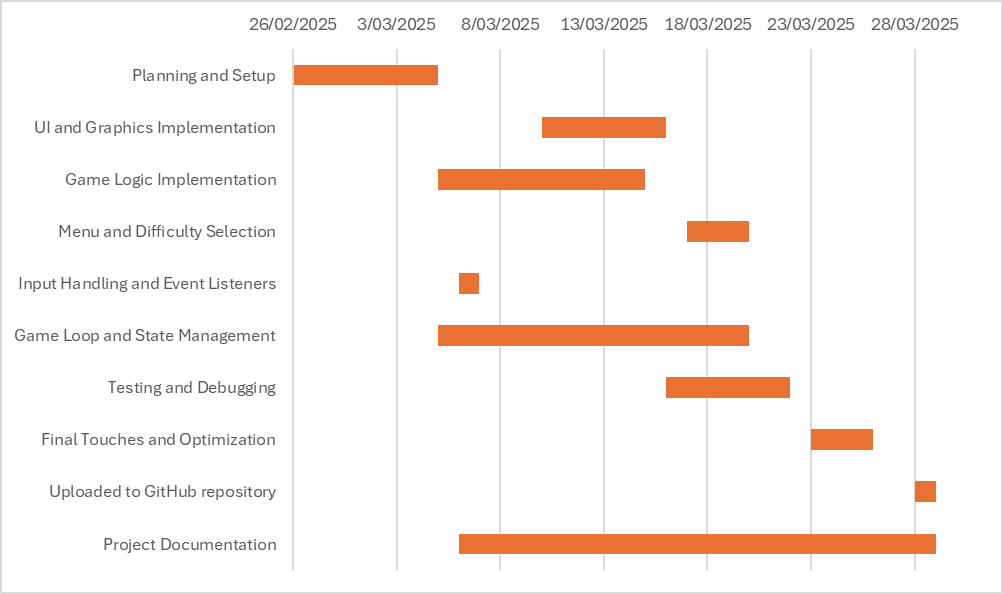
**HANGMAN PROJECT**

*Assessment Task 1 - By Kaleb Vong 11SENG1*

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** | |
| **Assessment Criteria 1**  Gantt Chart………………………………………………………………………………..  Resource Allocation……………………………………………………………………...  Budget……………………………………………………………………………………..  Workforce………………………………………………………………….………………  Storyboards……………………………………………………………………………….  Justification for Programming Language………………………………………………  **Assessment Criteria 2**  Class/Sequence Diagram……………………………………………………………….  Pseudocode……………………………………………………………………………….  IPO Context Diagram……………………………………………………….……………  Data Dictionary……………………………………………………………………………  **Assessment Criteria 3**  Planning Techniques……………………………………………………………………..  Risk Management Strategies……………………………………………………………  Divide and Conquer………………………………………………………………………  **Assessment Criteria 5**  Input Validation…………………………………………………………………………...  Output Encoding………………………………………………………………………….  Use of Official Python Libraries………………………………………………………… | 2  2  3  3  3  3  4  5  11  12  14  14  14  15  15  15 |

**Gantt Chart:**



**Resource Allocation:**

Throughout the development of this digital project, designed with the goal of educating younger generations about software development terminologies and vocabularies, many online resources were effectively utilised to fulfill this goal of education and to build the foundations of this game. The use of the **Pygame** library was strictly chosen due to its lightweight memory and RAM storage and its highly advanced methods for rendering graphics, detecting user inputs such as mouse clicks and movements and event management. Furthermore, the use of mostly pre-created assets from online tutorial from [Tech With Tim](https://www.youtube.com/watch?v=UEO1B_llDnc&t=9s) allowed for the developer’s time to be handled to be utilised to the fullest extent, with time being strictly used for coding the project rather than handcrafting assets or using generative AI for these assets.

**Budget:**

The budget allocated for the creation of this project was strictly **free** without any need of financial funding. Thanks to free open-source libraries such as Pygame that are extremely lightweight and are practically capable of running on any device that support **Python** code and assets pre-created by Tech With Tim, no money was needed to be spent on these particular aspects of this project. Furthermore, the word lists for each difficulty were handcrafted by the developer with the terminologies being taken from free digital sources such as online dictionaries and articles.

**Workforce:**

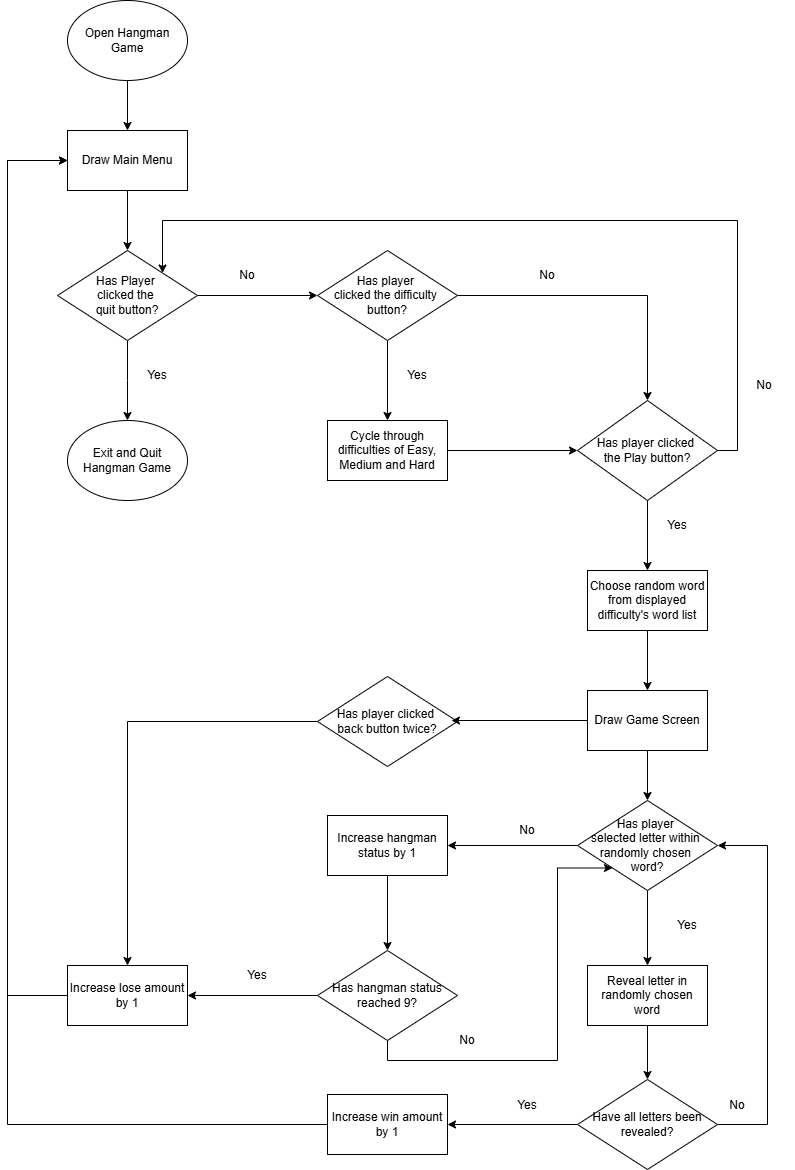
The workforce for this project was strictly developed only by **one developer**. As such, all roles not supplemented by other online sources (such as the role of **artist**) were solely partaken by this developer. Examples of the roles needed to be taken to create this project were **UI Designer**, **Main Coder** and **Playtester**.

**Storyboards:**

While there was no explicit story told due to the game's highly simple nature and primary goal of educating younger generations, a storyboard can still be identified within the design of the game. The designing of the main menu and game screen all required the development of wireframes before coding these interfaces into the game. Furthermore the positions and functions of the “Play”, “Difficulty” and “Quit” buttons all required planning before fully implementing into the code of the game. As such, storyboards for these elements were all utilised to aid in the development and implementation of these features within the game.

**Justification for Programming Language:**

During the main development of the code for my rendition of the classic Hangman Game, I (the sole developer) of the game had chosen to use Python in combination with the Pygame library due to their flexibility and specialty when creating such a program, widespread availability and readability of the language. Python allows easy and quick development of small games like my Education Hangman Game as it supports a vast amount of libraries such as Pygame, allowing it to be considered an extremely flexible language. Python allows for functionalities such as randomisation, file handling and Graphical User Interfaces to be implemented extremely easily with the use of its many pre-installed and community-made libraries. Furthermore, Python and pygame are both extremely lightweight applications that do not require much dedication from the system such as allocating memory and RAM use. As such, Python is extremely widespread throughout coding communities as even low-end devices can run such a coding language. Finally, python is extremely readable by human developers compared to many other programming languages, such as Java and C#. Python is coded with an extremely clean and simple syntax that allows developers to quickly and efficiently learn these formats of this language and start producing more applications coded in Python themselves. For example, coding in Python does not require a semicolon at the end of each line unlike many other programming languages. As such, all these reasons have influenced my decision to use Python when creating my rendition of the classic Hangman Game.

**Class/Sequence Diagram:**

**Pseudocode:**

Import pygame library

Import math library

Import random library

Initialise pygame

Get monitor resolution

Store width and height resolution in WIDTH variable and HEIGHT variable respectively

Set “win” variable and display pygame application in fullscreen

Set the application caption to “Educational Hangman Game!”

Set radius and gap of buttons

Set start width and height positions of buttons

Create list of letters

Set ASCII value for character A

Set x and y position of each letter of alphabet from left to right at the bottom of the screen in 13 columns and 2 rows

Set colour variables for WHITE, BLACK, GRAY, GREEN, YELLOW and RED

Set different pygame fonts for LETTERS, WORDS, TITLE, SUBTITLES and BUTTONS

Create image list for all hangman assets

Load correct hangman images depending on value from 1 to 10

Scale all hangman images corresponding to the player’s screen size

Set game variables for HANGMAN\_STATUS, WORDS LIST, WORD, GUESSED, DIFFICULTY and CURRENT\_STATE

Set stat variables for WIN\_AMOUNT and LOSE\_AMOUNT

Define “draw\_title” function

Render title “H A N G M A N G A M E” in font for TITLE and in BLACK

Set text x and y positions to be in the top middle of the player’s screen

Draw a line underneath text

Detect if player is in menu

If true, draw subtitle underneath title line saying "By: Kaleb Vong 11SENG1" in font for SUBTITLE and in BLACK

If false:

Render chosen difficulty in either GREEN for EASY, YELLOW for MEDIUM, or RED for HARD

Define “choose\_word” function:

Import word variable from game variable list

Detect if player has chosen EASY mode

Open “easy words” list file

Set “words” list variable to all words in opened list file

Else detect if the player has chosen MEDIUM mode

Open “medium words” list file

Set “words” list variable to all words in opened list file

Else detect if the player has chosen HARD mode

Open “hard words” list file

Set “words” list variable to all words in opened list file

Set word variable to random word inside words list variable

Define “draw” function

Import word variable from game variable list

Fill screen to white

Draw title by calling “draw\_title” function

Call stats function

Set display\_word variable as an empty string

If the letter is inside the chosen word and also in the guessed list, store the letter inside display\_word variable.

Else store each letter as “\_ “ inside display\_word variable Render display\_word variable on screen

For each letter of the alphabet

If unclicked and still visible

Draw a circle around each letter’s x and y positions

Draw the letter in its corresponding circle

Display corresponding hangman image on screen with hangman\_status variable

Draw back button in top left corner of player’s screen

Scale back button with player’s screen

Update pygame display

Define “display\_message” function

Delay sequence by 1000 ms

Fill screen with WHITE

Render either “win” or “lose” message depending on if the player has won or lost the game

Update pygame display

Delay sequence by 3000 ms

Define “menu” function

Fill screen with WHITE

Call draw\_title function

Call stats function

Draw three long gray rectangles on top of each other on the player’s screen

Label the top-most rectangle with “P L A Y” in WHITE

Label middle rectangle with “DIFFICULTY: “ in WHITE

Label bottom rectangle with “Q U I T” in white

Set default difficulty to EASY mode

Update pygame display

Define “diff\_name” function

Clear text display previously selected difficulty by hiding it with a grey rectangle

Render new difficulty selected in corresponding difficulty colour (GREEN = EASY, YELLOW = MEDIUM, RED = HARD)

Update pygame display

Define “stats” function

Import “win\_amount” and “lose\_amount” variables

Render “W I N S: “ at the bottom of player’s screen in BLACK and in SUBTITLE font

Render “ L O S S E S: “ at the bottom of player’s screen in BLACK and in SUBTITLE font

Render value stored in win\_amount variable next to “W I N S: “ in GREEN and in SUBTITLE font

Render value stored in lose\_amount variable next to “L O S S E S: “ in RED and in SUBTITLE font

Define “main” function

Import “hangman\_status”, “difficulty”, “word”, “guessed” and “current\_state” variables

Import “win\_amount” and “lose\_amount” variables

Set game to run at 60 FPS

Set clock variable to track amount of time game is running

Set “run” variable to true

Set “shown\_warning” variable to 0

Draw menu by calling menu function

Set current\_state variable to “menu”

Draw detection boxes for all menu buttons to detect collisions with mouse inputs

Draw detection box for in-game back button

Set up main game loop while game is running (run = True)

Set clock variable to the FPS the game is running at

Get all pygame events coded by default into pygame library

If “X” button on application is pressed, quit the game by setting “run = False”

If any mouse button is pressed (such as left-click, right-click, middle-click)

Store position of where mouse was clicked in m\_x and m\_y variables

If “current\_state” variable is “menu”

Quit game if mouse click detected inside detection box of quit button

If mouse click detected inside detection box of difficulty button

Increase “difficulty” variable by 1

If difficulty variable > 2 then reset back to 0

If difficulty variable matches 0

Call diff\_name function to draw EASY mode

Else if difficulty variable matches 1

Call diff\_name function to draw MEDIUM mode

Else if difficulty variable matches 2

Call diff\_name function to draw HARD mode

If mouse click detected inside detection box of play button

Change “current\_state” variable to “game”

Call choose\_word function to randomly select a word depending on selected difficulty

Call draw function to replace menu with game

If “current\_state” variable is “game”

If mouse click detected inside radius of circular letter button and it is visible

Set letter variable to false (thus making button invisible and showing in display\_word variable)

If letter clicked was not inside word

Increase hangman\_status by 1

Call draw function to update screen

If mouse click detected inside detection box of back button

If shown\_warning variable equal to 1

Increase lose\_amount by 1

Reset guessed letter list back to empty

Reset difficulty and hangman\_status back to 0

Set letter variable back to True

Set “current\_state” variable to “menu”

Draw menu by calling menu function

Set shown\_warning back to 0

If shown\_warning variable equal to 0 and current\_state is “game”

Render warning text

Update pygame display

Increase shown\_warning variable by 1

If mouse click was not detected inside detection box of back button and shown\_warning = 1

Reset shown\_warning variable back to 0

Set won variable to True

If a letter in display\_word remains unguessed and not in guessed variable list

Set won variable to False

Break game loop

If won

Display message of “You WON!” by calling the display\_message function

Increase win\_amount variable by 1

Reset guessed letter list back to empty

Reset difficulty and hangman\_status back to 0

Set letter variable back to True

Set “current\_state” variable to “menu”

Draw menu by calling menu function

If hangman\_status = 9

Display message of “You LOST!” by calling the display\_message function

Increase lose\_amount variable by 1

Reset guessed letter list back to empty

Reset difficulty and hangman\_status back to 0

Set letter variable back to True

Set “current\_state” variable to “menu”

Draw menu by calling menu function

Call main function

Terminate pygame library when no more game loop active

**IPO Context Diagrams:**

Play Button Functionality

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Position of Mouse Click * Bounding Box of Play button * “Current\_state” variable | 1. Detect if current\_state matches “menu” 2. Compare if the position of the mouse press is inside the bounding box of the rectangle | 1. Change “current\_state” variable to “game” 2. Run word selection process 3. Fill screen with white 4. Redraw title 5. Draw W/L statistics 6. Display image of first level of hangman status. 7. Draw chosen word on screen with “\_” representing unguessed letters |

Difficulty Button Functionality

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Position of Mouse Click * Bounding Box of Difficulty button * “Difficulty” variable value * “Current\_state” variable | 1. Detect if current\_state matches “menu” 2. Compare if the position of mouse press is inside the bounding box of the rectangle 3. Detect the value of the Difficulty variable | 1. Increase Difficulty Variable by 1 (If 2>, reset back to 0) 2. Draw name of difficulty depending on value of Difficulty variable (0 = Easy, 1 = Medium, 2 = Hard) |

Quit Button Functionality

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Position of Mouse Click * Bounding Box of Quit button * “Current\_state” variable | 1. Detect if current\_state matches “menu” 2. Compare if the position of the mouse press is inside the bounding box of the rectangle | 1. Set run variable to false, thus stopping the game loop 2. Close application |

Word Selection Process

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Value of Difficulty variable * Word string variable | 1. Detect if Difficulty = 0, fill words list variable with words from easy words list file 2. Else if Difficulty = 1, fill words list variable with words from medium words list file 3. Else if Difficulty = 3, fill words list variable with words from hard words list file 4. Choose a random word from all words within words list variable | 1. Word string will be populated with a single randomly chosen word |

Letter Button Functionality

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Position of Mouse Click * Radius of circle drawn over letter button | 1. Detect if mouse click position is within radius of circle drawn 2. Detect if letter chosen is in randomly chosen displayed word | 1. If letter is in displayed word, unhide all of that specific letter within that word 2. Else if letter is not in displayed word, increase hangman status by 1 |

Back Button Functionality

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Position of Mouse Click * Bounding Box of Back button * Value of “shown\_warning” variable * “Current\_state” variable | 1. Detect if current\_state matches “game” 2. Compare if the position of the mouse press is inside the bounding box of the rectangle 3. Detect value of “shown\_warning” variable | 1. If shown\_warning variable = 0, draw warning message on screen 2. If shown\_warning variable = 1, increase player lose\_amounts by 1 3. If shown\_warning variable = 1, set “current\_state” to “menu” 4. If shown\_warning variable = 1, redraw menu |

**Data dictionary:**

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Description |
| Hangman Status | Integer | Used to determine what image of hangman to load and how close a player is to losing a game. If hangman status = 9, the player loses the game. |
| Words | List | Used to store all words from specific word list chosen by difficulty variable |
| Word | String | Used to randomly select a word from all words in the words list variable. This word is then used as the word that needs to be guessed. |
| Guessed | List | Used to store all letters in the game that have been guessed. If all letters in the guessed list are also in the word trying to be guessed, the player wins the game. |
| Difficulty | Integer | Used to determine which word list the words list variable will populate from. Also used to determine what difficulty will be displayed next to the difficulty button and in-game title. |
| Current State | String | Used to track whether the player is playing a game or looking at the main menu. Buttons activate and deactivate depending on this string. |
| Win Amount | Integer | Used to store the total amount of wins the player has accumulated after playing. Displayed as the green number on the bottom of the screen. |
| Lose Amount | Integer | Used to store the total amount of losses the player has accumulated after playing. Displayed as the red number at the bottom of the screen. |

**Planning techniques**

Throughout the development of this educational take of the classic hangman game, I have always had the goal for the game’s design purpose to be for teaching computer related terminology to younger generations of students such as juniors and primary schoolers. As such, it was always meant to be a simplistic game with simple gameplay loop and consisted of a small scope of features - with the most advanced aspects being difficulty and a back button. However, even with this much smaller workload compared to the development processes of other games, each feature still required extensive planning on how I was going to implement it due to me being a sole developer for this project - making processes such as **divide and conquer** much harder.

**Risk management strategies**

Throughout the development of this educational hangman project, many risks regarding player experience and the overall health of the project had been identified and quelled to the best of my ability. For example, the tutorial I followed to build the foundations of this project had strictly defined the screen display to be 800 pixels wide and 600 pixels high. Due to such a small capture, people could easily become disinterested within the project as it could look unpolished and unfinished because of such negligent design. As such, to maintain the health of the project and make it look professional, I changed it so it will always display in fullscreen mode, with the drawn buttons and elements on screen being scaled and moved appropriately according to the player’s screen resolution. This in turn provides a much more immersive experience, as the entire screen is fully used to running and playing the educational hangman game.

Furthermore, another problem I found that was potentially harming the overall experience of the game was misclicking the back button. Players could accidently click the grey arrow button in the top left corner and be left confused with why they have just received a loss and how they are back at the main menu - especially since this program is for young students who are developing vocabularies. As such, I fixed this issue by making the back button display a noticeable and easily readable warning message to the player, not continuing and going back to the main menu unless the button is clicked again. This way, it will save players from accidentally misclicking this button and ending their games unpurposefully and as such being dissatisfied with the overall quality of the game.

**Divide and Conquer techniques**

Throughout the development of the code for this educational rendition of hangman, it is evident that the divide and conquer strategy had been utilised to make the game run most efficiently and have the code be extremely organised. Numerous functions splitting off from the main game loop is an example of divide and conquer being used, with game logic not being constrained into one large loop. Instead, the workload of the game loop has been **divided** to easily manageable and debuggable functions that can easily be read by humans. As such, the same function had been **conquered** more efficiently than if no division of this game loop had occurred.

**Input validation**

Within the code for my educational hangman game, input validation does exist in components, allowing the game to run smoothly without any risks of crashes or bugs. For example, using buttons on screen rather than your keyboard allows for only letters of the alphabet to be guessed, with no support for special characters such as hyphens and numbers. As such, it fully ensures that players are making only legitimate guesses and thus having the fullest potential to learn new vocabulary.

**Output encoding**

Within the code of my hangman game designed for younger generations, output encoding is \within my code to remove the extremely real risk of being targeted by malicious and dangerous injection attacks, potentially spreading computer viruses to the system that steal personal information and data. With the use of Pygame support fonts, these fonts are automatically protected by encoding, negating all risks of HTML or SQL injections from malicious third parties. As such, no virus will be downloaded unknowingly when installing this educational program.

**Use of Official Python APIs**

Within my code for this attempt at creating the classic hangman experience, only three APIs had been exclusively used to develop this project - Pygame, Math and Random. Both Math and Random are fully supported and ingrained within Python itself, with no need to install these APIs after the installation of Python. As such, these libraries are extremely secure with the official python program download from their website ensuring that you are receiving their product with no unwanted viruses or malware. However, the Pygame library is not hardcoded into the regular python installation, instead having it be completely community run. However, the library is generally still safe, with it being constantly maintained and updated by a full team of passionate developers and is exclusively installed with PIP, ensuring no tampering to your device is being done when installing Pygame.