

## **Dokumentacja techniczna systemu**

### **QuizVerse - Platforma do tworzenia i rozwiązywania quizów**

#### **Rozdział 1**

##### **Cel i zakres systemu**

###### **1.1 Cel projektu**

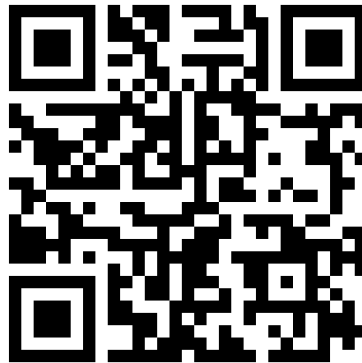
Celem systemu **QuizVerse** jest dostarczenie kompletnej platformy webowej umożliwiającej:

- tworzenie quizów przez zarejestrowanych użytkowników,
- przypisywanie quizów do kategorii tematycznych,
- definiowanie pytań oraz odpowiedzi,
- rozwiązywanie quizów przez użytkowników,
- automatyczne obliczanie wyników,
- zapisywanie historii rozwiązań,
- prowadzenie rankingów użytkowników,
- komentowanie i ocenianie quizów.

Aplikacja została zaprojektowana jako system rozproszony, w którym każdy główny komponent działa w osobnym kontenerze Docker. Komunikacja pomiędzy kontenerami odbywa się w ramach jednej, zdefiniowanej sieci Docker. Backend udostępnia API HTTP, frontend odpowiada za interfejs użytkownika, a baza danych przechowuje dane trwałe. Dodatkowo dostępny jest kontener z narzędziem administracyjnym pgAdmin, służącym do zarządzania bazą danych w trakcie developmentu.

Architektura nie zakłada bezpośredniego dostępu frontendu do bazy danych. Wszystkie operacje na danych realizowane są poprzez backend, co upraszcza kontrolę logiki biznesowej oraz bezpieczeństwo.

Projekt realizowany był przez pięcioosobowy zespół, przy wykorzystaniu platformy *GitHub* jako głównego narzędzia do zarządzania kodem źródłowym, wersjonowania zmian oraz koordynacji pracy zespołowej. Repozytorium projektu udostępniono pod adresem: <https://github.com/Xeliq/QuizVerse> (rys. 1.1.1). Zastosowanie systemu kontroli wersji Git umożliwiło równoległą pracę wielu programistów poprzez mechanizm gałęzi a także pełne śledzenie historii zmian. Platforma GitHub wspierała proces integracji kodu oraz ułatwiała identyfikację i rozwiązywanie problemów co znacząco usprawniło organizację pracy i zapewniło spójność rozwijanego oprogramowania.



Rys. 1.1.1. Kod QR zawierający adres repozytorium projektu

Ze względu na fakt, iż projekt ma charakter akademicki i jego celem jest demonstracja umiejętności programistycznych, a nie realizacja aplikacji dla rzeczywistego klienta, treści tekstowe prezentowane w interfejsie użytkownika (niebędące elementem kodu źródłowego) mają charakter przykładowy. W związku z tym przy ich tworzeniu wspomagano się narzędziem *ChatGPT*.

## 1.2 Zakres funkcjonalny

System obejmuje następujące obszary funkcjonalne:

- zarządzanie użytkownikami (rejestracja, logowanie, profil),
- zarządzanie quizami (CRUD),
- zarządzanie pytaniami i odpowiedziami,
- obsługę wyników quizów,
- system komentarzy i ocen,
- prezentację rankingów,
- obsługę kategorii,

- konteneryzację środowiska uruchomieniowego.

## Rozdział 2

### Architektura systemu

#### 2.1 Architektura logiczna

System oparty jest o architekturę **trójwarstwową**:

1. **Warstwa prezentacji (Frontend)**

Odpowiada za interakcję z użytkownikiem, renderowanie widoków oraz komunikację z API.

2. **Warstwa logiki biznesowej (Backend)**

Realizuje reguły biznesowe, walidację danych, autoryzację oraz dostęp do bazy danych.

3. **Warstwa danych (Database)**

Przechowuje trwałe dane aplikacji w relacyjnej bazie danych PostgreSQL.

Komunikacja pomiędzy frontendem a backendem odbywa się za pomocą protokołu HTTP i formatu JSON.

#### 2.2 Architektura fizyczna

System uruchamiany jest w środowisku kontenerowym Docker i składa się z następujących usług:

- kontener aplikacji backendowej (Laravel),
- kontener aplikacji frontendowej (Vue + Vite),
- kontener bazy danych PostgreSQL,
- kontener narzędzia administracyjnego pgAdmin.

Każda usługa działa w tej samej sieci Docker (quizverse), co umożliwia bezpośrednią komunikację pomiędzy kontenerami.

## Rozdział 3

### Środowisko uruchomieniowe i Docker

#### 3.1 Docker Compose - opis usług

##### 3.1.1 Usługa app (Backend)

- Technologia: PHP + Laravel
- Port: 8000
- Odpowiedzialność:
  - obsługa REST API,
  - logika biznesowa,
  - komunikacja z bazą danych,
  - autoryzacja użytkowników.

Kod aplikacji montowany jest jako wolumen, co umożliwia rozwój w trybie developerskim.

### **3.1.2 Usługa frontend**

- Technologia: Vue 3 + Vite
- Port: 5173
- Odpowiedzialność:
  - renderowanie interfejsu użytkownika,
  - obsługa routingu po stronie klienta,
  - komunikacja z API backendu.

### **3.1.3 Usługa db**

- Technologia: PostgreSQL 16
- Port: 5432
- Odpowiedzialność:
  - trwałe przechowywanie danych systemu.

Dane przechowywane są w wolumenie pgdata, co zapewnia ich trwałość.

### **3.1.4 Usługa pgadmin**

PgAdmin działa jako osobny kontener i służy wyłącznie do administracji bazą danych. Jest on dostępny przez przeglądarkę na porcie 5050. Połączenie z bazą realizowane jest wewnątrz tej samej sieci Docker.

- Technologia: pgAdmin 4
- Port: 5050
- Odpowiedzialność:
  - administracja bazą danych,
  - wizualny podgląd struktur i rekordów.

## Rozdział 4

### Baza danych

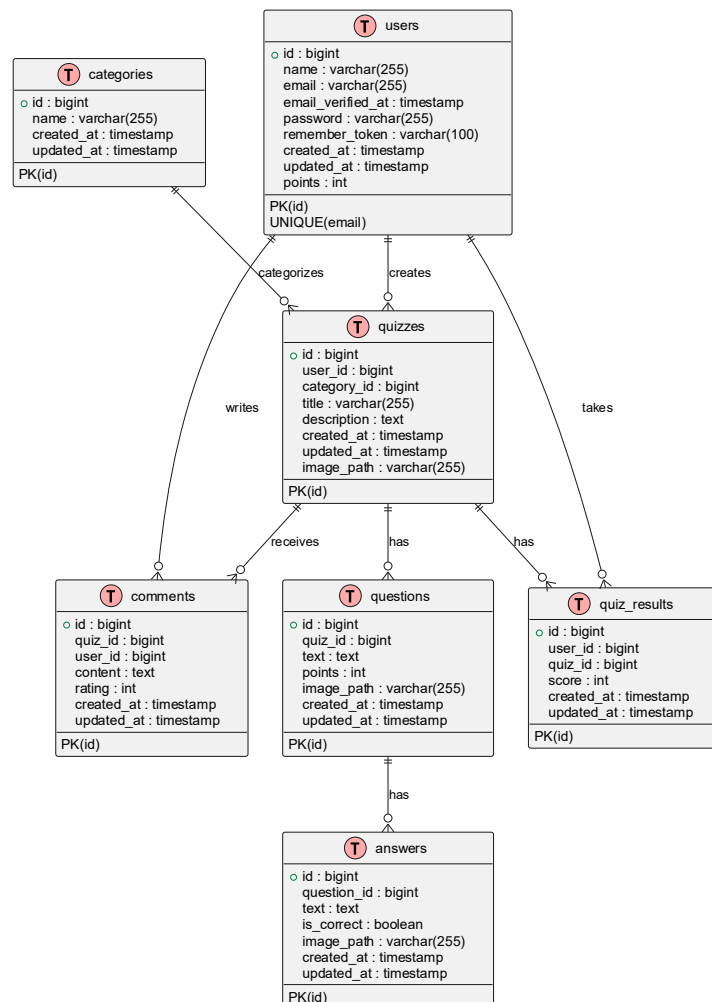
#### 4.1 Model danych

System wykorzystuje relacyjną bazę danych PostgreSQL. Model danych został zaprojektowany w sposób zapewniający:

- spójność referencyjną,
- eliminację redundancji,
- jednoznaczne relacje między encjami.

#### 4.2 Opis tabel

Na rysunku 4.2 przedstawiono schemat tabel bazy danych.



Rysunek 4.2. Schemat tabel bazy danych

##### 4.2.1 Tabela users

Przechowuje dane użytkowników systemu.

Kolumny:

- id - klucz główny,
- name - nazwa użytkownika,
- email - unikalny adres e-mail,
- password - hasło w postaci zaszyfrowanej,
- points - łączna liczba punktów użytkownika,
- created\_at, updated\_at.

Relacje:

- jeden użytkownik może utworzyć wiele quizów,
- jeden użytkownik może mieć wiele wyników quizów,
- jeden użytkownik może dodać wiele komentarzy.

#### **4.2.2 Tabela categories**

Przechowuje kategorie quizów.

Kolumny:

- id,
- name,
- created\_at, updated\_at.

Relacje:

- jedna kategoria może zawierać wiele quizów.

#### **4.2.3 Tabela quizzes**

Reprezentuje quizy tworzone przez użytkowników.

Kolumny:

- id,
- user\_id,
- category\_id,
- title,
- description,
- image\_path,
- created\_at, updated\_at.

Relacje:

- quiz należy do jednego użytkownika,

- quiz należy do jednej kategorii,
- quiz posiada wiele pytań,
- quiz posiada wiele wyników,
- quiz posiada wiele komentarzy.

#### **4.2.4 Tabela questions**

Przechowuje pytania przypisane do quizów.

Kolumny:

- id,
- quiz\_id,
- text,
- points,
- image\_path.

Relacje:

- pytanie należy do jednego quizu,
- pytanie posiada wiele odpowiedzi.

#### **4.2.5 Tabela answers**

Przechowuje odpowiedzi do pytań.

Kolumny:

- id,
- question\_id,
- text,
- is\_correct,
- image\_path.

Relacje:

- odpowiedź należy do jednego pytania.

#### **4.2.6 Tabela quiz\_results**

Przechowuje wyniki rozwiązywania quizów.

Kolumny:

- id,
- user\_id,

- quiz\_id,
- score,
- created\_at.

Relacje:

- wynik należy do użytkownika,
- wynik należy do quizu.

#### **4.2.7 Tabela comments**

Przechowuje komentarze i oceny quizów.

Kolumny:

- id,
- quiz\_id,
- user\_id,
- content,
- rating,
- created\_at.

## **Rozdział 5**

### **Backend - Laravel**

#### **5.1 Struktura backendu**

Backend aplikacji został zrealizowany przy użyciu frameworku Laravel. Odpowiada on za logikę biznesową, obsługę użytkowników, quizów, pytań, odpowiedzi oraz zapisywanie wyników. Backend udostępnia API, z którego korzysta frontend.

Struktura backendu jest zgodna z konwencjami Laravla, co oznacza wyraźny podział na kontrolery, modele, migracje oraz warstwę routingu. Modele Eloquent odwzorowują tabele bazy danych i definiują relacje pomiędzy encjami. Migracje służą do tworzenia i aktualizowania schematu bazy danych w sposób kontrolowany.

Backend odpowiada również za uwierzytelnianie użytkowników, walidację danych wejściowych oraz egzekwowanie integralności logicznej danych. Wszystkie operacje zapisu i odczytu z bazy realizowane są wyłącznie w tej warstwie.

Kluczowe elementy:

- modele Eloquent,
- kontrolery API,
- trasy API (routes/api.php),

- middleware autoryzacji (Sanctum).

## **5.2 Kontrolery - szczegółowy opis**

### **5.2.1 QuizController**

Odpowiedzialny za pełne zarządzanie quizami.

Zakres odpowiedzialności:

- pobieranie listy quizów użytkownika,
- pobieranie wszystkich quizów publicznych,
- tworzenie quizu wraz z pytaniami i odpowiedziami,
- edycja quizu,
- usuwanie quizu,
- zapisywanie wyników quizu,
- generowanie danych rankingowych.

### **5.2.2 QuestionController**

Odpowiada za zarządzanie pytaniami w ramach quizów.

Zakres odpowiedzialności:

- dodawanie pytań do quizu,
- walidacja uprawnień właściciela quizu,
- przypisywanie punktów do pytań.

### **5.2.3 AnswerController**

Odpowiada za obsługę odpowiedzi do pytań.

Zakres odpowiedzialności:

- dodawanie odpowiedzi,
- oznaczanie odpowiedzi poprawnych,
- sprawdzanie poprawności odpowiedzi.

### **5.2.4 CategoryController**

Odpowiada za zarządzanie kategoriami quizów.

Zakres odpowiedzialności:

- pobieranie listy kategorii,
- zwracanie kategorii w formacie do komponentów select,
- dodawanie nowych kategorii.

### 5.2.5 CommentController

Obsługuje system komentarzy i ocen quizów.

Zakres odpowiedzialności:

- pobieranie komentarzy dla quizu,
- dodawanie komentarzy przez użytkowników,
- obsługa ocen liczbowych.

### 5.2.6 QuizResultsController

Odpowiada za prezentację wyników quizów.

Zakres odpowiedzialności:

- pobieranie listy wyników użytkownika,
- pobieranie szczegółowego wyniku wraz z sumą punktów.

## 5.3 Szczegółowy opis API systemu QuizVerse

### 5.3.1 Informacje ogólne

Backend systemu QuizVerse udostępnia zestaw endpointów REST API opisanych zgodnie ze specyfikacją *OpenAPI 3 (OAS3)*. Dokumentacja API została wygenerowana przy użyciu narzędzia *Swagger UI* i jest dostępna w pliku:

*/storage/api-docs/api-docs.json*

API umożliwia pełną obsługę funkcjonalności systemu, w tym zarządzanie quizami, pytaniami, odpowiedziami, kategoriami, komentarzami oraz wynikami quizów.

### 5.3.2 Konwencje i założenia

- Wszystkie zapytania oraz odpowiedzi przesyłane są w formacie **JSON**, o ile nie zaznaczono inaczej.
- Identyfikatory przekazywane w ścieżkach URL ({id}, {quizId}, {questionId}, {commentId}) są typu *integer*.
- API wykorzystuje standardowe kody odpowiedzi HTTP:
  - 200 OK

- 201 Created
- 403 Forbidden
- 404 Not Found
- 422 Unprocessable Entity (błędy walidacji)

### 5.3.3 Answers - odpowiedzi do pytań

*POST /questions/is-correct/{id}*

#### **Sprawdzenie poprawności odpowiedzi**

Endpoint służy do sprawdzenia, czy wskazana odpowiedź jest poprawna.

Parametry ścieżki:

- id (integer) - identyfikator odpowiedzi

Możliwe odpowiedzi:

- 200 OK - status odpowiedzi
- 404 Not Found - odpowiedź nie została znaleziona

Przykładowa odpowiedź (200):

```
{
  "status": "success",
  "answer_id": 1,
  "question_id": 1,
  "is_correct": true
}
```

*POST /questions/{questionId}/answers*

#### **Dodanie odpowiedzi do pytania**

Endpoint umożliwia dodanie nowej odpowiedzi do pytania. Operacja dostępna wyłącznie dla właściciela quizu.

Parametry ścieżki:

- questionId (integer) - identyfikator pytania

Body (JSON):

```
{
  "text": "Warszawa",
  "is_correct": true,
}
```

```
    "image_path": "string"
  }
```

Możliwe odpowiedzi:

- 201 Created - odpowiedź została dodana
- 403 Forbidden - brak uprawnień

Przykładowa odpowiedź (201):

```
{
  "message": "Answer added successfully",
  "answer": {
    "id": 1,
    "question_id": 0,
    "text": "string",
    "is_correct": true,
    "image_path": "string"
  }
}
```

#### 5.3.4 Quizzes - quizy

*GET /all/quizzes*

##### **Lista wszystkich quizów**

Pobiera listę wszystkich quizów wraz z informacją o kategorii i liczbie rozwiązań.

Możliwe odpowiedzi:

- 200 OK - lista quizów

Przykładowa odpowiedź:

```
[
  {
    "id": 1,
    "user_id": 1,
    "category_id": 1,
    "title": "Historia Polski",
    "description": "string",
    "results_count": 5,
    "created_at": "2026-01-27T17:22:57.076Z",
    "updated_at": "2026-01-27T17:22:57.076Z"
  }
]
```

]

*GET /quizzes*

### **Lista quizów zalogowanego użytkownika**

Pobiera quizy utworzone przez aktualnie zalogowanego użytkownika.

Możliwe odpowiedzi:

- 200 OK

*POST /quizzes*

### **Utworzenie nowego quizu**

Tworzy nowy quiz wraz z pytaniami i odpowiedziami.

Body (JSON):

```
{
  "title": "Historia Polski",
  "description": "string",
  "category_id": 1,
  "questions": [
    {
      "text": "Kiedy został założony polski staat?",
      "points": 10,
      "answers": [
        {
          "text": "966",
          "is_correct": true
        }
      ]
    }
  ]
}
```

Możliwe odpowiedzi:

- 201 Created
- 422 Unprocessable Entity

*PUT /quizzes/{quiz}*

### **Aktualizacja quizu**

Aktualizuje podstawowe dane quizu.

Parametry ścieżki:

- quiz (integer) - identyfikator quizu

Możliwe odpowiedzi:

- 200 OK
- 403 Forbidden
- 404 Not Found

*GET /quizzes/{id}*

### **Pobranie quizu wraz z pytaniami**

Zwraca szczegółowe informacje o quizie, w tym pytania i odpowiedzi.

*DELETE /quizzes/{id}*

### **Usunięcie quizu**

Usuwa quiz. Operacja dostępna wyłącznie dla właściciela quizu.

*POST /quizzes/save-result*

### **Zapisanie wyniku quizu**

Zapisuje wynik rozwiązane quizu.

Body (JSON):

```
{
  "quiz_id": 1,
  "points": 85
}
```

*GET /get-ranking-data*

### **Dane rankingowe**

Pobiera dane wykorzystywane do wyświetlania rankingu użytkowników.

### 5.3.5 Categories - kategorie

*GET /categories*

#### **Lista kategorii**

Zwraca listę wszystkich kategorii quizów.

*GET /categories/select*

#### **Kategorie do pól wyboru**

Zwraca kategorie w uproszczonym formacie id => name.

### 5.3.6 Comments - komentarze

*GET /api/quizzes/{quizId}/comments*

#### **Lista komentarzy do quizu**

Pobiera komentarze przypisane do quizu, posortowane od najnowszych.

*POST /api/quizzes/{quizId}/comments*

#### **Dodanie komentarza**

Tworzy nowy komentarz przypisany do zalogowanego użytkownika.

*GET /api/quizzes/{quizId}/comments/rating*

#### **Średnia ocena quizu**

Zwraca średnią ocenę oraz liczbę komentarzy dla quizu.

*PUT /api/quizzes/{quizId}/comments/{commentId}*

#### **Aktualizacja komentarza**

Umożliwia edycję treści lub oceny komentarza przez jego właściciela.

*DELETE /api/quizzes/{quizId}/comments/{commentId}*

### Usunięcie komentarza

Usuwa komentarz. Operacja dostępna tylko dla autora komentarza.

### 5.3.7 Questions - pytania

*POST /quizzes/{quizId}/questions*

#### Dodanie pytania do quizu

Dodaje nowe pytanie do quizu. Operacja dostępna tylko dla właściciela quizu.

### 5.3.8 Quiz Results - wyniki quizów

*GET /quiz-results*

#### Lista wyników użytkownika

Pobiera wszystkie wyniki quizów zalogowanego użytkownika.

*GET /quiz-results/{id}*

#### Szczegóły wyniku quizu

Zwraca szczegółowe informacje o wyniku quizu, w tym sumę punktów.

## 5.4 Skrócony opis endpointów API w formie tabeli

Tabela 5.4.1 Opis endpointów API

Moduł	Metoda	Endpoint	Opis
Answers	POST	<i>/questions/is-correct/{id}</i>	Sprawdzenie poprawności odpowiedzi
Answers	POST	<i>/questions/{questionId}/answers</i>	Dodanie odpowiedzi do pytania
Quizzes	GET	<i>/all/quizzes</i>	Lista wszystkich quizów
Quizzes	GET	<i>/quizzes</i>	Lista quizów zalogowanego użytkownika
Quizzes	POST	<i>/quizzes</i>	Utworzenie nowego quizu
Quizzes	PUT	<i>/quizzes/{quiz}</i>	Aktualizacja danych quizu
Quizzes	GET	<i>/quizzes/{id}</i>	Pobranie quizu z pytaniami
Quizzes	DELETE	<i>/quizzes/{id}</i>	Usunięcie quizu
Quizzes	GET	<i>/category/quizzes/{id}</i>	Pobranie quizów z kategorii
Quizzes	POST	<i>/quizzes/save-result</i>	Zapisanie wyniku quizu

Quizzes	GET	<i>/get-ranking-data</i>	Pobranie danych rankingowych
Categories	GET	<i>/categories</i>	Lista kategorii
Categories	GET	<i>/categories/select</i>	Kategorie do pól wyboru
Comments	GET	<i>/api/quizzes/{quizId}/comments</i>	Lista komentarzy do quizu
Comments	POST	<i>/api/quizzes/{quizId}/comments</i>	Dodanie komentarza
Comments	GET	<i>/api/quizzes/{quizId}/comments/rating</i>	Średnia ocena quizu
Comments	GET	<i>/api/quizzes/{quizId}/comments/{commentId}</i>	Szczegóły komentarza
Comments	PUT	<i>/api/quizzes/{quizId}/comments/{commentId}</i>	Aktualizacja komentarza
Comments	DELETE	<i>/api/quizzes/{quizId}/comments/{commentId}</i>	Usunięcie komentarza
Questions	POST	<i>/quizzes/{quizId}/questions</i>	Dodanie pytania do quizu
Quiz Results	GET	<i>/quiz-results</i>	Lista wyników użytkownika
Quiz Results	GET	<i>/quiz-results/{id}</i>	Szczegóły wyniku quizu

## Rozdział 6

### Frontend - Vue 3

#### 6.1 Architektura frontendowa

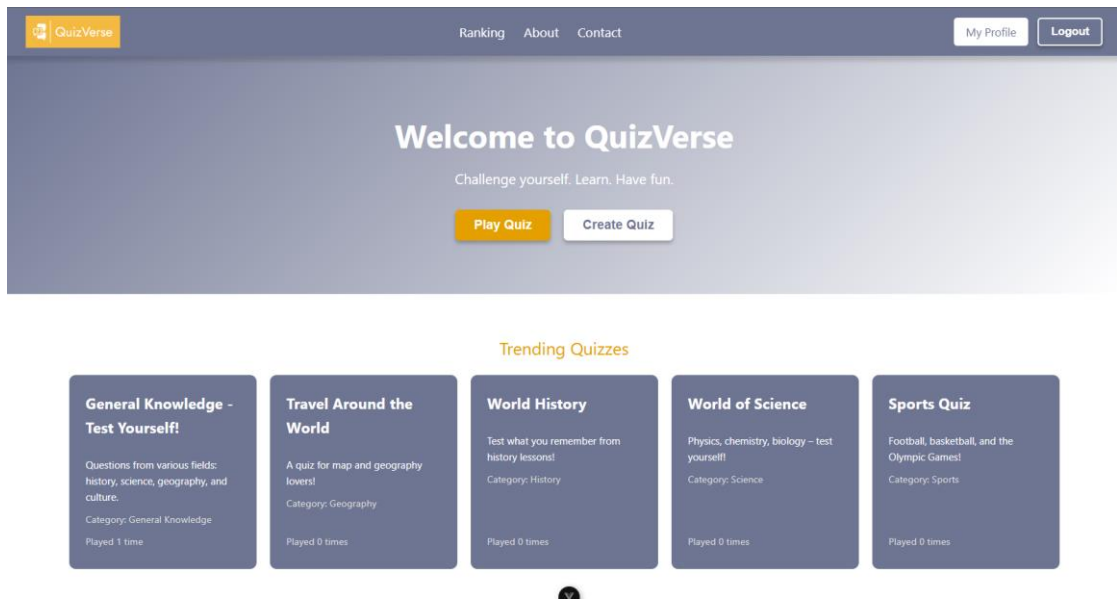
Frontend zbudowany jest jako SPA z wykorzystaniem:

- Vue 3 (Composition API),
- Vue Router,
- Axios,
- Chart.js (wizualizacja rankingów).

#### 6.2 Widoki aplikacji

##### HomeView

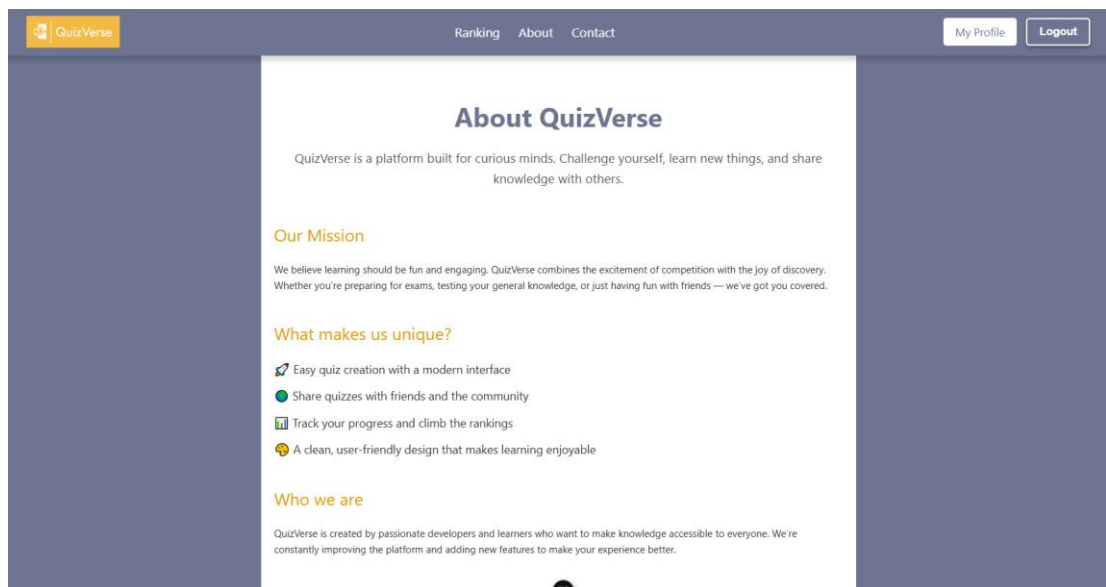
- ekran startowy,
- lista popularnych quizów,
- ranking użytkowników.



Rys. 6.2.1. Widok strony HomeView

## AboutView

- strona informacyjna o platformie QuizVerse,
- przedstawienie misji aplikacji i jej głównych założeń,
- lista kluczowych funkcjonalności (tworzenie quizów, statystyki, ranking),
- sekcja opisująca zespół twórców projektu.



Rys. 6.2.2 Widok strony AboutView

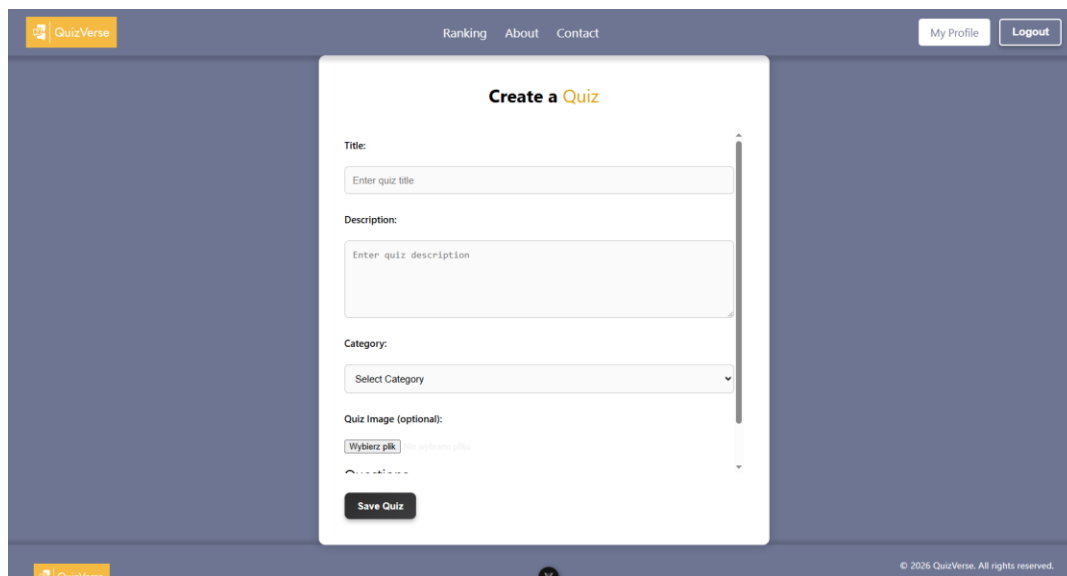
## ContactView

- formularz kontaktowy symulujący wysłanie wiadomości do zespołu QuizVerse,
- pola: imię, adres e-mail oraz treść wiadomości,
- podstawowa walidacja danych formularza,
- wyświetlenie alternatywnych danych kontaktowych (email, telefon).

Rys. 6.2.3. Widok strony ContactView

### CreateQuizView

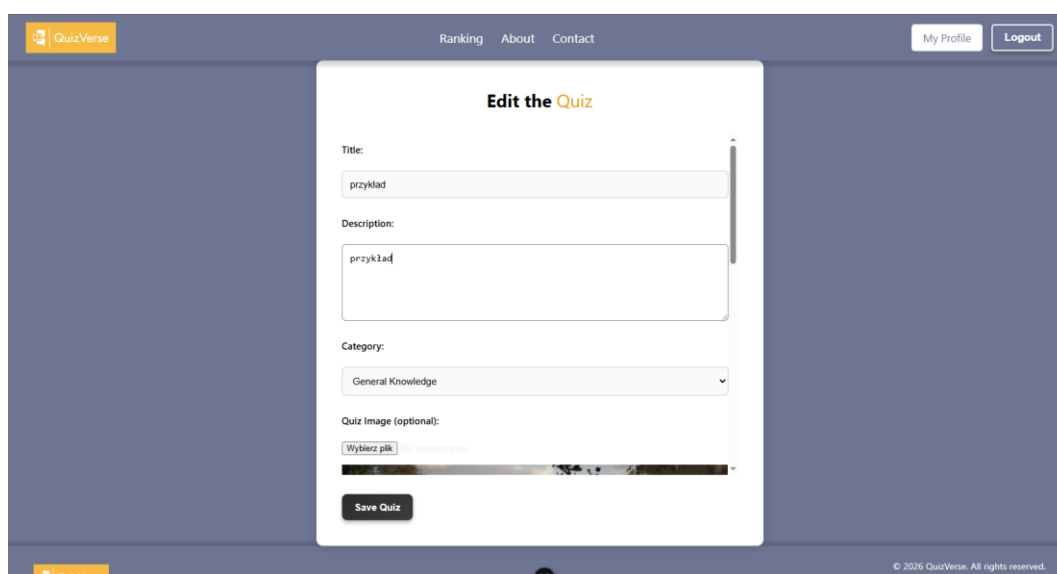
- formularz tworzenia nowego quizu,
- wprowadzanie tytułu, opisu oraz wybór kategorii,
- upload obrazu quizu (opcjonalnie) z podglądem,
- dynamiczne dodawanie i usuwanie pytań,
- ustalanie liczby punktów dla każdego pytania,
- dynamiczne dodawanie odpowiedzi z oznaczeniem poprawnych,
- upload obrazów dla pytań i odpowiedzi,
- wysyłanie danych do API przy użyciu FormData.



Rys. 6.2.4. Widok strony CreateQuizView

## EditQuizView

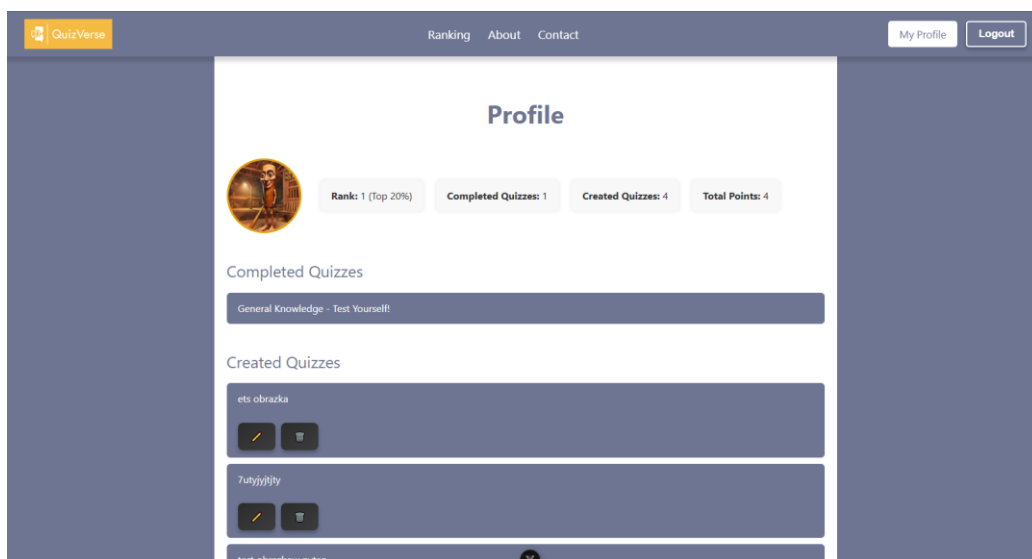
- edycja istniejącego quizu,
- wczytywanie aktualnych danych quizu z API,
- edycja tytułu, opisu, kategorii oraz obrazów,
- edycja pytań i odpowiedzi (dodawanie, usuwanie, zmiana treści),
- obsługa podglądu istniejących i nowych obrazów,
- zapis zmian poprzez metodę PUT.



Rys. 6.2.5. Widok strony EditQuizView

## ProfileView

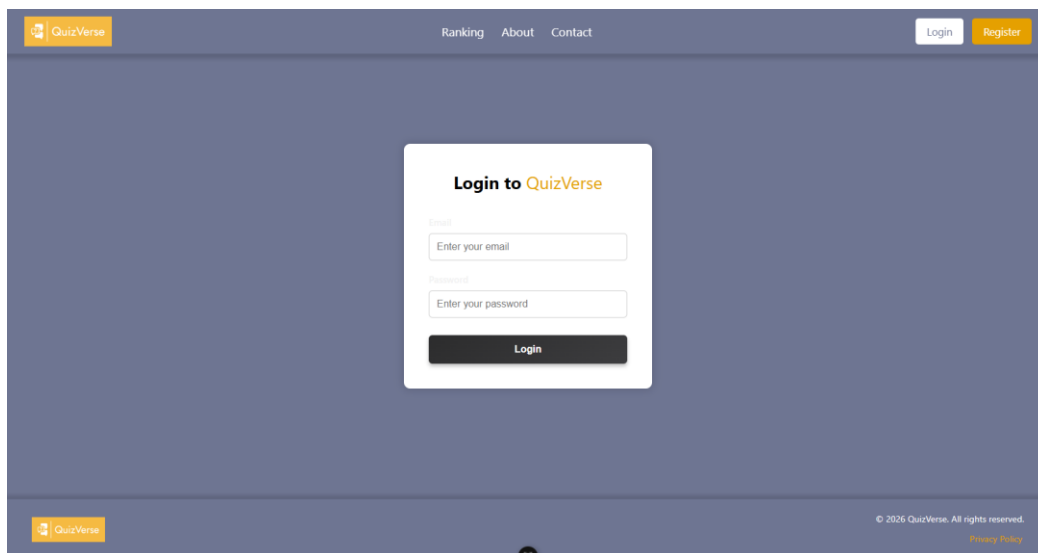
- profil użytkownika z danymi statystycznymi,
- wyświetlenie rangi, procentyla, liczby punktów,
- lista ukończonych quizów,
- lista quizów utworzonych przez użytkownika,
- możliwość edycji i usuwania własnych quizów,
- pobieranie danych profilu z API.



Rys. 6.2.6. Widok strony ProfileView

## LoginView

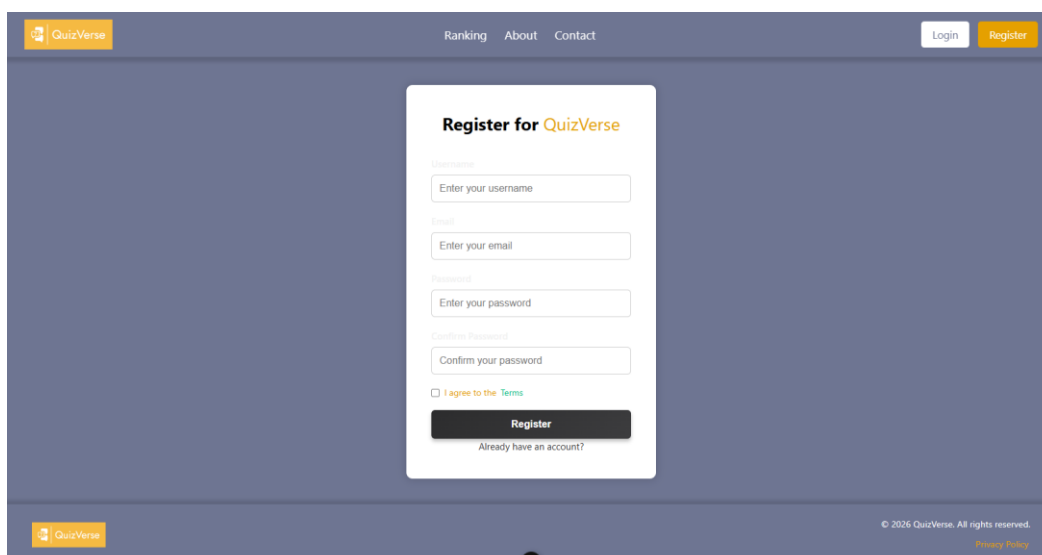
- formularz logowania użytkownika,
- autoryzacja przy użyciu adresu e-mail i hasła,
- obsługa błędów logowania,
- zapis tokenu autoryzacyjnego w localStorage,
- przekierowanie po poprawnym zalogowaniu.



Rys. 6.2.7. Widok strony LoginView

## RegisterView

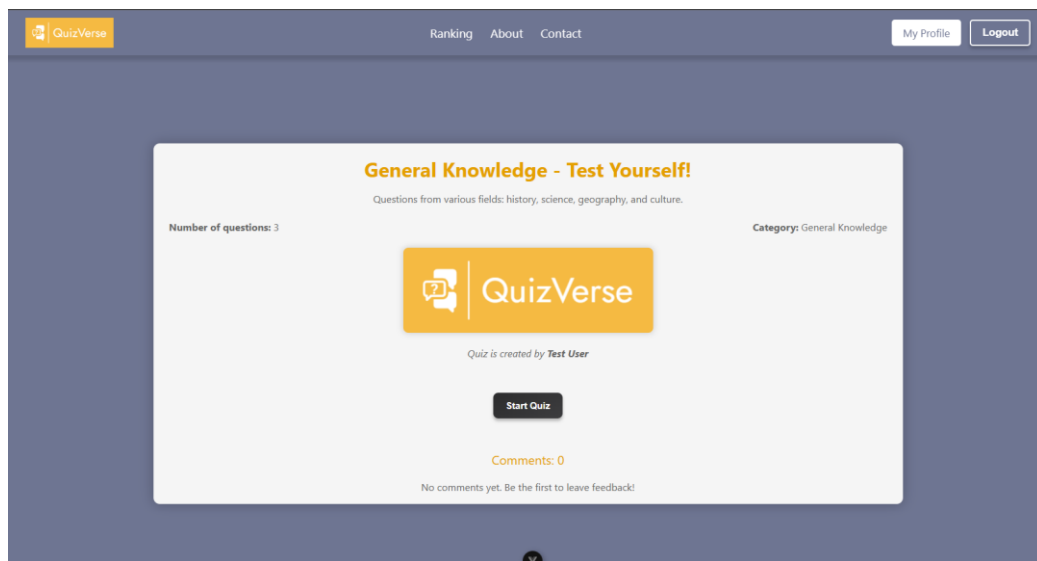
- rejestracja nowego konta użytkownika,
- wprowadzanie danych rejestracyjnych (email, hasło, itp.),
- walidacja danych po stronie klienta i serwera,
- utworzenie konta i możliwość późniejszego logowania.



Rys. 6.2.8. Widok strony RegisterView

## QuizPageView

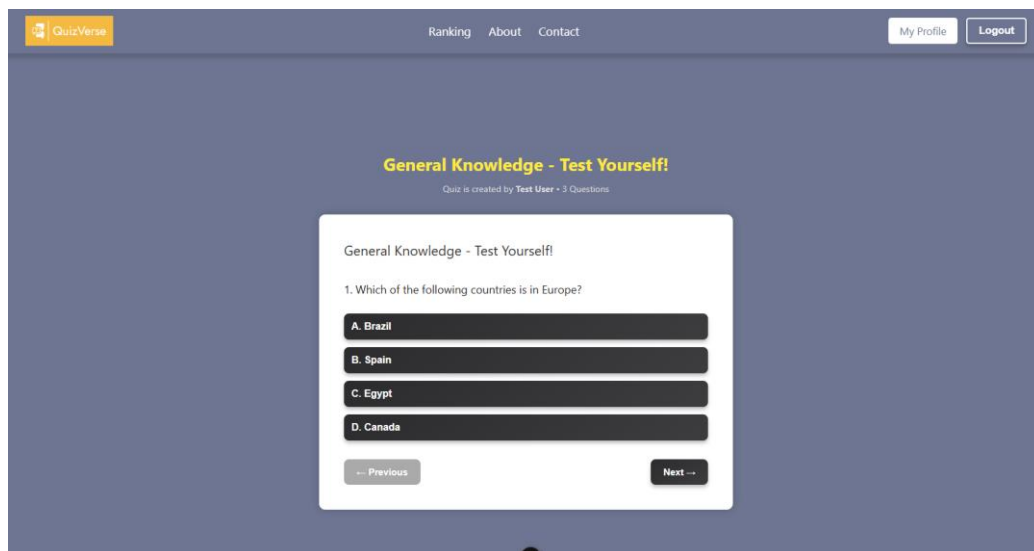
- szczegóły wybranego quizu,
- wyświetlenie opisu, kategorii, autora i liczby pytań,
- obsługa obrazu quizu (lub domyślnego),
- uruchomienie quizu,
- sekcja komentarzy i ocen użytkowników,
- obliczanie i prezentacja średniej oceny.



Rys. 6.2.9. Widok strony QuizPageView

## QuizView

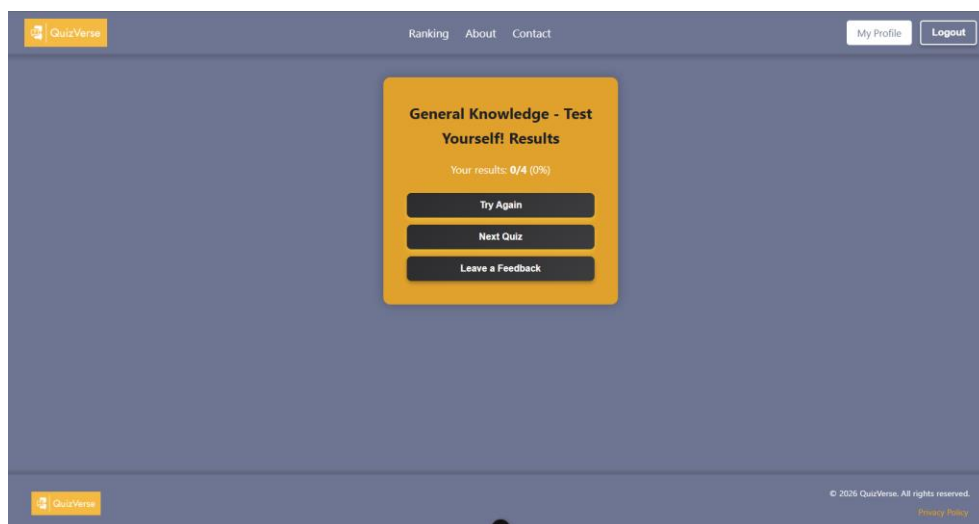
- widok rozwiązywania wybranego quizu,
- pobieranie danych quizu (pytania, odpowiedzi, autor) z API,
- wyświetlenie tytułu quizu oraz informacji o autorze i liczbie pytań,
- prezentacja pojedynczego pytania wraz z opcjonalnym obrazem,
- wyświetlenie listy odpowiedzi w formie przycisków,
- oznaczanie aktualnie wybranej odpowiedzi,
- nawigacja pomiędzy pytaniami (poprzednie / następne),
- zapamiętywanie odpowiedzi użytkownika dla każdego pytania,
- obliczanie wyniku na podstawie poprawnych odpowiedzi i punktów,
- wysłanie wyniku quizu do API po zakończeniu,
- przekierowanie do widoku wyników po zapisaniu rezultatu.



Rys. 6.2.10. Widok strony QuizView

## QuizResultsView

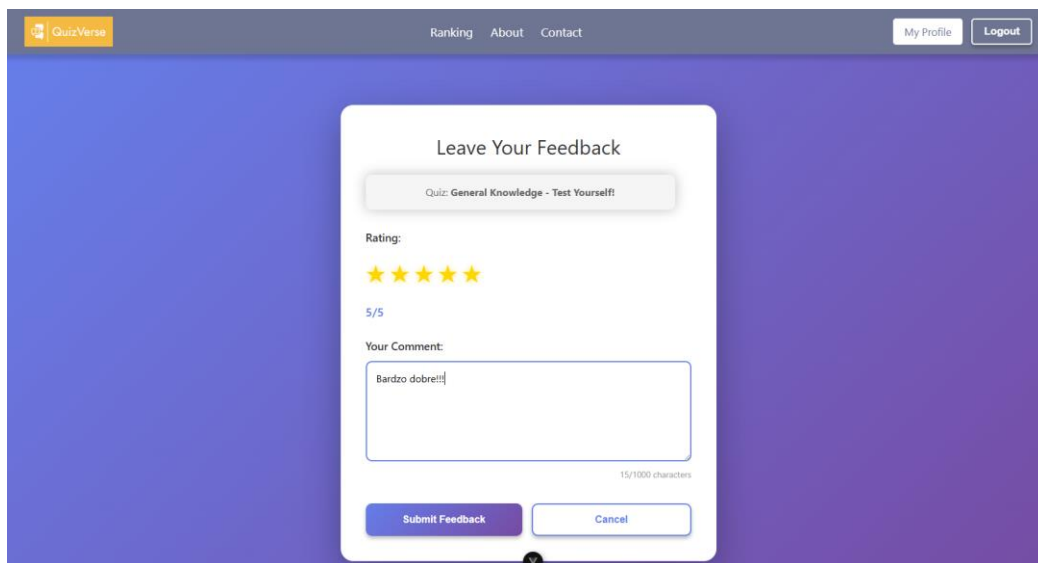
- podsumowanie wyników po ukończeniu quizu,
- wyświetlenie zdobytych punktów i procentowego wyniku,
- przyciski: ponowne rozwiązanie quizu, przejście do listy quizów,
- możliwość przejścia do formularza opinii.



Rys. 6.2.11. Widok strony QuizResultsView

## FeedbackView

- formularz wystawienia opinii o quizie,
- system ocen w skali 1-5 gwiazdek,
- pole komentarza z walidacją długości,
- powiązanie opinii z konkretnym quizem,
- wysłanie opinii do API,
- ekran potwierdzenia po zapisaniu opinii.



QuizVerse

Ranking About Contact

My Profile Logout

### Leave Your Feedback

Quiz: General Knowledge - Test Yourself!

Rating:

★★★★★

5/5

Your Comment:

Bardzo dobre!!!

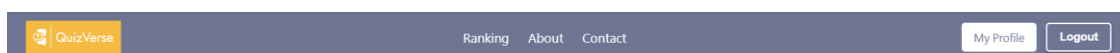
15/1000 characters

Submit Feedback Cancel

Rys. 6.2.12. Widok strony FeedbackView

## NavigationView (Navbar)

- nawigacja główna aplikacji,
- linki do rankingu, strony „About” i „Contact”,
- obsługa stanu zalogowania użytkownika,
- przyciski logowania, rejestracji lub profilu,
- menu mobilne (burger menu).



Rys. 6.2.13. Widok strony NavigationView

## FooterView

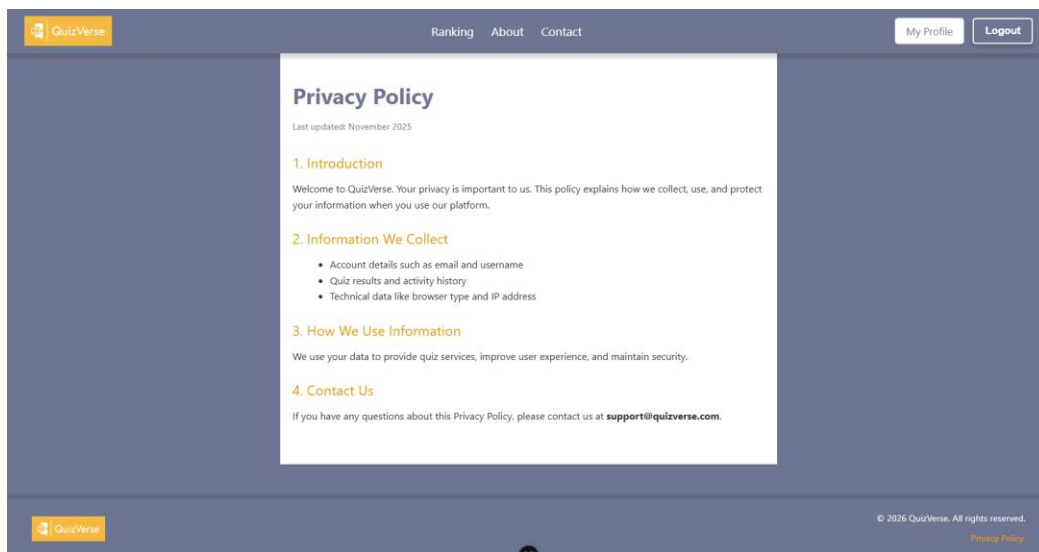
- stopka aplikacji,
- logo QuizVerse z możliwością powrotu na stronę główną,
- informacja o prawach autorskich,
- link do polityki prywatności.



Rys. 6.2.14. Widok strony FooterView

### PrivacyPolicyView

- strona polityki prywatności,
- opis zakresu zbieranych danych,
- informacje o sposobie przetwarzania danych,
- dane kontaktowe administratora.

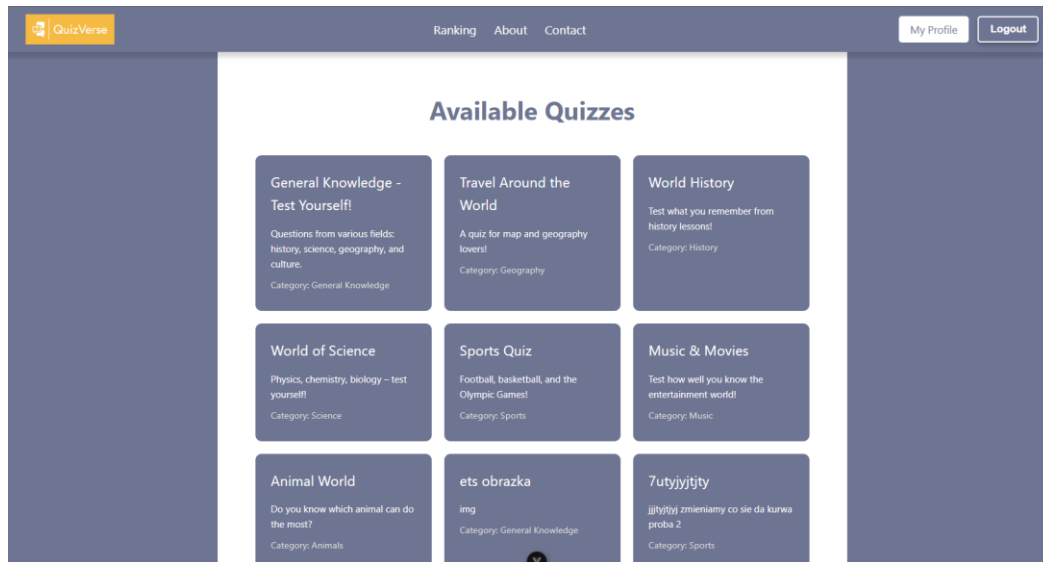


Rys. 6.2.15. Widok strony PrivacyPolicyView

### QuizzesView

- lista wszystkich dostępnych quizów w systemie,
- pobieranie danych quizów z API,
- wyświetlenie tytułu, opisu oraz kategorii quizu,

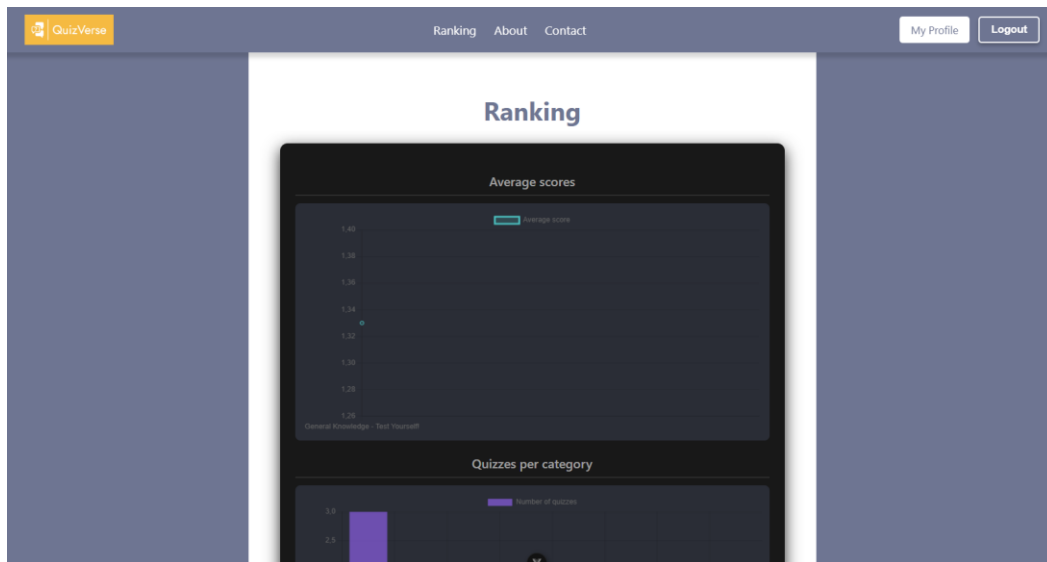
- obsługa stanów: ładowanie danych oraz błąd pobierania,
- przekierowanie do strony szczegółów quizu po kliknięciu karty,
- prezentacja quizów w formie czytelnych kart.



Rys. 6.2.16. Widok strony QuizzesView

## RankingView

- strona statystyk i rankingu użytkowników oraz quizów,
- wizualizacja danych przy użyciu wykresów Chart.js,
- wykres średnich wyników dla quizów,
- liczba quizów w poszczególnych kategoriach,
- ranking najlepszych użytkowników według zdobytych punktów,
- liczba pytań przypadających na quiz,
- statystyki poprawnych i błędnych odpowiedzi,
- pobieranie danych rankingowych z API,
- widok dostępny dla zalogowanych użytkowników.



Rys. 6.2.17. Widok strony RankingView

## RegisterView

- formularz rejestracji nowego użytkownika,
- pola: nazwa użytkownika, email, hasło, potwierdzenie hasła,
- wymóg akceptacji regulaminu i polityki prywatności,
- podstawowa walidacja danych po stronie klienta,
- obsługa błędów rejestracji zwracanych przez API,
- automatyczne zalogowanie po poprawnej rejestracji,
- zapis tokenu autoryzacyjnego w localStorage,
- link do widoku logowania dla istniejących użytkowników.

Rys. 6.2.18. Widok strony RegisterView

## Rozdział 7

### Zarządzanie projektem i organizacja pracy zespołu

#### 7.1 Metodyka organizacji pracy

W trakcie realizacji projektu **QuizVerse** zastosowano narzędzie **ClickUp** jako centralną platformę do planowania, koordynacji oraz monitorowania postępów prac zespołowych.

Projekt był realizowany przez **zespół pięcioosobowy**, w którym każdy członek odpowiadał za jasno zdefiniowany zakres obowiązków. ClickUp umożliwił jednoznaczny podział pracy, przypisanie odpowiedzialności oraz kontrolę terminowości realizowanych zadań.

#### 7.2 Statusy zadań

W celu kontroli postępów prac zastosowano jednolity zestaw statusów zadań, który odzwierciedlał rzeczywisty stan realizacji danego elementu projektu. Każde zadanie w ClickUp znajdowało się zawsze w jednym z poniższych statusów:

- **Do zrobienia**  
Zadanie zaplanowane, jeszcze nierozpoczęte.
- **W trakcie**  
Zadanie aktualnie realizowane przez przypisaną osobę.
- **Do weryfikacji**  
Zadanie ukończone implementacyjnie i oczekujące na sprawdzenie (code review, testy).
- **Zrobione**  
Zadanie zakończone, zweryfikowane i zaakceptowane.
- **Do poprawy**  
Zadanie wymagające poprawek po weryfikacji lub testach.

Zastosowanie tych statusów pozwoliło na szybkie określenie stanu projektu oraz identyfikację elementów wymagających uwagi.

#### 7.3 Kontrola postępu i komunikacja

ClickUp pełnił również rolę narzędzia komunikacyjnego w obrębie zespołu. Każde zadanie umożliwiało:

- dodawanie komentarzy technicznych,
- zgłaszanie problemów i uwag,
- informowanie o zmianach w implementacji,
- dokumentowanie decyzji projektowych.

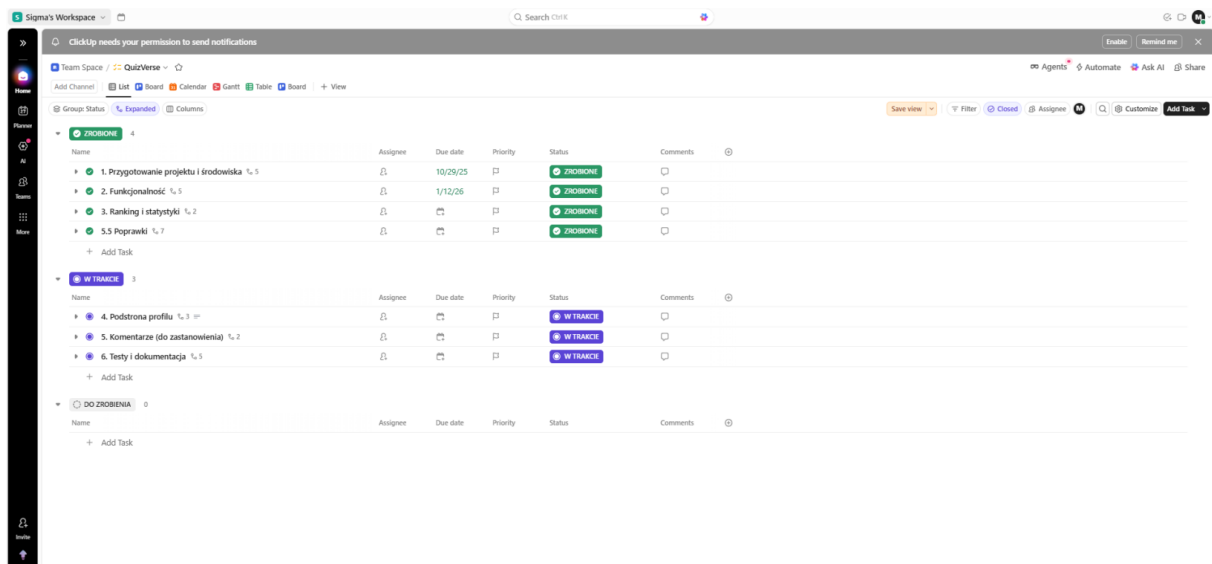
Dzięki temu wszystkie informacje dotyczące realizacji projektu znajdowały się w jednym, spójnym miejscu, co znacząco ułatwiało współpracę i ograniczało ryzyko nieporozumień.

## 7.4 Dokumentacja wizualna ClickUp

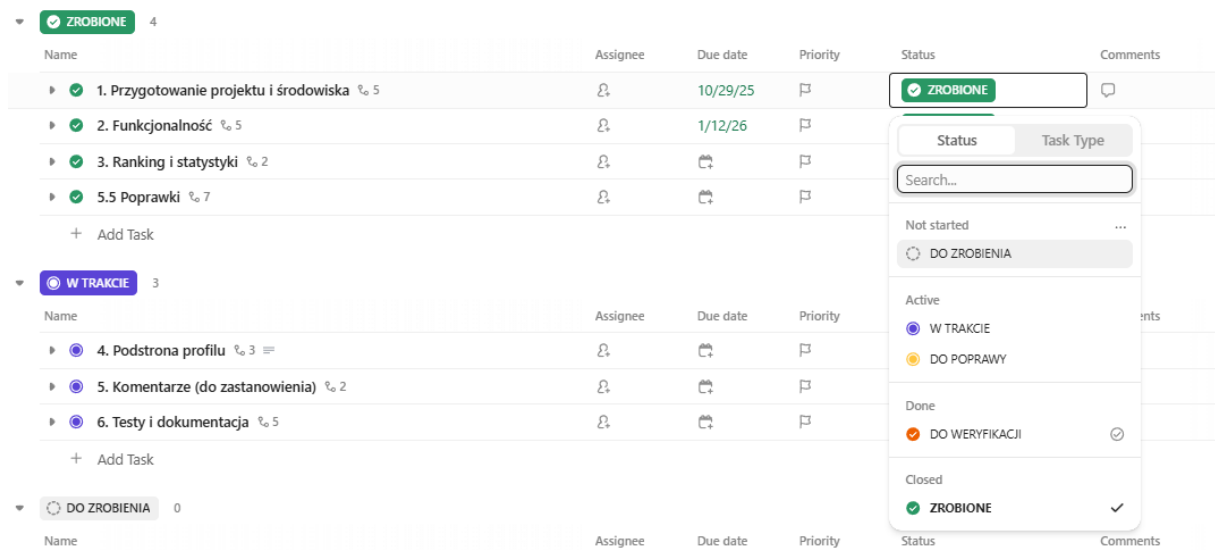
W celu pełniejszego udokumentowania procesu zarządzania projektem, do dokumentacji dołączone zostaną zrzuty ekranu z aplikacji ClickUp (rys. 7.4.1 - 7.4.3), przedstawiające między innymi:

- listę zadań projektu i przypisanie zadań do członków zespołu,
- widok statusów zadań
- harmonogramy oraz deadline'y.

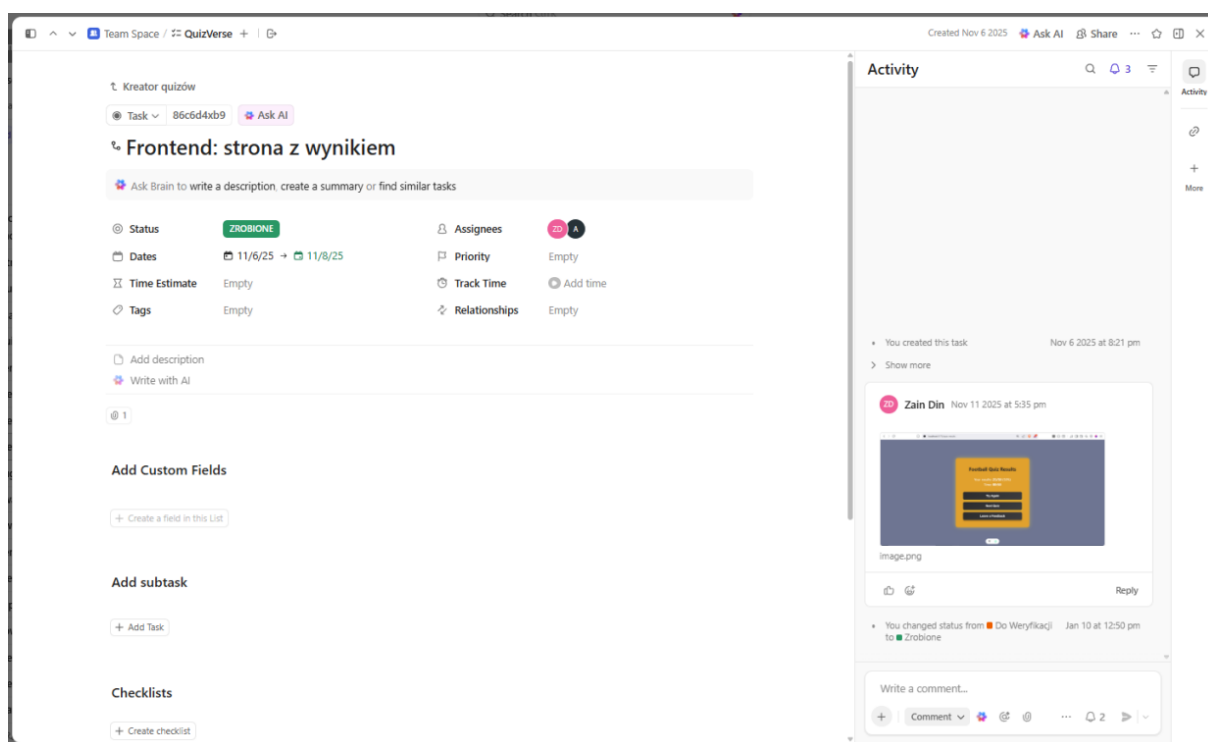
Materiały te stanowią potwierdzenie zastosowania profesjonalnego narzędzia do zarządzania projektem oraz ilustrują rzeczywisty proces organizacji pracy zespołowej.



Rys. 7.4.1. Lista zadań projektu i przypisanie zadań do członków zespołu



Rys. 7.4.2. Statusy zadań projektu



Rys. 7.4.3. Szczegóły zadania w ClickUp

## Rozdział 8

### Projektowanie graficzne interfejsu użytkownika

#### 8.1 Cel projektowania interfejsu

Proces projektowania interfejsu użytkownika w projekcie *QuizVerse* został przeprowadzony z wykorzystaniem narzędzia *Figma*, które posłużyło jako podstawowe środowisko do tworzenia koncepcji wizualnej aplikacji. Głównym celem tego etapu było zaprojektowanie czytelnego, spójnego oraz intuicyjnego interfejsu użytkownika jeszcze przed rozpoczęciem implementacji frontendowej.

Projektowanie graficzne poprzedzało etap programowania, co pozwoliło na ograniczenie liczby zmian w trakcie implementacji oraz zapewniło spójność wizualną całego systemu.

## 8.2 Etapy projektowania graficznego

Proces projektowania interfejsu w *Figmie* przebiegał etapowo i obejmował następujące kroki:

1. Analiza funkcjonalna systemu

Na podstawie wymagań funkcjonalnych określono zestaw widoków niezbędnych do prawidłowego działania aplikacji, takich jak strona główna, widok quizu, formularz tworzenia quizu, profil użytkownika oraz ekrany logowania i rejestracji.

2. Tworzenie makiet (wireframes)

W pierwszym etapie w *Figmie* przygotowano uproszczone makiety prezentujące układ elementów na poszczególnych ekranach, bez szczegółowej oprawy graficznej.

3. Projektowanie pełnych widoków UI

Następnie makiety zostały rozbudowane o elementy graficzne, kolory, typografię oraz komponenty interaktywne, tworząc pełny projekt interfejsu użytkownika.

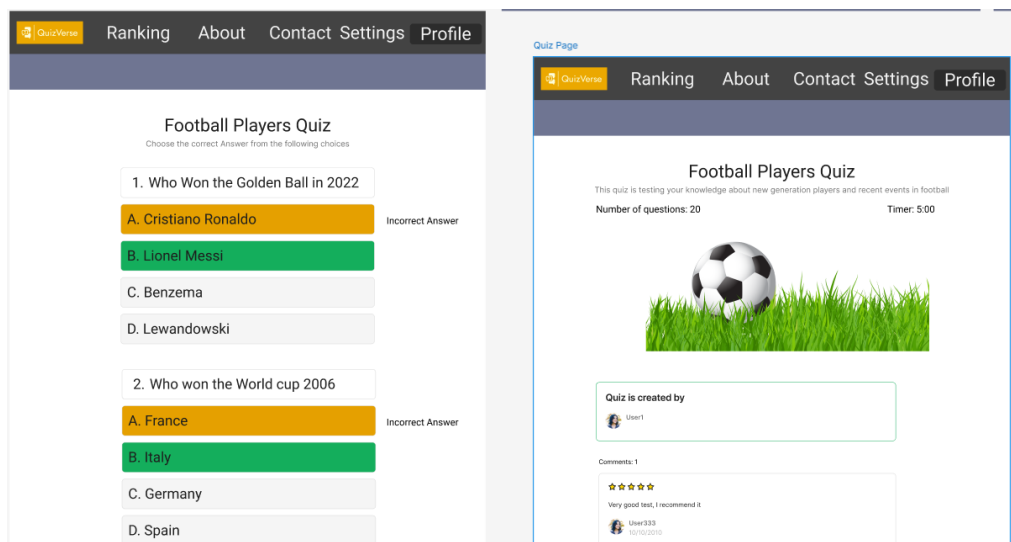
## 8.3 Przejście od projektu do implementacji

Po zakończeniu etapu projektowania graficznego w *Figmie* rozpoczęto implementację interfejsu użytkownika w technologii *Vue 3*. Każdy widok aplikacji był implementowany bezpośrednio na podstawie wcześniej przygotowanych projektów graficznych.

## 8.4 Dokumentacja wizualna

W ramach dokumentacji projektu zostaną dołączone zrzuty ekranu z narzędzia *Figma*, przedstawiające:

- widoki aplikacji na etapie projektowania,
- układ kluczowych ekranów systemu,



Rys. 8.4.1. Projekty przykładowych paneli aplikacji



Rys. 8.4.2. Projekt wszystkich paneli aplikacji

## Rozdział 9

### Testowanie aplikacji QuizVerse

#### 9.1 Środowisko testowe

W celu przeprowadzenia testów automatycznych aplikacji QuizVerse wykorzystano *Robot Framework* wraz z biblioteką *RequestsLibrary*, zapewniającą obsługę sesji HTTP i komunikację z API. Testy wykonywane były w wirtualnym środowisku *Pythona* (venv), co umożliwiało izolację zależności i powtarzalność wyników.

W celu prawidłowego wykonania testów, zarejestrowano użytkownika testowego (jeśli nie istniał wcześniej):

- email: test@test.test
- hasło: testtest

Testy uruchamiane były w katalogu głównym projektu quizverse poleceniem:

*robot -d results tests*

## 9.2 Dokumentacja Robot Framework Test Suite

Testy API aplikacji QuizVerse zostały opracowane jako kompleksowa, end-to-endowa sekwencja, obejmująca następujące obszary:

### Autoryzacja:

- Logowanie użytkownika i obsługa tokena (`{token}`), niezbędne dla wszystkich testów wymagających autoryzacji.

### Quizy:

- Przeglądanie dostępnych quizów, tworzenie nowego quizu, weryfikacja jego istnienia oraz sprawdzenie quizów należących do użytkownika.

### Kategorie:

- Pobieranie listy kategorii oraz mapowanie ID do nazw.

### Komentarze:

- Tworzenie, pobieranie, aktualizacja i usuwanie komentarzy.
- Sprawdzanie szczegółów komentarzy i sumarycznych ocen (`average_rating`, `total_comments`).

### Pytania i odpowiedzi:

- Dodawanie pytań do quizów, dodawanie odpowiedzi, weryfikacja poprawności odpowiedzi (`is_correct`).

### Testy negatywne:

- Próby dostępu do zasobów chronionych bez tokena (401/403),
- Pobranie nieistniejącego quizu (404),
- Próby tworzenia quizu z niepoprawnymi danymi (422),
- Sprawdzenie poprawności odpowiedzi dla nieistniejącej odpowiedzi (404),
- Niepoprawne zapisywanie wyników quizu (422).

### **Wyniki i rankingi:**

- Pobieranie wyników własnych quizów i danych rankingowych.

### **Czyszczenie danych:**

- Usuwanie stworzonych quizów i komentarzy po zakończeniu testów, aby środowisko testowe pozostało czyste.

## **9.3 Zależności i zmienne testowe**

Testy są zależne od kolejności wykonania:

1. Logowanie użytkownika musi być pierwsze ( $\${token}$ ).
2. Tworzenie quizu przed akcjami specyficznymi dla quizu (komentarze, pytania, odpowiedzi).
3. Tworzenie komentarza przed testami szczegółów, aktualizacji i usunięcia komentarza.
4. Tworzenie pytań i odpowiedzi przed testami is-correct.
5. Testy czyszczące (cleanup) powinny być ostatnie.

Podczas wykonywania testów tworzono i wykorzystywano zmienne współdzielone:

- $\${token}$  – token autoryzacyjny,
- $\${quiz\_id}$  – ID utworzonego quizu,
- $\${comment\_id}$  – ID utworzonego komentarza,
- $\${question\_id}$  – ID utworzonego pytania,
- $\${answer\_id}$  – ID utworzonej odpowiedzi.

## **9.4 Przykładowe przypadki testowe**

### **Autoryzacja**

- Login And Print Token: logowanie użytkownika, weryfikacja HTTP 200, zapis  $\${token}$ .

### **Quizy**

- Get All Quizzes: pobranie wszystkich quizów, weryfikacja HTTP 200.
- Create New Quiz: tworzenie quizu z pytaniami i odpowiedziami, zapis  $\${quiz\_id}$ .
- Verify Created Quiz Exists: sprawdzenie, że quiz istnieje i dane są zgodne z oczekiwaniami.

### **Kategorie**

- Get Categories: pobranie kategorii, weryfikacja statusu i obecności listy kategorii.

### **Komentarze**

- Create Quiz Comment: tworzenie komentarza, zapis  $\${comment\_id}$ .

- Get Quiz Comments: pobranie wszystkich komentarzy, weryfikacja liczby komentarzy.
- Update My Comment i Delete My Comment: aktualizacja treści komentarza i jego usunięcie.

### **Pytania i odpowiedzi**

- Add Question To Created Quiz i Add Answer To Question: dodanie pytania i odpowiedzi, zapis ID.
- Is Correct Should Return True: weryfikacja poprawności odpowiedzi.

### **Testy negatywne / błędy**

- Access Protected Endpoint Without Token: brak autoryzacji → 401/403.
- Get Nonexistent Quiz Returns 404: brak quizu → 404.
- Create Quiz Validation Returns 422: walidacja danych → 422.
- Is Correct Nonexistent Answer Returns 404: brak odpowiedzi → 404.
- Save Quiz Result Rejects Invalid Request: błędne zapisanie wyniku → 422.

### **Wyniki i rankingi**

- Get My Quiz Results: pobranie wyników własnych quizów, weryfikacja co najmniej jednego wyniku.
- Get Ranking Data: pobranie danych rankingowych, akceptacja HTTP 200 lub 201.

### **Czyszczenie danych**

- Delete Created Quiz: usunięcie utworzonego quizu, weryfikacja HTTP 200, logowanie odpowiedzi lub błędów.

## **Rozdział 10**

### **Podsumowanie**

System *QuizVerse* stanowi kompletną, skalowalną platformę edukacyjną o jasno zdefiniowanej architekturze, poprawnie zaprojektowanym modelu danych oraz pełnej separacji frontend-backend. Dokumentacja opisuje system w sposób umożliwiający jego dalszy rozwój, utrzymanie oraz wdrożenie produkcyjne.