A white background with blue text

AI-generated content may be incorrect.

**CHARLES DARWIN UNIVERSITY SYDNEY CAMPUS**Haymarket, NSW, Australia

**MASTER OF INFORMATION TECHNOLOGY (SOFTWARE ENGINEERING)**

**[PRT582] – SOFTWARE ENGINEERING PROCESS AND TOOLS**

**Submitted By:**

**Amsh Shrestha [S394695]**

**Submitted To:**

**Muhammad Rana**

Contents

[Figures 2](#_Toc207900703)

[Introduction 3](#_Toc207900704)

[Process 3](#_Toc207900705)

[Conclusion 6](#_Toc207900706)

# Figures

[Figure 1: Test-Driven Development 3](#_Toc207450840)

[Figure 2: Game Logic Implementation 4](#_Toc207450841)

[Figure 3: GUI Implementation 5](#_Toc207450842)

[Figure 4: Integration 6](#_Toc207450843)

# Introduction

The project hangman will provide two difficulty levels, Basic and Intermediate, before the start of the game. The player will have to guess a letter within 15 seconds or else they will lose a life as indicated by progressing hangman image. For every wrong letter, the life is deducted and for the right one, a point (score) will be awarded. If all the life runs out, correct answer will be shown and an option to start new game will be provided which will take players back to level selection interface. Incase player guess the word accurately; they will be congratulated.

The python programming language will be used for writing logic whereas one of its popular frameworks tkinter will be used for making User Interface (UI). Another framework, unittest will be used for Test-Driven Development (TDD).

Python was chosen due to its simplicity, easier to understand syntax and huge number of helpful packages such as NLTK. "unittest" which is python's built-in testing framework was used to completely test the game's logic and ensure that any modifications don't affect previously functional aspects.

# Process

1. **Test-Driven Development (TDD)**

Before implementing game logic, unit tests were written using “unittest” to cover all major game functionalities, including:

* Correct guess handling
* Wrong guess handling
* Win and lose conditions
* Reset functionality
* Handling invalid inputs and repeated guesses

A computer screen shot of a program

AI-generated content may be incorrect.

Figure 1: Test-Driven Development

1. **Game Logic Implementation**

The core game logic is implemented in hlogic.py:

* + Basic level: randomly selects words from a dictionary file.
  + Intermediate level: generates random phrases from the NLTK Brown corpus.
  + Tracks score, remaining tries, guessed letters, and handles game-over conditions.

A screen shot of a computer

AI-generated content may be incorrect.

Figure 2: Game Logic Implementation

1. **GUI Implementation**

The GUI is implemented using Tkinter in hgui.py:

* + Level selection menu.
  + Timer display with auto-decrement.
  + Tries and score display.
  + Hangman image panel that updates with wrong guesses.
  + Alphabet buttons for user input, as well as keyboard input support.
  + New game functionality upon game over.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3: GUI Implementation

1. **Integration**

* The main program (main.py) initializes the Tkinter root window and starts the Hangman GUI.
* Automated tests ensure all features work correctly before integration.

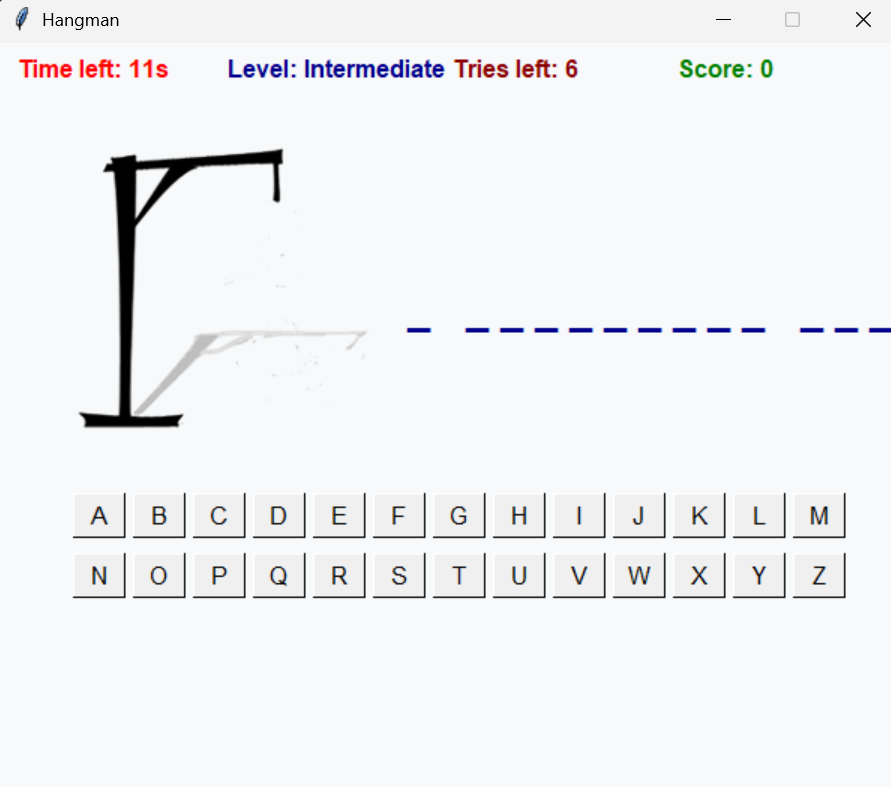


Figure 4: Integration

# Conclusion

TDD minimizes problems and guarantees robustness during feature development. It is easier to test and maintain when logic and GUI are kept apart. Simple yet effective tools for creating instructional games are offered by Python's Tkinter and NLTK libraries.

**GitHub Repository**

<https://github.com/ams-sth/Hangman-game.git>