A mini Project Report on

the Development of a Simple Desktop Wordle Application using Python and Tkinter

Submitted for partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE IN

ARTIFICIAL INTELLEGENCE

BY

ARYAN GOYAL	**25030421004**
DAKSH JAIN	**25030421007**
JASH SHAH	**25030421016**
SREEHARI S	**25030421045**

UNDER THE GUIDANCE OF

Dr. Dawa Chyophel Lepcha

॥वसुधैव कुटुम्बकम्॥

SAII (साई)

SYMBIOSIS ARTIFICIAL INTELLIGENCE INSTITUTE

Symbiosis Artificial Intelligence Institute

SAII(साई)
SYMBIOSIS ARTIFICIAL INTELLIGENCE INSTITUTE
॥वसुधैव कुटुम्बकम्॥

Certificate

This is to certify that the project work entitled "**the Development of a Simple Desktop Wordle Application using Python and Tkinter**" that is submitted by

ARYAN GOYAL	**25030421004**
DAKSH JAIN	**25030421007**
JASH SHAH	**25030421016**
SREEHARI S	**25030421045**

In partial fulfillment for the award of the degree of BACHELOR OF SCIENCE IN ARTIFICIAL INTELLIGENCE is a record of bonafide work carried out by them. The results embodied in this report have been verified and found satisfactory.

Declaration

We **ARYAN GOYAL** , **DAKSH JAIN** , **JASH SHAH** , and **SREEHARI S** hereby declare that the report of the Mini Project work entitled **“the Development of a Simple Desktop Wordle Application using Python and Tkinter”** which is being submitted to the **Symbiosis Artificial Intelligence Institute**, in partial fulfillment of the requirement for the award of the Degree of **BACHELOR OF SCIENCE IN ARTIFICIAL INTELLIGENCE**, is a bonafide record of the work carried out by us. The material contained in this report has not been submitted to any other University or Institution for the award of any degree.

Place: **Lavale,Pune**

Date:

(ARYAN GOYAL - 25030421004)

(DAKSH JAIN - 25030421007)

(JASH SHAH - 25030421016)

(SREEHARI S - 25030421045)

ABSTRACT

The proliferation of simple, accessible, web-based puzzle games has created a demand for lightweight, cross-platform desktop implementations. This mini-project focuses on developing a simplified version of the popular word-guessing game Wordle, henceforth referred to as **Wordle Lite**. The application is constructed entirely in **Python** using the **Tkinter** library for the Graphical User Interface (GUI). The core functionality involves generating a random 5-letter word and providing immediate, color-coded feedback to the user's guesses over a fixed number of attempts. The project successfully demonstrates competence in Python desktop application development, state management, and basic algorithmic design for game logic.

Keywords: Python, Tkinter, GUI, Game Development, Wordle, Simple Algorithm

CHAPTER 1: INTRODUCTION

1.1 General

Computer-based games, especially those categorized as puzzles, have become significant tools for user engagement and cognitive stimulation. The Wordle format, which challenges a player to guess a secret 5-letter word within six attempts using positional and inclusion feedback, represents a highly effective and minimal design. This project aims to recreate this essential, engaging experience within a self-contained desktop environment, eliminating the reliance on a web browser.

1.2 Problem Statement

While many web-based Wordle clones exist, there is a need for a readily executable, small-footprint desktop application that provides the fundamental game loop without external dependencies. The primary challenges are accurately mapping the game's core logic (the color-coded feedback system) to a **Python** function and successfully rendering and managing the interactive grid-based **GUI** using the native **Tkinter** framework.

1.3 Objectives

The objectives of this project were clearly defined as follows:

- To develop a functional desktop application using the **Python** programming language.
- To implement the GUI using the **Tkinter** library, featuring a `5 \times 8` grid (five letters wide, eight guesses deep).
- To design the core game algorithm, `get\_simple\_feedback`, to provide color-coded results (Green for correct position, Yellow for correct letter/wrong position, Gray for absent letter).
- To manage the game state, including the current guess row, the secret word selection, and the termination conditions (win/loss).

CHAPTER 2: LITERATURE REVIEW (ALGORITHMIC DESIGN)

The project's algorithmic focus is concentrated on the feedback function, which defines the core interaction of the game.

2.1 Game Concept and Rules

The game **Wordle Lite** follows the standard rules of its inspiration. A player attempts to guess a hidden 5-letter word. After each submission, the player receives feedback on each letter:

1. **Green:** The letter is correct and in the correct position.
2. **Yellow:** The letter is in the word but in the wrong position.
3. **Gray:** The letter is not in the secret word.

The player is allotted a maximum of ***N=8*** guesses, providing a slightly more forgiving environment than the original game.

2.2 Core Feedback Algorithm: *get_simple_feedback(secret_word,guess)*

The core logic is encapsulated in the function `get\_simple\_feedback`. This function iterates through the guessed word and compares each letter against the secret word.

The logic proceeds as follows:

- A default feedback list of 5 **Gray** values is initialized.
- The function iterates from index i=0 to 4 (since WORD_LENGTH = 5).
- **Positional Check (Green):** If guess[i] = secret_word[i], the feedback for that position is set to **Green** (#27AE60).
- **Inclusion Check (Yellow):** If the positional check fails, an inclusion check is performed: if guess[i] is present *anywhere* in the secret_word, the feedback is set to **Yellow** (#F39C12).
- If both checks fail, the color remains **Gray** (#95A5A6).

This approach offers a simplified Wordle logic suitable for a basic implementation.

CHAPTER 3: METHODOLOGY (IMPLEMENTATION)

The application was built following a standard Model-View-Controller (MVC) approach, where the `SimpleWordleApp` class acts as both the **Controller** and the **View**, and the internal variables manage the **Model** (Game State).

3.1 Programming Language and Framework

- **Programming Language:** Python 3.x was chosen for its readability, rapid development capabilities, and extensive library support.
- **Framework: Tkinter** was utilized as the standard, lightweight, and native Python library for GUI development, ensuring the application is highly portable across different operating systems without requiring external installations.

3.2 Key Class and Method Components

3.2.1 The `SimpleWordleApp` Class

This class inherits from `tk.Tk` and initializes the entire application. It holds the core state variables:

- `self.secret\_word`: The randomly selected target word.
- `self.current\_row`: The index of the current guess attempt (from 0 to 7).
- `self.cell\_labels`: A 2D array of `tk.Label` objects representing the visual grid.

3.2.2 `create\_widgets()`

This method is responsible for laying out the UI components using the `pack()` and `grid()` layout managers. A `Frame` is used to contain the 5×8 grid of `tk.Label` elements, which dynamically change text and background color as feedback.

3.2.3 `handle\_guess()`

This is the primary event-driven method, triggered by the "Guess!" button or the Return key. Its sequence of operations is:

1. **Input Validation:** Checks if the input is a 5-letter alphabetical word.
2. **Feedback Calculation:** Calls the `get\_simple\_feedback` function.
3. **UI Update:** Iterates through the current row's `tk.Label` objects, setting the text to the guessed letter and the background to the calculated feedback color.
4. **Game State Check:** Compares the guess to the `self.secret\_word` to check for a win, or increments `self.current\_row` to check for a loss.
5. **Termination:** Calls `end\_game()` upon win or loss.

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Game Interface and User Experience

The application presents a clean, centered interface. The use of custom colors (Green, Yellow) enhances visual clarity and contrast. The grid is clearly defined, and the text input field is automatically cleared after each submission, streamlining the user experience.

4.2 Game Flow and Win/Loss Conditions

The game successfully executes the core loop: input, validation, feedback, and row advancement.

- **Continue Game:** If the attempt number is less than the maximum guesses (8) and the word is not solved, the current row count is incremented (`self.current_row += 1`).
- **Win Game:** If the guess exactly matches the secret word, a victory message is displayed, and the game ends.
- **Loss Game:** If the current row count reaches the limit of 8 without solving the word, a loss message is displayed, and the game ends.

4.3 Limitations and Simplifications

The current implementation uses a **simple feedback algorithm** that does not fully handle edge cases involving duplicate letters in the secret word (e.g., guessing 'AARTH' when the secret word is 'CRANE'). In a strictly accurate Wordle implementation, the count of letters is tracked to prevent excessive 'Yellow' or 'Green' marking. This simplification was adopted to maintain the project's scope as a "Lite" desktop version. Furthermore, the vocabulary is limited to a small, hardcoded list for demonstration purposes.

CHAPTER 5: CONCLUSIONS

The **Wordle Lite** project successfully fulfilled all stated objectives, resulting in a functional, self-contained desktop application built on Python and Tkinter. The project validated the ability to translate game rules into simple programmatic logic and manage complex GUI state within a single-file structure.

The core achievement is the stable operation of the `handle_guess` method, which seamlessly integrates input handling, algorithmic processing, and real-time visual feedback.

The future scope for this project includes:

1. Implementing the **advanced duplicate-letter handling** logic for more accurate Wordle feedback.
2. Expanding the **vocabulary list** to thousands of words, potentially importing from a text file.
3. Adding a **keyboard interface** to allow users to type directly into the grid.

REFERENCES

[1] Python Software Foundation. (n.d.). *The Python Standard Library - Tkinter*.

[2] TkDocs. (n.d.). *Tkinter: A Graphical User Interface Toolkit for Python*.

.