WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Programowanie

Patryk Pieniążek
Nr albumu studenta 67174

Aplikacja desktopowa w języku Python

Prowadzący: dr inż. Leszek Gajecki

Projekt

Rzeszów 2024

Spis treści

Rozdział 1

Opisy, założenia i wymagania projektu

1.1 Założenia i cele projektu

Stworzenie interaktywnej aplikacji quizowej z graficznym interfejsem użytkownika, umożliwiającej użytkownikom rozwiązywanie quizów z różnych kategorii i o różnym poziomie trudności.

1.2 Główne założenia

- Pobieranie pytań z zewnętrznego API(Open Trivia Database)
- Możliwość wyboru liczby pytań, kategorii i poziomu trudności
- Prezentacja pytań i odpowiedzi w formie graficznej
- Zliczanie punktów i wyświetlanie wyniku końcowego

1.3 Wymagania techniczne

- Python (na ten moment 3.12.4)
- system Windows (customtkinter może nie działać na innych platformach)
- paczki pip: requests, customtkinter

1.4 Instrukcja uruchomienia aplikacji

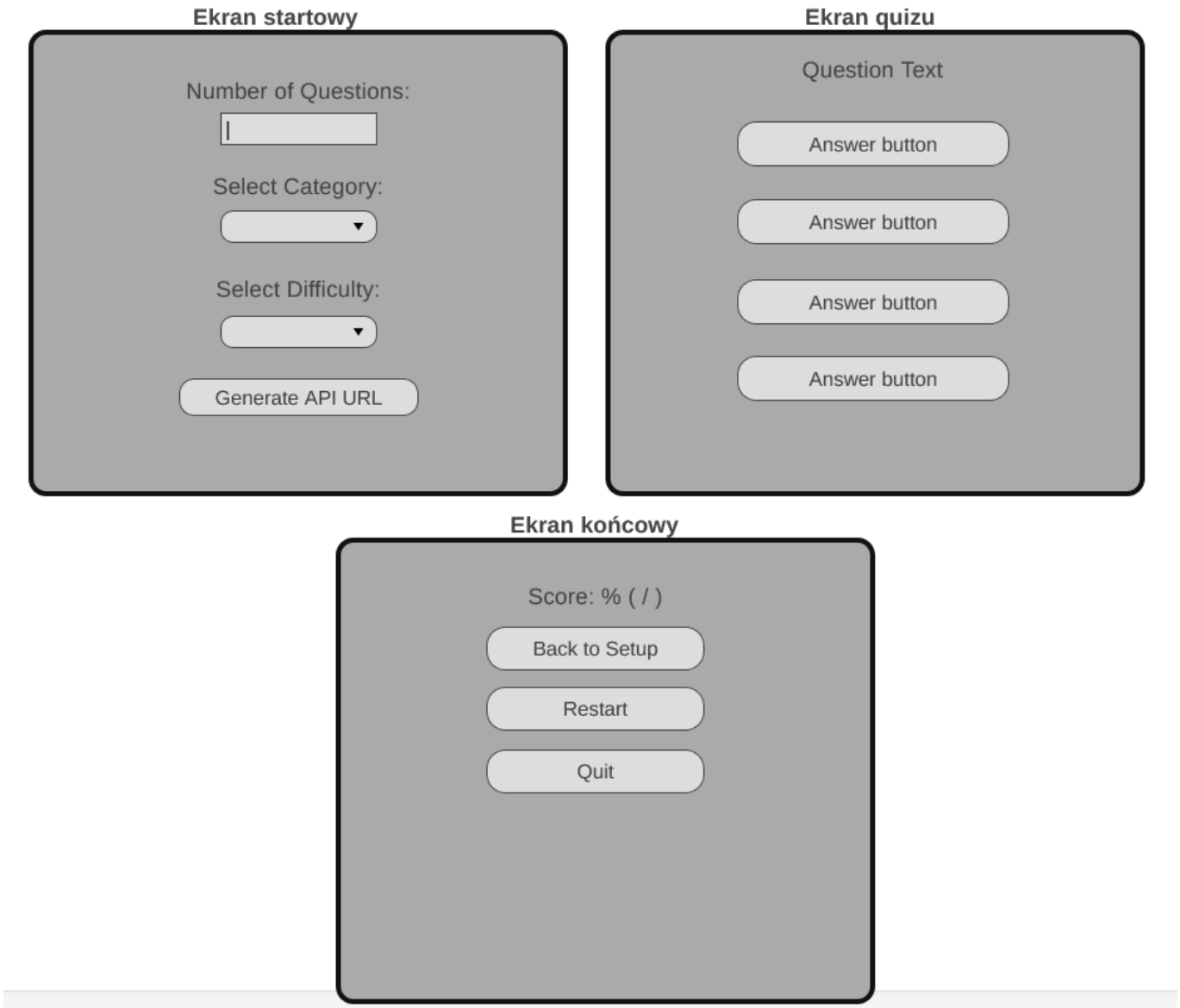
Aby poprawnie pobrać i uruchomić aplikację trzeba wykonać poniższe kroki:

- Pobrać interpreter Python z oficjalnej strony.
- W terminalu wykonać komendę "git clone https://github.com/ppieniazek/st1.git"
- Przejść do głównego folderu repozytorium i wykonać komendę "pip install -r requirements.txt"
- Uruchomić aplikację komendą "py main.py"

Rozdział 2

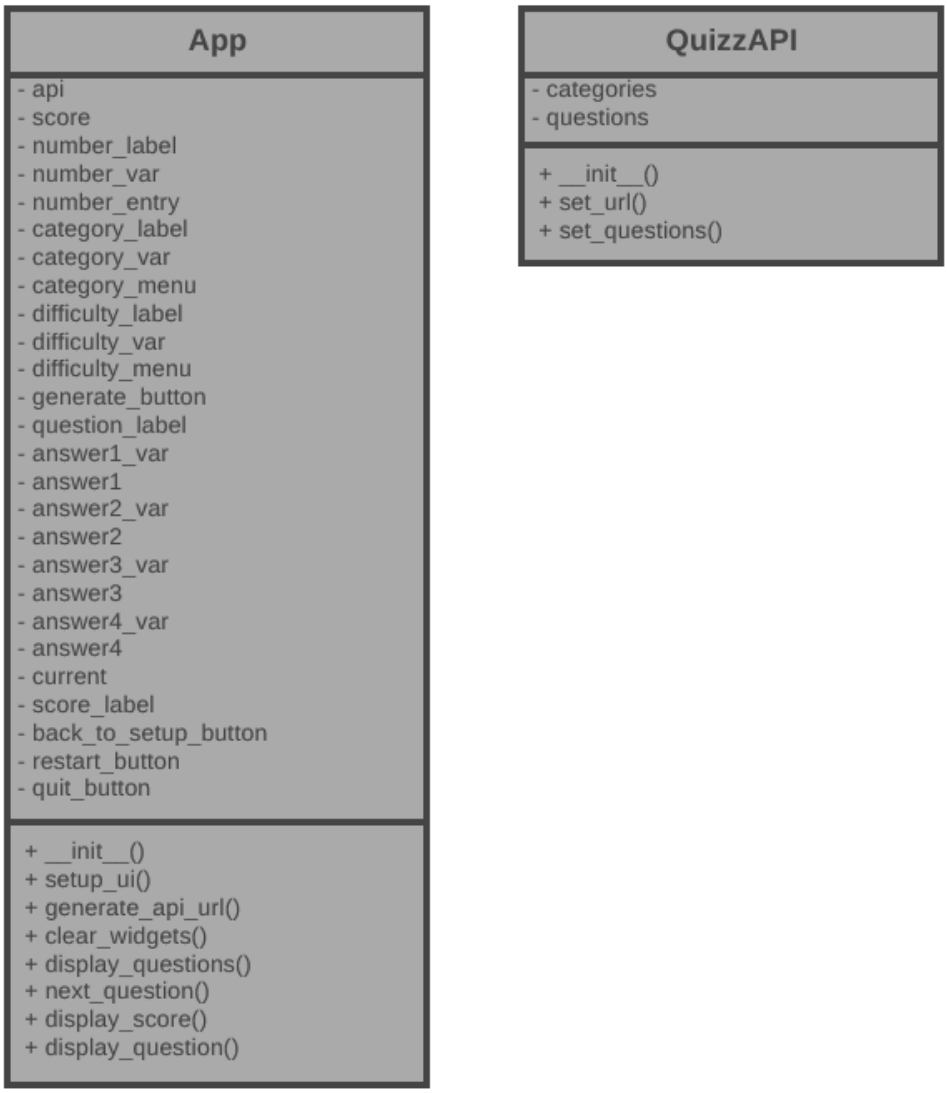
Projekt systemu i implementacja

2.1 Projekt GUI



Rysunek 2.1: Projekt GUI

2.2 Diagram klas



Rysunek 2.2: Diagram klas

2.3 Implementacja - ciekawsze elementy

```python
def set_questions(self):
    response = requests.get(self.url)
    questions = response.json()["results"]

    # Decoding html encoding
    for question in questions:
        question["question"] = html.unescape(question["question"])
        if "incorrect_answers" in question:
            question["incorrect_answers"] = [
                html.unescape(answer) for answer in question["incorrect_answers"]
            ]
        if "correct_answer" in question:
            question["correct_answer"] = html.unescape(question["correct_answer"])

    self.questions = questions
```

Rysunek 2.3: Pobieranie pytań z API

```python
def display_question(self):
    self.question_label.configure(text=self.api.questions[self.current]["question"])
    answers = self.api.questions[self.current]["incorrect_answers"][:]
    answers.append(self.api.questions[self.current]["correct_answer"])

    random.shuffle(answers)
    for answer, button_var in zip(
        answers,
        [
            self.answer1_var,
            self.answer2_var,
            self.answer3_var,
            self.answer4_var,
        ],
    ):
        button_var.set(answer)
```

Rysunek 2.4: Losowe wyświetlanie odpowiedzi

```python
def generate_api_url(self):
    try:
        q_num = int(self.number_entry.get())
    except ValueError:
        self.number_var.set("Enter a valid number!")
        return
    category = None if self.category_var.get() == "Any" else self.category_var.get()
    difficulty = (
        None if self.difficulty_var.get() == "Any" else self.difficulty_var.get()
    )

    self.api.set_url(q_num, category, difficulty)
    self.api.set_questions()

    self.clear_widgets()
    self.display_questions()
```

Rysunek 2.5: Generowanie url do API-1

```python
def set_url(self, q_num, category=None, difficulty=None):
    self.url += f"amount={q_num}&type=multiple"
    if category:
        for item in self.categories:
            if item["name"] == category:
                self.url += f"&category={item["id"]}"
                break

    if difficulty:
        self.url += f"&difficulty={difficulty.lower()}"
```

Rysunek 2.6: Generowanie url do API-2

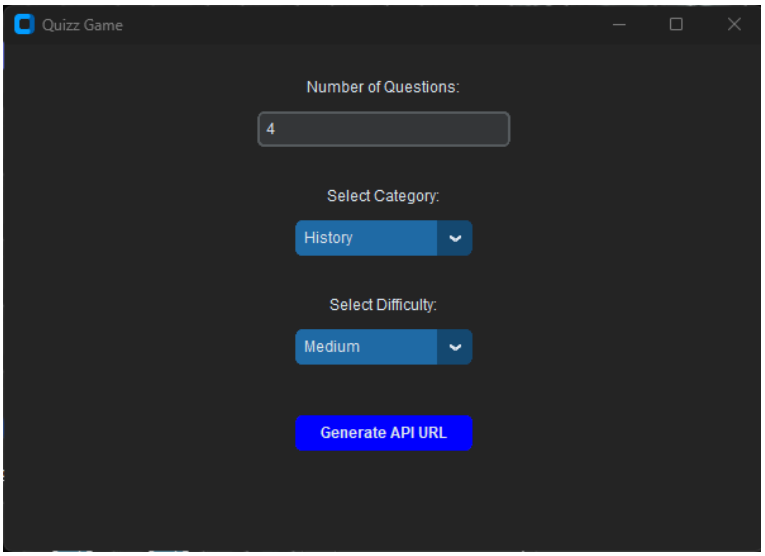
Rozdział 3

Działanie aplikacji

3.1 Screeny z jednego cyklu aplikacji

3.1.1 Menu startowe

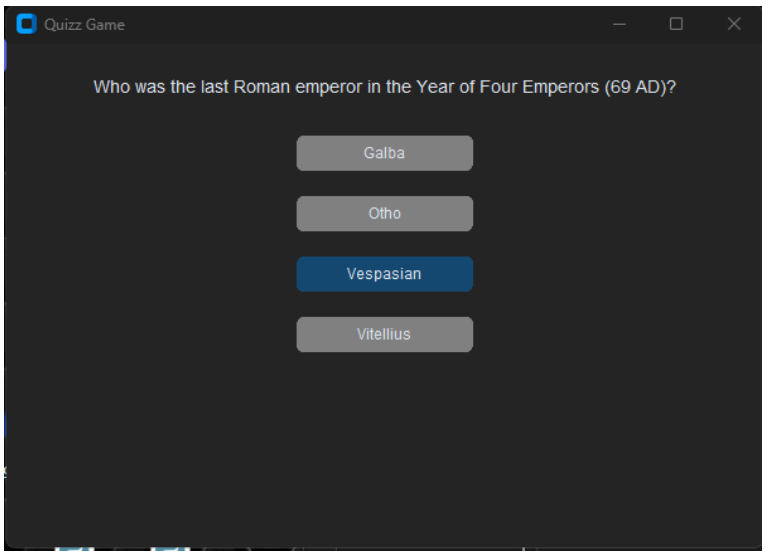
Przy starcie aplikacji wyświetlane jest menu z opcją wyboru liczby pytań, kategorii i trudności.



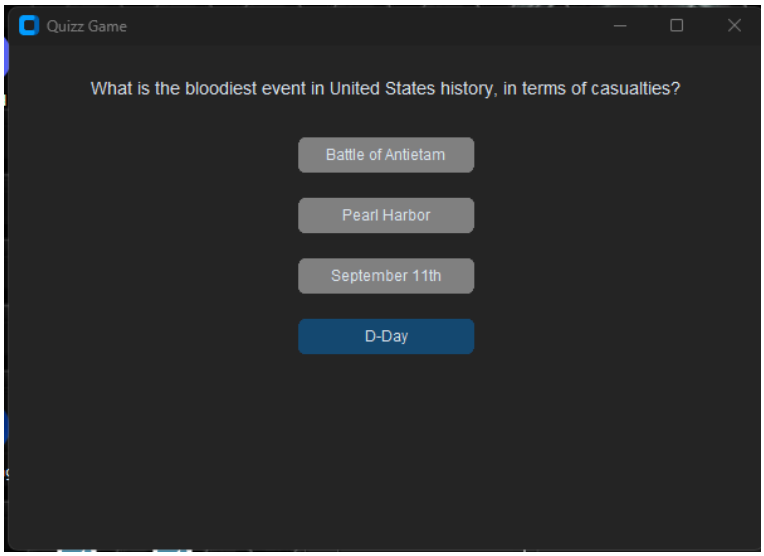
Rysunek 3.1: Startowe menu

3.1.2 Wyświetlanie pytań

Przy wyświetlaniu każdego pytania program czeka na kliknięcie w któryś z 4 przycisków odpowiedzi, po czym przechodzi do kolejnego pytania.



Rysunek 3.2: Wyświetlanie pytania-1

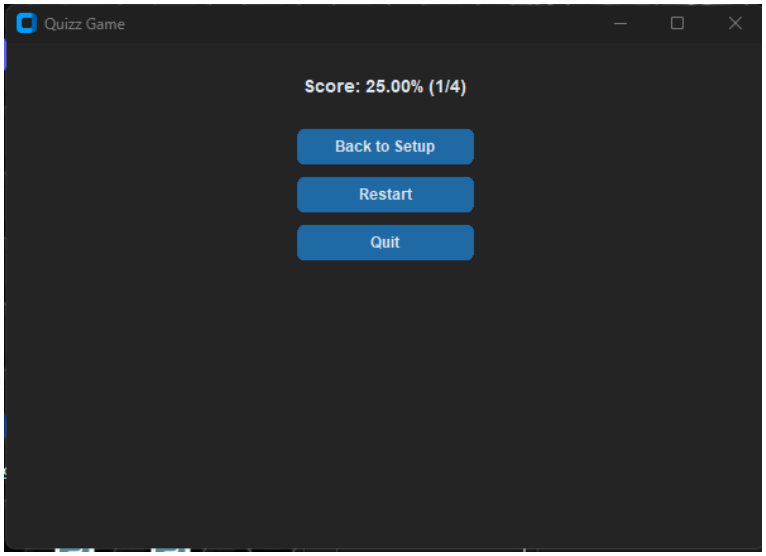


Rysunek 3.3: Wyświetlanie pytania-2

3.1.3 Menu końcowe

Po wyświetleniu wszystkich pytań aplikacja wyświetla wynik i 3 opcje dalszego działania:

- Przycisk powrotu do startowego menu
- Restart - czyli pobranie nowych pytań z tymi samymi parametrami wybranymi na początku
- Quit - wyjście



Rysunek 3.4: Końcowe menu

Podsumowanie

Aplikacja skutecznie realizuje podstawowe funkcje zgodnie z założeniami, umożliwiając użytkownikowi interakcję z zewnętrznym API do pobierania pytań quizowych. Komunikacja z API jest płynna i niezawodna, co stanowi solidną bazę dla funkcjonalności quizu. Jednakże, aplikacja w obecnej formie jest daleka od produktu końcowego o wysokiej jakości i wymaga dalszego rozwoju oraz dopracowania.

Bibliografia

[1] https://opentdb.com/ z dnia 28.06.2024

[2] https://github.com/TomSchimansky/CustomTkinter z dnia 28.06.2024

[3] https://docs.python-requests.org/ z dnia 28.06.2024

[4] https://www.youtube.com/watch?v=mop6g-c5HEY (fragmenty) z dnia 28.06.2024

Spis rysunków