**quiz\_brain.py:**

class QuizBrain:

def \_\_init\_\_(self, questions):

self.question\_no = 0

self.score = 0

self.questions = questions

self.current\_question = None

def has\_more\_questions(self):

"""To check if the quiz has more questions"""

return self.question\_no < len(self.questions)

def next\_question(self):

"""Get the next question by incrementing the question number"""

self.current\_question = self.questions[self.question\_no]

self.question\_no += 1

q\_text = self.current\_question.question\_text

return f"Q.{self.question\_no}: {q\_text}"

def check\_answer(self, user\_answer):

"""Check the user's answer against the correct answer and maintain the score"""

correct\_answer = self.current\_question.correct\_answer

if user\_answer.lower() == correct\_answer.lower():

self.score += 1

return True

else:

return False

def get\_score(self):

"""Get the number of correct answers, wrong answers, and score percentage."""

wrong = self.question\_no - self.score

score\_percent = int(self.score / self.question\_no \* 100)

return (self.score, wrong, score\_percent)

**quiz\_data.py:**

import requests

parameters = {

"amount": 5,

"type": "multiple"

}

response = requests.get(url="https://opentdb.com/api.php?amount=5&category=9&difficulty=medium&type=multiple", params=parameters)

question\_data = response.json()["results"]

**quiz\_ui.py:**

from tkinter import Tk, Canvas, StringVar, Label, Radiobutton, Button, messagebox

from quiz\_brain import QuizBrain

THEME\_COLOR = "#375362"

class QuizInterface:

def \_\_init\_\_(self, quiz\_brain: QuizBrain) -> None:

self.quiz = quiz\_brain

self.window = Tk()

self.window.title("iQuiz App")

self.window.geometry("850x530")

# Display Title

self.display\_title()

# Create a canvas for question text, and dsiplay question

self.canvas = Canvas(width=800, height=250)

self.question\_text = self.canvas.create\_text(400, 125,

text="Question here",

width=680,

fill=THEME\_COLOR,

font=(

'Ariel', 15, 'italic')

)

self.canvas.grid(row=2, column=0, columnspan=2, pady=50)

self.display\_question()

# Declare a StringVar to store user's answer

self.user\_answer = StringVar()

# Display four options (radio buttons)

self.opts = self.radio\_buttons()

self.display\_options()

# To show whether the answer is right or wrong

self.feedback = Label(self.window, pady=10, font=("ariel", 15, "bold"))

self.feedback.place(x=300, y=380)

# Next and Quit Button

self.buttons()

# Mainloop

self.window.mainloop()

def display\_title(self):

"""To display title"""

# Title

title = Label(self.window, text="iQuiz Application",

width=50, bg="green", fg="white", font=("ariel", 20, "bold"))

# place of the title

title.place(x=0, y=2)

def display\_question(self):

"""To display the question"""

q\_text = self.quiz.next\_question()

self.canvas.itemconfig(self.question\_text, text=q\_text)

def radio\_buttons(self):

"""To create four options (radio buttons)"""

# initialize the list with an empty list of options

choice\_list = []

# position of the first option

y\_pos = 220

# adding the options to the list

while len(choice\_list) < 4:

# setting the radio button properties

radio\_btn = Radiobutton(self.window, text="", variable=self.user\_answer,

value='', font=("ariel", 14))

# adding the button to the list

choice\_list.append(radio\_btn)

# placing the button

radio\_btn.place(x=200, y=y\_pos)

# incrementing the y-axis position by 40

y\_pos += 40

# return the radio buttons

return choice\_list

def display\_options(self):

"""To display four options"""

val = 0

# deselecting the options

self.user\_answer.set(None)

# looping over the options to be displayed for the

# text of the radio buttons.

for option in self.quiz.current\_question.choices:

self.opts[val]['text'] = option

self.opts[val]['value'] = option

val += 1

def next\_btn(self):

"""To show feedback for each answer and keep checking for more questions"""

# Check if the answer is correct

if self.quiz.check\_answer(self.user\_answer.get()):

self.feedback["fg"] = "green"

self.feedback["text"] = 'Correct answer! \U0001F44D'

else:

self.feedback['fg'] = 'red'

self.feedback['text'] = ('\u274E Oops! \n'

f'The right answer is: {self.quiz.current\_question.correct\_answer}')

if self.quiz.has\_more\_questions():

# Moves to next to display next question and its options

self.display\_question()

self.display\_options()

else:

# if no more questions, then it displays the score

self.display\_result()

# destroys the self.window

self.window.destroy()

def buttons(self):

"""To show next button and quit button"""

# The first button is the Next button to move to the

# next Question

next\_button = Button(self.window, text="Next", command=self.next\_btn,

width=10, bg="green", fg="white", font=("ariel", 16, "bold"))

# palcing the button on the screen

next\_button.place(x=350, y=460)

# This is the second button which is used to Quit the self.window

quit\_button = Button(self.window, text="Quit", command=self.window.destroy,

width=5, bg="red", fg="white", font=("ariel", 16, " bold"))

# placing the Quit button on the screen

quit\_button.place(x=700, y=50)

def display\_result(self):

"""To display the result using messagebox"""

correct, wrong, score\_percent = self.quiz.get\_score()

correct = f"Correct: {correct}"

wrong = f"Wrong: {wrong}"

# calculates the percentage of correct answers

result = f"Score: {score\_percent}%"

# Shows a message box to display the result

messagebox.showinfo("Result", f"{result}\n{correct}\n{wrong}")

**question\_model.py:**

class Question:

def \_\_init\_\_(self, question: str, correct\_answer: str, choices: list):

self.question\_text = question

self.correct\_answer = correct\_answer

self.choices = choices

**main.py:**

from question\_model import Question

from quiz\_data import question\_data

from quiz\_brain import QuizBrain

from quiz\_ui import QuizInterface

from random import shuffle

import html

question\_bank = []

for question in question\_data:

choices = []

question\_text = html.unescape(question["question"])

correct\_answer = html.unescape(question["correct\_answer"])

incorrect\_answers = question["incorrect\_answers"]

for ans in incorrect\_answers:

choices.append(html.unescape(ans))

choices.append(correct\_answer)

shuffle(choices)

new\_question = Question(question\_text, correct\_answer, choices)

question\_bank.append(new\_question)

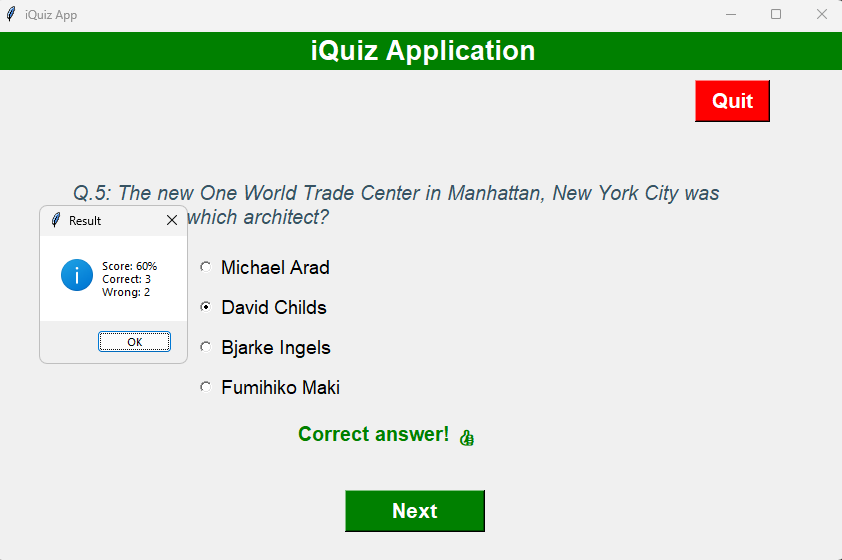
quiz = QuizBrain(question\_bank)

quiz\_ui = QuizInterface(quiz)

print("You've completed the quiz")

print(f"Your final score was: {quiz.score}/{quiz.question\_no}")

**OUTPUT:**

****