Software Engineering Assignment

By Jeremy Nguyen

Table of Contents:

Project Overview:

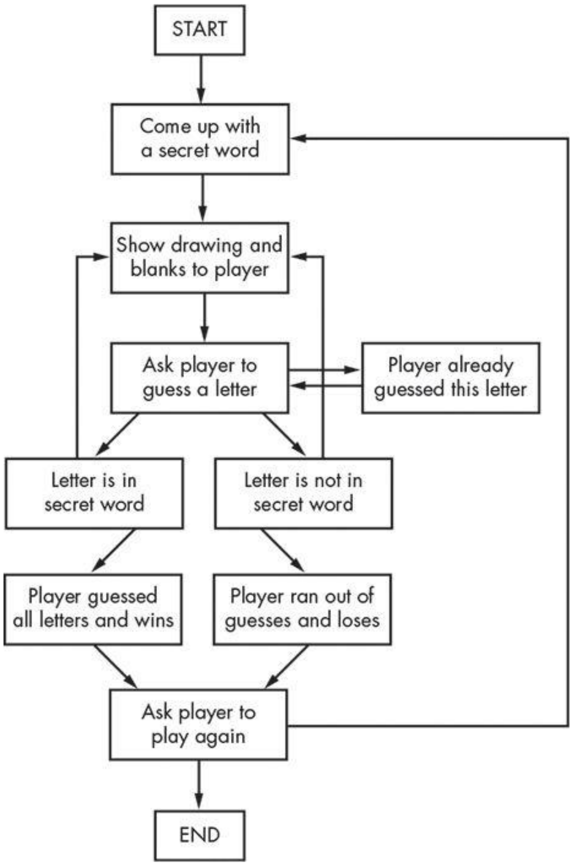
Development of a Hangman game using Python and Pygame. The game includes graphical elements, word selection with hints, and a scoring/reset system.

Development Log:

Day	**Task**	**Details**
Feb 27	Initial Setup	Installed Pygame and set up the display window. Defined game variables (window size, fonts, colors, and word dictionary).
Feb 29	UI and Button Setup	Created letter buttons using a loop and stored their positions. Designed fonts for title, word display, and hints.

Mar 1	Game Mechanics	Implemented word selection and hint display. Developed guessing logic, updating guessed letters and tracking mistakes.
Mar 5	Drawing Functions	Implemented functions to display letters, score, and Hangman images. Added hint and reset buttons with interactive functionality.
Mar 13	End Game Scenarios	Created win/lose conditions with an end screen. Implemented the ability to reset the game upon completion.
Mar 16	Testing & Bug Fixes	Fixed letter visibility issues after guessing. Adjusted UI layout for better readability. Ensured reset functionality properly restores game state.
Mar 20	Final Adjustments & Documentation	Improved button hit detection accuracy. Enhanced hint display and game feedback. Documented the code and completed this logbook.

Flowchart:


START
Come up with a secret word
Show drawing and blanks to player
Ask player to guess a letter
Player already guessed this letter
Letter is in secret word
Letter is not in secret word
Player guessed all letters and wins
Player ran out of guesses and loses
Ask player to play again
END

Modifications:

- **Hint System** – You added a hint feature that allows players to request a clue about the word. This enhances gameplay by aiding when players are stuck.
- **Scoring System** – A scoring mechanism was introduced where players earn points when they correctly guess a word. This adds an incentive for better performance.
- **Reset Feature with Limit** – Players can reset the game a limited number of times (tracked with reset_count), adding a strategic element to gameplay.
- **Improved UI & Interaction** – Buttons for hints and resets were added to the interface, along with graphical enhancements like font styling and word displays.

- **End Screen** – Instead of immediately restarting, the game now displays a win/lose screen, informing players about the outcome and the correct word.

Game Algorithm (Pseudocode):

```
// Initialize game variables
SET wordList = [list of words]
SET secretWord = RANDOM_CHOICE(wordList)
SET attemptsRemaining = 6
SET guessedLetters = []
SET wordCompletion = "_" * LENGTH(secretWord) // Initialize with underscores

// Game loop
WHILE attemptsRemaining > 0 AND wordCompletion != secretWord DO
  // Display current state
  DISPLAY "Word: " + wordCompletion
  DISPLAY "Attempts remaining: " + attemptsRemaining
  DISPLAY "Guessed letters: " + guessedLetters

  // Get user input
  GET userGuess FROM USER

  // Validate input
  IF userGuess IS NOT A SINGLE LETTER OR userGuess IS ALREADY IN guessedLetters THEN
    DISPLAY "Invalid input. Please try again."
    CONTINUE // Skip to the next iteration of the loop
  ENDIF

  // Add the guess to the list of guessed letters
  ADD userGuess TO guessedLetters

  // Check if the guess is correct
  IF userGuess IS IN secretWord THEN
    // Update wordCompletion
    FOR each position in secretWord DO
      IF userGuess IS AT THIS POSITION THEN
        SET wordCompletion[position] = userGuess
      ENDIF
    ENDFOR
  ELSE
    // Decrement attempts remaining
    SET attemptsRemaining = attemptsRemaining - 1
  ENDIF
ENDWHILE

// Determine and display the outcome of the game
IF wordCompletion == secretWord THEN
  DISPLAY "Congratulations! You guessed the word: " + secretWord
ELSE
  DISPLAY "You ran out of attempts. The word was: " + secretWord
ENDIF
```