PROJECT REPORT

CA - 2

CSM216

(PYTHON PROJECTS)

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Praneeth Reddy

12306037

K23CH – G2

ROLL NUMBER: 38

Submitted to:

Mr. AMAN KUMAR

LOVELY PROFESSIONAL UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

ACKNOWLEDGEMENT

I, Praneeth Reddy, a student of Bachelor of Technology under Computer Science and Engineering discipline with Data Sceince and Machine Learning specialization at Lovely Professional
University, Punjab, hereby declare that all the information furnished in this project report is based on my own work and is genuine.

Praneeth Reddy

12306037

TABLE OF CONTENTS

I. INTRODUCTION

The **Hangman Game** is a Python-based application designed to provide an interactive and engaging way to play the classic word-guessing game. This project uses the Tkinter framework to create an intuitive graphical user interface (GUI), offering an enjoyable experience for users who wish to test their word-guessing skills. The game is ideal for anyone looking for a fun and educational activity, and it serves as an introduction to programming concepts such as string manipulation, event handling, and GUI development.

In the traditional Hangman game, players guess letters in a hidden word with a limited number of incorrect guesses before they lose the game. This project addresses the challenges of implementing such a game by providing an easy-to-use interface with real-time feedback, making the game more accessible and user-friendly. It also adds a visual aspect to the traditional game, enhancing the overall experience.

This project implements a desktop application with the following features:

- **Word Guessing**: Players input a letter to guess a word, with immediate feedback provided for each guess.
- **Game Progress**: The current status of the word, including the correctly guessed letters and remaining blanks, is displayed dynamically.
- **Incorrect Guess Tracking**: The number of incorrect guesses is displayed, and the game ends when the player exceeds the maximum allowed wrong guesses.
- **Clear Feedback**: The application provides clear feedback about correct or incorrect guesses and announces the result when the game ends.

Implementation Details:

- **Programming Language**: Python is used as the primary programming language due to its simplicity and versatility, along with the powerful tkinter library for building GUI applications.
- **GUI Framework**: Tkinter is utilized to create an interactive and user-friendly interface, featuring buttons, labels, and text boxes to allow for smooth gameplay.
- **Game Logic**: The word selection, guess validation, and game status tracking are implemented directly in Python, providing a straightforward yet functional game experience.

This project serves as a fun and accessible tool for anyone interested in playing Hangman while also offering an opportunity to learn and practice Python programming and GUI development. It combines simplicity in design with interactive functionality, making it an engaging and enjoyable project for both developers and players.

All other details are mentioned below.

II. OBJECTIVES AND SCOPE OF THE PROJECT

OBJECTIVES:

The **Hangman Game** project aims to achieve the following objectives:

1. **Develop a Functional Word Guessing Game**: Create an interactive Hangman game where users can guess letters to form a word, with the game providing real-time feedback on correct and incorrect guesses.
2. **Create a User-Friendly GUI**: Implement a simple, visually appealing graphical user interface (GUI) using Python's Tkinter library, ensuring the game is easy to play and understand for users of all ages.
3. **Track Game Progress**: Incorporate features that track the current state of the game, including the progress of correct letter guesses, the number of incorrect guesses, and the overall game status (win or loss).
4. **Implement Validation and Feedback**: Ensure that the game provides proper validation of user inputs, including checking for previously guessed letters and informing the player of invalid or incorrect entries.
5. **Educate and Engage Users**: Provide an engaging platform for users to have fun while improving their vocabulary and word recognition skills, with an intuitive interface for easy interaction.

SCOPE:

The scope of this **Hangman Game** project includes the following aspects:

1. **Word Selection**: The game will randomly select a word from a predefined list of words. These words can be easily modified or expanded to add variety and difficulty.

2. **GUI Interface**: The project will be limited to a desktop application, and the primary interaction will occur through the GUI, which will include buttons, text fields, and labels.

3. **Game Mechanics**:
    - The game will allow players to guess one letter at a time.
    - The game will track both correct and incorrect guesses and update the display accordingly.
    - A maximum of 6 wrong guesses will be allowed before the game ends.
    - The game will notify players when they win (by guessing the entire word correctly) or lose (by exceeding the maximum incorrect guesses).

4. **Game Restart**: The project will allow users to play multiple rounds of Hangman without restarting the application, providing a new word for each round.

5. **Future Improvements** (Not currently implemented in the scope but can be added later):
    - **Difficulty Levels**: Adding different difficulty levels (easy, medium, hard) by varying the length or complexity of the words.
    - **Word Categories**: Incorporating word categories (e.g., animals, sports, countries) to provide users with a more diverse and challenging experience.

- **Visual Hangman Drawing**: Adding an illustration of the hangman being drawn as wrong guesses accumulate to visually represent the game status.

By focusing on these core aspects, the **Hangman Game** project provides a fun and simple educational tool while also offering opportunities for further expansion and enhancement in the future.

III. APPLICATION TOOLS

The development of the **Hangman Game** application utilizes a combination of tools and technologies to ensure efficient implementation and optimal functionality. These tools include:

1. Programming Language:

- **Python**: The core programming language used for its simplicity, readability, and strong support for GUI development. Python allows for easy implementation of game logic and user input handling.

2. Graphical User Interface (GUI) Framework:

- **Tkinter**: Used to create an interactive and user-friendly graphical interface for the Hangman game. Tkinter allows for seamless integration of buttons, labels, and text fields to provide an engaging user experience.

3. Game Logic and Randomization:

- **Random Module**: The Python random module is used to randomly select a word from a predefined list of words. This ensures that each game is unique and offers a new challenge with every round.

4. Development Environment:

- **Integrated Development Environment (IDE)**: Tools like **PyCharm**, **VS Code**, or **IDLE** are used to write, test, and debug the Python code. These IDEs provide helpful features like syntax highlighting, debugging, and project management.

5. Libraries and Modules:

- **Tkinter**: The primary library used for building the graphical interface.
- **tkinter.ttk**: Provides access to advanced Tkinter widgets, like styled buttons or entry fields, which enhance the appearance and functionality of the GUI.
- **random**: A Python module used to select the word to be guessed, ensuring variability in the game.

6. Error Handling and Debugging Tools:

- **Python Debugger (pdb)**: Used to troubleshoot and debug the game during development, ensuring the game functions as expected and handles edge cases properly.
- **Linting Tools**: Tools like **Pylint** or **Flake8** are used to ensure the code is clean, consistent, and follows Python coding standards, which improves readability and maintainability.

7. Testing and Deployment:

- **Manual Testing**: Used for verifying the functionality of the game, including word selection, correct letter guesses, and game over conditions.
- **PyInstaller**: Can be used to package the Hangman game as an executable file (.exe), allowing for easy distribution of the game on Windows systems without requiring the Python interpreter to be installed.

These tools ensure that the **Hangman Game** is developed with efficiency, usability, and reliability in mind, providing a solid foundation for future enhancements, such as adding more word categories or multiplayer capabilities.

IV. PROJECT DESIGN

The **Hangman Game** project is designed to provide a fun and interactive user experience while demonstrating fundamental programming concepts such as game logic, string manipulation, event handling, and GUI design. Below is a detailed breakdown of the project design, covering the system architecture, user interface, and game mechanics.

1. System Architecture

The Hangman Game is a **desktop application** built using **Python** and the **Tkinter** framework for the GUI. The overall architecture is divided into three key components:

1. **User Interface (UI)**: Handles user input and displays the current game state. This includes the display of guessed letters, the current word progress, and feedback messages.
2. **Game Logic**: The underlying logic that controls the game's flow, including word selection, guess validation, and game status (win/loss).
3. **Input Handling**: Manages user interactions, such as letter guesses and button clicks.

2. Functional Components

2.1 Game Logic

- **Word Selection**: The game starts by selecting a random word from a predefined list.
    - The random module is used to pick a word randomly from a list of words.
    - The length of the word determines how many blanks will be displayed to the user.
- **Guess Processing**:
    - The game accepts a letter guess from the player.
    - If the letter is part of the word, it is revealed in the correct positions; otherwise, it is counted as an incorrect guess.
    - The game tracks the number of incorrect guesses, and if the count reaches 6, the game ends in a loss.
- **Game State**:
    - The game updates the state after each guess, showing the current word (with underscores for unguessed letters) and the number of wrong guesses made.
    - The game ends when either the player guesses the word correctly or exceeds the maximum allowed wrong guesses (6).

2.2 User Interface (UI)

The GUI is built using **Tkinter**, which provides interactive widgets such as buttons, labels, and text entry fields. The user interface includes:

- **Word Display**:
    - A series of underscores (_) representing unguessed letters.
    - Correctly guessed letters are displayed in their respective positions.
- **Guess Entry**:
    - A text input field for players to enter their guesses.
- **Submit Button**:
    - A button that players click to submit their guess.
- **Incorrect Guesses Display**:
    - A label that shows the list of incorrect guesses.
- **Feedback**:
    - A label that updates to show whether the last guess was correct or incorrect.
- **Game Status**:
    - A label that displays messages such as "You Win!" or "Game Over!" when the game ends.
- **Reset Button**:
    - A button that allows players to start a new round with a fresh word.

2.3 Event Handling

- **Button Click**:
    - The game responds to button clicks by processing the letter guess entered by the player and updating the game state accordingly.
- **Game Feedback**:
    - After each guess, the game provides immediate feedback (correct or incorrect guess).
    - The game announces the outcome once it finishes, either by winning or losing.

3. Data Flow and User Interaction

Step 1: Start the Game

- The player opens the game, and the program selects a random word from the predefined list.

- The interface is initialized, displaying a series of underscores (_) based on the length of the selected word.

Step 2: Make a Guess

- The player enters a letter in the text input field and clicks the "Submit Guess" button.
- The system processes the guess:
    - If the guess is correct, the word's corresponding letters are revealed.
    - If the guess is incorrect, the number of incorrect guesses increases.

Step 3: Update Game Status

- After each guess, the system updates:
    - The current word display.
    - The list of incorrect guesses.
    - The number of remaining wrong guesses.

Step 4: Check for Game Over or Win

- If the player guesses all the letters correctly, the game ends with a win message.
- If the player reaches the limit of incorrect guesses (6 wrong guesses), the game ends with a loss message.

Step 5: Reset the Game

- The player can click the "Start New Game" button to reset the game, which selects a new word and resets all variables to their initial state.

5. Conclusion

The Hangman Game project is designed with a simple yet effective flow, focusing on user interactivity and game logic. The **Tkinter** GUI allows users to easily interact with the game, while the **Python** code manages the word selection, guess validation, and game state tracking. The design ensures that the game is easy to play while providing immediate feedback and maintaining a fun, engaging experience.

Future enhancements to the project could include features like:

- Adding difficulty levels with longer or more complex words.
- Implementing a visual representation of the "hangman" figure.
- Supporting multiple categories of words (e.g., animals, countries, etc.).
- Introducing multiplayer functionality for competitive play.

V. SUMMARY

The **Hangman Game** project is a Python-based desktop application designed to provide an interactive and engaging way to play the classic word-guessing game. Utilizing the Tkinter framework for the graphical user interface (GUI), this project offers an easy-to-use platform for players to guess letters in a hidden word, with immediate feedback on their progress. The game ensures an enjoyable experience by randomly selecting words and providing a visual representation of the number of incorrect guesses allowed before the game ends.

The game tracks both correct and incorrect guesses, updating the display dynamically to show the current status of the word and the number of wrong guesses made. When the player either guesses the word correctly or exceeds the maximum number of incorrect guesses, the game announces the result and disables further input.

Key Features:

- **Word Guessing**: Players guess letters, and the word's state is updated with correct guesses.
- **User Feedback**: The game provides feedback on whether a guessed letter is correct or incorrect, along with tracking the number of wrong guesses.
- **Game Reset**: Players can play multiple rounds, with a new word selected each time.
- **Interactive GUI**: The user interface is simple, intuitive, and interactive, making the game accessible for players of all ages.

Tools and Technologies:

- **Python**: The primary programming language used for both the game logic and the graphical interface.
- **Tkinter**: The GUI framework used to create the interactive application, with components like buttons, text fields, and labels.
- **Random Module**: Used to randomly select a word from a list for each game round.
- **Integrated Development Environment (IDE)**: IDEs like PyCharm or VS Code for coding, testing, and debugging.

This project serves as both a fun and educational tool, offering players an engaging way to play Hangman while also providing a hands-on opportunity to practice Python programming, event handling, and GUI design. Future enhancements could include adding difficulty levels, word categories, or even multiplayer functionality.

THANK YOU