Title: Word Guessing Game
Creators: Group 1 (Beatriz Glaser, Thuy Määttä, Elizabeth Witney, Kabir Bissessar, Sai Deepanker Kanjarla, Yeonwoo Yook, Rafal Ciechanski, Montero de la Plaza Jaime, Lim Harvey)
Course: Industrial Training

Technical Documentation

General Description:
The Word Guessing Game is an online web-based word game, where the user has five guesses to guess the word of the day. The word of the day is randomly selected from a list that consists of random five letter words from the English language. The game is run by and can be played via mobile devices if the sim card is connected to the server.

System Requirements:
The server needs to be an Ubuntu server with Python 3.7 installed. The recommended system requirements are:

- CPU: Minimum 1GHz
- RAM: Minimum 1GB
- Disk: Minimum 2.5GB

Required Packages:

- Datetime (Pre-installed with Python 3.7)
- Random (Pre-installed with Python 3.7)
- Flask (Must be installed using pip via the command line interface)

Structure:
The program runs off five different functions working together and then are called, these being…

- word_picker: selects a 'word of the day' from the list in the text file word_list.txt
- word_checker: checks if the input given by the user is correct and returns a string that indicates which letters are correct. It also ensures the input is of correct length and returns an error message if not.
- make_wordfile: create the word_list.txt file with valid (5 letter words) words. It reads words from a longer file, selects the valid ones and writes them on the new .txt file.
- index: set up the web-page of the game.
- test: run the main game. Saves user inputs, calls word_checker function and counts the number of guesses.

Algorithms:
Word checking algorithm
Each time a user submits a guess, the function that creates the feedback string first uses the python *datetime* package to get the current date. It then merges the day, month and year of the current date into one large number (that is, if the day is the 26th of May, 2022, it would turn the date into the number 26052022). This number is used as a seed for the python *random* package to randomly pick a word among the list of words given in the word_list.txt file, which contains

every five letter word in the English language. Because the seed changes every day, the randomizer will only pick a different word once the date has changed, thus giving the user the perception of a ‘word of the day’.

As for the feedback string, the function starts with a list of five ‘X’ characters. The first step it does is to cross-reference the word of the day with the user’s guess, and immediately changes any ‘X’ characters ‘+’ characters **if** there are letters in the user’s guess which are correct and in the right position. Once the correct letters have been identified, they are removed from the user’s guess and from a copy of the word of the day. Next, the function goes through every letter that remains in the guess, and if they are present in the copy of the word of the day, it replaces the corresponding letter in the feedback string with ‘-’ and deletes the letter from both the guess and the word of the day. Once all remaining letters have been gone through, the function returns the feedback string.

Files:
We utilize a text file named word_list.txt in our program which holds a list of different 5 letter words that will then be randomly selected on a daily rotation as a ‘word of the day’.

Product Highlights:
- Simplistic User Interaction
- Rotates the choses word from the word_list.txt text file to perceive a ‘word of the day’
- Instant and clear user feedback
- Limit to 5 guesses, adding to the user ‘game-like’ experience

Terminology:
CPU: Central Processing Unit
RAM: Random Access Memory
Disk: Storage drive for the computer
Datetime: Python library used to track dates and times as objects
Random: Python inbuilt library used to get random values (i.e. randomizer)
Seed: A number that is used to generate a sequence of random numbers.
Flask: Python module used for back-end development
Front-end development: development of a user graphical interface of a website
Back-end development: development of the server and the communication between the database and website/browser

Obstacles:
This experience was new to us as none of our team members had done such a big project nor used agile methodology before. For that reason, it was hard at first to estimate workload and create tasks. Our first and second sprints had to be mainly focused on research so we could understand what we were trying to create. A lot of this research raised more questions and confusion. This also led to poor team communication. We had trouble explaining and deciding

what should be done, because we were not familiar with our tasks and terminology in general. This often led to team members working individually, which was not the most efficient workflow. We also had to face obstacles regarding the technical aspects of the work. The project development required a lot of trial & error and research. This was more time consuming than expected.

Outcomes:
During this project we overcame the previously explained obstacles. Throughout the sprints, our communication improved significantly as the work became more familiar to us. This resulted in higher efficiency and better division of workload. We also had a better idea of how to estimate tasks and effort.
Throughout this project, we learned and now fully understand the connection between SIM cards and a server. Handling both the agile methodology and technical aspects was a great opportunity to experience the practical development of a real-life project.