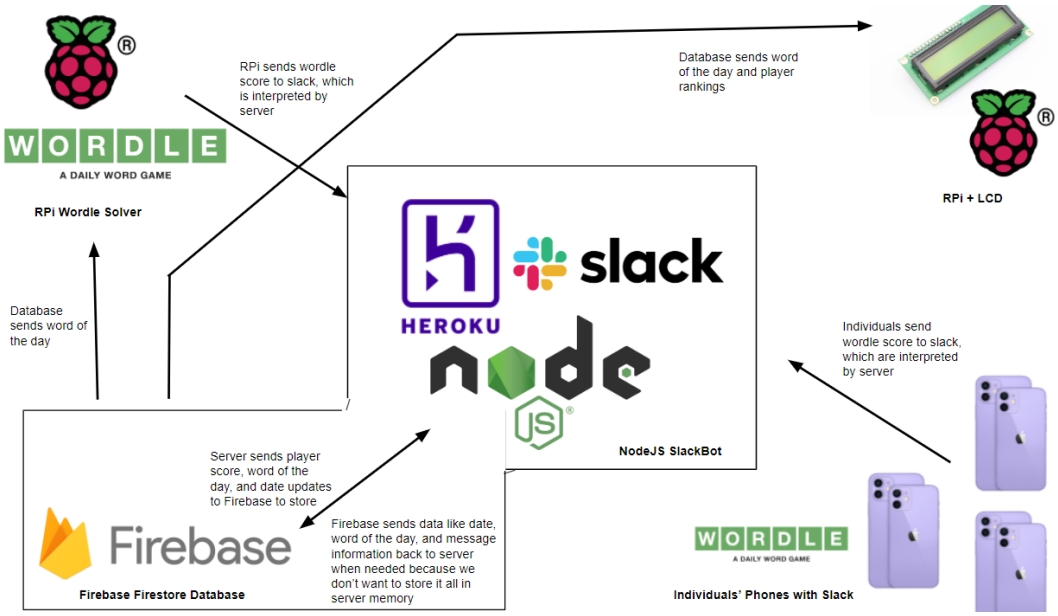
Team Members: Max Ginsberg & Riyana Gobin
Github Links:
https://github.com/usc-ee250-spring2022/lab2-max-riyana/tree/master/ee250/finalProj
mginsy/EE250WordleServer (github.com)

Final Project: Wordle Bot(s)

Description

Our project consists of three nodes. First, it has a virtual server, SlackBot, in the cloud which pulls Wordle scores from a club's Slack channel and stores them in a Firestore database (maybe a fourth node?). Then, there are two physical nodes: a bot that plays the daily Wordle and is capable of posting to the slack or terminal, and a raspberry pi that gets the scores of all our friends and displays them on an LCD screen. After these rankings are displayed, the word of the day will be displayed along with the meaning of the word using a WordAPI. Theoretically, physical nodes can span across every member in our slack channel as all of these people can interact with our SlackBot (fifth++++ nodes?????? :D).

Block Diagram



Description of components, platforms, protocols used, and processing/visualization techniques

Slackbot: The slackbot reacts to any message sent in a slack channel and can identify whether a valid command (starts with !) or a wordle score is sent. Different options for commands are things like !help, !init, or !leaderboard. These commands do things like direct users to create their accounts or view the current leaderboard. When a user's Wordle message is copy pasted and posted into the chat, the bot is able to parse it and record this data in Firebase categorized with their slack UserID. It recalculates their average score and saves this as well. It is also able to send a message every day at midnight congratulating the winners of the day (who had the

lowest score that day). It is also able to check if the day is not the same day as the day before and update its day-to-day parameters (daily scores, word of the day, etc.). One of these parameters, the word of the day, is obtained by opening a headless instance of chrome in Javascript and sending a command in the real Wordle website set to PST to obtain the new word. This new word is sent to the Firebase for storage that day. The Slack and Firebase APIs, which are doing the communication among all nodes, use HTTP to function.

Raspberry Pi Leaderboard: The RPi pulls the leaderboard of the game from the database. Students are able to send their scores into a slack channel and the Slackbot will store this information in this database, which can create a leaderboard that consists of everyone's names and accumulative scores over time. This will include the score of our Wordle Bot/Algorithm, named ShahinNazarian on the leaderboard. With the press of a button, the leader board will be iterated and displayed on the LCD screen, showing the person's name and score. Once the entire leaderboard is displayed, the word of the day will then be displayed as well as the meaning of the word. The meaning of the word was acquired through WordsAPI over the course of three days. We did not want to pay for the API, so we saved all the data we needed locally in a file called "words_data.txt" and reference that document now instead to get the data for free.

Wordle Solver Bot: The wordle solver first asks if you want to initialize or solve. Init is usually for admins in case of issue or startup, so solve is usually chosen. It first obtains the word of the day from Firebase. Then it starts with the word "Roate", which was mathematically calculated to be the most effective first word. Then, the bot has two choices: solve or fish. When solving, it is able to determine every possible solvable word left, then selects which of these choices would be the most statistically probable based on letter and position data. A score is assigned to this word and it is the highest scoring among all possibilities for solving. The bot can also fish: this is done by determining what word would give the most possible data in terms of letters (not position) based on what letters have already been confirmed and what words may be left. For example, if we know a word ends in -atch, it would be inefficient to guess Batch, Latch, Catch, then Patch. It would be more efficient to guess BLIMP, which would effectively guess all of them at the same time. A score is assigned to the best fishing word, then the fish score is compared to the solve score. The highest scoring word is chosen as the guess. The wordle bot is then able to compare the guess to the true word and give feedback, which will eliminate possible guesses. The list for possible fish guesses is every 5 letter english word, while the possible list for solves is every possible wordle word. This second list gets smaller over time as more information is gained.

Reflection, discussion of limitations that demonstrates insights to their cause and possible remediation, lessons learned

There was so much learned in this project. NodeJS, Javascript, Python, opening your own instance of chrome, AI, communication between programs and the internet, the list truly goes on. There were so many limitations that we overcome that became lessons, however, some of limitations of this project include graphical (we only print results to terminal), difference of timezone (only works perfectly accurately if everyone is on PST), and error logging (we can only read errors if logged into server when error occurs).