PROJECT CONTRIBUTION REPORT

GENPLAY ARCADE

Name: Kalaivani S

Date: 5th April 2025

Mentor: Dr. P. Vijayakumari

Problem Statement:

Gen AI Interactive Learning Game

PROBLEM STATEMENT

Traditional education struggles with engagement and personalized learning. This project aims to create an interactive learning game powered by Gen AI to address these issues. The goal is to develop a dynamic tool that:

- Personalizes learning through adaptive content and feedback.
- Increases engagement via gamification and interactive scenarios.
- Facilitates deeper understanding by using Gen AI to generate learning content.

The challenge is to effectively integrate Gen AI into a game, ensuring educational value and ethical considerations. Success would revolutionize learning by creating engaging, personalized educational experiences.

THE PROCESS

PROJECT DEVELOPMENT AND IMPLEMENTATION

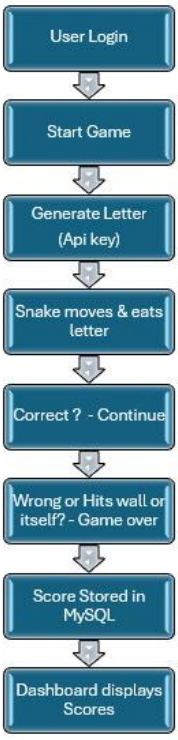
• **User Authentication and Profile Management:**
A secure login and sign-up system was developed to ensure authorized access. MySQL was used to store user credentials and game progress, allowing players to track their scores. Hashing techniques were implemented to protect user passwords, ensuring data security.

• **Gameplay Mechanics and Letter Generation:**
The core gameplay was designed using Pygame, where the snake moves to eat letters in the correct order to form a word. An API key was integrated to dynamically generate letters, ensuring a varied and challenging experience for players. The game logic was structured to handle different conditions such as valid letter selection, incorrect choices, and snake movements. After eating the entire word, the word along with its parts of speech and meaning is displayed on the screen.

• **Game Over Conditions and Score Management:**
The game was programmed to end under specific failure conditions: if the snake ate the wrong letter, collided with itself, or hit the wall. Upon game over, the player's score was stored in the MySQL database, allowing for future retrieval and analysis.

• **Integration of Pygame for Game Functionality:**
Pygame was used to handle real-time game rendering, user inputs, and event handling. Smooth animations and interactions were developed to enhance the user experience. The movement of the snake, collision detection, and word formation logic were implemented within the Pygame framework.

• **Dashboard Development and Score Visualization:**
A dashboard was created to display player statistics, including high scores and game performance. MySQL queries were used to fetch and display stored scores, providing an overview of player progress. Data visualization techniques were applied to show trends and performance metrics.



SKILL DEVELOPED

- Implemented game mechanics, collision detection, and event handling using Pygame.
- Fetched random letters dynamically using API integration.
- • Stored and retrieved player scores using MySQL.
- Created an interactive dashboard with Streamlit.
- Designed logic for word formation and game-over conditions.
- Managed game speed and random letter placement effectively.

PYTHON PACKAGES USED

- Pygame
- Streamlit
- MySQL. Connector
- Google.generativeai

CODE SNIPPETS

To connect to a database:

```
def get_db_connection():
    connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="password",# Change to your MySQL password
        database="genplayarcade"
    )
    return connection
```

To configure api key:

```
# Initialize Gemini API
    GENAI_API_KEY = "API_KEY"  # Replace with your Gemini API key
    genai.configure(api_key=GENAI_API_KEY)
    model = genai.GenerativeModel('gemini-2.0-flash')
```

To display the meaning and parts of speech of the word:

```
def get_word_details(word):
    prompt = f"Give me the part of speech and a simple meaning of the word '{word}' in this format:\nPart of Speech: ...\nMeaning: ..."
    response = model.generate_content(prompt)
    details = response.text.strip()

# Basic parsing for part of speech and meaning
    part_of_speech = "Unknown"
    meaning = "No meaning found."

    for line in details.split('\n'):
        if line.lower().startswith("part of speech"):
            part_of_speech = line.split(":")[-1].strip()
        elif line.lower().startswith("meaning"):
            meaning = line.split(":")[-1].strip()

    return part_of_speech, meaning
```

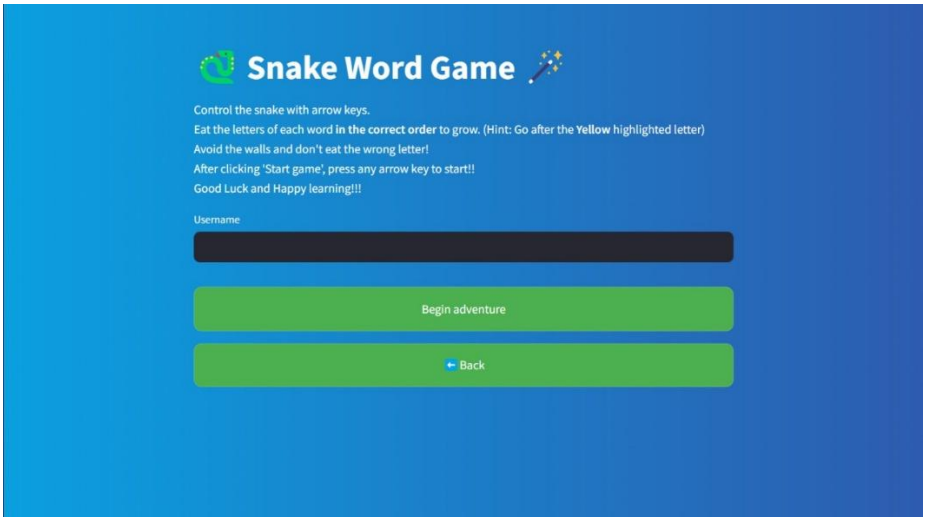
To play the snake game:

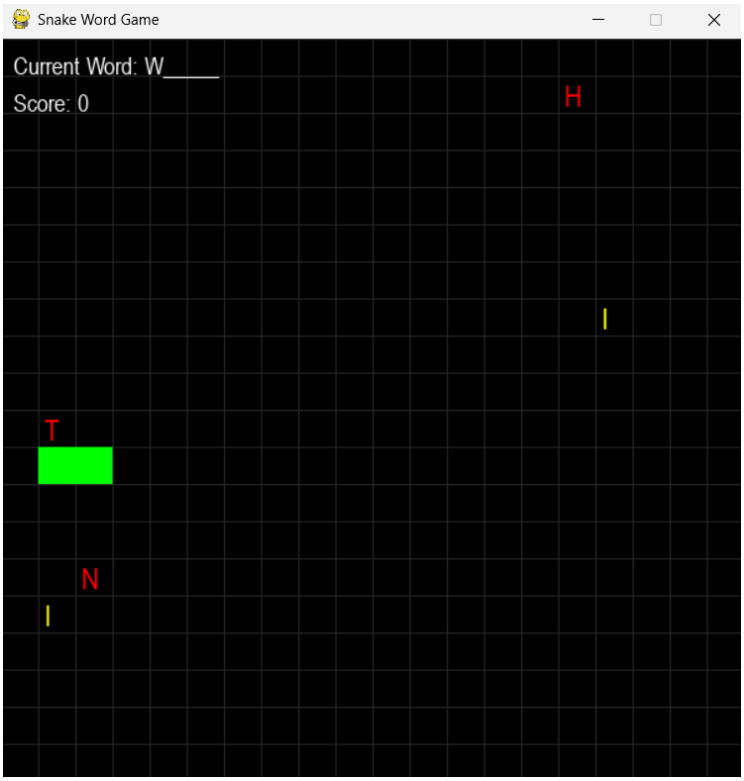
```
# Key handling for movement
    if keys[pygame.K_LEFT] and direction != (BLOCK_SIZE, 0):
        direction = (-BLOCK_SIZE, 0)
    elif keys[pygame.K_RIGHT] and direction != (-BLOCK_SIZE, 0):
        direction = (BLOCK_SIZE, 0)
    elif keys[pygame.K_UP] and direction != (0, BLOCK_SIZE):
        direction = (0, -BLOCK_SIZE)
    elif keys[pygame.K_DOWN] and direction != (0, -BLOCK_SIZE):
        direction = (0, BLOCK_SIZE)
```

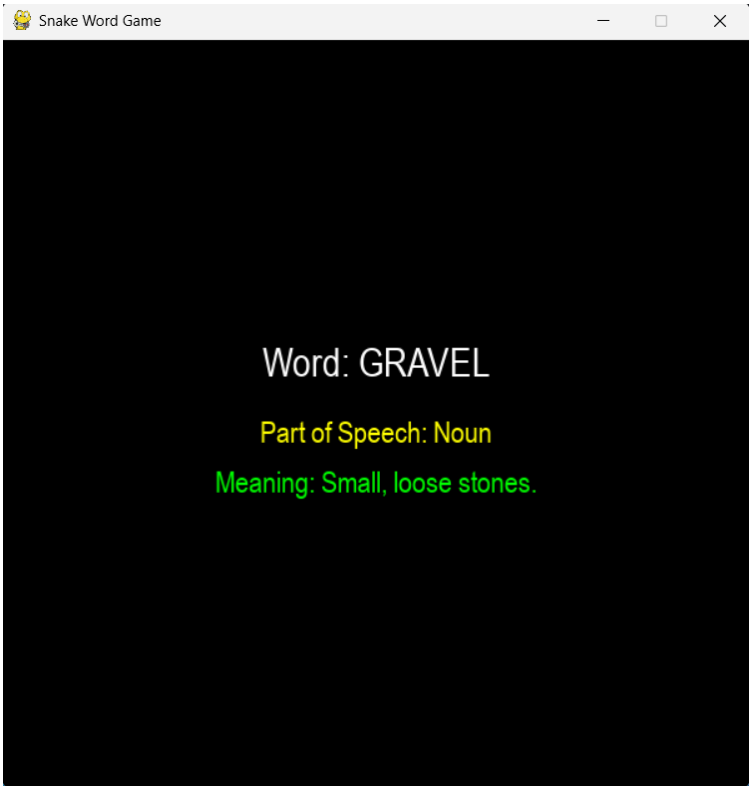
To highlight the expected word:

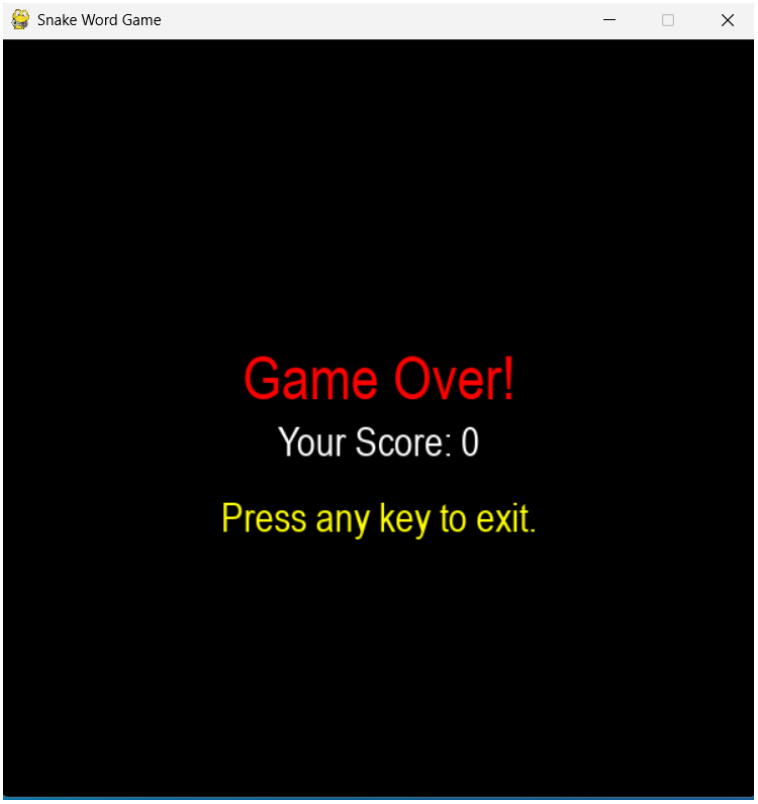
```
# Draw letters
font = pygame.font.SysFont('Arial', 24)
for pos, letter in letters.items():
# Highlight the next expected letter in YELLOW
    if letter == current_word[letter_index]:
        text = font.render(letter, True, YELLOW)
    else:
        text = font.render(letter, True, RED)

    win.blit(text, (pos[0] + 5, pos[1] + 2))
```

OUTPUT:






Snake Word Game
Word: GRAVEL
Part of Speech: Noun
Meaning: Small, loose stones.


Snake Word Game
Game Over!
Your Score: 0
Press any key to exit.