Development Plan

Title: Words, s.a.m.s (team name)
The code will be written in C, and implemented in a GUI.

High Level Description:

What kind of app, what will it do, how will the user use it?
Based on the New York Times game Wordle, our style inspired game, Words, is built in C with a Graphical User Interface. We are aiming to have a similar visual representation, such as the display and colour coordination. However, our game will allow the user to choose the length of the word ranging from 5 to 8 letters. Depending on the number of letters the user chooses the amount of guesses they will have; if they don't choose, it'll default to 6. A list of words will be held in text files depending on the length of the word. Later a word will randomly be chosen from the list, for the user to guess. The user will use it by launching it from their computer and be presented with a menu.

What is the context in which this app will be used (i.e., characteristics of the expected users and the environment in which they will use it)?
The game is designed for everyone for their own enjoyment, regardless of their skill set. As well, this game can help users refresh and expand their vocabulary in a fun approach. It can be for those who enjoy puzzles and simple games. The users will be able to play this game on their own computers and in any environment.

What constraints are there on your design (i.e., performance, memory, user interface, development time and resources, etc.)?

- Technical knowledge:
    - As our group does not have any experience with GUI's, our final program may not completely align with what we have envisioned. By what we can learn in the time being, along with testing, there may be adjustments on what features are included.
- Time constraints:
    - As there is a time schedule for when the program is due, combined with the GUI hopeful representation, it will determine how our GUI will appear visually.
    - If the C code is not complete and does not fit all the standards we wish, we will not be able to implement the GUI.
        - We rather have stronger code, than a mediocre GUI.

Team members (names, student numbers, GitHub ids, and role titles):

- Sriha Kanthasamy, 400571386, sriha-k, **Text File Lead**
- Michael Mondaini, 400591695, SharkieBite, **Checking Correct Placement Lead**
- Sabrina Leung, 400575507, BrinaLeung, **Checking Repeating word, and Choosing word Lead**
- Axell Panganiban, 400588161, axell8560, **Letter in Word Lead**

Team Roles and Deliverables:

- Sriha: Check if the word exist in the list (function)
    - The checkWord function is dependent on which word length the user wants in Words. Depending on the chosen length, the text file with words of the specified length will be opened. From here, the guess that the user enters, will be checked against the entire text file, to ensure if that word exists; feedback will be displayed. From here, after the other functions are created, the rest of the authentications on the user guess will be validated. An altercation that will be made to this function is to use another function to ensure a valid length of word is chosen, guaranteeing an error of opening a text file does not occur.
- Michael: Correct letter, correct placement, fully correct (function)
    - My role, Checking Correct Placement Lead, is responsible for checking correct letter placement within the program. So far in my role I have designed the checkWordPlacement function which checks if the user's letters in their guess are in the correct placement as in the word. If the user letter is in the correct placement as the word, the function will output the character, "G", representing the green colour that would be displayed. This function will later be modified to return a value instead of outputting a value, so it can be used by other functions in the program.
- Sabrina: Check if repetitive word (function), Read text file, and randomly choose a word in text file function
    - checkRepeat function takes a 2D array holding the previous guesses and compares the current guess to the previous guesses to see if the user has guessed the word before. It returns true or false based on the results.
    - chooseWord function takes an integer, the number of letters in the word, and opens the appropriate file. It randomly selects a file and returns the chosen word.
- Axell: If the letter is in the word (right letter, wrong placement)
    - checkLetter function checks if the guess is the right length, and if it is, checks if each letter in the word. Prints letters as yellow if the letter is in the word but wrong placement, prints letters as red if the letter isn't in the word. It also takes in the number of guesses the user has as the argument, and if the number of guesses is used up, you lose.
- Everyone:
    - Main class:
        - Intro message
        - Help flag
        - Ask user for number of letters in the word they want to guess, if no choice default to 5
        - Ask the user for the number of guesses they would like, if no response default to 6
        - Call the specific functions to check the guess
        - Output the answers
        - And end game

    - Pipeline:
        - Test cases, individual for each functions created by members
        - Test cases for the whole game all together
        - Build

The Increments:

- First increment
    - Every group member has their own individual functions to complete. This is determined by splitting up the different functional necessities and making programs to target them.
    - **Done by November 12, 2025, lab 9.2.**
- Second increment
    - Main
        - Put all the functions together, to get a working game. Make adjustments based on what the outputs.
    - Add flags to individual functions as well as the entire game.
    - Create a README.md
    - **Done by November 26, 2025; increment delivered in lab 11.2.**
- Setting up the pipeline
    - Create a functional, working build for the pipeline.
    - Build test cases for individual functions, as well as the entirety of the game and ensure they are all passed.
    - Update all necessary files and plans.
    - **Done by December 3, 2025, lab 12.2**