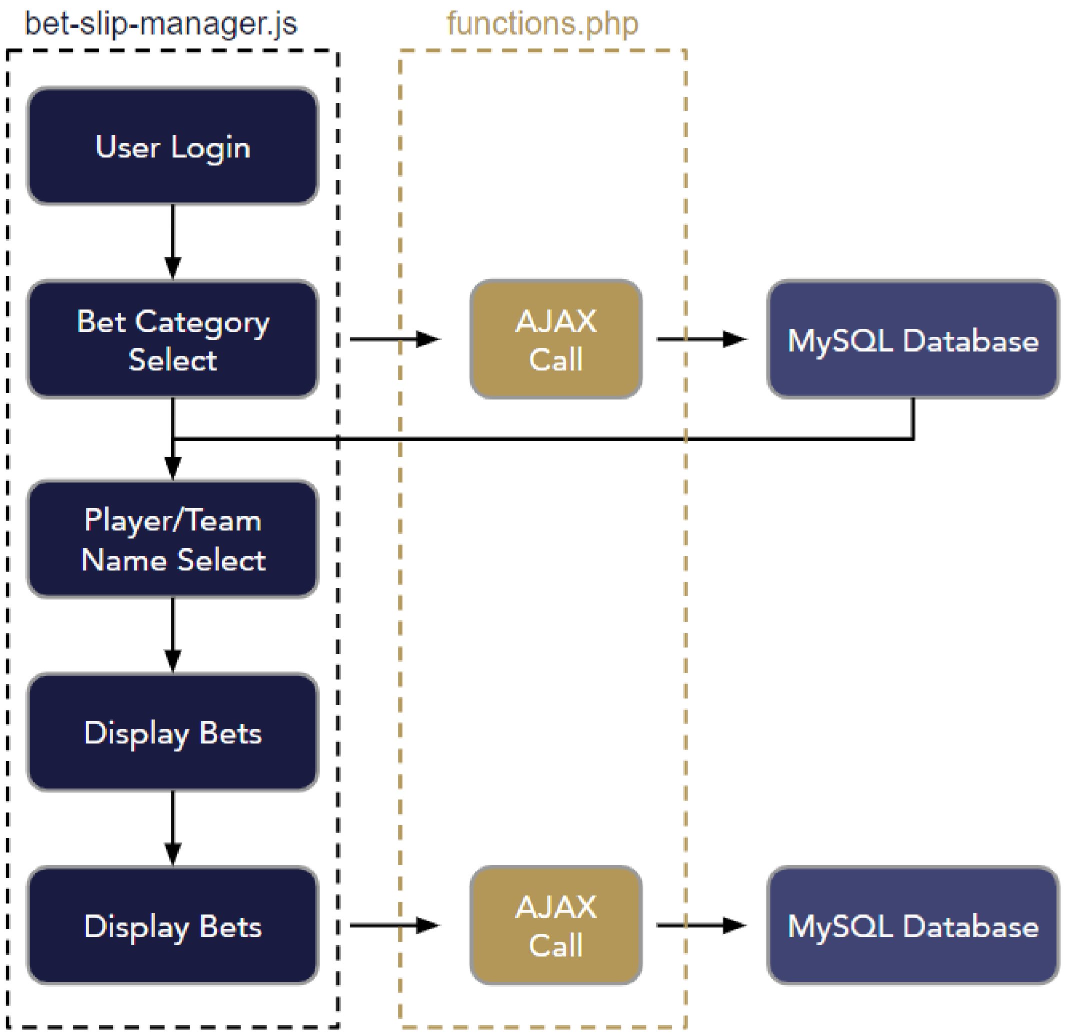
Gamblingscreen.com



Enter Screen Name: John Noonan
Select Sport: Baseball
Select Category: Game Outcomes
Select Type: Moneyline
Winning Team:
Baltimore Orioles (BAL)
Boston Red Sox (BOS)
Chicago White Sox (CWS)
Chicago Cubs (CHC)
Wager ($):

GamblingScreen.com is a website that allows users to enter their desired bet slip information. It is a domain that was purchased on a GoDaddy Web Hosting server. We decided to use this product for our website because it allowed us to attach a hidden MySQL Database and hidden files to the website domain. The files 'template-bet-slip.php' and 'bet-slip-manager.js' are the files that contain the HTML code for our website page. The file 'functions.php' contains the code that changes how the pages and files of our web server interact with each other. The files, header-custom.php, header-footer.php, header.css, and styles.css are all code files that have no function other than to make our website's user interface better looking and more consumer friendly.

Bet-slip-manager.js and functions.php



The JavaScript file ‘bet-slip-manager.js’ is the front-end controller that manages the states of all forms on the website. The script starts with the class ‘BetSlipManager’ that loads the configuration for all the possible betting options. The form begins with general options that are general to all user bet slips such as the user’s screen name, sport selection, bet category, and bet type. For each of these forms, the event listeners are initialized. Then, each form is set to appear once the higher-level form is submitted. In other words, once the user logins, the sport form is dynamically populated with possible sports. Then, once a sport is selected, the possible bet categories are populated. Finally, once the user selects a bet category, the possible bet types are populated.

The website was designed in this way so that it would be as intuitive as possible for new users. We wanted to ensure that users of our product were not overwhelmed by tons of forms that need to be filled out. This system is also advantageous because it ensures that all forms are filled out correctly, mitigating the potential risk of user error. Also, we chose this approach for this section of the website because it generalizes possible bet slips. By simplifying bets into bet category and bet type, we were able use a simple HTML cascading dropdown without needing any additional coding complexities. This is achieved in the functions ‘handleLogin()’, ‘handleSportChange()’, ‘handleCategoryChange()’, and ‘updateForm()’. Each of these functions dynamically creates and displays the desired form based on the previous form’s input.

After the user selects a bet type, ‘handleTypeChange()’ creates a new, single bet details form for the situation. The details form has a text input for either a player name or team name, and multiple possible additions based on the bet type that the user selects.

All possible forms are populated using a class named config. We did this to modularize our code and make adding to our code significantly easier. By using the class config, we can easily modify or add new sport, bet categories, or bet types without having to change any of the methods in ‘BetSlipManager’. Also, the class config allows us to make unique details form for every user selection.

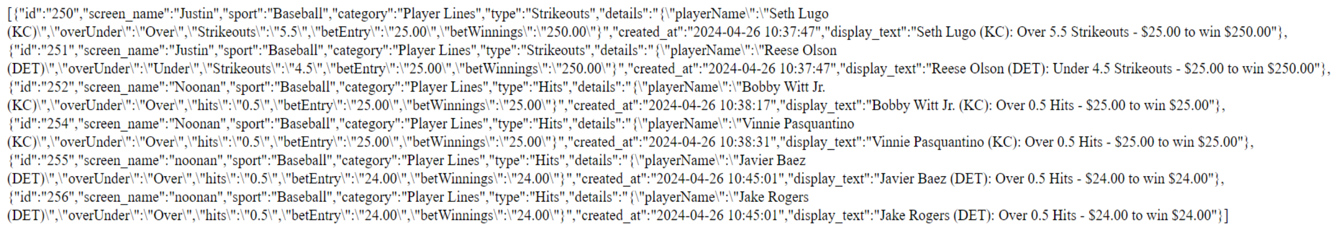
Once the bet details form appears, users can enter the team or player that they are betting on. In our website’s SQL database, there are files that have every sports team’s name and their updated rosters. To make the team names and player names accessible from our website, the functions ‘fetchAndSetupTeamNames()’ and ‘fetchAndSetupPlayerNames()’ make an AJAX post request via ‘functions.php’, which fetches the needed information from the server. Once there is a successful data retrieval from the server, the functions ‘setupAutocompleteForWinningTeam()’ and ‘setupAutocompleteForPlayerNames()’ create a form that allows website users to pick a name from the database.

We felt that this was a necessary addition to our user interface. We wanted to remove the possibility of user error, either through typos or selecting an improper player. Including the autocomplete form also ensures that the user information is entered in the format that we need it to be in for future processing.

Once the user enters all of their betting information, a button becomes visible that says “Add to Bet Slip”. When the button is pressed, the user’s choices for each form are displayed at the bottom of the screen. For each additional bet that the users choose to add, they can press the button again and cause additional bets to be displayed. Additionally, for each bet that is displayed there is a “Remove” button that will allow the user to remove that bet from their slip. Once the users choose the bets they want, they can press the button “Upload Bets” which calls the function uploadBets(). uploadBets() serializes the bet slip data and uses another AJAX handle to send it to the server for future processing. The function also handles the response to inform the user of success or failure.

The “Add to Bet Slip” button and “Upload Bets” are separated to prevent duplicate handles being sent to our server. We felt it was important that displayBets() showed users exactly what information users were sending to the database. We also wanted to give users the opportunity to see their selections before uploading so that they would have the opportunity to remove bets. It would have been significantly more difficult for us to allow users to delete bets from the server than it was to let them remove them before uploading.

getData.php



```
[{"id":"250","screen_name":"Justin","sport":"Baseball","category":"Player Lines","type":"Strikeouts","details":"{\"playerName\":\"Seth Lugo
(KC)\",\"overUnder\":\"Over\",\"Strikeouts\":\"5.5\",\"betEntry\":\"25.00\",\"betWinnings\":\"250.00\"}","created_at":"2024-04-26 10:37:47","display_text":"Seth Lugo (KC): Over 5.5 Strikeouts - $25.00 to win $250.00"},
{"id":"251","screen_name":"Justin","sport":"Baseball","category":"Player Lines","type":"Strikeouts","details":"{\"playerName\":\"Reese Olson
(DET)\",\"overUnder\":\"Under\",\"Strikeouts\":\"4.5\",\"betEntry\":\"25.00\",\"betWinnings\":\"250.00\"}","created_at":"2024-04-26 10:37:47","display_text":"Reese Olson (DET): Under 4.5 Strikeouts - $25.00 to win $250.00"},
{"id":"252","screen_name":"Noonan","sport":"Baseball","category":"Player Lines","type":"Hits","details":"{\"playerName\":\"Bobby Witt Jr.
(KC)\",\"overUnder\":\"Over\",\"hits\":\"0.5\",\"betEntry\":\"25.00\",\"betWinnings\":\"25.00\"}","created_at":"2024-04-26 10:38:17","display_text":"Bobby Witt Jr. (KC): Over 0.5 Hits - $25.00 to win $25.00"},
{"id":"254","screen_name":"Noonan","sport":"Baseball","category":"Player Lines","type":"Hits","details":"{\"playerName\":\"Vinnie Pasquantino
(KC)\",\"overUnder\":\"Over\",\"hits\":\"0.5\",\"betEntry\":\"25.00\",\"betWinnings\":\"25.00\"}","created_at":"2024-04-26 10:38:31","display_text":"Vinnie Pasquantino (KC): Over 0.5 Hits - $25.00 to win $25.00"},
{"id":"255","screen_name":"noonan","sport":"Baseball","category":"Player Lines","type":"Hits","details":"{\"playerName\":\"Javier Baez
(DET)\",\"overUnder\":\"Over\",\"hits\":\"0.5\",\"betEntry\":\"24.00\",\"betWinnings\":\"24.00\"}","created_at":"2024-04-26 10:45:01","display_text":"Javier Baez (DET): Over 0.5 Hits - $24.00 to win $24.00"},
{"id":"256","screen_name":"noonan","sport":"Baseball","category":"Player Lines","type":"Hits","details":"{\"playerName\":\"Jake Rogers
(DET)\",\"overUnder\":\"Over\",\"hits\":\"0.5\",\"betEntry\":\"24.00\",\"betWinnings\":\"24.00\"}","created_at":"2024-04-26 10:45:01","display_text":"Jake Rogers (DET): Over 0.5 Hits - $24.00 to win $24.00"}]
```

To pull bet information from our server, we created the file ‘getData.php’ on the web server. getData.php reads through the table of information that it receives from gamblingscreen.com, and it displays it as a text file. ‘getData.php’ is embedded into the html of our web server, meaning that the text file can be displayed (as shown above) from the URL: ‘gamblingscreen.com/getData.php’.

We decided to display the information from the server publicly at the address because it is the easiest way to get the information from our database to our ESP32. Directly connecting our ESP32 to read from our server would have required us to install additional libraries and plug-ins to our project, drastically increasing its complexity. Our data does not require any privacy, so putting it on getData.php allows us to bypass those complexities and the need for credentials.

3.3.2.4. Website Design and Testing

The website design began with ensuring that we would have a place to store data and access it with relative ease. The GoDaddy Web Hosting server was the first place we looked, and it gave us the power to do everything we needed. We knew we would be able to make AJAX handles in functions.php, and it would handle data transfer from a website to a data server.

Once we knew GoDaddy was a viable server option, we made the corresponding purchase and began programming our website itself. To start, we followed the Cascading Dropdown tutorial from W3Schools in HTML. This gave us a good start to generate forms that were dependent on the results of a previous form. The simple dropdown gave us the exact needed functionality for the user’s ability to select sport, bet category, and bet type. Then we wrote the necessary method to create the bet details form based off of the bet type. As we added more and more bet types, the complexity increased, and we created the config class to simplify form generation. At this stage, we tested all possible bet types that we wanted users to have the option to choose from. We were successful and confident, we did not do any more testing with the form generation itself.

Once the forms populated as desired, our next step was to display the form inputs for the user to see. We implemented and tested the displayBet() function until bets were properly displayed on our website. Also, at this stage we added the "Remove" button which hid any bets in displayBet() from the screen.

Our next step was to create the "Upload Button" and add the AJAX handler to functions.php. The upload button successfully sent all the information we wanted to the server, so little testing was needed. At this point in time, we considered the website subsystem at or near completion.

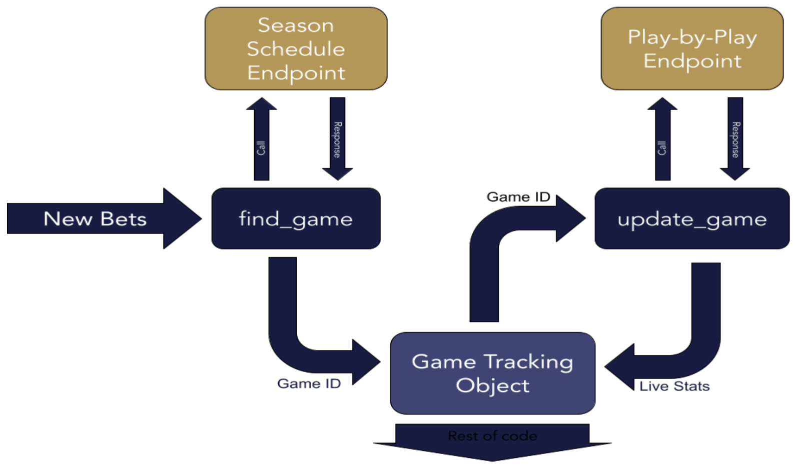
When we began integrating the website with the API and display, we realized that it would be easier for us to parse the website's information if we limited the user's responses to certain forms. To do this, we changed many of the forms from text inputs to dropdown options. This made taking the values from the forms consistent and easier to use logically. Our next step was to make sure that users were selecting teams and players in the proper form. First, we created two tables in our SQL server: one that contained all player names, and one that contained all team names. Then we created a new AJAX handle that sent that particular information from the database to the website. We tested by adding new bets on the websites and determined that the forms were populating as desired.

3.3.3. API Subsystem

3.3.3.1. API Requirements

- Must be able to pull from sportradar.us to get live player and team information
- Needs to read information onto ESP32 and parse into processible inputs

3.3.3.2. API Flowchart



3.3.3.3. API Function and Design

This subsystem retrieves and stores live data about an ongoing MLB game. To retrieve the live data, it uses SportRadar's MLB API, and to store the data the code includes the Game and Player classes.

The Game and Player classes were created to store the relevant information retrieved from the API. These two classes have members that hold any information that could be relevant to a bet in the system. The Player class is the simpler of the two, consisting only of a constructor based on the player's name, private members that include the player's name and stats, and public getting and setting methods to interact with the private members. This class allows individual player stats to be tracked based solely on the player's name.

The Game class is much larger and more complex than the player class and has a wide variety of members and methods. The class is designed to hold the current and past state of the game (i.e. the current score, current inning, past inning scores, current number of outs, live player stats, whether the game is finished) and the unchanging elements of the game (i.e. the SportRadar API game ID, team names, etc.). The main Game constructor used creates a Game based solely on its ID. This ID is used later to populate the Game with the right stats. The key methods in the Game class are the getters and setters of information, the add and find Player methods, the Game reset method, and the retrieve_stat method. Similar to the Player class, the getters and setters get and set the simpler members of the Game class (e.g. team names, number of outs, scores, etc.). Any getter or setter that has to do with a player stat also includes the player name and inning state as an input to ensure the class is accessing the correct player in the correct roster. The add and find Player methods are similar to getters and setters, but they cater to the way players are found and added to the roster vector. Because Players are created based on their name as they are found either in the Play-by-Play API call or a user's bet and there is a chance that these players' names are repeated in those contexts, there is a risk that a Player could be added to a roster twice. To avoid this, the Game class uses the findPlayer method to make sure a Player is not in the home or away roster vector before it adds the Player with the addPlayer method. The Game reset method is there because of the way Games are updated in the update_game function. Since Games are updated sequentially and any given part of the Play-by-Play API response is liable to change each time it's called (due to plays being called back, booth reviews, etc.), games are "updated" by resetting and repopulating all of Game information. The Game reset method is a simple call that clears all the variables except for the game ID so that this kind of updating can be performed. Finally, the retrieve_stat method is in place so that the rest of the code can use one function to get any stat it needs from a given Game. It takes in strings of the desired stat, relevant team name, and relevant player (if applicable). Given those three parameters, it locates and returns the desired stat as a string in format that is useful in the context of the other code.

The information from the API is received in XML format. To reduce memory usage, in the find_game and update_game functions (the two functions which make SportRadar API calls) the XML response is read one line at a time. An example of what one of these lines of XML looks like can be seen here.

```
    <home name="Phillies" market="Philadelphia" abbr="PHI" id="2142e1ba-3b40-445c-b8bb-f1f8b1054220" runs="3" hits="7"
errors="1"/>
```

To make the extraction of relevant information from these lines of XML easier, the parseString and parseDate functions were implemented.

The parseString function takes in a line of XML and a target string. It searches the line of XML for the target string, and once found extracts and returns whatever is inside the next two quotation marks after the target string. For example, to find the abbreviation PHI in the XML string shown above, one would call parseString(line, “abbr=”). Because these XML responses are standardized in how they arrange and send information, the parseString function allows the code to extract any piece of information the SportRadar API provides.

The parseDate function is similar to the parseString function with a few key changes. All of the dates/times listed in the XML responses are in the form YYYY-MM-DDTHH:MM:SS+00:00, where the time is expressed as UTC time. This time/date structure appears after the string “scheduled=”, and in the context of the rest of the code, only the date is needed. Because of this, instead of searching for an ending quotation mark after “scheduled=”, it just takes the next ten characters after the first quotation mark and calls that the date. If the date and time were given in Pacific Coast Time, the date extracted in that process would work for every game. However, since the date and time are based on UTC time, many of the later evening games appear to occur on the next day. To deal with this, extra if statements were added to check if the listed hour was 7 or less; if it was less, the day of the date was reduced by one to match the actual date in American time zones. This way, accurate dates could be pulled from API calls and compared to the dates used in the rest of the code.

Once a new bet is placed and retrieved from gamblingscreen.com, the code holds the abbreviation of the team associated with the bet. For example, if a user placed a bet on Shohei Ohtani to have more than 1.5 hits, the code would hold onto the String “LAD”. This team abbreviation string is passed to the find_game function which takes in the SportRadar API key, the current date, and the team abbreviation.

The find_game function makes an API call to SportRadar’s MLB Regular Season Schedule endpoint. This call returns Strings of XML that contain information about every MLB regular season game. Using the parseString and parseDate functions, each line of the API call’s response is filtered until the correct game is found. It filters first by date, ignoring any game that has a scheduled date other than the one brought into the find_game function. It is important to note here that SportRadar’s API refers to any time-related event in UTC time rather than any American time zone. This means that the listed dates of some games are (to an American) incorrect, and must be adjusted based on the

UTC hour listed in the API call. This adjustment is made in the parseDate function. Once a game with the correct date is found, the code reassigns the string game_id to be the 20 character game ID within SportRadar's API system and sets the bool foundDateMatch to true. This game ID is eventually what will be returned by the find_game function; however, more parsing must be done before the code is sure it has the correct game ID. It only grabs the game ID at this stage because it appears in the same line of the XML response as the scheduled date of the game. At this point, the code has found a game ID associated with the correct date and it needs to check if the game involves the correct teams. In a separate if statement, as the code continues to read the API response line by line, if it finds a line that contains a home or away team abbreviation and the bool foundDateMatch is true, it checks to see if the team abbreviations from the API response match the one passed to the input of the find_game function. If it does match, the while loop that runs through the entire API response breaks, and the most recent game ID (which is the game ID associated with the correct date and the correct team) is returned. If the team names are not found on that game, the bool foundDateMatch is reset as soon as a game with the wrong date is found (which is usually the next game in the API response). Once the game ID is found, a new Game pointer is created with it. Here is an example of what a game in the regular season schedule looks like in the XML response.

```
      <game id="0015384e-cd17-46f6-88de-5f8b503e77c9" status="scheduled" coverage="full" game_number="1" day_night="D"
scheduled="2024-06-27T19:45:00+00:00" home_team="a7723160-10b7-4277-a309-d8dd95a8ae65" away_team="55714da8-fcaf-4574-8443-
59bfb511a524" double_header="false" entry_mode="STOMP" reference="745324">
        <venue name="Oracle Park" market="San Francisco" capacity="41915" surface="grass" address="24 Willie Mays Plaza"
city="San Francisco" state="CA" zip="94107" country="USA" id="2d7542f5-7b80-49f7-9b24-c53ffdc75af6" field_orientation="E"
stadium_type="outdoor" time_zone="US/Pacific">
          <location lat="37.7784199" lng="-122.3906212"/>
        </venue>
        <home name="Giants" market="San Francisco" abbr="SF" id="a7723160-10b7-4277-a309-d8dd95a8ae65"/>
        <away name="Cubs" market="Chicago" abbr="CHC" id="55714da8-fcaf-4574-8443-59bfb511a524"/>
        <broadcast/>
      </game>
```

Once a Game pointer exists, it is populated with information by the update_game function. The update_game function starts by making an API call to that specific game's Play-by-Play. The Play-by-Play endpoint follows each action that takes place during an MLB game. This includes facts about the weather, lineup changes, at-bats, pitches, outs, errors, and much more. After resetting the Game object, the XML response is parsed line by line to extract all the stats that can be stored in the Game. The parsing algorithm uses a switch-case statement that divides information based on how it appears in the XML response. For instance, once it finds the overall "game level" it knows that the only thing it can expect next is an overall score or an inning. Then, once it knows it's supposed to be reading an overall score or inning, it can expect information about a home or away score, inning type, etc. The update_game function has a few tracking variables to be able to correctly update the Game object based on the state of the game. For instance, it flips the bool inningState any time it changes from the top to the bottom of an inning and vice versa. Various strings hold pitcher, hitter, and runner names as well. At the end of the function, there is a check to make sure that the full XML response has been parsed through as the WiFi connection sometimes falters and part of the response is lost. If the parsing algorithm does not make it to the end of the response, it tries again until it gets it right.

3.3.3.4. API Design Process and Testing

SportRadar's API was chosen for this project because we were able to use free trials of it while we developed the full system. It was also chosen because it has extremely up-to-date information about live sports. In fact, the API gets Play-by-Play updates faster than a live broadcast of the game.

Once it was chosen, we spent a few weeks seeing how viable it was to make API calls on the ESP32. After testing how both the XML-style and Json-style responses were handled by the ESP32, we decided to go with the XML format for our API calls. This was because the Json objects that we would have had to implement on the ESP32 would have been extremely large and cumbersome with respect to the rest of the code. Additionally, even though the XML format was somewhat clunky and contained lots of unnecessary information, at that early stage we were able to imagine ways of parsing through the response to get the data we needed. Looking back, Json might have been a viable option, but we chose XML because we knew we would be able to implement it.

Once we decided on the XML format for the API calls, we started figuring out how to parse out and store the information we wanted from the calls. We came up with the Game and Player classes to store the information. We chose to use classes with private members instead of structs because we weren't sure exactly how we would need to use them later on in the project and we thought it would be unwise to allow direct changes to certain members (like the Game ID). Parsing all of the information we needed was a slow process, but the more stats we learned how to extract, the easier it became to extract others. SportRadar's documentation for their API was extremely helpful if figuring out how to find very specific information in API responses that looked like this:

```
      <at_bat hitter_id="f2a70df3-0bff-48e3-92ae-970c2a103fcf" id="9627f4b4-8bc9-4e76-94bb-a59cf391325d" hitter_hand="R"
pitcher_id="581fc086-8ee2-4b14-b3fb-edf358111186" pitcher_hand="R" sequence_number="28">
        <hitter preferred_name="J.T." first_name="Jacob" last_name="Realmuto" jersey_number="10" id="f2a70df3-0bff-48e3-
92ae-970c2a103fcf" full_name="J.T. Realmuto"/>
        <pitcher preferred_name="Jacob" first_name="Jacob" last_name="Waguespack" jersey_number="67" id="581fc086-8ee2-
4b14-b3fb-edf358111186" full_name="Jacob Waguespack"/>
        <description>J.T. Realmuto homers to left field. Kyle Schwarber scores.</description>
        <pitch status="official" id="6a2c4522-1f84-4e58-91b3-f734a5c693ce" outcome_id="kKS" created_at="2024-03-
25T16:16:18+00:00" updated_at="2024-03-25T16:17:20+00:00" sequence_number="29" official="true">
          <wall_clock start_time="2024-03-25T16:16:13+00:00" end_time="2024-03-25T16:16:30+00:00"/>
          <flags is_ab_over="false" is_bunt="false" is_hit="false" is_wild_pitch="false" is_passed_ball="false"
is_double_play="false" is_triple_play="false"/>
          <count balls="0" strikes="1" outs="2" pitch_count="1"/>
          <pitcher pitch_type="FA" pitch_speed="94.8" pitch_zone="3" pitcher_hand="R" hitter_hand="R" pitch_count="8"
id="581fc086-8ee2-4b14-b3fb-edf358111186" pitch_x="47" pitch_y="82" preferred_name="Jacob" first_name="Jacob"
last_name="Waguespack" jersey_number="67" full_name="Jacob Waguespack"/>
          <hitter preferred_name="J.T." first_name="Jacob" last_name="Realmuto" jersey_number="10" id="f2a70df3-0bff-48e3-
92ae-970c2a103fcf" full_name="J.T. Realmuto"/>
          <mlb_pitch_data speed="94.8" strike_zone_top="3.255" strike_zone_bottom="1.643" zone="1" code="FF"
description="Four-Seam Fastball">
```

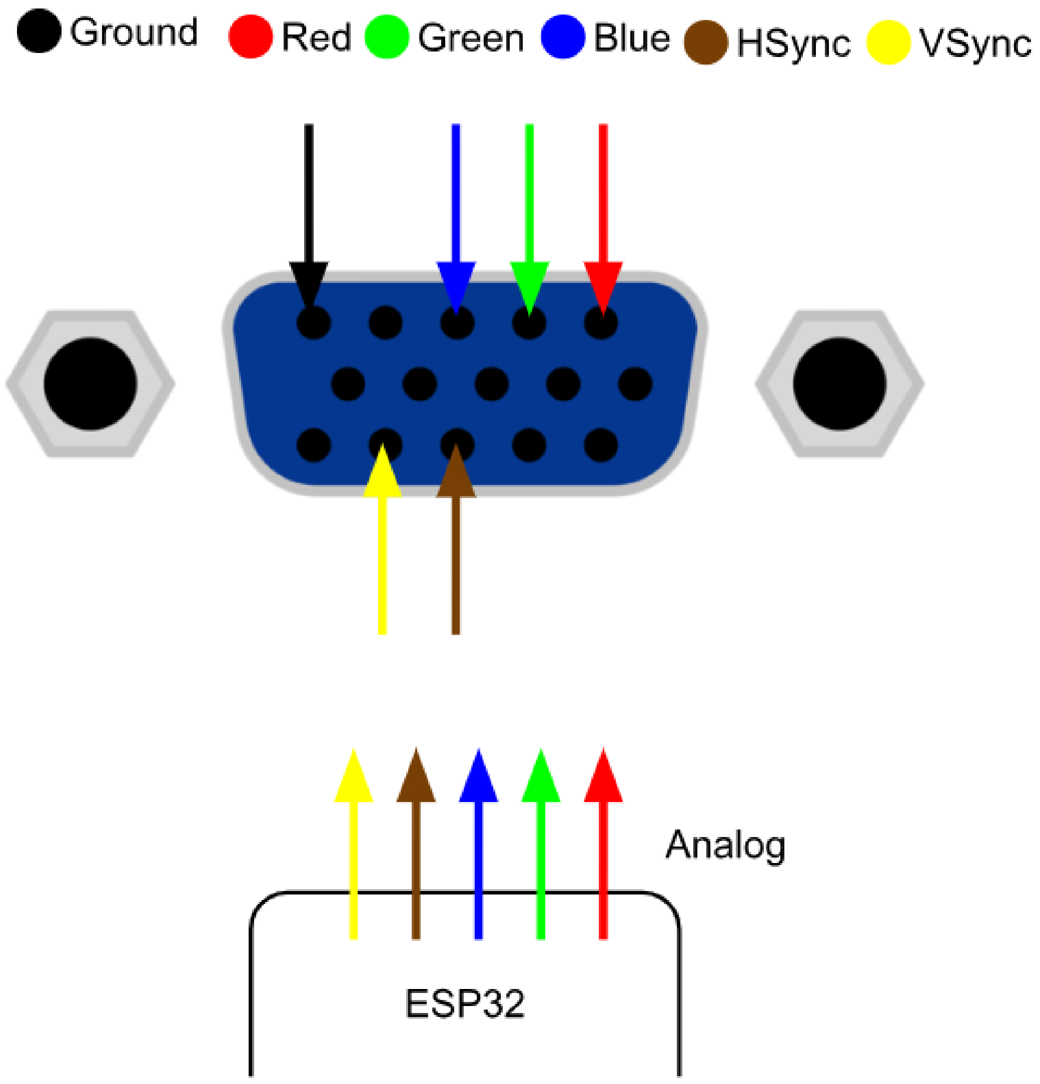
We developed various functions to make parsing things easier and incorporated more stats into the parsing algorithm as the demo deadline approached. Additionally, we streamlined the code to use fewer API calls per game, which allowed us to get the most value out of our trial keys, as we only had 1000 calls per key.

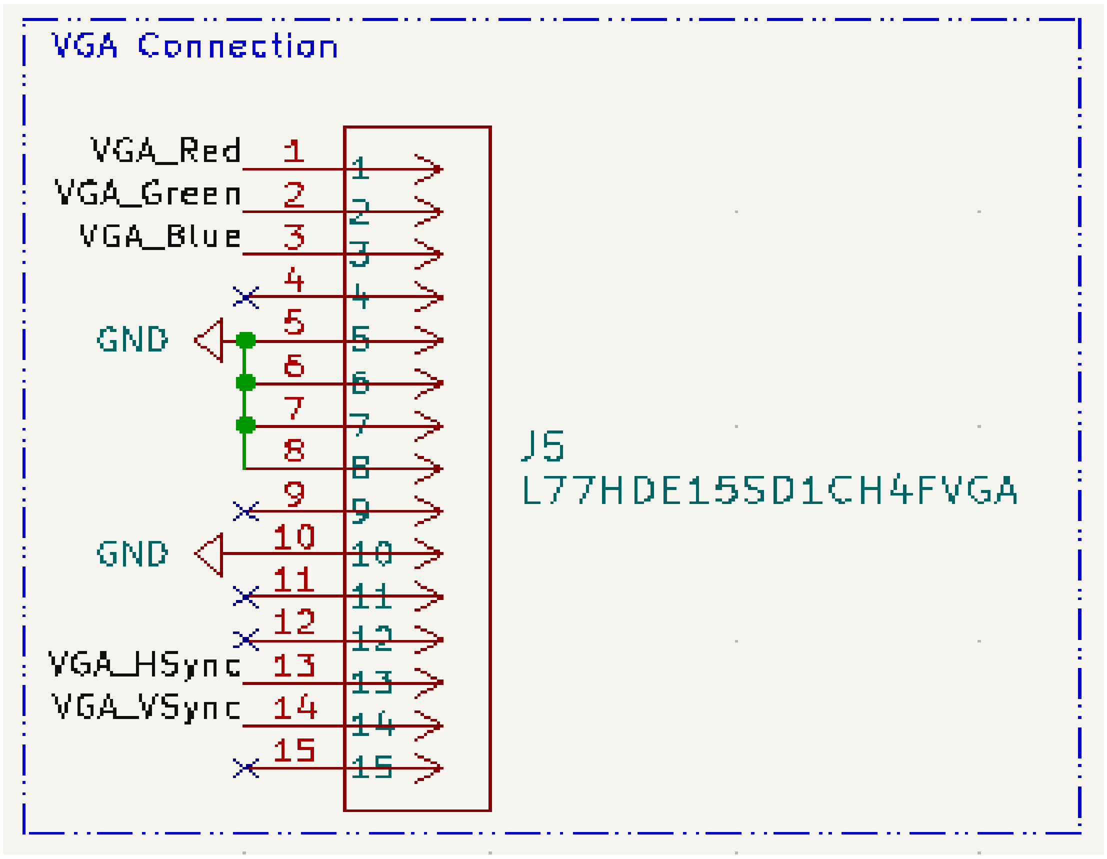
3.3.4. Display Subsystem

3.3.4.1. Display Subsystem Requirements

- Must have a clean graphic that is easy to read for users
- Must continuously loop to show multiple bet slips and update to show live bets

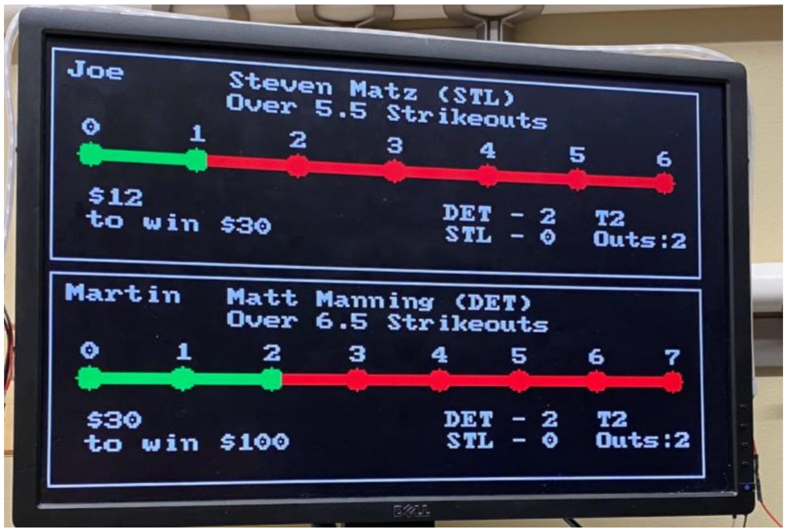
3.3.4.2. Display Schematic

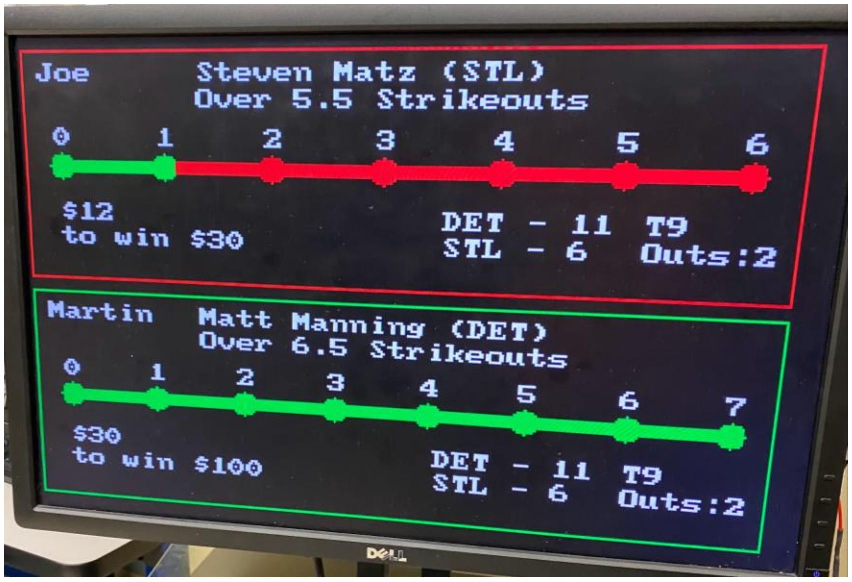




3.3.4.3. Display Function, Testing, and Design

The display subsystem is able to update the progress of your bets in real time, it uses a 320x200 resolution with an arcade-inspired design to display the most important information of your bet slips whether a game is on or not (however, user experience is better with a simultaneous game accompanying it), and is able to display information from multiple concurrent games.




Joe Steven Matz (STL)
Over 5.5 Strikeouts
0 1 2 3 4 5 6
$12
to win $30
DET - 11 T9
STL - 6 Outs:2

Martin Matt Manning (DET)
Over 6.5 Strikeouts
0 1 2 3 4 5 6 7
$30
to win $100
DET - 11 T9
STL - 6 Outs:2

The pivotal function that governs the display is called BetBox which is coded using the self-titled library Bitluni that can program VGA. To understand what the function does, we should look at the example images., we can see it displays the name of the bettor (Joe), the team or player that is subject of the bet (Steven Matz), the bet details (Over 5.5 strikeouts), a progress bar with the current status of the bet on green/red depending on over/under, the betted monet and the potential win ($12 to win $30), and overall game information like teams, score, inning, and current outs. Additionally, to enhance user experience, the outline of the box can change color depending on the outcome of the bet, it will turn green if positive, and red if negative.

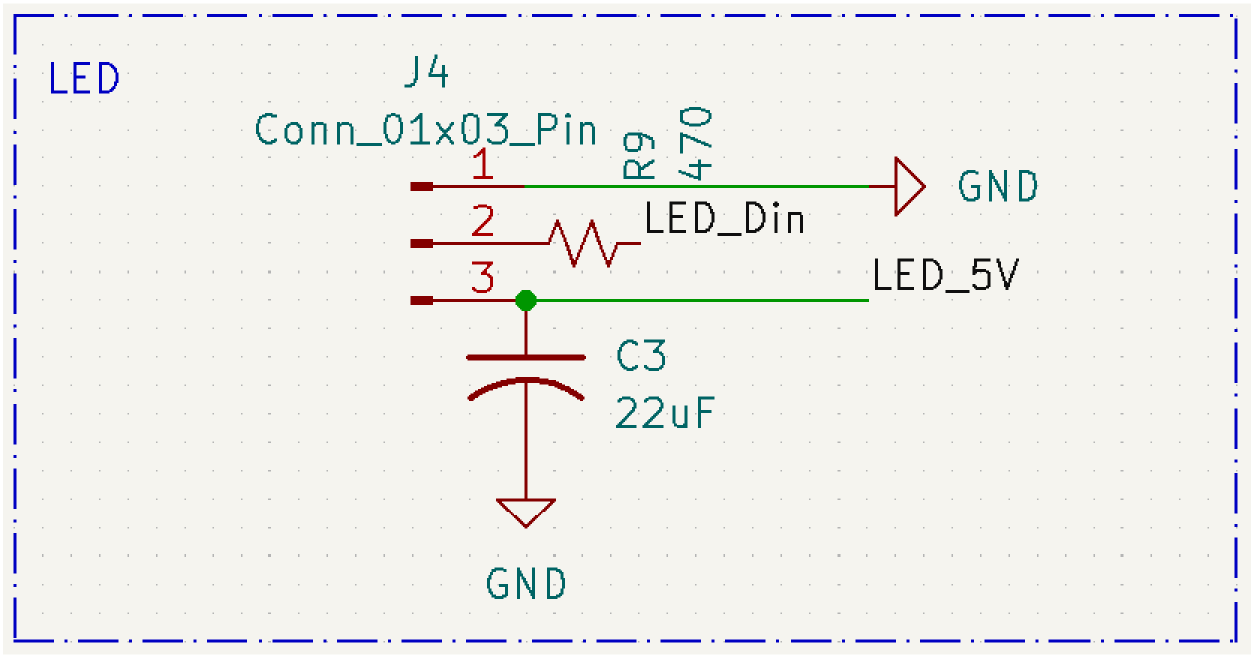
The design of the display was largely based on the need to show the user the most updated information, whether it is information from the game they are watching that they missed, or from games that are happening at the same time. To correctly carry out this function, we searched for programming libraries that were compatible with an ESP32 microcontroller. To demonstrate the viability of the library we connected a monitor using a VGA connection (this is a I2S-based protocol that sends an analog signal to the monitor) and were able to display basic objects on the screen, proving its feasibility. We tested the limits of the resolution and variety of colors, and were able to use a maximum of 320x200 pixel resolution and 3-bit color resolution after doing our integration tests, by itself the screen could reach 800x600 pixel resolution; however, due to discovering memory limitations, it uses the current resolution and is still able to present a great user experience.

3.3.5. LEDs Subsystem

3.3.5.1. LED Requirements

- Display results of the betting information with visual stimulus using various colors

3.3.5.2. LED Schematic



3.3.5.3. LEDs Function

The LEDs intended function is to give users additional visual feedback when a bet hits. For example, if the bet hits the LEDs will all flash green and if the bet loses the LEDs will all flash red. This also is proportional to how the bets hit by if the entire bet hits or loses, and not just increments, it will flash for a second and then turn off. For this to work as intended we needed to download the Adafruit Neopixel library and manipulate the functions to work as intended. The LEDs needed to work independently, so functions for both win and loss were needed. This was done by initializing the Neopixel first in the setup otherwise the LEDs would not function properly. To show the lights these functions would be called within a bet win function in conjunction with the other subsystems when they were needed to run.

For the LEDs they are the WS2812B Adafruit Neopixel 1 m long light strip. This is because it is a LED strip that is made for home projects, so we believe that it is an LED strip that would be beneficial to use because there is a lot of documentation about it on the Adafruit Website.

3.3.5.4. LEDs Design

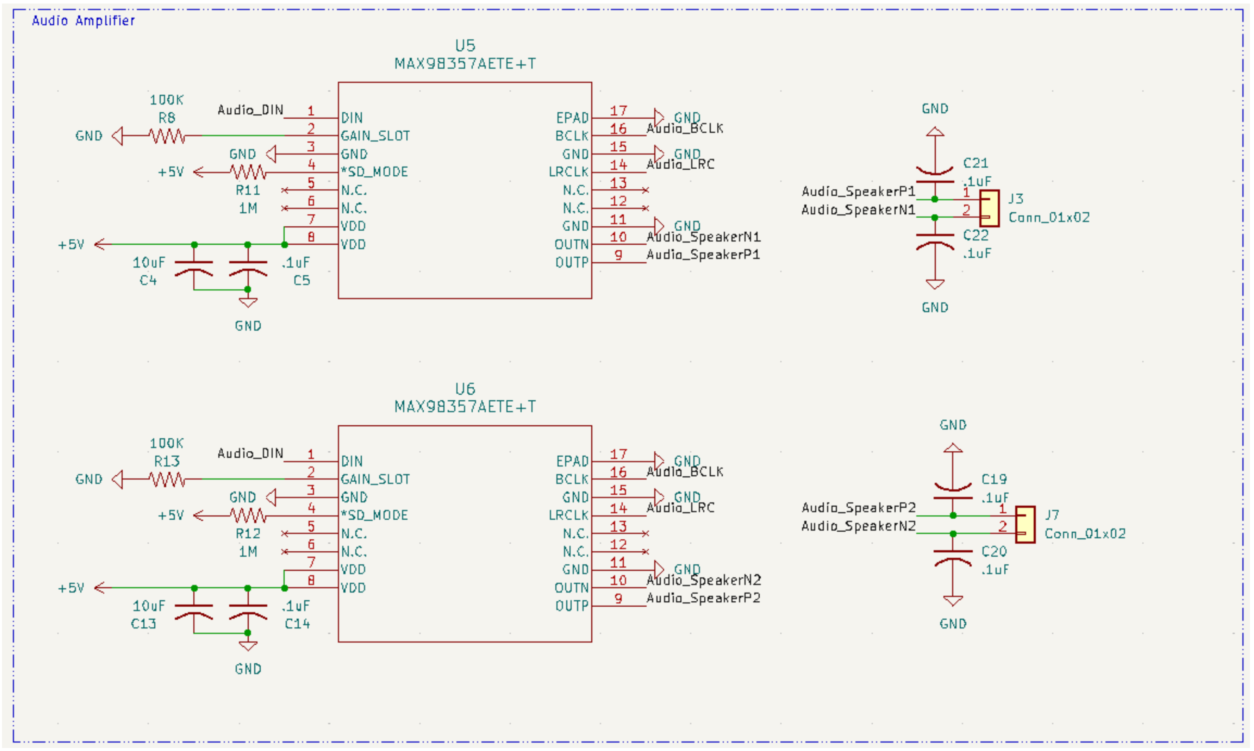
The LED Design is mainly based upon what was found on the Adafruit Neopixel Uberguide online where it talks about how to connect the LEDs correctly for proper use. The LEDs take in 5 V and need 1 GPIO pin from the ESP32 to work and by manipulating the wires that came with the LED strip we were able to make each input: 5V, GND, and DataIn correspond to header pins that were added to the board. Since the Neopixel only needed 1 GPIO pin to connect all the colors it was able to display all the colors we needed without problem when connected. We set the brightness to be lower upon small wins and higher upon big wins. This was done through dedicated library functions.

3.3.6. Speaker Subsystem

3.3.6.1. Speaker Requirements

- Provide audio feedback dependent on the current/ending status of the bets

3.3.6.2. Speaker Schematic



3.3.6.3. Speaker Function and Design

The speaker's intended function is to give users audio feedback based on updates to their bet. For example, if a bet is lost, a losing sound effect plays. On the other end, if a bet is won then a positive sound effect plays. This also includes playing proportional sounds to smaller updates such as a team scoring or a player getting a hit. For this to work as intended, each of the desired sounds needs to be converted to proper header files. To achieve this, we first downloaded the desired sound effects into a mp3 format. Then, the mp3 file was imported into Audacity where it was converted to and exported as a 16-bit wav file at a sampling frequency of 8kHz. Using HxD, a program for hex editing, the new wav file was converted to a hex value where it was then copied as C code. Last, the code was transferred into the proper VisualStudioCode project as a header file. These header files are then called as appropriate.

To play the audio, we decided to use an I2S 98357MAX amp as the basis. It was chosen because it provided the proper voltage requirements and the necessary I2S channel. For the speakers, a more compact 4", 3W, 4 ohm design was purchased and implemented to the system. Volume is set by the manufacturer and cannot be changed.

3.3.6.4. Speaker Design Process and Testing

The audio subsystem initially consisted of a breakout board wired to an I2S 98257 amp. A simple software play function successfully played all audio files. After verifying all the audio files using the simple play function, we then assigned each sound to a different outcome and placed bets that would correspond to those outcomes happening. Conditions for the bets to win were then hard coded in to elicit the appropriate sound. Each sound played at the correct time and tested satisfactorily.

4. System Integration Testing

We began our integrated subsystem testing with the display and tried to integrate our web-sourced data into a user-friendly display. To do this, we constructed the “betSlips” class. The class held all of the variables from a singular bet slip and was constructed by parsing data from ‘getData.php’ which was accessed by connecting our ESP32 to a WiFi client at ‘gamblingscreen.com/getdata.php’. Subsequently, we developed the ‘betBox()’ function to manage the display of information fetched from this PHP script. Initial tests confirmed a successful connection to a Wifi client and the ability to display the needed numbers from an unique bet slip.

Following this, we refined ‘betBox()’ to display only the relevant variables from each bet slip, rearranging the display format to suit our intended interface. Also, we moved the call for ‘betBox()’ from the ‘setup()’ function to the ‘loop()’ function. In the ‘loop’ function we implemented a for-loop to iterate through all bet slips in the class, displaying only two bets at a time. This decision was driven by our aim to optimize screen space and ensure all bets were displayed in some sort of cycle.

To further improve the user interface, we created the ‘displayProgress()’ function which visually represents statistical progress through a color-changing bar. As the user’s desired stat increases, the bar shifts from red to green. Once fully satisfied with the UI design, we began to integrate the SportRadar API.

To introduce the SportRadar API, we decided to make the call in the function ‘betBox()’. We decided to flow the code in this way so that the API call occurs while the code is in the process of displaying the information from a particular bet. This allowed us to implement the variables from the ‘betSlips” class into the API retrieval, which made it easier to retrieve the data that we wanted. We began the implementation by initializing the ‘Game’ class to fetch live game data, starting with testing the retrieval of game scores via the ‘retrieveStat()’ function. This function consolidates scores, innings, and outs into a single string. Once the string is retrieved from the API, a parsing process extracts the specific game details that are needed for the display. We chose to have one universal function handle data retrieval because it was more efficient than having to create multiple functions for each stat type and bet type. We tested this extensively until the game scores, game inning, and number of outs were being displayed to the screen as desired.

As the display began reflecting live stat updates accurately, we began to implement the LEDs and audio subsystems for sound and visual feedback. We implemented conditional statements within ‘betBox()’ to trigger one of four different functions: ‘betWin()’, ‘betLoss()’, ‘positiveChange()’, or ‘negativeChange()’. The function that was called was dependent on what numbers were retrieved from the API. Each of these functions are associated with respective LED and sound signals - flashing green lights and positive sounds for winning or advantageous changes, and red lights with negative sounds for losses or detrimental events. We began testing this implementation by ensuring that the proper function was being called for the proper conditions by using serial monitor prints.

Once we confirmed the conditions were correct, the codes from the speaker subsystem and LED subsystem were placed into the functions. The functions were encapsulated within 'results.h' and 'results.cpp' to make the handling of feedback responses easy to update. Extensive testing at this stage, confirmed that the display, lights, and audio updated appropriately in real-time.

5. User's Manual

5.1. Product Installation

Step 1: Unboxing and Safety Check

1) Carefully unbox all components, and check for damage in any of the components.
    a) If any damage is noted, contact customer support for a further assistance

Step 2: Connecting the Screen

1) Place the auxiliary screen near your television
2) Connect the VGA cable from the auxiliary screen to the corresponding PCB board for the betting console

Step 3: Powering the Device

1) Connect the barrel plug of the power adapter ot the power input port on the PCB
2) Plug the other end of the power adapter into a 120V power outlet.
3) Plug in your monitor into a different 120V power outlet.

Step 4: Using Auxiliary Screen

1) Once installation is complete, the screen may now be used for the tracking of bets

5.2. Product Setup

Step 1: Check Installation

1) Ensure all connections are secured and properly installed in accordance with Section 5.1

Step 2: Placing Bets

1) Place bets on the accredited sportsbook of your choosing, complying with all state and federal laws
2) Log into GamblingScreen.com

3) Go to Add Bets on the top right corner of the website
4) Enter screen name of your choice
5) Select your desired sport
6) Select your desired category (e.g. Game outcome)
7) Select your type of bet (e.g. moneyline, spread, player prop, etc.)
8) Select player or team as appropriate
9) Enter the amount wagered and odds from the sportsbook the bet was placed
10) Add this to the bet slip
    Steps 5) through 9) may be repeated as desired to match parlays placed
    When satisfied, click on upload to screen. You should see the bets populate the auxiliary screen

Step 3: Gamble Responsibly

1) If you or someone you know has an issue with gambling, call 1-800-Gambler

5.3. How to Tell if Product is Working

The user can tell if the product is working if every part of the product is working in conjunction. The User will see the bets they placed on the screen, if a bet hit they will see proper lights and sound cues to notify them. If the bet misses or loses the user will also be notified by corresponding lights and sound cues with these bets displayed on the screen.

5.4. Troubleshooting

If the product is not working as expected use the following steps to troubleshoot:

Step 1: Check Display Output

1) Turn on your television and betting screen
2) If the screen lights up and displays the bets this indicates that it is running properly

Step 2: Verify the Connections

1) Ensure that the VGA is securely connected to at both ends: the auxiliary screen and the PCB
2) Double check to make sure that the barrel plug is connected to the PCB and the monitor and barrel plug are connected to 120V outlets

Step 3: Check Display Settings

1) On the monitor check that the display outlet is VGA and not HDMI, this will cause the monitor to not display properly

Step 4: Test with Betting Software

1) Open the betting application and check that the bets you placed are displaying on the auxiliary screen
2) Next check if the bets you placed are tracking correctly and that they are working in real time

Step 5: Check Display Resolution

1) Check if it is 320x200 if not change to that or else screen will not place properly

Step 6: Last Resort

1) Contact 1-(331)-684-7968 if none of the previous steps fix the issue.

6. To-Market Design Changes

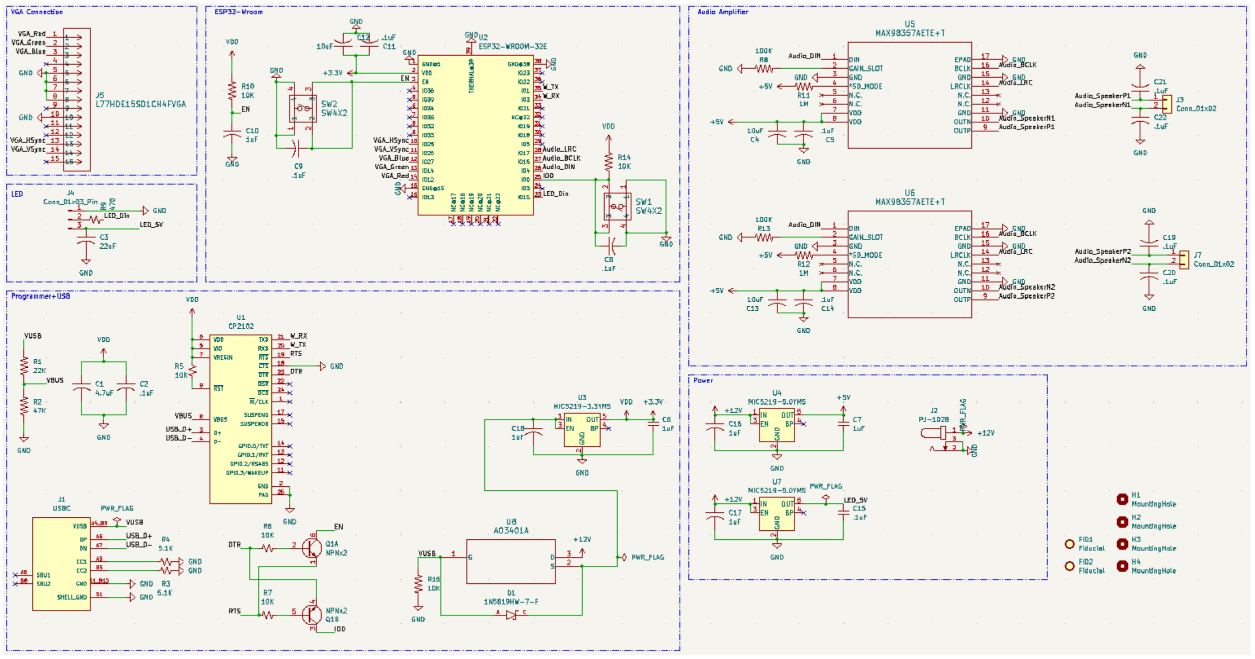
The first update to the project would be a stronger processor. The gambling screen was severely limited by the small amount of flash memory. More flash memory would allow us to display more bets at a time. It would also allow for a better UI and higher resolution. Another potential solution for this is a dedicated processor for the display. These notions would require more money and more testing. As for speakers, volume control from the user's phone would be more pleasant than that of a consistent sound. A dedicated app for placing these bets or integration with sportsbooks would also be much more convenient than logging onto a website and would greatly enhance the user's experience. Another issue with sound was that the power system used for the speakers was not effective for more than one speaker. Therefore, the power system would need to be improved to handle multiple speakers depending on how one might have their auxiliary monitor setup. Next there should be a dedicated LED PWM driver like the WS2812 that would offload the CPU from needing to manage every change in LED state. Last, before this would be sent to the market a company would need to buy a full subscription to SportRadar, so users would not run out of API keys when they use the product. An alternative to that would be Team Gamblers working out some sort of deal with SportRadar regarding API keys.

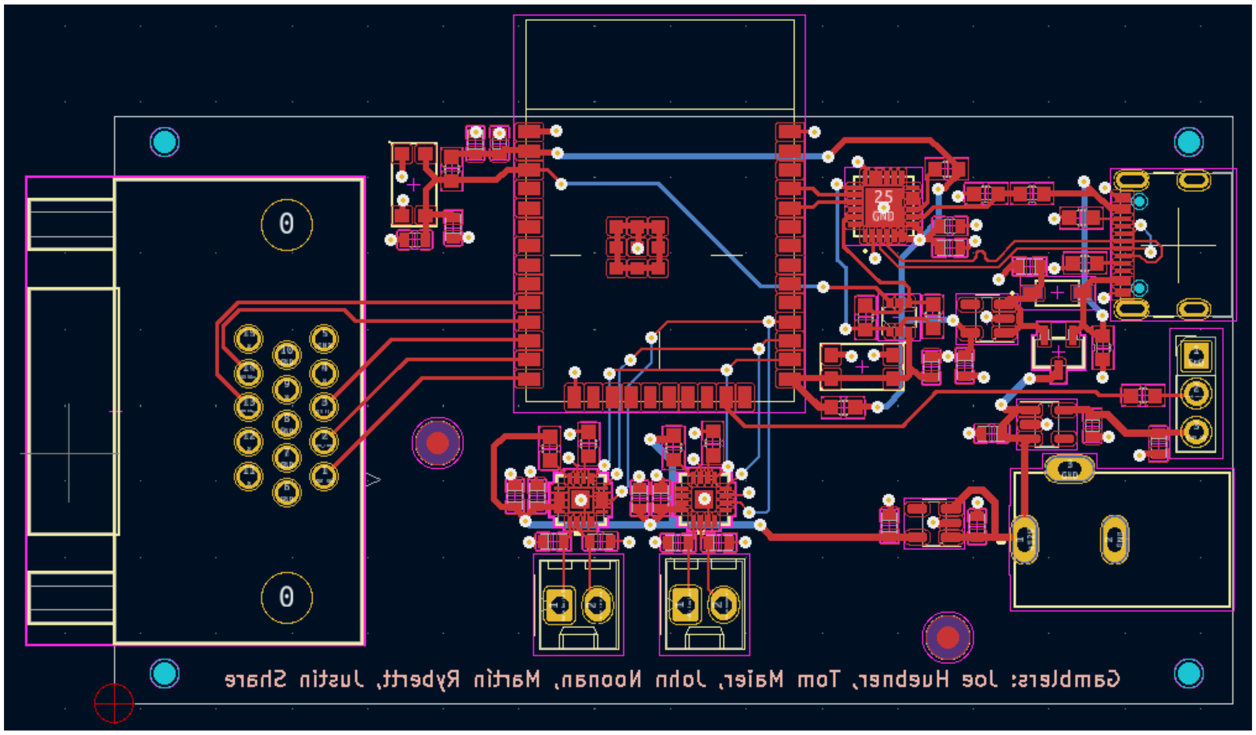
7. Conclusion

In summary, team Gamblers was not content with the way that sports matches were being viewed. Your average viewer spent too much time looking down at their phone trying to track statistics thereby inhibiting viewers from enjoying the game itself. This went on to inspire the idea of an auxiliary screen that may be personalized to track the statistics the user wants to see regardless of which game they are physically watching. The team identified the major issues that would need to be addressed and split up from there. The first and most obvious challenge was a display itself. We needed to decide on how large it would be and what the UI would look like. It had to be something that could be referenced quickly but would not detract from watching the game on the main screen. This led to the creation of the screen detailed in section 3.3.4. Next up was figuring out how to power all of this. We quickly determined that power would be a variable to solve for later as we didn't even know how much we needed yet. Picking at the idea more made us realize that no one would want a boring old screen that just displays information, it needed to have some pizzazz to it. Ideas for LED lights and installed speakers were tossed around. The team ultimately decided that these would work best in tandem with one another. Lights were added to add visual feedback based on the status of the statistics while speakers did the same thing but for audio feedback. Lighting is detailed more thoroughly in section 3.3.5 while the audio portion can be found in section 3.3.6. This settled most of the hardware but we still needed to figure out how we were going to make this idea happen. We needed a website. This would provide the medium necessary for users to update their betting information. Data from the website would then be uploaded and displayed on the auxiliary screen via a wifi enabled ESP32 microcontroller and a VGA cable. GoDaddy web services were used to create this website. A self explanatory format was developed and subsequently launched. Reference section 3.3.2 for more details. Reflecting on everything, we had: a display for showing the necessary information, a way to upload the information you wanted displayed, lights, and sounds. Power was still a wildcard because it relied on everything else being finished. The one thing we were missing was how we were going to actually get the display to update accurately. A company called SportRadar was contacted to assist with this. They are who ESPN and all the major sportsbooks get their stats from. SportRadar tracks every metric that our project was looking to track. Working out a deal with them provided access to their API. This API came in a giant text file every 10 seconds. We were able to decode this into a more usable format that allowed our screen to update as necessary every 10 seconds. See section 3.3.3 for more details regarding the API. Now we were able to finish with the power subsystem. We determined that we needed 120V wall power for the display and 5V for the ESP32. Power adapters were acquired and voltage regulators installed into the system. Power is listed in section 3.3.1. Bringing all of these systems together involved designing a low-profile way for potential customers to use the product. Schematics from all of our components were compiled and fitted to a printed circuit board design. This board was installed into a 3D printed housing and allows for seamless connection between the project, the main screen, and the auxiliary display.

8. Appendix

8.1. Hardware Schematics





Gamblers: Joe Huebner, Tom Maier, John Noonan, Martin Rybett, Justin Share

8.2. Components and Datasheets

ESP32 Microcontroller	Crimp
Barrel Plug Connection for Power	VGA connector
5V Voltage Regulator	I2S Amp
3.3V Voltage Regulator	40V 1A Diode
Molex Connector	30V 4A Mosfet
Molex Header Connection	4Ohm 3W Speakers
12V 1A Power Adapter	LED Light Strip

8.3. Website Software Files

8.3.1. template-bet-slip.php

```
<?php
/*
Template Name: Template Bet Slip Page
*/

get_header('custom'); ?>

<div class="form-container">
   <label for="screenName" class="form-label" >Enter Screen Name:</label>
   <input type="text" id="screenName" name="screenName" class="form-select">
   <button id="loginButton" class='login-button' type="button">Login</button>
</div>

<div id="bet-form" class="form-container" style="display:none;">
   <div class="form-header">
   </div>
   <label for="sportSelect" class="form-label" >Select Sport:</label>
   <select id="sportSelect" class="form-select" name="sportSelect">
      <option value="">Select a Sport</option>
      <option value="Baseball">Baseball</option>
   </select>
   <div id="categoryForm" style="display:none;">
      <label for="categoryForm" ">Select Category:</label>
   </div>
   <div id="typeForm" style="display:none;">
      <label for="typeForm" >Select Type:</label>
```

```
    </div>
    <div id="detailsForm" style="display:none;">
        <label for="detailsForm" >Enter Details:</label>
    </div>
    <div id="bettingFormContainer"></div>
    </div>
    <div class="clear"></div>
    <div id="betOutput"></div>

    <div id="parlayOutput" style="display:none;"></div>
</div>

<button id="uploadButton" class="upload-button">Upload to Screen</button>

<?php get_footer('custom'); ?>
```

8.3.2. template-view-bet.php

```
<?php
/**
 * Template Name: Template View Bet
 */

get_header('custom'); ?>

<div id="primary" class="content-area">
   <main id="main" class="site-main" role="main">

       <?php echo do_shortcode('[wpdataaccess pub_id="2"]'); ?>

   </main><!-- #main -->
</div><!-- #primary -->


<?php get_footer('custom'); ?>
```

8.3.3. bet-slip-manager.js

```
jQuery(document).ready(function($) {
  class BetSlipManager {
    constructor(config) {
      this.config = config;
      this.betSlip = [];
      this.initEventListeners();
    }

    initEventListeners() {
      $('#loginButton').on('click', this.handleLogin.bind(this));
      $('#sportSelect').on('change', this.handleSportChange.bind(this));
      $('body').on('change', '#categorySelect', this.handleCategoryChange.bind(this));
      $('body').on('change', '#typeSelect', this.handleTypeChange.bind(this));
      $('#addParlayButton').on('click', this.displayParlay.bind(this));
      $('#uploadButton').on('click', this.uploadBets.bind(this));
    }

    hideForm(formId) {
      $('#' + formId).hide().empty();
    }

    handleLogin() {
      const screenName = $('#screenName').val().trim();
      if (screenName) {
        this.screenName = screenName;
        $('#loginForm').hide();
        $('#bet-form').show();
    } else {
        alert('Please enter a screen name.');
    }
}

handleSportChange(event) {
    const sport = $(event.target).val();
    this.hideForm('categoryForm');
    this.hideForm('typeForm');
    this.hideForm('detailsForm');
    if (sport) {
        this.updateCategoryForm(sport);
    }
}

handleCategoryChange(event) {
    const category = $(event.target).val();
    const sport = $('#sportSelect').val();
    this.hideForm('typeForm');
    this.hideForm('detailsForm');
    if (category) {
        this.updateTypeForm(sport, category);

        if (sport === "Baseball" && category === "Game Outcomes") {
            this.fetchAndSetupTeamNames();
        }
        if (sport === "Baseball" && category === "Player Lines") {
            this.fetchAndSetupPlayerNames();
        }
    }
}



handleTypeChange(event) {
    const type = $(event.target).val();
    const sport = $('#sportSelect').val();
    const category = $('#categorySelect').val();
    this.hideForm('detailsForm');
    if (type) {
        this.showDetailsForm(sport, category, type);
    }
}

updateCategoryForm(sport) {
    const categoryLabel = $('<label/>', {
         'for': 'categorySelect',
        text: 'Select Category:',
        class: 'form-label'
    });

    const categorySelect = $('<select></select>', { id: 'categorySelect' }).addClass('form-
select');

    categorySelect.append(this.createOption('Select a Category', ''));
    $.each(this.config[sport].categories, (category) => {
```

```
        categorySelect.append(this.createOption(category, category));
    });
   $('#categoryForm').empty().append(categoryLabel).append(categorySelect).show();

}

 fetchAndSetupTeamNames() {
$.ajax({
    type: 'POST',
    url: betSlipManagerParams.ajaxurl,
    data: {
        'action': 'get_sd46_teamnames'
    },
    success: (response) => {
        if (response.success) {
            this.setupAutocompleteForWinningTeam(response.data);
        } else {
            console.error('Failed to load team names.');
        }
    },
    error: (jqXHR, textStatus, errorThrown) => {
        console.error('Error fetching team names:', textStatus, errorThrown);
    }
});
}
```

```
setupAutocompleteForWinningTeam(teamnames) {
    const winningTeamInput = $('#winningTeam');
    const teamNamesArray = teamnames.map(team => team.teamname);
    winningTeamInput.autocomplete({
        source: teamNamesArray,
        minLength: 0
    }).focus(function () {
        $(this).autocomplete("search", "");
    });
}



    updateTypeForm(sport, category) {
        const typeLabel = $('<label/>', {
            'for': 'typeSelect',
            text: 'Select Type:',
            class: 'form-label' // Add any additional classes for styling
        });

        const typeSelect = $('<select></select>', { id: 'typeSelect' }).addClass('form-select');
        typeSelect.append(this.createOption('Select a Type', ''));
        $.each(this.config[sport].categories[category], (type) => {
            typeSelect.append(this.createOption(type, type));
        });

        $('#typeForm').empty().append(typeLabel).append(typeSelect).show();
        }


    fetchAndSetupPlayerNames() {
    $.ajax({
        type: 'POST',
        url: betSlipManagerParams.ajaxurl,
        data: {
            'action': 'get_sd46_playernames'
        },
        success: (response) => {
            if (response.success) {
                this.setupAutocompleteForPlayerName(response.data);
            } else {
                console.error('Failed to load team names.');
            }
        },
        error: (jqXHR, textStatus, errorThrown) => {
            console.error('Error fetching team names:', textStatus, errorThrown);
        }
    });
}

    setupAutocompleteForPlayerName(playernames) {
        const playerNameInput = $('#playerName');
        id="playerName"
        const playerNamesArray = playernames.map(player => player.playername);
        playerNameInput.autocomplete({
            source: playerNamesArray,
            minLength: 0
        }).focus(function () {
        $(this).autocomplete("search", "");
        });
    }


    showDetailsForm(sport, category, type) {
        const detailsForm = $('#detailsForm');
        detailsForm.empty();

        const details = this.config[sport].categories[category][type].details;
        $.each(details, (index, detail) => {
            const label = $('<label></label>').text(detail.label).addClass('form-label');
        let inputElement;

        if (detail.type === 'select') {
            inputElement = $('<select></select>', { id: detail.id, name: detail.name });
              $.each(detail.options, (index, optionValue) => {
            inputElement.append(this.createOption(optionValue, optionValue));
        });

        }
        else if (detail.id === "winningTeam") {

            inputElement = $('<input>', {
```

```
    type: 'text',
    id: 'winningTeam',
    name: 'winningTeam',
    class: 'ui-autocomplete-input',
    placeholder: detail.placeholder
});
}

else if (detail.id === "playerName") {
    inputElement = $('<input>', {
    type: 'text',
    id: 'playerName',
    name: 'playerName',
    class: 'ui-autocomplete-input',
    placeholder: detail.placeholder
});
}

else {
inputElement = $('<input>', {
type: detail.type,
id: detail.id,
name: detail.name,
placeholder: detail.placeholder
});
}

detailsForm.append(label).append(inputElement);
```

```
    if (detail.id === "winningTeam") {
        this.fetchAndSetupTeamNames();
    }

    if (detail.id === "playerName") {
        this.fetchAndSetupPlayerNames();
    }



    });

    detailsForm.show();

    this.showBettingForm();

}


showBettingForm() {

    const bettingFormContainer = $('#bettingFormContainer');
    bettingFormContainer.empty();

    const betEntryLabel = $('<label/>', {
        'for': 'betEntry',
        text: 'Wager ($):',
        class: 'form-label bet-entry-label'
```

```
});

const dollarSignLabel = $('<span/>', {
   text: '$',
   class: 'dollar-sign'
});

const betEntryInput = $('<input/>', {
   id: 'betEntry',
   type: 'text',
   class: 'form-select bet-entry-input',
   placeholder: ' '
});

const oddsLabel = $('<label/>', {
   'for': 'odds',
   text: 'Bet Odds (+/-):',
   class: 'form-label'
});

const oddsInput = $('<input/>', {
   id: 'odds',
   type: 'text',
   class: 'form-select odds-input',
   placeholder: ' '
});

bettingFormContainer.append(betEntryLabel, betEntryInput, oddsLabel, oddsInput);
```

```
    const addButton = $('<button></button>', {
         text: 'Add to Bet Slip',
        type: 'button',
    click: () => {
        this.addToBet(false);
    }
    }).addClass('add-button');

    bettingFormContainer.append(addButton);
    bettingFormContainer.show();

}

addToBet() {
    const sport = $('#sportSelect').val();
    const category = $('#categorySelect').val();
    const type = $('#typeSelect').val();

    const details = {};

    $('#detailsForm :input').each(function() {
    const input = $(this);
    if (input.attr('type') !== 'button') { // Ignore button inputs
        details[input.attr('id')] = input.val();
    }
    });
```

```
const betEntry = parseFloat($('#betEntry').val());
const odds = parseFloat($('#odds').val());
let betWinnings;

if (!isNaN(odds) && odds > 0 && !isNaN(betEntry)) {

   if (odds > 0) {

      betWinnings = betEntry * (odds / 100);

   }
   else {

      betWinnings = betEntry * (100 / Math.abs(odds));
   }
}

details['betEntry'] = betEntry.toFixed(2);
details['betWinnings'] = betWinnings.toFixed(2);

const displayTextFormat = this.config[sport].categories[category][type].displayText;
let displayText = displayTextFormat;
for (const detailKey in details) {
   displayText = displayText.replace(`{${detailKey}}`, details[detailKey]);
}

this.betSlip.push({ sport, category, type, details, isParlay, displayText });
```

```
        this.displayBet();

    }

    displayBet() {
        $('#betOutput').empty();
        this.betSlip.forEach((bet, index) => {
        const displayTextFormat =
this.config[bet.sport].categories[bet.category][bet.type].displayText;
            let displayText = displayTextFormat;

            for (const detailKey in bet.details) {
                displayText = displayText.replace(`{${detailKey}}`, bet.details[detailKey]);
            }

            const betDisplay = $('<p></p>').addClass('display-bet-text').text(displayText);
            const removeBtn = $('<button></button>', {
                text: 'Remove',
                click: () => {
                    this.betSlip.splice(index, 1); // Remove this bet from the slip.
                    this.displayBet(); // Refresh the display.
                }
            }).addClass('remove-button');

            $('#betOutput').append(betDisplay.append(removeBtn));
            }
     });
    }
```

```
uploadBets() {
    const data = {
        action: 'upload_bets',
        bets: JSON.stringify(this.betSlip),
        screenName: this.screenName
    };

    $.ajax({
        type: 'POST',
        url: betSlipManagerParams.ajaxurl,
        data: data,
        success: function(response) {
            console.log('Success:', response);
            alert('Bet slip has been successfully uploaded to the display.');
        },
        error: function(xhr, status, error) {
            console.error('Error:', error);
            alert('Failed to upload bet slip.');
        }
    });
}


hideForm(formId) {

    $('#' + formId).hide().empty();
    }

    createOption(text, value) {

        return $('<option></option>').val(value).text(text);

    }

    replaceFormContent(formId, element) {

      $('#' + formId).empty().append(element).show();

    }


}


    const config = {
    Baseball: {
      categories: {
        "Game Outcomes": {
          "Moneyline": {
            details: [
              {id: "winningTeam", label: "Winning Team:", type: "text", placeholder: " "},
            ],
            displayText: "{winningTeam} Moneyline   -   ${betEntry} to win ${betWinnings}"
          },
          "Spread": {
            details: [
              {id: "winningTeam", label: "Winning Team:", type: "text", placeholder: " "},
              {id: "spread", label: "Spread:", type: "select",
                options: [-6.5, -5.5, -4.5, -3.5, -2.5, -1.5, "+1.5", "+2.5", "+3.5", "+4.5", "+5.5",
"+6.5"], placeholder: " "


              },
            ],
            displayText: "{winningTeam} {spread}   -   ${betEntry} to win ${betWinnings}"
          },
        },
        "Player Lines": {
          "Hits": {
            details: [
              {id: "playerName", label: "Player Name:", type: "text", placeholder: " "},
              {id: "overUnder", label: "Over/Under:", type: "select", options: ["Over",
"Under"]},
              {id: "hits", label: "Hits:", type: "select",
              options: [0.5, 1.5, 2.5, 3.5 ], placeholder: " "},
            ],
            displayText: "{playerName}: {overUnder} {hits} Hits   -   ${betEntry} to win
${betWinnings}"
          },


          "RBIs": {
            details: [
              {id: "playerName", label: "Player Name:", type: "text", placeholder: " "},
              {id: "overUnder", label: "Over/Under:", type: "select", options: ["Over",
"Under"]},
                {id: "RBIs", label: "RBIs:", type: "select",
                options: [0.5, 1.5, 2.5, 3.5], placeholder: " "},
            ],
            displayText: "{playerName}: {overUnder} {RBIs} RBIs  -  ${betEntry} to win
${betWinnings}"
        },



        "Home Runs": {
            details: [
                {id: "playerName", label: "Player Name:", type: "text", placeholder: " "},
                {id: "overUnder", label: "Over/Under:", type: "select", options: ["Over",
"Under"]},
                {id: "homeRuns", label: "Home Runs:", type: "select", options: [0.5, 1.5],
placeholder: " "},
            ],
            displayText: "{playerName}: {overUnder} {homeRuns} Home Runs  -
${betEntry} to win ${betWinnings}"
        },


        "Walks": {
            details: [
                {id: "playerName", label: "Player Name:", type: "text", placeholder: " "},
                {id: "overUnder", label: "Over/Under:", type: "select", options: ["Over",
"Under"]},
                {id: "walks", label: "Walks:", type: "select",
                options: [1.5, 2.5, 3.5, 4.5], placeholder: " "},
            ],
            displayText: "{playerName}: {overUnder} {walks} Walks  -  ${betEntry} to win
${betWinnings}"
```

```
            },



            "Strikeouts": {
                details: [
                    {id: "playerName", label: "Player Name:", type: "text", placeholder: " "},
                    {id: "overUnder", label: "Over/Under:", type: "select", options: ["Over",
"Under"]},
                    {id: "Strikeouts", label: "Strikeouts:", type: "select", options: [3.5, 4.5, 5.5, 6.5,
7.5, 8.5, 9.5, 10.5, 11.5],placeholder: " "},
                ],
                displayText: "{playerName}: {overUnder} {Strikeouts} Strikeouts  -  ${betEntry}
to win ${betWinnings}"
            },
        },
    },
  },
};
  const betSlipManager = new BetSlipManager(config);


});
```

8.3.4. getData.php

```
<?php
$servername = "localhost";
$username = "gamblers";
$password = "gamblers";
$dbname = "GamblingData"; // Website database credentials

$conn = new mysqli($servername, $username, $password, $dbname); // connects to database

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM `sd46_bet_slips`"; // reads from bet slips
$result = $conn->query($sql);

$data = array();
if ($result->num_rows > 0) {
   while($row = $result->fetch_assoc()) { // prints bet slips to file
      $data[] = $row;
   }
   // Encode data to JSON
   $jsonData = json_encode($data);
   // Echo JSON data
   echo $jsonData;
} else {
    echo "0 results";
}
$conn->close();
?>
```

8.3.5. functions.php

```
<?php


 update_post_meta(58, '_wp_page_template', 'template-bet-slip.php');
  update_post_meta(60, '_wp_page_template', 'template-view-bet.php');



function enqueue_custom_scripts() {
        wp_enqueue_script('bet-slip-manager', get_template_directory_uri() . '/js/bet-slip-
        manager.js', array('jquery'), '1.0', true);
   wp_localize_script('bet-slip-manager', 'betSlipManagerParams', array(
        'ajaxurl' => admin_url('admin-ajax.php')
        ));
   wp_enqueue_script('jquery-ui-autocomplete');
}

add_action('wp_enqueue_scripts', 'enqueue_custom_scripts');

function theme_enqueue_styles() {
   wp_enqueue_style('main-styles', get_stylesheet_uri());
}

add_action('wp_enqueue_scripts', 'theme_enqueue_styles');



add_action('wp_ajax_get_sd46_teamnames', 'get_sd46_teamnames');
add_action('wp_ajax_nopriv_get_sd46_teamnames', 'get_sd46_teamnames');
add_action('wp_ajax_upload_bets', 'handle_upload_bets');
add_action('wp_ajax_nopriv_upload_bets', 'handle_upload_bets');

function get_sd46_teamnames() {
   global $wpdb;
   $table_name = $wpdb->prefix . 'teamnames'; // This assumes your table name is prefixed with
the WP prefix
   $query = "SELECT DISTINCT teamname FROM {$table_name}";
   $teamnames = $wpdb->get_results($query);

   if (!empty($teamnames)) {
      wp_send_json_success($teamnames);
   } else {
      wp_send_json_error('No team names found.');
   }

   wp_die();
}

add_action('wp_ajax_get_sd46_playernames', 'get_sd46_playernames');
add_action('wp_ajax_nopriv_get_sd46_playernames', 'get_sd46_playernames');

function get_sd46_playernames() {
   global $wpdb;
   $table_name = $wpdb->prefix . 'playernames';
   $query = "SELECT DISTINCT playername FROM {$table_name}";
   $playernames = $wpdb->get_results($query);

   if (!empty($playernames)) {
        wp_send_json_success($playernames);
    } else {
        wp_send_json_error('No player names found.');
    }

    wp_die();
}

add_action('wp_ajax_get_sd46_teamnames', 'get_sd46_teamnames');
add_action('wp_ajax_nopriv_get_sd46_teamnames', 'get_sd46_teamnames');



function theme_setup() {
    register_nav_menus(array(
        'header-menu' => 'Header Menu'
    ));
}
add_action('init', 'theme_setup');

function register_theme_menus() {
    register_nav_menus(array(
        'footer-menu' => __('Footer Menu', 'text_domain'),
    ));
}
add_action('init', 'register_theme_menus');

function handle_upload_bets() {
```

```
error_log(print_r($_POST, true));

$bets = json_decode(stripslashes($_POST['bets']), true);
if (empty($bets) || !is_array($bets)) {
    wp_send_json_error('Invalid bets data provided.');
    wp_die();
}
global $wpdb;

$screenName = sanitize_text_field($_POST['screenName']);
$bets = json_decode(stripslashes($_POST['bets']), true);

if (empty($bets) || !is_array($bets)) {
    wp_send_json_error('Invalid bets data provided.');
    wp_die();
}

foreach ($bets as $bet) {
    $sport = sanitize_text_field($bet['sport']);
    $category = sanitize_text_field($bet['category']);
    $type = sanitize_text_field($bet['type']);
    $details = wp_json_encode($bet['details']);
    $displayText = sanitize_text_field($bet['displayText']);
    $insert_result = $wpdb->insert(
        $wpdb->prefix . 'bet_slips',

        array(
            'screen_name' => $screenName,
                'sport' => $sport,
                'category' => $category,
                'type' => $type,
                'details' => $details,
                'display_text' => $displayText,
            ),
            array('%s', '%s', '%s', '%s', '%s', '%s') // Data format
        );

        // Check for successful insertion
        if ($insert_result === false) {
            wp_send_json_error('Failed to insert bet slip: ' . $wpdb->last_error);
            wp_die();
        }
    }

    wp_send_json_success('Bet slips processed successfully.');
    wp_die();
}

function upload_bets_callback() {
    // Check if the request method is POST
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        // Get the data from the POST request
        $data = $_POST['data'];

        echo json_encode(array('success' => true, 'data' => $data));
    }
    wp_die();
}
add_action('admin_post_add_custom_bet', 'handle_custom_form_submission');
```

8.3.6. header-custom.php

```
<?php
/**
 * The custom header for our theme.
 */
?>
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
    <meta charset="<?php bloginfo('charset'); ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="profile" href="http://gmpg.org/xfn/11">
    <?php wp_head(); ?>
</head>
<body <?php body_class(); ?>>
<header class="custom-site-header">
    <div class="custom-header-content">
        <img src="<?php echo get_template_directory_uri(); ?>/textlogo.png" alt="Lib Image"
class="custom-header-logo">
        <p class="custom-header-title"> </p>
        <?php
        if (has_nav_menu('header-menu')) {
            wp_nav_menu(array(
                'theme_location' => 'header-menu',
                'menu_class'     => 'custom-header-nav',
                'container_class' => 'custom-header-menu-container'
            ));
        }
        ?>
    </div>
</header>
```

8.3.7. footer-custom.php

```
<?php
/**
 * The custom footer for our theme.
 */
?>
<footer class="site-footer">
   <div class="footer-content">
      <div class="footer-text-container">
         <p class="footer-title"> </p>
         <P class="footer-rights">All rights reserved.</P>
         <p class="footer-text"> Martin, Tom, John, Joe, and Justin are not liable for any
gambling addictions or monetary losses resulting from the use of this product. Please gamble
responsibly. </p>
      </div>
      <img src="<?php echo get_template_directory_uri(); ?>/footerlogo.png" alt="Logo"
class="footer-logo">
   </div>
</footer>


      <?php
      wp_footer();
      ?>
   </body>
</html>
```

8.3.8. style.css

```
.page-template {
   background-color: #b29758 !important;
}

.content-area, .site-main {
   background-color: #b29758 !important;
}

.dataTables_length {
   color: #000000;
   background-color: #b29758;
   font-size: 1vw;
}
.dataTables_filter {
   color: #000000;
   background-color: #b29758;
   font-size: 1vw;
}
.dataTables_processing {
   color: #000000;
   background-color: #b29758;
   font-size: 1vw;
}
.stripe tr {
   color: #ffffff;
   background-color: #191c40;
```

```
    font-size: 2vw !important;
}
.odd {
    color: #000000 !important;
    background-color: #ffffff !important;
    font-size: 1.8vw !important;
}
.even {
    color: #ffffff !important;
    background-color: #404472 !important;
    font-size: 1.8vw !important;
}

.ui-autocomplete {
    color: #ffffff;
    max-height: 12vw;
    overflow-y: auto;
    overflow-x: hidden;
    font-size: 0.8em;
    list-style-type: none;
    width: 30vw;
    padding-left: 0;
    font-size: 2vw;
}

.ui-menu-item {
    color: #ffffff;
    width: auto;
    font-size: 2vw;
}

.ui-menu-item .ui-menu-item-wrapper {
    padding: .2vw ;
    color: ffffff;
    background: #191c40;
    border-bottom: .1vw solid #191c40;
}

.ui-menu-item .ui-menu-item-wrapper.ui-state-active,
.ui-menu-item .ui-menu-item-wrapper.ui-state-hover {
    background: #404472;
    color: #ffffff;
    cursor: pointer;
}
.form-select {
    margin-top: .5vw;
    color: #191c40;
    font-size: 2vw !important;
    height: 2.5vw;
    padding-right: 1vw;
    margin-bottom:.5vw;
    margin-left: 1vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
```

```
   background-color: #ffffff !important;
   border: .2vw solid #191c40;
}

.form-label {
   font-size: 2vw;
   margin-left: 1.5vw;
   margin-right: .4vw;
   margin-bottom:.5vw;
}

.login-button {
   margin-top: 0;
   color: #ffffff;
   height:2.8vw;
   font-size: 2vw;
   background-color: #191c40;
   opacity: 1;
   cursor: pointer;
   vertical-align: middle;
   border: .2vw solid #ffffff;
}
.login-button:hover {
  border: .3vw solid #ffffff;
  background-color: #5c5f7c;
}
.add-button {
   color: #ffffff;
```

```
    height:2.8vw;
    font-size: 2vw;
    margin-left: 1vw;
    border: .2vw solid #ffffff;
    background-color: #191c40;
    opacity: 1;
    cursor: pointer;
    vertical-align: middle;
    width: 20vw;
}

.add-button:hover {
  border: .3vw solid #ffffff;
  background-color: #5c5f7c;
}

.upload-button {
  color: #ffffff;
  float: none;
  display: block;
  width: calc(40%);
  font-size: 2.1vw;
  text-transform: capitalize;
  text-align: center;
  height: 3vw;
  background-color: #191c40;
  border: .2vw solid #ffffff;
  margin-left: 28vw;
  margin-bottom: 2vw;
  margin-top: 1vw;
}
.upload-button:hover {
    border: .3vw solid #ffffff;
    background-color: #5c5f7c;
}

.remove-button {
    color: #ffffff;
    height:2vw;
    font-size: 1.5vw;
    background-color: #191c40;
    border: .1vw solid #FFFFFF;
    opacity: 1;
    cursor: pointer;
    vertical-align: middle;
    position: absolute;
    right: calc(10%);
}

.remove-button:hover {
  border: .15vw solid #ffffff;
  background-color:#5c5f7c;
}

.display-bet-text {
    height:3v2;
    font-size: 2vw;
    margin-left: 2vw;
    background-color: #191c40;;
    cursor: pointer;
    width: 83vw;
    vertical-align: middle;
    padding-left:1vw;
    padding-top: 1vw;
    padding-right: 10vw;
    border: .2vw solid #ffffff
}

#winningTeam {
    color: #191c40;
    width: 27vw;
    padding-left:.5vw;
    background-color: #ffffff;
    font-size: 2vw;
    height: 2.5vw;
    padding-right: .1vw;
    margin-top: .5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    margin-bottom: 1vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
```

```
}

#playerName {
    color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
   font-size: 2vw;
    height: 2.5vw;
    padding-right: .1vw;
    margin-left: .5vw;
    margin-right: .5vw;
    margin-bottom: 1vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 35vw;
}

#betEntry {
    color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
   font-size: 2vw;
    padding-right: .1vw;
    margin-left: .5vw;
    margin-right: .5vw;
    margin-top: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 6vw;
    height: 2.5vw;
}

#odds {
    color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
   font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
     margin-top: -.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
        margin-top:.5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 6vw;
    height: 2.5vw;
}

#overUnder {
    color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
   font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
    height: 2.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 8vw;
     margin-top: -.5vw;
}

#Strikeouts {
        color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
    font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
     margin-top: -.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 7vw;
    height: 2.5vw;
}

#RBIs  {
        color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
    font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
     margin-top: -.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 6vw;
    height: 2.5vw;
}

#hits {
        color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
    font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
     margin-top: -.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 6vw;
}

#spread {
    color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
    font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
    margin-top: -.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 6.5vw;
}

#walks {
        color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
    font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
     margin-top: -.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 6vw;
}

#homeRuns {
        color: #191c40;
    padding-left:.5vw;
    background-color: #ffffff;
    font-size: 2vw;
    height: 2.8vw;
    padding-right: .1vw;
     margin-top: -.5vw;
    margin-left: .5vw;
    margin-right: .5vw;
    vertical-align: middle;
    opacity: 1;
    cursor: pointer;
    border: .2vw solid #191c40;
    width: 6vw;
}

.bet-entry-label {
    display: inline-block;
}
.dollar-sign {
    margin-right: .5vw;
    font-size: 20vw;
}

.bet-entry-input {
    display: inline-block;
    width: 40vw;
}

.odds-input {
    width: 40vw;
}

.site-footer {
    background-color: #ffffff;
    padding: 1vw;
    padding-left: 1.5vw;
    box-sizing: border-box;
}

.footer-content {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.footer-text-container {
    max-width: calc(40%);
}

.footer-title {
    font-weight: bold;
    color: #333;
    margin-right: auto;
}

.footer-text {
    color: #000000;
    font-size: 1vw;
    margin-top: .5vw;
    word-wrap: break-word;
}
```

```
.footer-rights {
        color: #000000;
    font-size: 1vw;
    margin-top: .5vw;
}

.footer-logo {
    /*height: auto;*/
    position: absolute;
    right: calc(5%);
    width: calc(50%);
}
```

8.3.9. header.css

```
.custom-site-header {
  background-color: #ffffff;
  padding-top: 1vw;
  padding-left: 1vw;
  padding-right: 2vw;
  padding-bottom: 2vw;
  display: flex;
  align-items: center;
  justify-content: space-between;
}

.custom-header-content {
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: start;
}

.custom-header-logo {
  /*transform: scale(4);*/
  width: calc(40%);
  height: auto;
  margin-right: 5vw;
}

.custom-header-title {
  color: #333;
  margin-right: auto;
  font-weight: bold;
}

.custom-header-nav {
  list-style: none;
  display: flex;
}

.custom-header-nav li {
  padding: 1vw;
}

.custom-header-nav a {
  text-decoration: none;
  color: #191c40;
  font-size: 1.5vw;
  font-weight: bold;
}

.custom-header-nav a:hover {
  color: #5c5f7c;
  outline: .3vw solid #5c5f7c;
}

.custom-header-menu-container {
  display: flex;
  align-items: center; }
```

8.4. API Files

```
#include <Arduino.h>

#include <ESP32Lib.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include <WiFiClientSecure.h>

#include <vector>

#include <ctime>

#include <map>

#include <string>

#include "results.h"




const char* ssid = "SDNet";

const char* password = "CapstoneProject";

const char* host = "api.sportradar.us";

const int httpsPort = 80; // Use port 443 for HTTPS

const char* apiKey = "dIf2bFcXi01nQ0rdfcPuUaJDqOMhkHXi4yaGEl13";




String parseString(const String& line, const String& target);

String parseDate(const String& line, const String& target);




class Player;

class Game;
class Player {

  private:

      String fullName;

      int runs;

      int hits;

      int walks;

      int totalBases;

      int singles;

      int doubles;

      int triples;

      int homeRuns;

      int grandSlams;

      int stolenBases;

      int strikeOuts;

      int RBIs;


  public:

      // Default constructor

      Player() : fullName(""), runs(0), hits(0), walks(0), totalBases(0),

                 singles(0), doubles(0), triples(0), homeRuns(0), grandSlams(0),

                 stolenBases(0), strikeOuts(0), RBIs(0){}


      // Constructor that only sets fullName, others to 0

      Player(String fn) : fullName(fn), runs(0), hits(0), walks(0),
totalBases(0),

                          singles(0), doubles(0), triples(0), homeRuns(0),
grandSlams(0),
                          stolenBases(0), strikeOuts(0), RBIs(0) {}


      // Parameterized constructor

      Player(String fn, int r, int h, int w, int tb, int s, int d, int t, int hr,

             int gs, int sb, int so, int rbi) : fullName(fn), runs(r), hits(h),
walks(w),

                                       totalBases(tb), singles(s), doubles(d),

                                       triples(t), homeRuns(hr), grandSlams(gs),

                                       stolenBases(sb), strikeOuts(so), RBIs(rbi)
{}


      // Destructor

      ~Player() {}


      // Getters

      String getFullName() const { return fullName; }

      int getRuns() const { return runs; }

      int getHits() const { return hits; }

      int getWalks() const { return walks; }

      int getTotalBases() const { return totalBases; }

      int getSingles() const { return singles; }

      int getDoubles() const { return doubles; }

      int getTriples() const { return triples; }

      int getHomeRuns() const { return homeRuns; }

      int getGrandSlams() const { return grandSlams; }

      int getStolenBases() const { return stolenBases; }

      int getStrikeOuts() const { return strikeOuts; }
```

```
    int getRBIs() const { return RBIs; }


    // Setters

    void setFullName(String fn) { fullName = fn; }

    void setRuns(int r) { runs = r; }

    void setHits(int h) { hits = h; }

    void setWalks(int w) { walks = w; }

    void setTotalBases(int tb) { totalBases = tb; }

    void setSingles(int s) { singles = s; }

    void setDoubles(int d) { doubles = d; }

    void setTriples(int t) { triples = t; }

    void setHomeRuns(int hr) { homeRuns = hr; }

    void setGrandSlams(int gs) { grandSlams = gs; }

    void setStolenBases(int sb) { stolenBases = sb; }

    void setStrikeOuts(int so) { strikeOuts = so; }

    void setRBIs(int rbi) { RBIs = rbi; }


    bool isEqualTo(const Player& other) const {

        return this->fullName == other.fullName

               && this->runs == other.runs

               && this->hits == other.hits

               && this->walks == other.walks

               && this->totalBases == other.totalBases

               && this->singles == other.singles

               && this->doubles == other.doubles

               && this->triples == other.triples
```

```
                 && this->homeRuns == other.homeRuns

                 && this->grandSlams == other.grandSlams

                 && this->stolenBases == other.stolenBases

                 && this->strikeOuts == other.strikeOuts

                 && this->RBIs == other.RBIs;

      }



};

class Game{

  private:

    String ID;

    String homeTeamName;

    String awayTeamName;

    String homeID;

    String awayID;

    int homeScore;

    int awayScore;

    std::vector<int> homeInningScores;

    std::vector<int> awayInningScores;

    std::map<String,Player> homeRoster;

    std::map<String,Player> awayRoster;

    String inningState;

    int outs;

    bool finished;
```

```
public:


  // Default constructor

  Game() {}


  // Constructor that initializes based on Teams

  Game(const String& id)

  :ID(id) {

    homeInningScores.resize(9);

    awayInningScores.resize(9);

    finished = false;

  }



  // Destructor

  ~Game() {

      // Clean up resources, if any were allocated

  }


  void reset() {

      // Resetting team names and IDs

      homeTeamName = "";

      awayTeamName = "";

      homeID = "";

      awayID = "";

```

```
        // Resetting scores

        homeScore = 0;

        awayScore = 0;



        // Clearing inning scores

        homeInningScores.clear();

        awayInningScores.clear();

        homeInningScores.resize(9, 0);  // Assuming standard 9 innings, setting
all to 0

        awayInningScores.resize(9, 0);



        // Clearing player rosters

        homeRoster.clear();

        awayRoster.clear();

    }



    void addPlayer(const Player& p, const String& name, const String& team_id){

      if(team_id == homeID){

        homeRoster[name] = p;

        //Serial.println("Added player not on roster to " + homeTeamName + ": " +
name);

      }

      else if(team_id == awayID){

        awayRoster[name] = p;

        //Serial.println("Added player not on roster to " + awayTeamName + ": " +
name);

      }
```

```
}


bool findPlayer(const String& name, const String& team_id){

  if(team_id == homeID){

    if (homeRoster.count(name) > 0){

      return true; // return 1 if player is on home team

    }

  }

  if(team_id == awayID){

    if (awayRoster.count(name) > 0){

      return true; // return 2 if player is on away team

    }

  }


  return false;

}


// Set Home Team's score

void setHomeScore(const int score){

  homeScore = score;

}


// Set Away Team's score

void setAwayScore(const int score){

  awayScore = score;
}

void setOuts(int o){
  outs = o;
}

int getOuts(){
  return outs;
}

void endGame(){
  finished = true;
}

bool checkProgress(){
  return finished;
}

// Get Home Team Score
String getHomeScore(){
  return String(homeScore); // Convert int to string
}

// Get Away Team Score
String getAwayScore(){
  return String(awayScore); // Convert int to string
```

```
}

void setHomeName(const String& name){homeTeamName = name;}

void setAwayName(const String& name){awayTeamName = name;}

String getHomeName(){
  return homeTeamName;
}

String getAwayName(){
  return awayTeamName;
}

void setHomeID(const String& ID){homeID = ID;}

void setAwayID(const String& ID){awayID = ID;}

String getHomeID(){
  return homeID;
}

String getAwayID(){
  return awayID;
}
```

```
String getGameId(){return ID;}


void setInningState(bool state, int inningCount){

  if(state){

    inningState = "B" + String(inningCount);

    }

  else{

    inningState = "T" + String(inningCount);

    }

};


void setInningScoreHome(int inningCount, const int score) {

  if (inningCount > homeInningScores.size()) {

    homeInningScores.resize(inningCount);

  }

  homeInningScores.at(inningCount-1) = score;

}


void setInningScoreAway(int inningCount, const int score) {

  if (inningCount > awayInningScores.size()) {

    awayInningScores.resize(inningCount);

  }

  awayInningScores.at(inningCount-1) = score;

}
```

```
void updateHitCount(const String& hitterName, const bool& topBottom){

  if (topBottom == false){
    // away
    int totalHits = awayRoster[hitterName].getHits() + 1;
    awayRoster[hitterName].setHits(totalHits);
  }
  if (topBottom == true){
    // home
    int totalHits = homeRoster[hitterName].getHits() + 1;
    homeRoster[hitterName].setHits(totalHits);
  }
}

void updateSOCount(const String& pitcherName, const bool& topBottom){
  if (topBottom == false){
    // home
    int totalSOs = homeRoster[pitcherName].getStrikeOuts() + 1;
    homeRoster[pitcherName].setStrikeOuts(totalSOs);
  }
  if (topBottom == true){
    // away
    int totalSOs = awayRoster[pitcherName].getStrikeOuts() + 1;
    awayRoster[pitcherName].setStrikeOuts(totalSOs);
  }
}
```

```
void updateRBICount(const String& hitterName, const bool& topBottom){
  if (topBottom == false){
    // away
    int totalRBIs = awayRoster[hitterName].getRBIs() + 1;
    awayRoster[hitterName].setRBIs(totalRBIs);
  }
  if (topBottom == true){
    // home
    int totalRBIs = homeRoster[hitterName].getRBIs() + 1;
    homeRoster[hitterName].setRBIs(totalRBIs);
  }
}

void updateRunCount(const String& runnerName, const bool& topBottom) {
  if (topBottom == false){
    // away
    int totalRuns = awayRoster[runnerName].getRuns() + 1;
    awayRoster[runnerName].setRuns(totalRuns);
  }
  if (topBottom == true){
    // home
    int totalRuns = homeRoster[runnerName].getRuns() + 1;
    homeRoster[runnerName].setRuns(totalRuns);
  }
}
```

```
void updateSingleCount(const String& hitterName, const bool& topBottom) {
    if (topBottom == false){
    // away
    int totalSingles = awayRoster[hitterName].getSingles() + 1;
    awayRoster[hitterName].setSingles(totalSingles);
  }
  if (topBottom == true){
    // home
    int totalSingles = homeRoster[hitterName].getSingles() + 1;
    homeRoster[hitterName].setSingles(totalSingles);
  }
}

void updateDoubleCount(const String& hitterName, const bool& topBottom) {
    if (topBottom == false){
    // away
    int totalDoubles = awayRoster[hitterName].getDoubles() + 1;
    awayRoster[hitterName].setDoubles(totalDoubles);
  }
  if (topBottom == true){
    // home
    int totalDoubles = homeRoster[hitterName].getDoubles() + 1;
    homeRoster[hitterName].setDoubles(totalDoubles);
  }
}
```

```
void updateTripleCount(const String& hitterName, const bool& topBottom) {
  if (topBottom == false){
    // away
    int totalTriples = awayRoster[hitterName].getTriples() + 1;
    awayRoster[hitterName].setTriples(totalTriples);
  }
  if (topBottom == true){
    // home
    int totalTriples = homeRoster[hitterName].getTriples() + 1;
    homeRoster[hitterName].setTriples(totalTriples);
  }
}

void updateHRCount(const String& hitterName, const bool& topBottom) {
    if (topBottom == false){
    // away
    int totalHRs = awayRoster[hitterName].getHomeRuns() + 1;
    awayRoster[hitterName].setHomeRuns(totalHRs);
  }
  if (topBottom == true){
    // home
    int totalHRs = homeRoster[hitterName].getHomeRuns() + 1;
    homeRoster[hitterName].setHomeRuns(totalHRs);
  }
}
```

```
    void updateStealCount(const String& runnerName, const bool& topBottom, const
int& bases){
      if (topBottom == false){
        // away
        int totalSteals = awayRoster[runnerName].getStolenBases() + 1;
        awayRoster[runnerName].setStolenBases(totalSteals);
      }
      if (topBottom == true){
        // home
        int totalSteals = homeRoster[runnerName].getStolenBases() + 1;
        homeRoster[runnerName].setStolenBases(totalSteals);
      }
    }

    void updateWalkCount(const String& Name, const bool& topBottom) {
      if (topBottom == false){
        // away
        int totalWalks = awayRoster[Name].getWalks() + 1;
        awayRoster[Name].setWalks(totalWalks);
      }
      if (topBottom == true){
        // home
        int totalWalks = homeRoster[Name].getWalks() + 1;
        homeRoster[Name].setWalks(totalWalks);
      }
    }
```

```
void printGameDetails() {
    Serial.println("Game ID: " + ID);
    Serial.println("Current Score:");
    Serial.print("Home: " + homeTeamName + " - ");
    Serial.println(homeScore);
    Serial.print("Away: " + awayTeamName + " - ");
    Serial.println(awayScore);

    Serial.println("Inning Scores:");
    for (size_t i = 0; i < homeInningScores.size(); ++i) {
        Serial.print("Inning ");
        Serial.print(i + 1);
        Serial.print(": Home: ");
        Serial.print(homeInningScores[i]);
        Serial.print(" Away: ");
        Serial.println(awayInningScores[i]);
    }

    // Printing details of the home roster players
    Serial.println("Home Team Roster:");
    for (auto& pair : homeRoster) {
        Serial.print("Player: ");
        Serial.println(pair.first);
        printPlayerDetails(pair.second);
    }

    // Printing details of the away roster players
    Serial.println("Away Team Roster:");
    for (auto& pair : awayRoster) {
        Serial.print("Player: ");
        Serial.println(pair.first);
        printPlayerDetails(pair.second);
    }
}

// Helper function to print details of a player
void printPlayerDetails(const Player& player) {
    Serial.print("Full Name: ");
    Serial.println(player.getFullName());
    Serial.print("Runs: ");
    Serial.println(player.getRuns());
    Serial.print("Hits: ");
    Serial.println(player.getHits());
    Serial.print("Walks: ");
    Serial.println(player.getWalks());
    //Serial.print("Total Bases: ");
    //Serial.println(player.getTotalBases());
    Serial.print("Singles: ");
    Serial.println(player.getSingles());
    Serial.print("Doubles: ");
```

```
        Serial.println(player.getDoubles());

        Serial.print("Triples: ");

        Serial.println(player.getTriples());

        Serial.print("Home Runs: ");

        Serial.println(player.getHomeRuns());

        //Serial.print("Grand Slams: ");

        //Serial.println(player.getGrandSlams());

        //Serial.print("Stolen Bases: ");

        //Serial.println(player.getStolenBases());

        Serial.print("Strike Outs: ");

        Serial.println(player.getStrikeOuts());

        Serial.print("RBIs: ");

        Serial.println(player.getRBIs());

    }



    String retrieveStat(const String& stat, const String& player, const String&
bettingTeam) {

      Serial.println("Beginning of retrievestat");









      int playerStat;

      int statType;
      if(stat == "Score"){statType = 1;}

      if(stat == "Hits"){statType = 2;}

      if(stat == "RBIs"){statType = 3;}

      if(stat == "Strikeouts"){statType = 4;}

      if(stat == "Home Runs"){statType = 5;}

      if(stat == "Walks"){statType = 6;}



      switch(statType){

        case 1:

          if (homeTeamName == bettingTeam){

            return inningState + ", Betting Team: " + bettingTeam + " - " +
String(homeScore) + ", Other Team: " + awayTeamName + " - " + String(awayScore);

          }

          else{

            return inningState + ", Betting Team: " + bettingTeam + " - " +
String(awayScore) + ", Other Team: " + homeTeamName + " - " + String(homeScore);

          }

          break;

        case 2:


          if (homeTeamName == bettingTeam){

            playerStat = homeRoster[player].getHits();

            return player + " Hits: " + String(playerStat);

          }

          else{

            playerStat = awayRoster[player].getHits();
            return player + " Hits: " + String(playerStat);
          }
          break;
        case 3:
          if (homeTeamName == bettingTeam){
            playerStat = homeRoster[player].getRBIs();
            return player + " RBIs: " + String(playerStat);
          }
          else{
            playerStat = awayRoster[player].getRBIs();
            return player + " RBIs: " + String(playerStat);
          }
          break;
        case 4:
          if (homeTeamName == bettingTeam){
            playerStat = homeRoster[player].getStrikeOuts();
            return player + " Strikeouts: " + String(playerStat);
          }
          else{
            playerStat = awayRoster[player].getStrikeOuts();
            return player + " Strikeouts: " + String(playerStat);
          }
          break;
        case 5:
          if (homeTeamName == bettingTeam){
            playerStat = homeRoster[player].getHomeRuns();
```

```
            return player + " Home Runs: " + String(playerStat);
          }
          else{
            playerStat = awayRoster[player].getHomeRuns();
            return player + " Home Runs: " + String(playerStat);
          }
          break;
          case 6:
          if (homeTeamName == bettingTeam){
            playerStat = homeRoster[player].getWalks();
            return player + " Walks: " + String(playerStat);
          }
          else{
            playerStat = awayRoster[player].getWalks();
            return player + " Walks: " + String(playerStat);
          }
          break;
        default:
          break;
      }
      return "Invalid stat retrieval call.";
    }

};

std::vector<Game*> allGames = {};
```

```
int parseNumberFromString(String input) {
    String numberString = "";
    bool foundNumber = false;

    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c >= '0' && c <= '9') {
            foundNumber = true;
            numberString += c;
        } else if (foundNumber) {
            break;
        }
    }
    int number = numberString.toInt();
    return number;

}



// Function declarations

String getDate(void);
void connectWifi(const char* ssid, const char* password);
String find_game(const char* apiKey, const String& date, const String& teamAbbr);
bool update_game(Game* currGame, const char* apiKey);
```

```
Game* init_game(const String& betTeamAbbr, const char* apiKey);

String get_new_stat(Game* currGame, const String& stat, const String& player,
const String& bettingTeam);


String parseString(const String& line, const String& target) {

    int pos = line.indexOf(target);

    if (pos != -1) { // Corrected comparison

        int startPos = line.indexOf('\"', pos) + 1; // Note the use of single
quotes for the character literal

        int endPos = line.indexOf('\"', startPos);

        if (startPos != -1 && endPos != -1) { // Additional check for robustness

            return line.substring(startPos, endPos);

        }

    }

    return String(""); // Return an empty String instead of String(NULL)

}


String parseDate(const String& line, const String& target) {

    int pos = line.indexOf(target);

    String actualDate = "";

    String utcHour = "";

    if (pos != -1) { // Corrected comparison

        int startPos = line.indexOf('\"', pos) + 1; // Note the use of single
quotes for the character literal

        int endPos = line.indexOf('T', startPos);

        if (startPos != -1 && endPos != -1) { // Additional check for robustness

            actualDate = line.substring(startPos, endPos);
        }
    }
    if (pos != -1){
      int startPos = line.indexOf('T', pos) + 1; // Note the use of single quotes
for the character literal
      int endPos = line.indexOf(':', startPos);
      if (startPos != -1 && endPos != -1) { // Additional check for robustness
            utcHour = line.substring(startPos, endPos);
        }
    }
    int hour = utcHour.toInt();
    if (hour < 7){
      String day = actualDate.substring(8,10);
      int actualDay = day.toInt() - 1;
      actualDate = actualDate.substring(0,8) + String(actualDay);
    }
    return actualDate; // Return an empty String instead of String(NULL)
}

String getDate() {
    // Obtain current time
    std::time_t t = std::time(nullptr);

    // Convert to local time structure
    std::tm* now = std::localtime(&t);

    // Manually adjust for Pacific Standard Time (PST, UTC-8)
    // Note: Add the offset instead of subtracting to correct the underflow issue

    now->tm_hour -= 8; // Adjusting hours directly in the tm structure


    // Normalize the tm struct in case the adjustments cross day boundaries

    mktime(now); // Adjusts the time structure in case of under/overflow


    // Buffer to store the formatted date

    char buffer[11];


    // Format the date as YYYY-MM-DD

    strftime(buffer, sizeof(buffer), "%Y-%m-%d", now);


    // Return the formatted date as an Arduino String

    return String(buffer);

}


String find_game(const char* apiKey, const String& date, const String& teamAbbr){



  const char* host = "api.sportradar.us";

  const int httpsPort = 80; // Use port 443 for HTTPS

  String regSeasonUrl = "/mlb/trial/v7/en/games/2024/REG/schedule.xml?api_key=" +
String(apiKey);




    Serial.print("Connecting to ");
    Serial.println(host);

    if (!client.connect(host, httpsPort)) {

        Serial.println("Connection failed!");

        return "";

    }

    client.print(String("GET ") + regSeasonUrl + " HTTP/1.1\r\n" + "Host: " +
host + "\r\n" + "User-Agent: BuildFailureDetectorESP32\r\n" + "Connection:
close\r\n\r\n");

    Serial.println(String("GET ") + regSeasonUrl);

    Serial.println("Request sent");

    while (client.connected()) {

        String line = client.readStringUntil('\n');

        if (line == "\r") {

            Serial.println("Headers received");

            break;

        }

    }


    String game_id = "";


    bool foundDateMatch = false;




    while(!client.available()){}


    while (client.available()) {

      String line = client.readStringUntil('\n');
      if (line.indexOf("game id=") != -1) {

        String rawDate = parseDate(line, "scheduled=");

        String inProgress = parseString(line, "status=");

        if(rawDate == date && inProgress != "closed"){

          foundDateMatch = true;

          game_id = parseString(line, "game id=");

        }

        else{

          foundDateMatch = false;

        }

      }

      if (foundDateMatch == true && (line.indexOf("home name=") != -1 ||
line.indexOf("away name=") != -1 )) {

        Serial.println(line);

        if (parseString(line, "abbr=") == teamAbbr){

          break;

        }

      }

    }



    if (game_id == ""){

      Serial.println("I'm an ESP32 and I fucked up getting the game ID again");

    }


  return game_id;
```

```
}

bool update_game(Game* currGame, const char* apiKey) {



  // Your function's logic here, adapted for task use

  String game_id = currGame->getGameId();


  const char* host = "api.sportradar.us";
  const int httpsPort = 80; // Use port 443 for HTTPS
  String url = "/mlb/trial/v7/en/games/" + String(game_id) + "/pbp.xml?api_key="
+ String(apiKey);

    Serial.print("Connecting to ");
    Serial.println(host);
    if (!client.connect(host, httpsPort)) {
        Serial.println("Connection failed!");
        return false;
    }
    client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " + host +
"\r\n" + "User-Agent: BuildFailureDetectorESP32\r\n" + "Connection:
close\r\n\r\n");
    Serial.println(String("GET ") + url);
    Serial.println("Request sent");

    while (client.connected()) {

        String line = client.readStringUntil('\n');

        if (line == "\r") {

            Serial.println("Headers received");

            break;

        }

    }


  currGame->reset();



  int parsingState = 0;

  bool inningState;


  String teamName;

  int runs;

  int inningCount;

  String playerName;

  String pitcherName;

  String hitterName;

  String runnerName;

  String homeID;

  String awayID;

  bool hitState;
```

```
String temp1;

String temp2;

String temp3;

int tempInt;



bool foundBottom;



while(!client.available()){}


while (client.available()){

  String line = client.readStringUntil('\n');


  switch(parsingState){

    case 0: // Game Level

      //Serial.println("At Game Level!");

      if(line.indexOf("<game ") != -1){

        // Get game progress

        temp1 = parseString(line, "status=");


        // Set game progress

        if(temp1 == "closed" || temp1 == "canceled"){

          currGame->endGame();

        }

        parsingState = 0;
      }
      if(line.indexOf("<scoring>") != -1){
        // Go to Total Score Level
        parsingState = 1;
      }
      if(line.indexOf("<inning number") != -1){
        // Get relevant info
        temp1 = parseString(line, "number=");

        // Set relevant info
        inningCount = temp1.toInt();

        // Go to Inning Level
        parsingState = 2;
      }
      if(line.indexOf("</game>") != -1){
        // API call over
        parsingState = 100;
        break;
      }
      break;
    case 1: // Total Score Level
      //Serial.println("At Total Score Level!");
      if(line.indexOf("<home") != -1){
        // Get relevant info
        teamName = parseString(line, "abbr=");
        homeID = parseString(line, "id=");

        temp1 = parseString(line, "runs=");



        // Set relevant info

        runs = temp1.toInt();

        currGame->setHomeName(teamName);

        currGame->setHomeScore(runs);

        currGame->setHomeID(homeID);



        // Set parsingState

        parsingState = 1;

      }

      if(line.indexOf("<away") != -1){

        // Get relevant info

        teamName = parseString(line, "abbr=");

        awayID = parseString(line, "id=");

        temp1 = parseString(line, "runs=");



        // Set relevant info

        runs = temp1.toInt();

        currGame->setAwayName(teamName);

        currGame->setAwayScore(runs);

        currGame->setAwayID(awayID);



        // Set parsingState

        parsingState = 1;
```

```
        }
        if(line.indexOf("</scoring>") != -1){


          // Set parsingState
          parsingState = 2;
        }
        break;
      case 2: // Inning Level
        //Serial.println("At Inning Level!");
        if(line.indexOf("<scoring>") != -1){


          // Go to Inning Score Level
          parsingState = 3;
        }
        if(line.indexOf("<inning_half") != -1){
          // Get relevant info
          temp1 = parseString(line, "type=");



          // Set relevant info
          if(temp1 == "T"){
            foundBottom = false;
            inningState = false;


          }
          else {
```

```
        inningState = true;

        foundBottom = true;


      }

      currGame->setInningState(inningState, inningCount);


      // Go to Inning Half Level

      parsingState = 4;

    }

    if(line.indexOf("</inning>") != -1){

      // Go to Game Level

      if(foundBottom == true){

        inningState = false;

        currGame->setInningState(inningState, inningCount);

      }

      parsingState = 0;

    }

    break;

  case 3: // Inning Score Level

    //Serial.println("At Inning Score Level!");

    if(line.indexOf("<home") != -1){

      // Get relevant info

      temp1 = parseString(line, "runs=");


      // Set relevant info

      runs = temp1.toInt();
```

```
          currGame->setInningScoreHome(inningCount, runs);


          // Set parsingState

          parsingState = 3;

        }

        if(line.indexOf("<away") != -1){

          // Get relevant info

          temp1 = parseString(line, "runs=");


          // Set relevant info

          runs = temp1.toInt();

          currGame->setInningScoreAway(inningCount, runs);


          // Set parsingState

          parsingState = 3;

        }

        if(line.indexOf("</scoring>") != -1){

          // Go back to Inning Level

          parsingState = 2;

        }

        break;

      case 4: // Inning Half Level

        if(line.indexOf("<at_bat") != -1){


          foundBottom = false;

          // Go to At Bat Level
          parsingState = 5;

        }

        if(line.indexOf("<lineup") != -1){

          // Get relevant info

          playerName = parseString(line, "full_name=");

          temp1 = parseString(line, "team_id=");



          //Serial.println("New player in lineup: " + playerName);

          // Set relevant info

          if(!currGame->findPlayer(playerName, temp1)){

            Player tempPlayer(playerName);

            currGame->addPlayer(tempPlayer, playerName, temp1);

          }



          // Set parsingState

          parsingState = 4;

        }

        if(line.indexOf("</inning_half>") != -1){

          // Go back to Inning Level

          parsingState = 2;

        }

        break;

      case 5: // At Bat Level

        if(line.indexOf("<pitcher ") != -1){

          // Get relevant info

          pitcherName = parseString(line, "full_name=");
```

```
          // set parsingState
          parsingState = 5;
        }
        if(line.indexOf("<hitter ") != -1){
          // Get relevant info
          hitterName = parseString(line, "full_name=");

          // set parsingState
          parsingState = 5;
        }
        if(line.indexOf("<pitch ") != -1){
          // Get relevant info
          temp1 = parseString(line, "outcome_id=");
          temp2 = parseString(line, "status=");

          // Set relevant info
          // If the pitch is overturned it gets shown twice, but we only care
about the official call, so skip overturned pitches
          if(temp2 == "overturned"){
            parsingState = 5;
            break;
          }
          if(temp1 == "aHR"){
            currGame->updateHRCount(hitterName, inningState);
            currGame->updateHitCount(hitterName, inningState);
          }
          if(temp1 == "aT" || temp1 == "aTAD4" || temp1 == "oTT3" || temp1 ==
"oTT4"){
            currGame->updateTripleCount(hitterName, inningState);
            currGame->updateHitCount(hitterName, inningState);
          }
          if(temp1 == "aD" || temp1 == "aDAD3" || temp1 == "aDAD4" || temp1 ==
"oDT2" || temp1 == "oDT3" || temp1 == "oDT4"){
            currGame->updateDoubleCount(hitterName, inningState);
            currGame->updateHitCount(hitterName, inningState);
          }
          if(temp1 == "aS" || temp1 == "aSAD2" || temp1 == "aSAD3" || temp1 ==
"aSAD4" || temp1 == "oST1" || temp1 == "oST2" || temp1 == "oST3" || temp1 ==
"oST4"){
            currGame->updateSingleCount(hitterName, inningState);
            currGame->updateHitCount(hitterName, inningState);
          }

          // Go to Pitch Level
          parsingState = 6;
        }
        if(line.indexOf("<lineup ") != -1){
          // Get relevant info
          playerName = parseString(line, "full_name=");
          temp1 = parseString(line, "team_id=");

          // Set relevant info
          if(!currGame->findPlayer(playerName, temp1)){
            Player tempPlayer(playerName);
            currGame->addPlayer(tempPlayer, playerName, temp1);
          }

          // Set parsingState
          parsingState = 5;
        }
        if(line.indexOf("</at_bat>") != -1){
          // Go back to Inning Half Level
          parsingState = 4;
        }
        //if(line.indexOf("<steal ") != -1){
          // Go to Steal Level
        //  parsingState = 7;
        //}
        break;
      case 6: // Pitch Level
        if(line.indexOf("<runners>") != -1){
          // Go to Pitch Runners Level
          parsingState = 8;
        }
        if(line.indexOf("<count ") != -1){
          // Get relevant info
          temp1 = parseString(line, "balls=");
          temp2 = parseString(line, "strikes=");
          temp3 = parseString(line, "outs=");
          // Set relevant info
          // Walk count updater
          if(temp2.toInt() == 3){
            currGame->updateSOCount(pitcherName, inningState);
          }
          if(temp1.toInt() == 4){
            currGame->updateWalkCount(pitcherName, !inningState);
            currGame->updateWalkCount(hitterName, inningState);
          }
          currGame->setOuts(temp3.toInt());
          // Set parsingState
          parsingState = 6;
        }
        if(line.indexOf("</pitch>") != -1){
          // Go back to At Bat Level
          parsingState = 5;
        }
        break;
      case 7: // Steal Level
        if(line.indexOf("<runners>") != -1){
          // Go to Steal Runners Level
        }
        break;
      case 8: // Pitch Runners Level
        if(line.indexOf("<runner ") != -1){
          // Get relevant info
          runnerName = parseString(line, "full_name=");
          temp1 = parseString(line, "outcome_id=");


          // Set relevant info
          if(temp1 == "ERN" || temp1 == "eRN" || temp1 == "ERNu" || temp1 ==
"eRNu" || temp1 == "URN" || temp1 == "uRN" || temp1 == "SB4u" || temp1 ==
"SB3AD4u" || temp1 == "SB2AD4u" ){
            currGame->updateRunCount(runnerName, inningState);
            if(temp1 == "ERN" || temp1 == "ERNu" || temp1 == "URN"){
              currGame->updateRBICount(hitterName, inningState);
            }
          }


          // Set parsingState
          parsingState = 8;
        }
        if(line.indexOf("</runners>") != -1){
          // Go back to Pitch Level
          parsingState = 6;
        }
        break;
    }
  }


  if (parsingState != 100){
    Serial.println("Didn't reach end of API call!! NOOOOOOOOOO!!!!");
    return false;
  }
  else{
    Serial.println("Game successully updated.");
    return true;
  }
}

Game* init_game(const String& betTeamAbbr, const char* apiKey){
  String date = getDate();
  date = "2024-04-30";
  Serial.println("Date: " + date);
  String gameId = "";
  while(gameId == ""){gameId = find_game(apiKey, date, betTeamAbbr);}
  Serial.println("Game ID: " + gameId);
  delay(1000);
  Serial.println(esp_get_free_heap_size());
  Game* currGame = new Game(gameId);
  Serial.println("Set Game Details!");
  Serial.println(esp_get_free_heap_size());
  return currGame;
}

String get_new_stat(Game* currGame, const String& stat, const String& player, 
const String& bettingTeam){
  String send_to_screen = currGame->retrieveStat(stat, player, bettingTeam);
  return send_to_screen;
}
```

```
void setup() {
Serial.begin(115200);



  WiFi.begin(ssid, password);


  Serial.println("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("Connected to WiFi");
  configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
  delay(2000);

  // Define the URL to read from
  const char* url = "http://gamblingscreen.com/getdata.php";
  NeoPixel.begin();


}
```

8.5. Sound Software Files

8.5.1. sound.h

```
#include <Arduino.h>

#include <stdio.h>

#include <stdlib.h>

#include "driver/i2s.h" // Library of I2S routines, comes with ESP32 standard
install

#include "Bruins_Goal_Horn.h"

#include "Sad_Trambone.h"

#include "Sonic_Ring.h"

#include "Curb_Loser.h"


#define USE_I2S_SPEAKER_OUTPUT

#define I2S_SPEAKER_SERIAL_DATA GPIO_NUM_4

#define I2S_SPEAKER_SERIAL_CLOCK GPIO_NUM_16

#define I2S_SPEAKER_LEFT_RIGHT_CLOCK GPIO_NUM_17


static i2s_port_t i2s_num = I2S_NUM_0; // i2s port number



struct WavHeader_Struct

{

    //   RIFF Section

    char RIFFSectionID[4]; // Letters "RIFF"

    uint32_t Size;          // Size of entire file less 8
    char RiffFormat[4];    // Letters "WAVE"


    //   Format Section

    char FormatSectionID[3]; // letters "fmt"

    uint32_t FormatSize;     // Size of format section less 8

    uint16_t FormatID;       // 1=uncompressed PCM

    uint16_t NumChannels;    // 1=mono,2=stereo

    uint32_t SampleRate;     // 44100, 16000, 8000 etc.

    uint32_t ByteRate;       // =SampleRate * Channels * (BitsPerSample/8)

    uint16_t BlockAlign;     // =Channels * (BitsPerSample/8), effectivly the
size of a single sample for all chans.

    uint16_t BitsPerSample;  // 8,16,24 or 32


    // Data Section

    char DataSectionID[4]; // The letters "data"

    uint32_t DataSize;     // Size of the data that follows

};


// speaker settings




// Shutdown line if you have this wired up or -1 if you don't

#define I2S_SPEAKER_SD_PIN -1






//    HEADERS    //
```

```
bool validWAVData(WavHeader_Struct *Wav);

void dumpWAVHeader(WavHeader_Struct *Wav);

void play(const unsigned char *WavFile);


//    FUNCTION DEFINITIONS    //


bool validWAVData(WavHeader_Struct *Wav);

void dumpWAVHeader(WavHeader_Struct *Wav);

void play(const unsigned char *WavFile);
```

8.5.2. sound.cpp

```
#include"sound.h"




WavHeader_Struct WavHeader;



bool validWAVData(WavHeader_Struct *Wav)
{
    if (memcmp(Wav->RIFFSectionID, "RIFF", 4) != 0)
    {
        Serial.print("Invlaid data - Not RIFF format");
        return false;
    }
    if (memcmp(Wav->RiffFormat, "WAVE", 4) != 0)
    {
        Serial.print("Invlaid data - Not Wave file");
        return false;
    }
    if (memcmp(Wav->FormatSectionID, "fmt", 3) != 0)
    {
        Serial.print("Invlaid data - No format section found");
        return false;
    }
    if (memcmp(Wav->DataSectionID, "data", 4) != 0)
    {
        Serial.print("Invlaid data - data section not found");
        return false;
    }
    if (Wav->FormatID != 1)
    {
        Serial.print("Invlaid data - format Id must be 1");
        return false;
    }
    if (Wav->FormatSize != 16)
    {
        Serial.print("Invlaid data - format section size must be 16.");
        return false;
    }
    if ((Wav->NumChannels != 1) & (Wav->NumChannels != 2))
    {
        Serial.print("Invlaid data - only mono or stereo permitted.");
        return false;
    }
    if (Wav->SampleRate > 48000)
    {
        Serial.print("Invlaid data - Sample rate cannot be greater than 48000");
        return false;
    }
    if (Wav->BitsPerSample != 16)
    {
        Serial.print("Invlaid data - Only 16 bits per sample permitted.");
        return false;
    }
    return true;
}

void dumpWAVHeader(WavHeader_Struct *Wav)
{
    if (memcmp(Wav->RIFFSectionID, "RIFF", 4) != 0)
    {
        Serial.print("Not a RIFF format file - ");
        Serial.print(Wav->RIFFSectionID);
        return;
    }
    if (memcmp(Wav->RiffFormat, "WAVE", 4) != 0)
    {
        Serial.print("Not a WAVE file - ");
        Serial.print(Wav->RiffFormat);
        return;
    }
    if (memcmp(Wav->FormatSectionID, "fmt", 3) != 0)
    {
        Serial.print("fmt ID not present - ");
        Serial.print(Wav->FormatSectionID);
        return;
    }
    if (memcmp(Wav->DataSectionID, "data", 4) != 0)
    {
        Serial.print("data ID not present - ");
        Serial.print(Wav->DataSectionID);
        return;
    }

    // All looks good, dump the data
    Serial.print("Total size :");
    Serial.println(Wav->Size);
    Serial.print("Format section size :");
    Serial.println(Wav->FormatSize);
    Serial.print("Wave format :");
    Serial.println(Wav->FormatID);
    Serial.print("Channels :");
    Serial.println(Wav->NumChannels);
    Serial.print("Sample Rate :");
    Serial.println(Wav->SampleRate);
    Serial.print("Byte Rate :");
    Serial.println(Wav->ByteRate);
    Serial.print("Block Align :");
    Serial.println(Wav->BlockAlign);
    Serial.print("Bits Per Sample :");
    Serial.println(Wav->BitsPerSample);
    Serial.print("Data Size :");
    Serial.println(Wav->DataSize);
}
```

```
void play(const unsigned char *WavFile)
{
    unsigned const char *TheData;
    uint32_t DataIdx = 0;            // index offset into "TheData" for current
data t send to I2S
    memcpy(&WavHeader, WavFile, 44); // Copy the header part of the wav data into
our structure
    dumpWAVHeader(&WavHeader);       // Dump the header data to serial, optional!
    if (validWAVData(&WavHeader))
    {
        i2s_set_sample_rates(i2s_num, WavHeader.SampleRate); // set sample rate
        TheData = WavFile + 44;                              // set to start of
data

        while (DataIdx <= WavHeader.DataSize)
        {
            uint8_t Mono[4];           // This holds the data we actually send to
the I2S if mono sound
            const unsigned char *Data; // Points to the data we are going to send
            size_t BytesWritten;       // Returned by the I2S write routine, we
are not interested in it

            // The WAV Data could be mono or stereo but always 16 bit, that's a
data size of 2 byte or 4 bytes
            // Unfortunatly I2S only allows stereo, so to send mono we have to
send the mono sample on both left and right
            // channels. It's a bit of a faf really!
            if (WavHeader.NumChannels == 1) // mono
            {
                Mono[0] = *(TheData + DataIdx); // copy the sample to both left
and right samples, this is left
                Mono[1] = *(TheData + DataIdx + 1);
                Mono[2] = *(TheData + DataIdx); // Same data to the right channel
                Mono[3] = *(TheData + DataIdx + 1);
                Data = Mono;
            }
            else // stereo
                Data = TheData + DataIdx;


            i2s_write(i2s_num, Data, 4, &BytesWritten, portMAX_DELAY);
            DataIdx += WavHeader.BlockAlign; // increase the data index to next
next sample
        }
    }
}
```

8.6. LED and Audio Condition Software Files

8.6.1. results.h

```
#ifndef RESULTS_H

#define RESULTS_H


#include <Adafruit_NeoPixel.h>

#include <Arduino.h>

#include "sound.h"


#define PIN_NEO_PIXEL 15

#define NUM_PIXELS 60     // The number of LEDs (pixels) on NeoPixel LED strip

extern Adafruit_NeoPixel NeoPixel;




void winLED();

void loseLED();

void turnOffLED();

void playBetWin();

void playBetLoss();

void playPositiveChange();

void playNegativeChange();






#endif
```

8.6.2. results.cpp

```
#include "results.h"









void winLED(){
        NeoPixel.clear();
        NeoPixel.fill(NeoPixel.Color(0,255,0), 0, NUM_PIXELS);
        NeoPixel.setBrightness(50);
        NeoPixel.show();
}

void loseLED(){
        NeoPixel.clear();
        NeoPixel.fill(NeoPixel.Color(255,0,0), 0, NUM_PIXELS);
        NeoPixel.setBrightness(50);
        NeoPixel.show();

}

void turnOffLED(){
        NeoPixel.clear();
        NeoPixel.fill(NeoPixel.Color(0,0,0), 0, NUM_PIXELS);
        NeoPixel.show();
}



void playBetWin() {
        Serial.println("BET WIN");
        winLED();
        play(Bruins_Goal_Horn);
        turnOffLED();

}

void playBetLoss() {
        Serial.println("BET LOSS");
        loseLED();
        play(Curb_Loser);
        turnOffLED();

}

void playPositiveChange() {
        Serial.println("BET POSITIVE CHANGE");
        winLED();
        play(Sonic_Ring);
        delay(1000);
        turnOffLED();
}
```

```
void playNegativeChange() {

    Serial.println("NEGATIVE CHANGE");

    loseLED();

    play(Sad_Trambone);

    delay(1000);

    turnOffLED();


}
```

8.7. Integrated System Software Files

8.7.1. main.cpp

```
#include <Arduino.h>

#include <ESP32Lib.h>

#include <Ressources/Font6x8.h>

#include <Ressources/CodePage437_8x8.h>

#include <WiFi.h>

#include <HTTPClient.h>

#include <WiFiClientSecure.h>

#include <ArduinoJson.h>

#include <Adafruit_NeoPixel.h>

#include <vector>

#include <ctime>

#include <map>

#include <string>

#include "results.h"


#define PIN_NEO_PIXEL 15

#define NUM_PIXELS 60


//extern CRGB leds[NUM_PIXELS];

Adafruit_NeoPixel NeoPixel(NUM_PIXELS, PIN_NEO_PIXEL,NEO_GRB + NEO_KHZ800);


WiFiClient client;

HTTPClient http;
```

```
// Sound declarations
//static i2s_port_t i2s_num = I2S_NUM_1; // i2s port number


const i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_TX),
    .sample_rate = 44100, // Note, this will be changed later
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = (i2s_comm_format_t)(I2S_COMM_FORMAT_STAND_I2S),
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1, // high interrupt priority
    .dma_buf_count = 8,                      // 8 buffers
    .dma_buf_len = 1024,                     // 1K per buffer, so 8K of buffer
space
    .use_apll = 0,
    .tx_desc_auto_clear = true,
    .fixed_mclk = -1};


const i2s_pin_config_t pin_config = {
    .bck_io_num = I2S_SPEAKER_SERIAL_CLOCK,
    .ws_io_num = I2S_SPEAKER_LEFT_RIGHT_CLOCK,
    .data_out_num = I2S_SPEAKER_SERIAL_DATA,
    .data_in_num = I2S_PIN_NO_CHANGE};
```

```
const char* ssid = "SDNet";

const char* password = "CapstoneProject";

const char* host = "api.sportradar.us";

const int httpsPort = 80; // Use port 443 for HTTPS

const char* apiKey = "dIf2bFcXi01nQ0rdfcPuUaJDqOMhkHXi4yaGEl13";

int displayIndex = 0; // Index of the first BetSlip to display


void updateBetSlipsFromServer();




class BetSlips {

public:

  int id;

  String displayName;

  String sport;

  String betCategory;

  String betType;

  String playerName;

  String teamName;

  String teamAbbrv;

  String overUnder;

  String hits;

  String spread;

  String RBIs;

  String Strikeouts;

  String homeRuns;
  String walks;

  String displayText;

  String datePlaced;

  double betEntry;

  double betOdds;

  double betWinnings;

  Game* gamePointer;

  int currentStatNumber;

  int desiredStat;

  int bettingScore;

  int opposingScore;

  bool betFinished;

  bool gameOver;

  bool winBet;


BetSlips() {}


BetSlips(const JsonObject& obj) {

    id = obj["id"].as<int>();

    displayName = obj["screen_name"].as<String>();

    sport = obj["sport"].as<String>();

    betCategory = obj["category"].as<String>();

    betType = obj["type"].as<String>();

    if (obj["details"].is<String>()) {

      String details = obj["details"].as<String>();

      DynamicJsonDocument detailsDoc(256);
      deserializeJson(detailsDoc, details);

      JsonObject detailsObj = detailsDoc.as<JsonObject>();

      if (betCategory == "Game Outcomes") {

        teamName = detailsObj["winningTeam"].as<String>();


        int startPos = teamName.indexOf('(');

        int endPos = teamName.indexOf(')');

        if (startPos != -1 && endPos != -1) {

          teamAbbrv = teamName.substring(startPos + 1, endPos);

        }

      } else if (betCategory == "Player Lines") {

        playerName = detailsObj["playerName"].as<String>();


        int startPos = playerName.indexOf('(');

        int endPos = playerName.indexOf(')');

        if (startPos != -1 && endPos != -1) {

          teamAbbrv = playerName.substring(startPos + 1, endPos);

        }

      }

      playerName = detailsObj["playerName"].as<String>();

      teamName = detailsObj["winningTeam"].as<String>();

      overUnder = detailsObj["overUnder"].as<String>();

      hits = detailsObj["hits"].as<String>();

      spread = detailsObj["spread"].as<String>();

      RBIs  = detailsObj["RBIs"].as<String>();

      Strikeouts  = detailsObj["Strikeouts"].as<String>();
```

```
      homeRuns = detailsObj["homeRuns"].as<String>();

      walks = detailsObj["walks"].as<String>();


    }

  displayText = obj["display_text"].as<String>();

  datePlaced = obj["created_at"].as<String>();


    int dollarSignIndex = displayText.indexOf('$');

    int toWinIndex = displayText.indexOf("to win $");




    if (dollarSignIndex != -1 && toWinIndex != -1) {


      int betEntryEndIndex = displayText.indexOf(' ', dollarSignIndex);

      String betEntryString = displayText.substring(dollarSignIndex + 1,
betEntryEndIndex);

      betEntry = betEntryString.toFloat();


      /

      String betWinningsString = displayText.substring(toWinIndex + 8);

      betWinnings = betWinningsString.toFloat();

    }

    gamePointer = NULL;

    currentStatNumber = 0;

    desiredStat = 1;

    bettingScore = 0;

    opposingScore = 0;
    betFinished = false;

    gameOver = false;

    winBet = false;

  }





  void print() {

    Serial.println("Bet Slip:");

    Serial.println("ID: " + id);

    Serial.println("Display Name: " + displayName);

    Serial.println("Sport: " + sport);

    Serial.println("Team Abbreviation: " + teamAbbrv);

    Serial.println("Spread: " + spread);

    Serial.println("Team Name: " + teamName);

    Serial.println("Player Name: " + playerName);

    Serial.println("Bet Category: " + betCategory);

    Serial.println("Bet Type: " + betType);

    Serial.println("Over/Under: " + overUnder);

    Serial.println("hits: " + hits);

    Serial.println("Strikeouts: " + Strikeouts);

    Serial.println("RBIs: " + RBIs);

    Serial.printf("Bet Entry: %f\n", betEntry);

    Serial.printf("Bet Odds: %f\n", betOdds);

    Serial.printf("Bet Winnings: %f\n", betWinnings);

  }
```

```
};

std::vector<BetSlips> betSlips;
std::vector<Game*> allGames = {};

int parseNumberFromString(String input) {
    String numberString = "";
    bool foundNumber = false;

    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c >= '0' && c <= '9') {
            foundNumber = true;
            numberString += c;
        } else if (foundNumber) {
            break;
        }
    }
    int number = numberString.toInt();
    return number;

}

void updateBetSlipsFromServer() {
  // Make the HTTP GET request as done in the setup() function


  http.begin("http://gamblingscreen.com/getdata.php");

  int httpCode = http.GET();


  if (httpCode > 0) {

    String payload = http.getString();

    DynamicJsonDocument doc(2048); // Ensure the document is large enough

    deserializeJson(doc, payload);

    JsonArray array = doc.as<JsonArray>();

    Serial.println("Processing bet slips:");

    for(JsonVariant var : array) {

      JsonObject obj = var.as<JsonObject>();

      BetSlips betSlip(obj);

      Serial.println("BET ID:");

      Serial.println(betSlip.id);

      // Check if this betSlip is already in betSlips

      auto it = std::find_if(betSlips.begin(), betSlips.end(), [&betSlip](const
BetSlips& b) {

        return b.id == betSlip.id;

      });


      // If the betSlip is new, add it to the vector

      if (it == betSlips.end()) {

        betSlips.push_back(betSlip);

        Serial.print("Added new bet slip with ID:");

        Serial.print(betSlip.id);
      }
    }
  } else {
    Serial.printf("HTTP GET request failed, code: %d\n", httpCode);
  }



}

DynamicJsonDocument doc(1024);
//pin configuration
const int redPin = 12;
const int greenPin = 14;
const int bluePin = 27;
const int hsyncPin = 25;
const int vsyncPin = 26;

VGA3BitI vga;
int y;
int indent;



// Function declarations

void displayProgress(int desiredStat, int actualStat, int indent, int x, int y) {
  int circleSpacing = x / desiredStat; // Calculate spacing between circles
  for (int i = 0; i <= desiredStat; i++) {
    int circleX = indent + i * circleSpacing; // Calculate x-coordinate of circle
    uint32_t circleColor;

    // Determine circle color based on actualStat
    if (i < actualStat + 1) {
      circleColor = vga.RGB(0, 255, 0); // Green color for circles with index
less than actualStat
    } else {
      circleColor = vga.RGB(255, 0, 0); // Red color for other circles
    }

    // Draw circle at (circleX, y) with radius 5 and specified color
    vga.fillCircle(circleX, y, 5, circleColor);

    vga.setCursor(circleX - 3, y - 15); // Position text above the circle
    vga.setTextColor(vga.RGB(0xffffff));
    vga.print(i); // Print the number text

  }

  // Draw rectangles to connect circles
  for (int i = 0; i < desiredStat; i++) {
    int circleX = indent + i * circleSpacing; // Calculate x-coordinate of circle
    uint32_t rectangleColor;

    // Determine rectangle color based on the color of the circle to its right
    if (i < actualStat) {
      rectangleColor = vga.RGB(0, 255, 0); // Green color if the next circle is
green
    } else {
      rectangleColor = vga.RGB(255, 0, 0); // Red color if the next circle is red
    }

    // Draw rectangle to connect circles
    vga.fillRect(circleX + 5, y - 2, circleSpacing - 5, 5, rectangleColor);
  }
}

void underDisplayProgress(int desiredStat, int actualStat, int indent, int x, int
y) {
  int circleSpacing = x / desiredStat; // Calculate spacing between circles

  for (int i = 0; i <= desiredStat; i++) {
    int circleX = indent + i * circleSpacing; // Calculate x-coordinate of circle
    uint32_t circleColor;

    // Determine circle color based on actualStat
    if (i < actualStat + 1) {
      circleColor = vga.RGB(255, 0, 0); // red color for circles with index less
than actualStat
    } else {
      circleColor = vga.RGB(0, 255, 0); // green color for other circles
    }

    // Draw circle at (circleX, y) with radius 5 and specified color
    vga.fillCircle(circleX, y, 5, circleColor);

    vga.setCursor(circleX - 3, y - 15); // Position text above the circle
    vga.setTextColor(vga.RGB(0xffffff));
    vga.print(i); // Print the number text

  }

  // Draw rectangles to connect circles
  for (int i = 0; i < desiredStat; i++) {
    int circleX = indent + i * circleSpacing; // Calculate x-coordinate of circle
    uint32_t rectangleColor;

    // Determine rectangle color based on the color of the circle to its right
    if (i < actualStat) {
      rectangleColor = vga.RGB(255, 0, 0); // Green color if the next circle is
green
    } else {
      rectangleColor = vga.RGB(0, 255, 0); // Red color if the next circle is red
    }

    // Draw rectangle to connect circles
    vga.fillRect(circleX + 5, y - 2, circleSpacing - 5, 5, rectangleColor);
  }
```

```
}


void betBox2(BetSlips &betSlip, int x, int y) {


  int boxWidth = 305;

  int boxHeight = 92;

  vga.setFont(CodePage437_8x8);

  vga.setTextColor(vga.RGB(0xffffff)); // Text color




  // Create bet box

  if (betSlip.betFinished == true && betSlip.winBet == true) {

    vga.fillRect(x, y, boxWidth, boxHeight, vga.RGB(0x00ff00));

    vga.fillRect(x + 1, y + 1, boxWidth - 2, boxHeight - 2, vga.RGB(0x000000));
// green if bet win

  }

else if (betSlip.betFinished == true && betSlip.winBet == false) {

    vga.fillRect(x, y, boxWidth, boxHeight, vga.RGB(0x0000ff)); // Background
color

   vga.fillRect(x + 1, y + 1, boxWidth - 2, boxHeight - 2, vga.RGB(0x000000)); //
red if bet loses

  }

  else {

    vga.fillRect(x, y, boxWidth, boxHeight, vga.RGB(0xffffff)); // Background
color

    vga.fillRect(x + 1, y + 1, boxWidth - 2, boxHeight - 2, vga.RGB(0x000000));
// white if bet is ongoing

  }
```

```
vga.setCursor(x + 7, y + 5);



// Display Name
const char* displayNameChar = betSlip.displayName.c_str();
vga.print(displayNameChar);



String teamNameStr = betSlip.teamName;
String playerNameStr = betSlip.playerName;
String betCategoryStr = betSlip.betCategory;
String betTypeStr = betSlip.betType;
String overUnderStr = betSlip.overUnder;
String strikeoutsStr = betSlip.Strikeouts;
String RBIsStr = betSlip.RBIs;
String hitsStr = betSlip.hits;
String homeRunsStr = betSlip.homeRuns;
String walksStr = betSlip.walks;

String datePlaced = betSlip.datePlaced;



int actualStat = 0;
String player;
int index = playerNameStr.indexOf('(');
if (index != -1) {

  player = playerNameStr.substring(0, index-1);

}



  if (betCategoryStr == "Player Lines") {


    vga.setCursor(x + 75, y+5);


    vga.print(playerNameStr.c_str());

    vga.setCursor(x + 75, y+15);

    vga.print(overUnderStr.c_str());

    vga.print(" ");


    if (betSlip.hits != "null") {

      vga.print(hitsStr.c_str());

      vga.print(" ");

      //betSlip.desiredStat = betSlip.hits.toInt();

      betSlip.desiredStat = round(atof(betSlip.hits.c_str()));

    }


    if (betSlip.Strikeouts != "null") {

      vga.print(strikeoutsStr.c_str());

      vga.print(" ");

      //betSlip.desiredStat = betSlip.Strikeouts.toInt();

      betSlip.desiredStat = round(atof(betSlip.Strikeouts.c_str()));
    }



    if (betSlip.RBIs != "null") {
      vga.print(RBIsStr.c_str());
      vga.print(" ");
      //betSlip.desiredStat = betSlip.RBIs.toInt();
      betSlip.desiredStat = round(atof(betSlip.RBIs.c_str()));
    }
    if (betSlip.homeRuns != "null") {
      vga.print(homeRunsStr.c_str());
      vga.print(" ");
      //betSlip.desiredStat = betSlip.homeRuns.toInt();
      betSlip.desiredStat = round(atof(betSlip.homeRuns.c_str()));
    }
    if (betSlip.walks != "null") {
      vga.print(walksStr.c_str());
      vga.print(" ");
      //betSlip.desiredStat = betSlip.walks.toInt();
      betSlip.desiredStat = round(atof(betSlip.walks.c_str()));
    }

    vga.print(betTypeStr.c_str());

  actualStat = 0;
```

```
}
    else{
      vga.setCursor(x + 75, y+5);
      vga.print(teamNameStr.c_str());
      vga.setCursor(x + 75, y+15);
      vga.print(betTypeStr.c_str());
      if(betSlip.spread != "null") { // if spread
          vga.setCursor(x + 75, y+15);
          vga.print(betTypeStr.c_str());
          vga.print(" ");
          if(betSlip.spread.toInt() > 0) { // if positive spread
            vga.print("+");
            vga.print(betSlip.spread.toInt());
            //betSlip.desiredStat = betSlip.spread.toInt();
            betSlip.desiredStat = round(atof(betSlip.spread.c_str()));
          }
          else {
          vga.print(betSlip.spread.toInt()); // if negative spread
          //betSlip.desiredStat = betSlip.spread.toInt();
          betSlip.desiredStat = round(atof(betSlip.spread.c_str()));
          }
      }
      else { // If Moneyline
        vga.setCursor(x + 75, y+15);
        vga.print(betTypeStr.c_str());
      }
    actualStat = 0;



    }


    vga.setCursor(x+15,y+60);
  vga.print("$");
  vga.print(betSlip.betEntry);
  vga.setCursor(x+15,y+70);
  vga.print("to win $");
  vga.print(betSlip.betWinnings);







    Serial.println("about to try initializing games");
      String tempTeamName = betSlip.teamAbbrv;
      Serial.println(tempTeamName);
      Game* tempGame;
      bool teamFound = false;
      if(allGames.size() == 0){
        Serial.println("allGames is empty");
          tempGame = init_game(tempTeamName, apiKey);
          while(!update_game(tempGame, apiKey)){delay(500);}
```

```
          allGames.push_back(tempGame);

          Serial.println("added frist game");

      }

      for (int gameIter = 0; gameIter < allGames.size(); gameIter++){

        //Serial.println("In first For loop");


        if((allGames[gameIter]->getHomeName() == tempTeamName ||
allGames[gameIter]->getAwayName() == tempTeamName)){

          teamFound = true;

        }


      }

      if(!teamFound){

        tempGame = init_game(tempTeamName, apiKey);

        while(!update_game(tempGame, apiKey)){delay(500);}

        allGames.push_back(tempGame);

      }


      for (int gameIter = 0; gameIter < allGames.size(); gameIter++){

        //Serial.println("In second for loop");

        if(betSlip.gamePointer == NULL && (allGames[gameIter]->getHomeName() ==
tempTeamName || allGames[gameIter]->getAwayName() == tempTeamName)){

            betSlip.gamePointer = allGames[gameIter];

            Serial.println("added gamePointer to betSlip");

        }


      }
```

```
   //Serial.println("If this is the only message, its so over :(");

   Game* currGame = betSlip.gamePointer;



  Serial.println("Player Name: " + playerNameStr + ", team name:" +
tempTeamName);

   String testDisplay = get_new_stat(currGame, "Score", playerNameStr,
tempTeamName);


  Serial.println(testDisplay);




String inning;

  String bettingTeam;

int bettingTeamScore;

String opposingTeam;

int opposingTeamScore;


int inningEnd = testDisplay.indexOf(",");


// Indices for extracting "Betting Team"

int bettingTeamStart = testDisplay.indexOf("Betting Team: ") + 14; // Start after
"Betting Team: "

int bettingTeamEnd = testDisplay.indexOf(" -", bettingTeamStart);
```

```
// Indices for extracting "Opposing Team"

int opposingTeamStart = testDisplay.indexOf("Other Team: ") + 12; // Start after
"Other Team: "

int opposingTeamEnd = testDisplay.indexOf(" -", opposingTeamStart);


// Extract team names

inning = testDisplay.substring(0, inningEnd);

Serial.println("Inning string");

Serial.println(inning);

bettingTeam = testDisplay.substring(bettingTeamStart, bettingTeamEnd);

opposingTeam = testDisplay.substring(opposingTeamStart, opposingTeamEnd);


// Indices for extracting scores

int bettingTeamScoreStart = testDisplay.indexOf("- ", bettingTeamEnd) + 2;

int bettingTeamScoreEnd = testDisplay.indexOf(",", bettingTeamScoreStart);

int opposingTeamScoreStart = testDisplay.lastIndexOf("- ") + 2;


// Extract scores as strings

String bettingTeamScoreStr = testDisplay.substring(bettingTeamScoreStart,
bettingTeamScoreEnd);

String opposingTeamScoreStr = testDisplay.substring(opposingTeamScoreStart);


// Convert score strings to integers

bettingTeamScore = bettingTeamScoreStr.toInt();

opposingTeamScore = opposingTeamScoreStr.toInt();


  vga.setCursor(x+175,y+60);
  vga.print(bettingTeam.c_str());

  vga.print(" - ");

  vga.print(bettingTeamScore);

  vga.setCursor(x+175,y+70);

  vga.print(opposingTeam.c_str());

  vga.print(" - ");

  vga.print(opposingTeamScore);


  vga.setCursor(x+250, y+60);

  vga.print(inning.c_str());

  vga.setCursor(x+250, y+70);

  String currOuts = "Outs:" + String(currGame->getOuts());

  vga.print(currOuts.c_str());





   //betSlip.gameOver = tempGame.checkProgress();
//////////////////////////////////////////////////////////////// GAME PROGRESS


  if (betCategoryStr == "Game Outcomes" && betTypeStr == "Moneyline") {


      if(bettingTeamScore > opposingTeamScore) { //Sets bar for MoneyLine if team
is winning

        actualStat = actualStat;

        displayProgress(1, 1,  18, (boxWidth-30), y+45);

      }

      else {
        actualStat = 0;

        displayProgress(1, actualStat,  18, (boxWidth-30), y+45);

      }



      if (bettingTeamScore > betSlip.bettingScore && opposingTeamScore ==
betSlip.opposingScore) {


          playPositiveChange(); // play positive change and update scores

          betSlip.bettingScore = bettingTeamScore;

          betSlip.opposingScore = opposingTeamScore;

      }

      else if (opposingTeamScore > betSlip.opposingScore && bettingTeamScore ==
betSlip.bettingScore) {


          playNegativeChange(); // play negative change and update scores

          betSlip.bettingScore = bettingTeamScore;

          betSlip.opposingScore = opposingTeamScore;

      }

      else {

          betSlip.bettingScore = bettingTeamScore; //update scores regardless

          betSlip.opposingScore = opposingTeamScore;

      }




      if (betSlip.gameOver == true && actualStat == 1 && betSlip.betFinished ==
false) {

        playBetWin();

        betSlip.betFinished = true; // updates betslip to finish so sound is not
played again


      }

      else if (betSlip.gameOver == true && actualStat == 0 && betSlip.betFinished
== false) {


        playBetLoss();

        betSlip.betFinished = true; // updates betslip to finish so sound is not
played again

      }


  }




  else if (betCategoryStr == "Player Lines" && betTypeStr == "Hits") {

    String hitsDisplay = get_new_stat(currGame, "Hits", player,
betSlip.teamAbbrv);

    actualStat = parseNumberFromString(hitsDisplay);


     if (actualStat == betSlip.desiredStat && overUnderStr == "Over" &&
betSlip.betFinished == false) { // If a over bet hits for the first time

        betSlip.betFinished = true;

        betSlip.winBet = true;

        betSlip.currentStatNumber = actualStat;

        playBetWin();

     }
    else if (actualStat == betSlip.desiredStat && overUnderStr  == "Under" &&
betSlip.betFinished == false) { // if an under bet fails for the first time
        betSlip.betFinished = true;
        betSlip.winBet = false;
        betSlip.currentStatNumber = actualStat;
        playBetLoss();
    }
    else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Over") {
// If stat occurs and over bet (POSITIVE)
        betSlip.currentStatNumber = actualStat;
        playPositiveChange();
    }
    else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Under")
{ // If stat occurs in under bet (BAD)
        betSlip.currentStatNumber = actualStat;
        playNegativeChange();
    }
    else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Over") { // If game ends without over hitting
        betSlip.betFinished = true;
        playBetLoss();
    }
    else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Under") { // If game ends with under hitting
        betSlip.betFinished = true;
        betSlip.winBet = true;
        playBetWin();
    }
```

```
     else {
        betSlip.currentStatNumber = actualStat;
     }









  }
  else if (betCategoryStr == "Player Lines" && betTypeStr == "Strikeouts") {
    String strikeoutDisplay = get_new_stat(currGame, "Strikeouts", player,
betSlip.teamAbbrv);
    actualStat = parseNumberFromString(strikeoutDisplay);

    if (actualStat == betSlip.desiredStat && overUnderStr == "Over" &&
betSlip.betFinished == false) { // If a over bet hits for the first time
        betSlip.betFinished = true;
        betSlip.winBet = true;
        betSlip.currentStatNumber = actualStat;
        playBetWin();
     }
     else if (actualStat == betSlip.desiredStat && overUnderStr  == "Under" &&
betSlip.betFinished == false) { // if an under bet fails for the first time
        betSlip.betFinished = true;
        betSlip.winBet = false;
        betSlip.currentStatNumber = actualStat;
        playBetLoss();

     }

     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Over") {
// If stat occurs and over bet (POSITIVE)

        betSlip.currentStatNumber = actualStat;

        playPositiveChange();

     }

     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Under")
{ // If stat occurs in under bet (BAD)

        betSlip.currentStatNumber = actualStat;

        playNegativeChange();

     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Over") { // If game ends without over hitting

        betSlip.betFinished = true;

        playBetLoss();

     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Under") { // If game ends with under hitting

        betSlip.betFinished = true;

        betSlip.winBet = true;

        playBetWin();

     }

     else {

        betSlip.currentStatNumber = actualStat;

     }


  }
  else if (betCategoryStr == "Player Lines" && betTypeStr == "RBIs") {
    String rbiDisplay = get_new_stat(currGame, "RBIs", player,
betSlip.teamAbbrv);
    actualStat = parseNumberFromString(rbiDisplay);

    if (actualStat == betSlip.desiredStat && overUnderStr == "Over" &&
betSlip.betFinished == false) { // If a over bet hits for the first time
        betSlip.betFinished = true;
        betSlip.winBet = true;
        betSlip.currentStatNumber = actualStat;
        playBetWin();
     }
     else if (actualStat == betSlip.desiredStat && overUnderStr  == "Under" &&
betSlip.betFinished == false) { // if an under bet fails for the first time
        betSlip.betFinished = true;
        betSlip.winBet = false;
        betSlip.currentStatNumber = actualStat;
        playBetLoss();
     }
     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Over") {
// If stat occurs and over bet (POSITIVE)
        betSlip.currentStatNumber = actualStat;
        playPositiveChange();
     }
     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Under")
{ // If stat occurs in under bet (BAD)
        betSlip.currentStatNumber = actualStat;
        playNegativeChange();
     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Over") { // If game ends without over hitting

        betSlip.betFinished = true;

        playBetLoss();

     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Under") { // If game ends with under hitting

        betSlip.betFinished = true;

        betSlip.winBet = true;

        playBetWin();

     }

     else {

        betSlip.currentStatNumber = actualStat;

     }


  }

    else if (betCategoryStr == "Player Lines" && betTypeStr == "Home Runs") {

    String homeRunDisplay = get_new_stat(currGame, "Home Runs", player,
betSlip.teamAbbrv);

    actualStat = parseNumberFromString(homeRunDisplay);


    if (actualStat == betSlip.desiredStat && overUnderStr == "Over" &&
betSlip.betFinished == false) { // If a over bet hits for the first time

        betSlip.betFinished = true;

        betSlip.winBet = true;

        betSlip.currentStatNumber = actualStat;

        playBetWin();
     }

     else if (actualStat == betSlip.desiredStat && overUnderStr  == "Under" &&
betSlip.betFinished == false) { // if an under bet fails for the first time

        betSlip.betFinished = true;

        betSlip.winBet = false;

        betSlip.currentStatNumber = actualStat;

        playBetLoss();

     }

     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Over") {
// If stat occurs and over bet (POSITIVE)

        betSlip.currentStatNumber = actualStat;

        playPositiveChange();

     }

     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Under")
{ // If stat occurs in under bet (BAD)

        betSlip.currentStatNumber = actualStat;

        playNegativeChange();

     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Over") { // If game ends without over hitting

        betSlip.betFinished = true;

        playBetLoss();

     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Under") { // If game ends with under hitting

        betSlip.betFinished = true;

        betSlip.winBet = true;

        playBetWin();
     }
     else {
        betSlip.currentStatNumber = actualStat;
     }
  }
    else if (betCategoryStr == "Player Lines" && betTypeStr == "Walks") {
    String walksDisplay = get_new_stat(currGame, "Walks", player,
betSlip.teamAbbrv);
    actualStat = parseNumberFromString(walksDisplay);


    if (actualStat == betSlip.desiredStat && overUnderStr == "Over" &&
betSlip.betFinished == false) { // If a over bet hits for the first time
        betSlip.betFinished = true;
        betSlip.winBet = true;
        betSlip.currentStatNumber = actualStat;
        displayProgress(betSlip.desiredStat, betSlip.currentStatNumber, 18,
(boxWidth-30), y+45);
        playBetWin();
     }
     else if (actualStat == betSlip.desiredStat && overUnderStr  == "Under" &&
betSlip.betFinished == false) { // if an under bet fails for the first time
        betSlip.betFinished = true;
        betSlip.winBet = false;
        betSlip.currentStatNumber = actualStat;
        underDisplayProgress(betSlip.desiredStat, betSlip.currentStatNumber, 18,
(boxWidth-30), y+45);
        playBetLoss();
     }
     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Over") {
// If stat occurs and over bet (POSITIVE)

        betSlip.currentStatNumber = actualStat;

        displayProgress(betSlip.desiredStat, betSlip.currentStatNumber, 18,
(boxWidth-30), y+45);

        playPositiveChange();

     }

     else if (actualStat > betSlip.currentStatNumber && overUnderStr == "Under")
{ // If stat occurs in under bet (BAD)

        betSlip.currentStatNumber = actualStat;

        underDisplayProgress(betSlip.desiredStat, betSlip.currentStatNumber, 18,
(boxWidth-30), y+45);

        playNegativeChange();

     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Over") { // If game ends without over hitting

        betSlip.betFinished = true;

        displayProgress(betSlip.desiredStat, betSlip.currentStatNumber, 18,
(boxWidth-30), y+45);

        playBetLoss();

     }

     else if (betSlip.gameOver == true && actualStat < betSlip.desiredStat &&
overUnderStr == "Under") { // If game ends with under hitting

        betSlip.betFinished = true;

        betSlip.winBet = true;

        underDisplayProgress(betSlip.desiredStat, betSlip.currentStatNumber, 18,
(boxWidth-30), y+45);

        playBetWin();

     }
     else {
        betSlip.currentStatNumber = actualStat;
     }
  }
  else if (betCategoryStr == "Game Outcomes" && betTypeStr == "Spread") {
    if (betSlip.desiredStat < 0) { // If - Spread
      actualStat = bettingTeamScore - opposingTeamScore; // winning by less than
spread
      Serial.println(betTypeStr);
      Serial.println(actualStat);
      if (actualStat > -1*betSlip.desiredStat) { // winning by more than spread
        actualStat = -1*betSlip.desiredStat;
      }
      else if (actualStat < 0) { // if losing
        actualStat = 0;
      }
    }
    else { // If +Spread
      actualStat = bettingTeamScore - opposingTeamScore;

      if (actualStat > 0) { // max bar if winning by enough
        actualStat = betSlip.desiredStat;
      }
      else if (bettingTeamScore < opposingTeamScore && actualStat*-1 <
betSlip.desiredStat) { // losing by less than spread
        actualStat = opposingTeamScore - bettingTeamScore;
    }
    else { // losing by more than spread

        actualStat = 0;


    }

  }

  }


  if (betSlip.desiredStat < 0 && betSlip.overUnder == "Over") {

      int finalNumber = betSlip.desiredStat*-1;

      displayProgress(betSlip.desiredStat*-1, actualStat, 18, (boxWidth-30),
y+45);

  }

  else if (betSlip.overUnder == "Over"){

      int finalNumber = betSlip.desiredStat;

      displayProgress(betSlip.desiredStat, actualStat, 18, (boxWidth-30), y+45);

  }

  else if (betSlip.overUnder == "Under"){

      int finalNumber = betSlip.desiredStat;

      underDisplayProgress(betSlip.desiredStat, actualStat, 18, (boxWidth-30),
y+45);

  }





}


void setup() {
```

```
Serial.begin(115200);

Serial.println(esp_get_free_heap_size());

vga.init(vga.MODE320x200, redPin, greenPin, bluePin, hsyncPin, vsyncPin);

  Serial.println(esp_get_free_heap_size());


    i2s_driver_install(i2s_num, &i2s_config, 0, NULL);

    i2s_set_pin(i2s_num, &pin_config);


  WiFi.begin(ssid, password);


  Serial.println("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED) {

    delay(1000);

    Serial.print(".");

  }

  Serial.println("Connected to WiFi");

  configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");

  delay(2000);


  // Define the URL to read from

  const char* url = "http://gamblingscreen.com/getdata.php";

  NeoPixel.begin();


}


void loop() {
    Serial.println("before updating from server");

    updateBetSlipsFromServer();

    Serial.println("after updating from server");


    // Clear the screen before displaying new BetSlips to avoid overwriting text

    vga.clear();


    // Display BetSlips at specified positions

    if(betSlips.size() < 2){

      Serial.println("looping if less than 2 bets");

      for (int i = 0; i < betSlips.size(); i++) {

        betBox2(betSlips[i], 2, 5);

      }

    }

    else{

    for (int i = 0; i < 2; i++) {

      Serial.println("looping if enough bets");

      int index = (displayIndex + i) % betSlips.size(); // Ensure the index wraps
around

      if (!betSlips.empty()) { // Check to prevent division by zero

        if (i == 0) betBox2(betSlips[index], 2, 5);

        else if (i == 1) betBox2(betSlips[index], 2, 102);

      }

    }

    }

    delay(2000);
    for (int iter = 0; iter < allGames.size(); iter++){

      while (!update_game(allGames[iter], apiKey)){delay(500);}

    }
    delay(10000);

    // Increment the displayIndex by 2 to prepare for the next group of BetSlips
    displayIndex += 2;
    if (displayIndex >= betSlips.size()) displayIndex = 0; // Wrap around if
necessary
  //}

  // Include any other loop logic here
}
```