A purple logo with white text

Description automatically generated

**HIT137 SOFTWARE NOW**

Assignment 3

Semester 2, 2024

Casuarina Campus

**Professor: Dr. Thuseethan Selvarajah**

**Group Name: CAS 068**

|  |  |
| --- | --- |
| **Group members** | **Student ID** |
| Sangay Jamtsho | S374529 |
| Thi Thuy Trang Tran | S375041 |
| Cristine Casindac Mansinares | S377798 |
| Shrey Kumar | S381119 |

**TABLE OF CONTENT**

[**QUESTION 1:** 3](#_Toc180046089)

[1.1 IDEA 3](#_Toc180046090)

[1.2 KEY COMPONENTS 3](#_Toc180046091)

[1.3 MAIN FEATURES 7](#_Toc180046092)

[1.4 HOW IT WORKS 7](#_Toc180046093)

[**QUESTION 2:** 9](#_Toc180046094)

[2. 1 IDEA: 9](#_Toc180046095)

[2.2 GAME ENVIRONMENT SET-UP 9](#_Toc180046096)

[2.3 GAME VARIABLE 10](#_Toc180046097)

[2.4 GAME MECHANICS 10](#_Toc180046098)

[2.5 GAME LOOP 11](#_Toc180046099)

[2.6 GAME TERMINATION 11](#_Toc180046100)

[2.7 MAIN FEATURES 12](#_Toc180046101)

[2.8 CONCLUSION 12](#_Toc180046102)

[**QUESTION 3:** 13](#_Toc180046103)

[3.1 TEAMWORK GANTT CHART 13](#_Toc180046104)

[3.2 GITHUB’S LINK 14](#_Toc180046105)

# **QUESTION 1:**

A close up of text

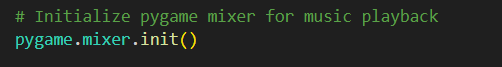
Description automatically generated

* 1. IDEA

Creating the basic music player application with usage of libraries including:

* **Tkinter:** for creating of graphical user interface (GUI)
* **Filedialog (from tkinter):**
* **Messagebox (from tkinter):** for showing pop-up notifications, warning or information messages to user
* **PIL (Pillow):** for handing and displaying image (buttons)
* **A black screen with white text

  Description automatically generatedPygame:** for managing music playback (loading, playing, controlling audio file)
  1. KEY COMPONENTS
* **The Pygame.mixer.init** : function initializes the Pygame mixer module, enabling the program to handle audio playback such as loading and playing sound files.



* **MediaData Class:** Hold and manage list of media file that provide functionality to add and retrieve files from the list. Encapsulation was implemented here. The MediaData class secures the media list with a private attribute, media list. It offers controlled methods to add and access media files, ensuring the list remains intact.

**A screen shot of a computer program

Description automatically generated**

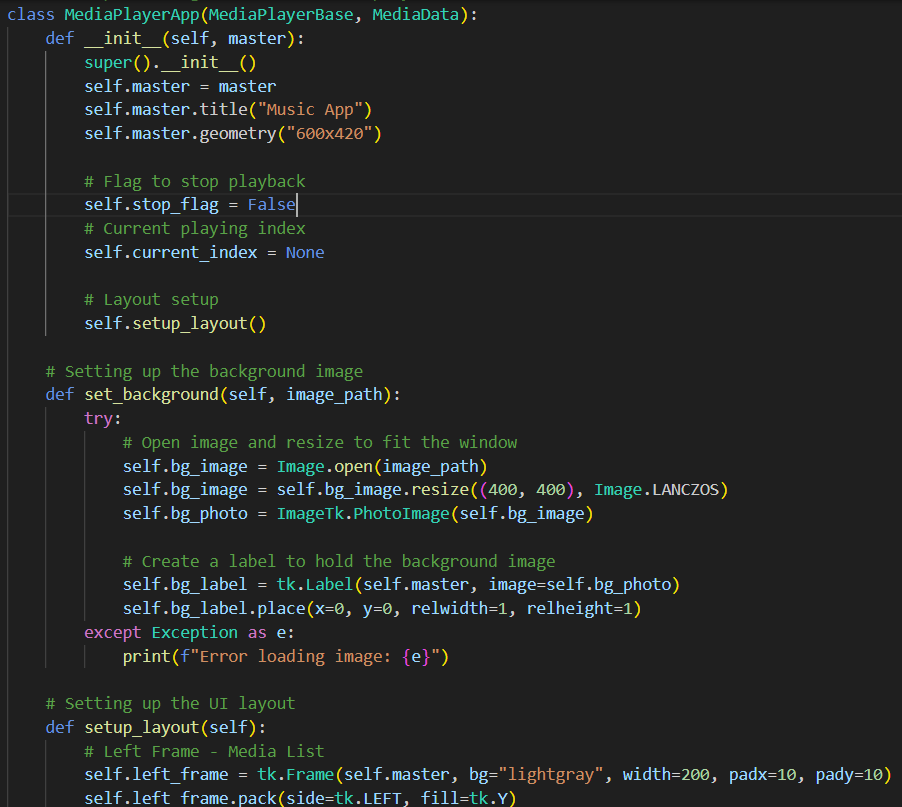
* **MediaPlayerBase Class:** define the method play\_media and stop\_media. The overriding method was implemented here. In MediaPlayerApp, the methods play\_media and stop\_media override the abstract definitions in MediaPlayerBase, delivering specific behaviors for starting and halting media playback.

**A screen shot of a computer error

Description automatically generated**

* **MediaPlayerApp Class:** this is main class, responsible for graphical interface and media player functionality and interact with uses via buttons, listbox and volume control. The MediaPlayerApp class uses multiple inheritance, deriving from both MediaPlayerBase and MediaData. This allows it to manage media data effectively while also handling.

A screen shot of a computer program

Description automatically generated****

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

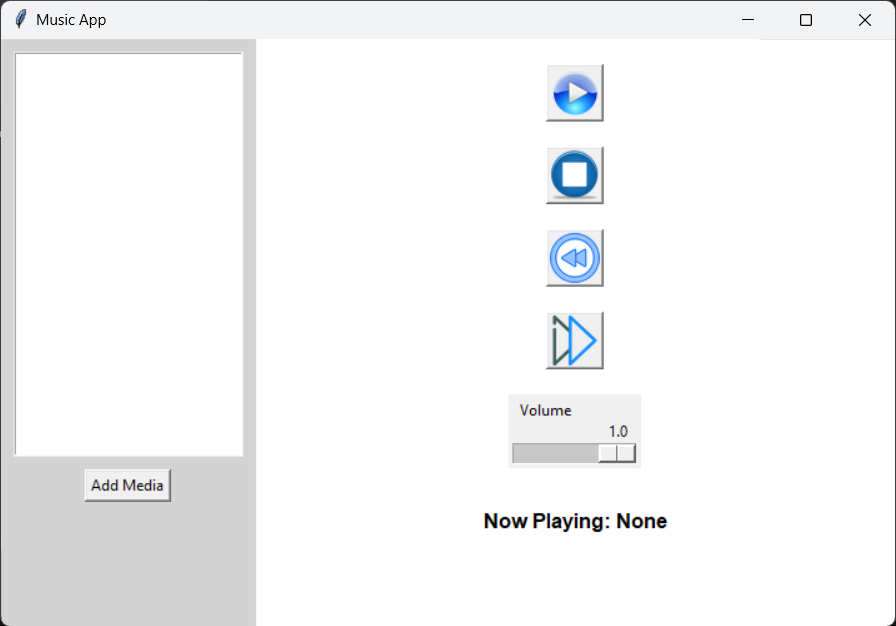
Description automatically generated

* Main functions to run the tkinter app **def main ()**: : creating the main window with tk.Tk(), starting the media player app, keeping the window open with root.mainloop(), and running the program only when the script is run directly with if \_\_name\_\_ == "\_\_main\_\_":.

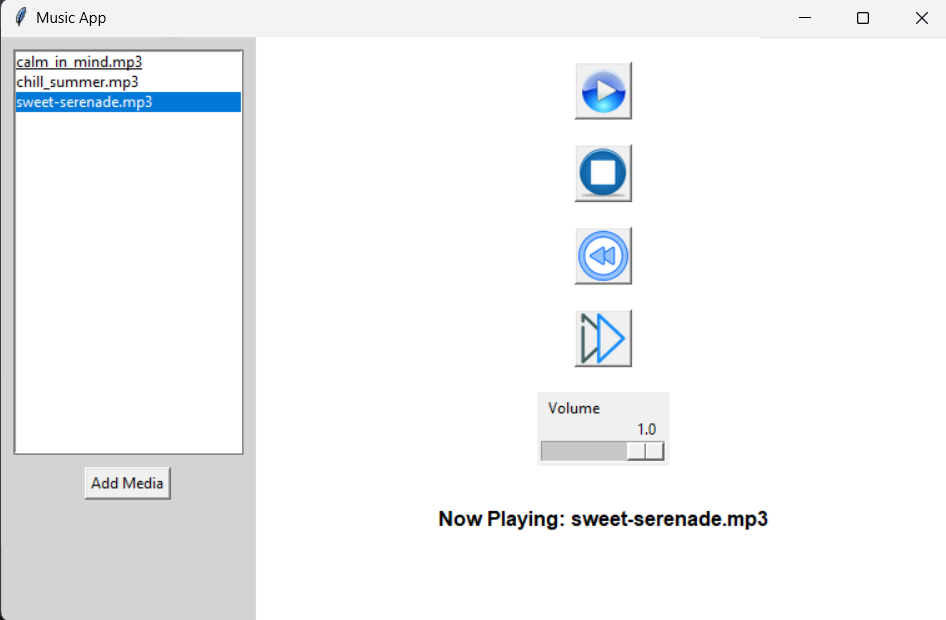
A screenshot of a computer

Description automatically generated

* 1. MAIN FEATURES
* **Media list:** The user can add and manage MP3 file or playback
* **Play, stop and navigation functions:** users can play, stop and move between media file easily
* **Volume control:** Users can control the volume during playback
* **Notifications:** Messagebox inform users when important actions occur or information messages.
  1. HOW IT WORKS
* Load the mp3 file and image button into the same folder before running the python code.
* Run the python code and simple, user-friendly interface appear like below:



* **UI Layout**: The application is split into two main sections:
* A left panel that shows a list of media files the user adds.
* A right panel containing the playback controls (play, stop, next, previous), a volume controller and display the current track
* **Adding Media**: User can click the “Add Media” button to browse and select MP3 file and the file name will be listed on listbox for playback.



* **Playing Media**: After selecting a file from the list and clicking the “play” button , the application use “pygame” to play the chosen media and the file is displayed as well as playback begin.
* **Stopping Media**: When the users click “stop” button , the music stops and display “Now playing: None”
* **Navigating between Media**: User can use the “Next” button  or “Previous” button A blue and white button

  Description automatically generated to navigate between the file in the mp3 list.
* **Volume control**: The user can adjust the volume of the playback using the slider to update the volume.
* **Messagebox Notifications**: will show some important notifications:
* Warning: when users try to play music without file
* Information messages: when users reach the first or the last track in the playlist

# **QUESTION 2:**

## 2. 1 IDEA:

"Zod Runner" is a 2D side-scroller game developed using the pygame library. The player navigates through multiple levels, avoiding enemies and collecting coins. The game includes an escalating difficulty level and ends when the player loses all lives or successfully completes all levels. Below is a detailed report of the game's structure, mechanics, and functionality based on the provided code.

* 1. GAME ENVIRONMENT SET-UP
* **Library Imports**:
* pygame: Core library for handling game graphics, sounds, and event loops.
* Configuration (as config): A custom module that holds configuration values like screen width and height.
* Custom imports (Enemy, Collectible, engine, context): These handle enemy behaviors, collectible objects, game mechanics, and context-specific elements (e.g., images, surfaces).
* A screen shot of a computer

  Description automatically generatedSystem imports (exit, choice, randint): Used for basic operations like exiting the game and randomizing in-game elements.
* **Display Setup:**
* A Pygame display is created with dimensions from the configuration file (config.WIDTH, config.HEIGHT).
* The game window is titled "Zod Runner."
* A pygame.time.Clock() object regulates the game's frame rate.

A computer screen with text

Description automatically generated with medium confidence

* 1. GAME VARIABLE
* **Game State:**
* game\_over: Boolean flag to track if the game is over.
* start\_time: Tracks the start of the game to compute scores.
* score, hit\_score: Score variables to track player performance.
* life: The number of lives the player has (initially 2).
* current\_level: Tracks the player's progress through the levels (starts at 1).
* current\_enemies: Tracks the number of enemies spawned in each level.
* enemies\_level: A list defining the number of enemies per level.
* obstacle\_timer: A custom event that triggers the creation of obstacles and coins at regular intervals (800 ms).
  1. GAME MECHANICS

**Rendering Functions:**

* game\_over\_render(life=0):
* Displays the game-over screen when the player's lives reach 0.
* Blits the background, game over text, and optionally a "player dead" image.
* Calls the engine.render\_life() function to display the player's remaining lives.
* Shows the player's score and life count on screen.
* game\_render(start\_time, life, hit\_score):
* The primary game rendering function.
* Draws and updates the player, obstacles, and collectible coins.
* Checks for collisions:
* **Obstacle collisions**: Reduces the player's life upon collision.
* **Coin collisions**: Increases the player's score when coins are collected.
* Updates the score based on time elapsed and coins collected.
* Displays the score, remaining lives, and level on the screen.
* Handles game-over logic when the player's life reaches 0.
* render\_levels(levels):
* Calls the engine.render\_levels() function to display the current game level on screen.

**Collision Detection:**

* Obstacle Collision: When the player collides with an enemy (detected via engine.obstacle\_collision\_sprite()), they lose a life. If life drops to 0, the game ends.
* Coin Collection: Collectibles (coins) are detected via engine.coin\_collision\_sprite(). Each collected coin increases the player's hit score, which is added to the total score.
  1. GAME LOOP

The game operates within a while loop that continues as long as the current level is less than or equal to the maximum level (4 levels defined). Hereʼs a breakdown of the loop:

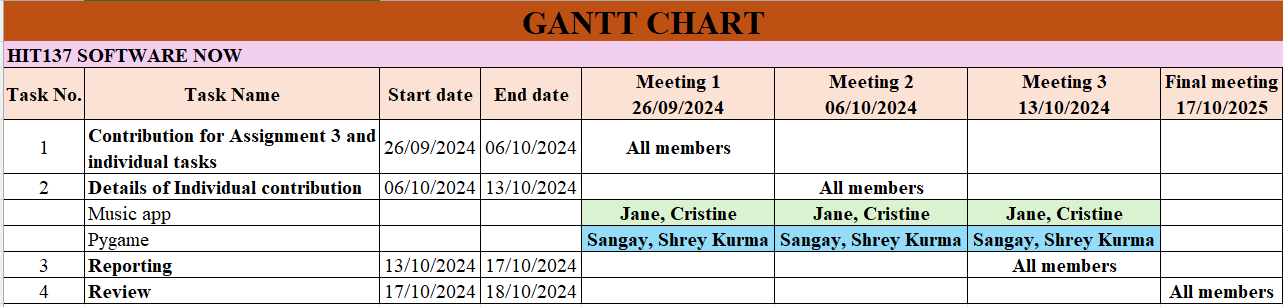
* **Event Handling:**
* The game listens for events like quitting the game, keypresses, and mouse clicks.
* If the game is over, pressing the space bar or clicking the mouse restarts the game with default values (resetting lives, score, and levels).
* **Obstacle and Coin Spawning:**
* The custom obstacle\_timer event triggers every 800 ms. When this event fires, the game spawns enemies or coins based on the current state.
* Enemies spawn until a set limit is reached per level. After the levelʼs enemy quota is filled, the game progresses to the next level.
* Coins are randomly spawned in the game using a random number of types and positions.
* **Rendering and Updates:**
* If the game is not over, the game\_render() function updates the game elements (player, obstacles, coins) and checks for collisions. The game screen is updated at a rate of 60 frames per second (clock.tick(60)).
  1. GAME TERMINATION
* **Game Over State:**
* When the player loses all lives (life == 0), the game-over screen is displayed, and the game halts.
* The game waits for player input to either restart the game or quit.
* **Exit Conditions:**
* The player can quit the game by closing the window or by reaching the last level, where the game ends after a brief delay.
  1. MAIN FEATURES
* **Player Controls:**
* The player can control the game using the keyboard or mouse to restart when the game ends.
* **Obstacles and Collectibles:**
* Enemies (obstacles) and coins (collectibles) spawn periodically based on random choices.
* **Game Levels:**
* The game includes 4 progressively harder levels, each with an increasing number of enemies.
* **Score Tracking:**
* The score is calculated based on both coin collection and time survived. The hit score (from collected coins) is combined with the elapsed time to create the total score.

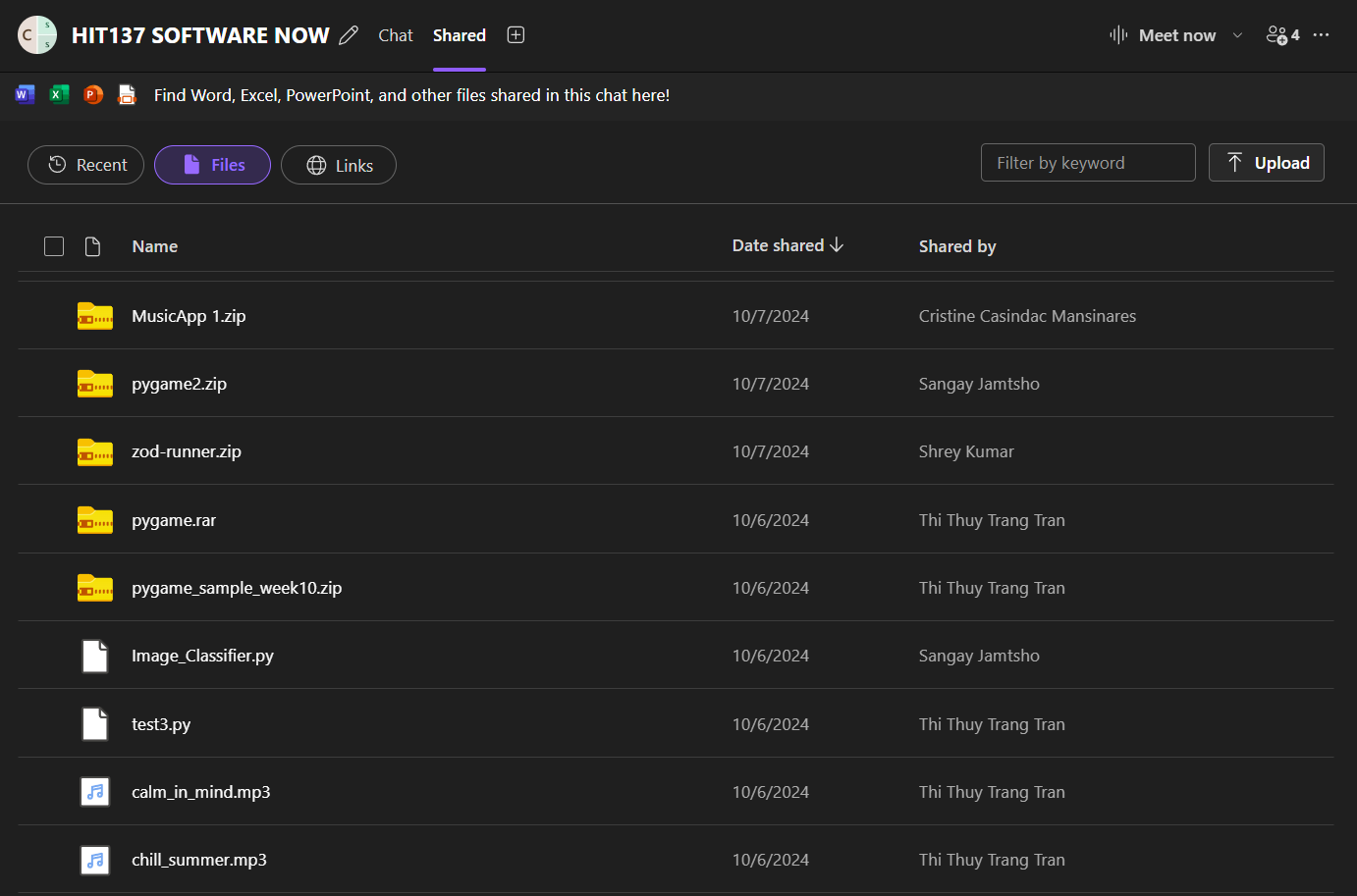
## 2.8 CONCLUSION

"Zod Runner" provides a simple yet engaging side-scrolling experience with basic player mechanics, obstacle dodging, and score collection. The increasing difficulty with each level and life-based game-over system adds an element of challenge. The modular structure of the game (with separate engine, Enemy, and Collectible modules) suggests flexibility for future expansion, such as adding more levels, enemy types, or collectibles.

# A close-up of a message Description automatically generated**QUESTION 3:**

## 3.1 TEAMWORK GANTT CHART



**Sharing information in teamwork on the Assignment 3**

## 3.2 GITHUB’S LINK

Link: [Janetran91bio/HIT137\_Assignment3\_Group\_Cas068 (github.com)](https://github.com/Janetran91bio/HIT137_Assignment3_Group_Cas068)

