FLAG GAME SUMMARY REPORT

İLSU ERDAL-200209011

AHMET ALPEREN AKSOY-200209042

CONTENTS

INTRODUCTION OF THE PROJECT

The "Flag Guessing Game" stands as a captivating and educational quiz application crafted in Python, leveraging the Tkinter library for its development. This interactive game immerses users in flag-related questions and images, prompting them to select the correct answer from a set of multiple options. The primary objective behind this initiative is to spark curiosity among children about the diverse world they inhabit. By exposing them to flags from various countries, the aim is to instill a sense of tolerance and appreciation for global diversity, fostering an outlook that embraces different cultures and societies.

Furthermore, the game seeks to contribute to the enhancement of visual memory as children engage in the process of learning through an enjoyable and interactive experience. Beyond its educational aspect, the "Flag Guessing Game" also serves as a platform to provide an effective and entertaining examination experience. Through this project, we aspire to create a valuable tool that not only promotes cultural awareness but also makes the learning process engaging and memorable for young minds.

PROCESS

First,we did research to make the idea of exams fun for children and decided on our topic.

Secondly, we made our planning and task distribution.

Then, we started our coding, coding took about 2 days for completion.

Then,we analyzed and corrected the mistakes made.

Then, we prepared the project format as shown above.

TOOLS USED

NO	RESOURCES	DESCRIPTIONS
1	Programming Language	Python 3.9
2	GUI library	Tkinter
3	Image Processing	Pillow(PIL)
4	GUI Styling	ttkboostrap
5	Version Control	Git
6	Computer	At two computer run MacOs M1 12.0.1 , Ram 16 GB,

DESCRIPTION OF ENTIRE PROJECT

Flag Guessing Game has been meticulously crafted to offer users an engaging and entertaining quiz experience focused on the fascinating world of flags. Developed with a dynamic approach, the game works by randomly selecting questions from a carefully selected data set. Each question is carefully paired with an illustrative image of that flag, enhancing the visual learning experience.

If the user wants to learn about country flags and improve himself before starting the game, he can get a quick look at the country flags with a small exercise option.

Upon starting the game, users are presented with a series of flag-related queries and asked to choose the correct answer from a series of cleverly shuffled options for added challenge. One of the most important features of the game is the instant feedback mechanism. Users get quick answers, allowing them to learn and understand each flag, creating a deeper connection with the diverse world of international symbols.

The exam progresses gradually, moving smoothly from one question to the next until all questions are successfully answered. At the end of the test, users are presented with a final score that provides a comprehensive overview of their performance throughout the game.

This game serves not only as a source of entertainment but also as an effective educational tool. By combining entertainment and learning elements, the Flag Guessing Game aims to engage users, improve their knowledge of flags around the world, and contribute to a richer and more engaging learning experience. Thoughtful design and interactive features make this game an ideal choice for those looking for an engaging and educational trivia adventure.

WORKING PRINCIPLES

The project works according to the following principles:

- **Game Model Selection:** Starts the quiz or mini exercise.

- **Random Question Selection:** The game randomly selects questions to keep the quiz dynamic.

- **Scrambled Answer Options:** Options for each question are shuffled to prevent pattern recognition.

- **Instant Feedback:** Users receive instant feedback on the accuracy of their answers.

- **Score Tracking:** The user's score is updated after each correct answer.

- **Next Question Functionality:** After the user answers, the test moves on to the next question.

MOTIVATION

The driving force behind the development of the Guessing Flag Game was rooted in the desire to offer users more than just a quiz; it was crafted to provide an interactive and educational journey. Beyond being a mere test of flag knowledge, the game aspires to be a captivating learning experience, ensuring that users not only enjoy the process but also enhance their understanding of flags from around the world.

One of the key objectives is to make learning engaging and enjoyable. By incorporating visually appealing flag images, the game aims to create a seamless blend of entertainment and education, fostering a deeper connection with the subject matter. The random selection of questions ensures a diverse and challenging experience, making each round unique and informative.

Moreover, the Guessing Flag Game serves as a practical platform for users to enhance their Python programming skills, with a particular emphasis on graphical user interface (GUI) development using the Tkinter library. This aspect of the project allows users, especially those keen on programming, to apply and reinforce their coding abilities in a real-world scenario. In essence, the Guessing Flag Game goes beyond being a traditional quiz application; it's a multifaceted initiative that seeks to cultivate knowledge, spark curiosity, and provide a practical learning ground for Python enthusiasts. Whether users are focused on expanding their understanding of flags or honing their programming prowess, this game aims to deliver a holistic and enjoyable experience.

```python
import tkinter as tk
from tkinter import messagebox, ttk
import PIL
from ttkbootstrap import Style
from PIL import Image, ImageTk, ImageOps
import random
from quiz_data import quiz_data

def show_question():
    global current_question
    global used_questions
    global question_timer

    # Check if all questions have been used
    if len(used_questions) == len(quiz_data):
        messagebox.showinfo("Quiz Completed",
                            "Quiz Completed! Final score: {}/{}".format(score, len(quiz_data)))
        root.destroy()
        return

    # Choose a random question that hasn't been used yet
    while True:
        current_question = random.randint(0, len(quiz_data) - 1)
        if current_question not in used_questions:
            used_questions.add(current_question)
            break

    question = quiz_data[current_question]

    # Load the image
    image = Image.open(question["question"])

    # Calculate the border size based on the loaded image size
    max_border_size = 10  # Maximum border size
    border_size = min(max_border_size, image.width // 20, image.height // 20)

    # Add a black border around the image
    image = ImageOps.expand(image, border=border_size, fill=(0, 0, 0))

    # Resize the image as needed
    image = image.resize((200, 150), PIL.Image.Resampling.LANCZOS)  # Adjust the size as needed

    imgtk = ImageTk.PhotoImage(image)

    # Set the image to the label
    qs_label.config(image=imgtk)
    qs_label.image = imgtk
```

```
    # Shuffle the choices for the current question
    choices = question["choices"]
    random.shuffle(choices)

    # Display the shuffled choices on the buttons
    for i in range(len(choices)):
        choice_btns[i].config(text=choices[i], state="normal")

    # Clear the feedback label and disable the next button
    feedback_label.config(text="")
    next_btn.config(state="disabled")

    # Start the question timer
    question_timer = root.after(10000, time_up)

def check_answer(choice):
    # Cancel the question timer
    root.after_cancel(question_timer)

    # Get the current question from the quiz_data list
    question = quiz_data[current_question]
    selected_choice = choice_btns[choice].cget("text")

    # Check if the selected choice matches the correct answer
    if selected_choice == question["answer"]:
        # Update the score and display it
        global score
        score += 1
        score_label.config(text="Score: {}/{}".format(score, len(quiz_data)))
        feedback_label.config(text="Correct!", foreground="green")
    else:
        feedback_label.config(text="Incorrect!", foreground="red")

    # Disable all choice buttons and enable the next button
    for button in choice_btns:
        button.config(state="disabled")
    next_btn.config(state="normal")

def time_up():
    # If time is up, move to the next question
    feedback_label.config(text="Time's up!", foreground="red")
    next_question()
```

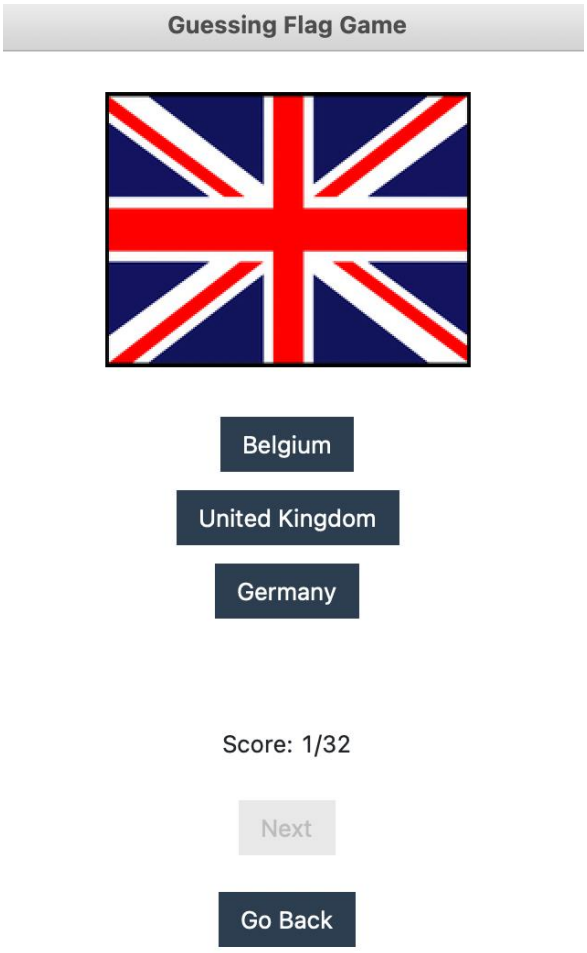
```
# Function to move to the next question
def next_question():
    # If all questions have been used, show the final score and end the quiz
    if len(used_questions) == len(quiz_data):
        messagebox.showinfo("Quiz Completed",
                            "Quiz Completed! Final score: {}/{}".format(score, len(quiz_data)))
        root.destroy()
    else:
        show_question()

# Function to switch to the quiz game window
def play_game():
    play_button.pack_forget()  # Hide the "Play Game" button
    initial_frame.pack_forget()
    game_frame.pack()
    show_question()  # Start the first question when the game begins

# Function to go back to the initial screen
def go_back():
    game_frame.pack_forget()
    play_button.pack()  # Show the "Play Game" button
    initial_frame.pack()

# Function to display information about how to play the game
def how_to_play():
    messagebox.showinfo("How to Play", "Guess the flag based on the given image. Select the correct answer within 10 seconds to earn points. Try to answer as many
        questions as you can!")

# Create the main window
root = tk.Tk()
root.title("Guessing Flag Game")
root.geometry("600x500")
style = Style(theme="flatly")

# Initial screen frame
initial_frame = ttk.Frame(root)
initial_frame.pack(pady=50)

# Create "PLAY" button to start the game
play_button = ttk.Button(initial_frame, text="PLAY", command=play_game)
play_button.pack(pady=10)

# Create "How to Play" button
how_to_play_button = ttk.Button(initial_frame, text="How to Play", command=how_to_play)
how_to_play_button.pack(pady=10)
```

```
# Quiz game frame
game_frame = ttk.Frame(root)

# Create the question label
qs_label = ttk.Label(
    game_frame,
    anchor="center",
    wraplength=500,
    padding=10
)
qs_label.pack(pady=10)

# Create the choice buttons
choice_btns = []
for i in range(3):
    button = ttk.Button(
        game_frame,
        command=lambda i=i: check_answer(i)
    )
    button.pack(pady=5)
    choice_btns.append(button)

# Create the feedback label
feedback_label = ttk.Label(
    game_frame,
    anchor="center",
    padding=1
)
feedback_label.pack(pady=10)

# Initialize the score
score = 0

# Create the score label
score_label = ttk.Label(
    game_frame,
    text="Score: 0/{}".format(len(quiz_data)),
    anchor="center",
    padding=1
)
score_label.pack(pady=10)
```

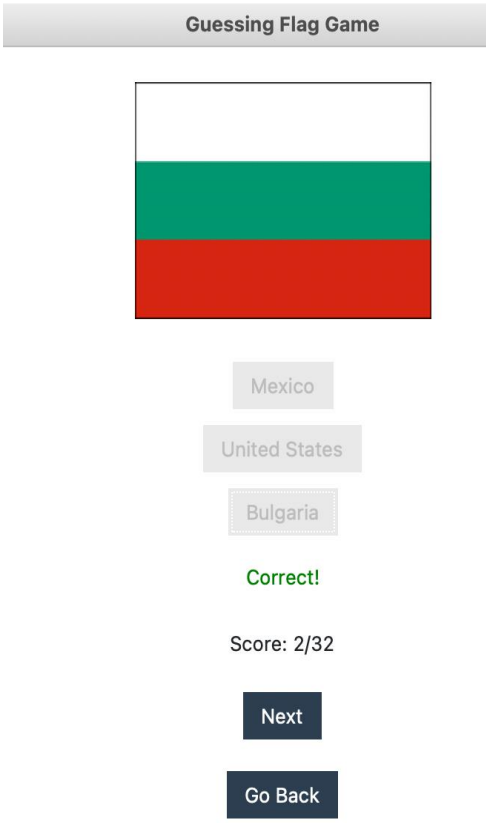
```
)
next_btn.pack(pady=10)

# Create the "Go Back" button
go_back_btn = ttk.Button(
    game_frame,
    text="Go Back",
    command=go_back
)
go_back_btn.pack(pady=10)

# Initialize the current question index
current_question = 0

# Track used questions
used_questions = set()

# Variable to store the question timer ID
question_timer = None

# Start the main event loop
root.mainloop()
```

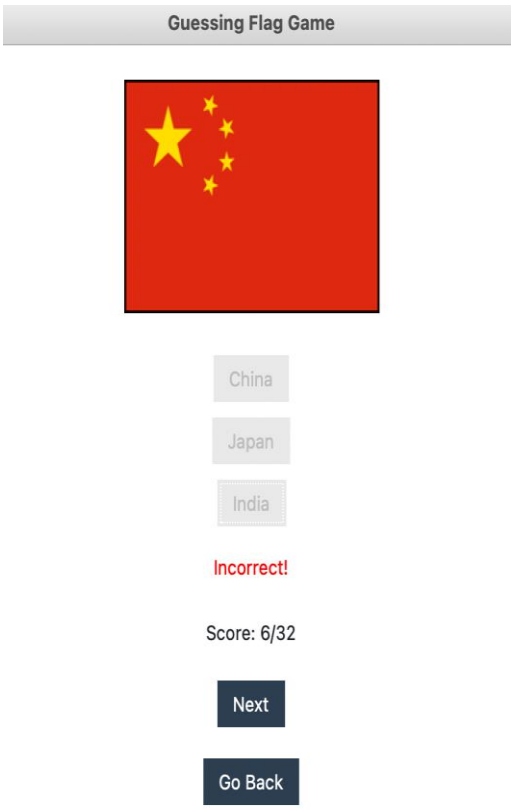
START SCREEN

WHEN CLICK ON PLAY BUTTON

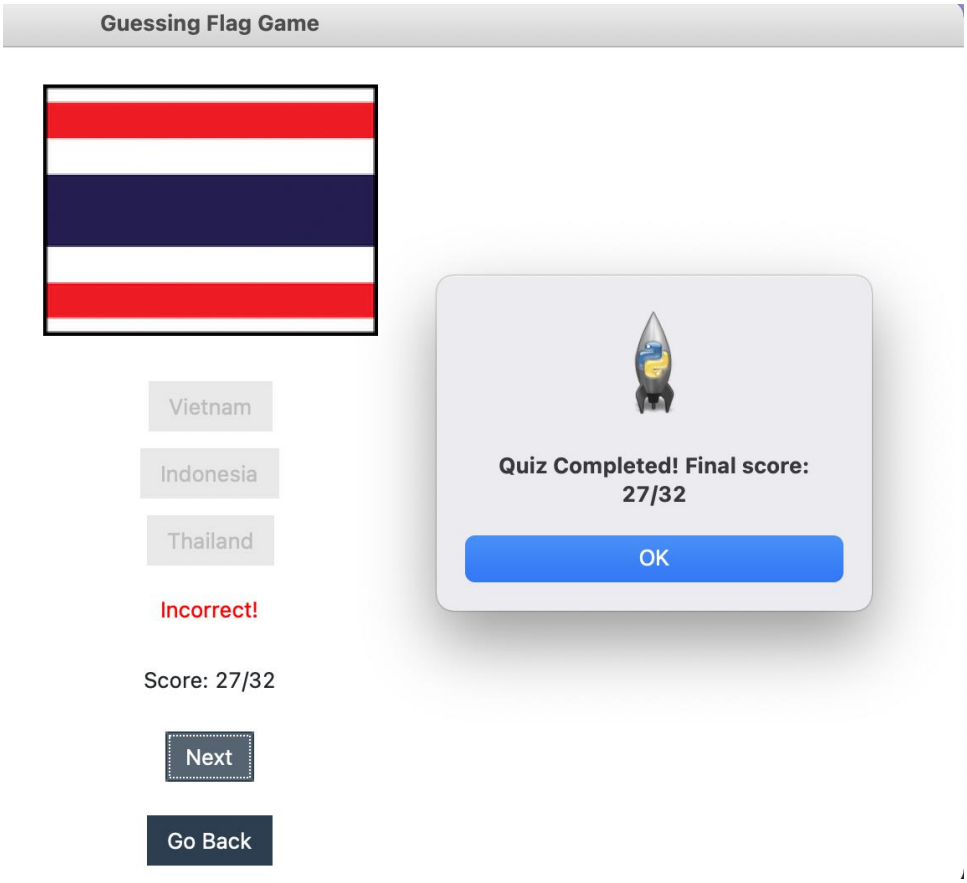


WHEN CLICK THE CORRECT ANSWER	WHEN CLICK THE WRONG ANSWER





FINISH THE GAME



PART OF EXERCİSE

```python
import tkinter as tk
from tkinter import ttk, messagebox
from ttkbootstrap import Style
import PIL.Image
import random
from quiz_data import quiz_data
from PIL import Image, ImageTk, ImageOps
class Exercise:
    def __init__(self, root):
        self.root = root
        self.setup_ui()

    def setup_ui(self):
        self.frame = ttk.Frame(self.root)
        self.frame.pack(pady=50)

        # Create the question label
        self.qs_label = ttk.Label(
            self.frame,
            anchor="center",
            wraplength=500,
            padding=10
        )
        self.qs_label.pack(pady=10)

        # Create the answer label with a larger font size
        self.answer_label = ttk.Label(
            self.frame,
            anchor="center",
            padding=1,
            font=("Helvetica", 31)  # Adjust the font size as needed
        )
        self.answer_label.pack(pady=10)

        # Create the next button
        self.next_btn = ttk.Button(
            self.frame,
            text="Next",
            command=self.show_question,
            state="normal"
        )
        self.next_btn.pack(pady=10)

        # Initialize the current question index
        self.current_question = 0

```

```
        # Track used questions
        self.used_questions = set()

        # Show the first question
        self.show_question()
    def show_question(self):
        # Check if all questions have been used
        if len(self.used_questions) == len(quiz_data):
            messagebox.showinfo("Exercise Completed", "Exercise Completed!")
            self.root.destroy()
            return

        # Choose a random question that hasn't been used yet
        while True:
            self.current_question = random.randint(0, len(quiz_data) - 1)
            if self.current_question not in self.used_questions:
                self.used_questions.add(self.current_question)
                break

        # Load the image
        image = Image.open(quiz_data[self.current_question]["question"])

        # Calculate the border size based on the loaded image size
        max_border_size = 10  # Maximum border size
        border_size = min(max_border_size, image.width // 20, image.height // 20)

        # Add a black border around the image
        image = ImageOps.expand(image, border=border_size, fill=(0, 0, 0))

        # Resize the image as needed
        image = image.resize((200, 150), Image.LANCZOS)  # Adjust the size as needed

        imgtk = ImageTk.PhotoImage(image)

        # Set the image to the label
        self.qs_label.config(image=imgtk)
        self.qs_label.image = imgtk

        # Set the answer to the label
        answer_text = " {}".format(quiz_data[self.current_question]["answer"])
        self.answer_label.config(text=answer_text)

        # Enable the next button
        self.next_btn.config(state="normal")

```

```
91
92  # Create the main window
93  root = tk.Tk()
94  root.title("Exercise Game")
95  root.geometry("600x500")
96  style = Style(theme="flatly")
97
98  # Create an instance of the Exercise class
99  exercise_game = Exercise(root)
100
101 # Start the main event loop
102 root.mainloop()
103
```

South Africa

Next

Go Back

REFERENCES

1.Python Official Documentation:
https://www.python.org/doc/
2.Tkinter Documentation:
https://docs.python.org/3/library/tkinter.html
3.Pillow Documentation:
https://pillow.readthedocs.io/en/stable/
4.ttkbootstrap Documentation:
https://github.com/TkinterEP/ttkbootstrap