QuizVerse API – Robot Framework Test Suite Documentation

Overview

This Robot Framework suite validates core QuizVerse API flows using RequestsLibrary. It covers:

- Authentication (login + token handling)
- Quiz browsing and quiz creation
- Quiz retrieval and ownership checks
- Categories endpoints
- Comments lifecycle (create, list, details, update, delete)
- Questions and answers (create + correctness check)
- Negative tests for authorization, not found, and validation errors
- Results and ranking endpoints
- Cleanup (delete created resources)
- The suite is designed as an end-to-end flow, where later tests reuse variables created earlier (token, quiz_id, comment_id, etc.).

Configuration

Libraries

- RequestsLibrary – HTTP session management and requests
- Collections – dictionary/list helpers
- json – JSON parsing via Evaluate json.loads(…)

Variables

- `${BASE_URL}: http://localhost:8000`
- `${EMAIL}: test@test.test`
- `${PASSWORD}: testtest`

Shared Suite Variables Created During Execution

- `${token}` – authentication token saved after login
- `${quiz_id}` – ID of quiz created in Create New Quiz
- `${comment_id}` – ID of comment created in Create Quiz Comment
- `${question_id}` – ID of question created in Add Question To Created Quiz
- `${answer_id}` – ID of answer created in Add Answer To Question

Execution Flow & Dependencies

This suite is order-dependent:

1. Login must run first to generate `${token}`
2. Quiz creation must happen before quiz-specific actions (comments/questions/answers)
3. Comment creation must happen before comment detail/update/delete tests
4. Question and answer creation must happen before is-correct tests
5. Cleanup tests should run last

If you run a single test in isolation, it may fail due to missing suite variables.

Test Cases (What Each Test Does)

Authentication

Login And Print Token

- Sends login request with `${EMAIL}` and `${PASSWORD}`
- Verifies HTTP 200
- Extracts token or access_token from JSON response
- Saves `${token}` as a suite variable
- Logs token to console

Quizzes

Get All Quizzes

- Calls GET `/api/all/quizzes`
- Verifies HTTP 200
- Logs each quiz title returned

Create New Quiz

- Calls POST `/api/quizzes` with:
  - title, description, category_id
  - questions array with answers array
- Verifies HTTP 200 or 201
- Saves `${quiz_id}` from response

Logs response and created quiz ID

Verify Created Quiz ExistsVerify Created Quiz Exists

Calls GET `/api/quizzes/${quiz_id}`

- Verifies HTTP 200
- Handles two possible response formats:
  - `{ "quiz": {...} }` or direct `{...}`
- Checks quiz id and title match expected values

Get My Quizzes

- Calls GET `/api/quizzes`
- Verifies HTTP 200
- Logs each quiz as id - title
- Asserts `${quiz_id}` exists in the returned list

Categories

Get Categories

- Calls GET `/api/categories`
- Verifies HTTP 200
- Asserts status == „success” and categories exists
- Logs each category id - name

Get Categories Select

- Calls GET `/api/categories/select`
- Verifies HTTP 200
- Asserts status == „success” and categories exists (id=>name map)

Comments

Create Quiz Comment

- Calls POST `/api/quizzes/${quiz_id}/comments`
- Verifies HTTP 200 or 201
- Validates response fields (id, content, rating)
- Saves `${comment_id}` as a suite variable

Get Quiz Comments

- Calls GET `/api/quizzes/${quiz_id}/comments`
- Verifies HTTP 200
- Logs each comment row: id | user_name | rating | content
- Asserts at least one comment exists

Get Comment Details

- Calls GET `/api/quizzes/${quiz_id}/comments/${comment_id}`
- Verifies HTTP 200
- Checks comment belongs to `${quiz_id}` and has correct id

Update My Comment

- Calls PUT `/api/quizzes/${quiz_id}/comments/${comment_id}`
- Verifies HTTP 200
- Checks updated content and rating match expected values

Delete My Comment

- Calls DELETE `/api/quizzes/${quiz_id}/comments/${comment_id}`
- Verifies HTTP 200

Get Comments Rating Summary

- Calls GET `/api/quizzes/${quiz_id}/comments/rating`
- Verifies HTTP 200
- Checks presence of average_rating and total_comments
- Confirms quiz_id matches `${quiz_id}`

Questions & Answers

Add Question To Created Quiz

- Calls POST `/api/quizzes/${quiz_id}/questions`
- Verifies HTTP 201
- Saves `${question_id}` as suite variable
- Confirms returned quiz_id matches `${quiz_id}`

Add Answer To Question

- Calls POST `/api/questions/${question_id}/answers`
- Verifies HTTP 201
- Saves `${answer_id}` as suite variable

Is Correct Should Return True

- Calls POST `/api/questions/is-correct/${answer_id}`
- Verifies HTTP 200
- Asserts:
  - status == „success”
  - is_correct == true
  - returned IDs match `${answer_id}` and `${question_id}`

Negative / Error Handling Tests

Access Protected Endpoint Without Token

- Calls protected GET `/api/quizzes` without Authorization header
- Uses Run Keyword And Ignore Error to prevent suite failure on 401/403
- Expects request to fail with 401 or 403
- Logs request headers and error message

Get Nonexistent Quiz Returns 404

- Calls GET `/api/quizzes/99999999`
- Expects failure with 404
- Logs error output

Create Quiz Validation Returns 422

- Attempts invalid quiz creation (empty title + empty questions list)
- Expects failure with 422
- Logs request payload and error output

Is Correct Nonexistent Answer Returns 404

- Calls POST `/api/questions/is-correct/99999999`
- Expects failure with 404
- Logs error output

Save Quiz Result Rejects Invalid Request

- Calls POST `/api/quizzes/save-result`
- This is intentionally a negative test expecting 422
- Logs request payload and error output

Results & Ranking

Get My Quiz Results

- Calls GET `/api/quiz-results`
- Verifies HTTP 200
- Asserts at least one result exists

Get Ranking Data

- Calls GET `/api/get-ranking-data`

Accepts HTTP 200 or 201 (depending on API behavior)

Cleanup

Delete Created Quiz

- Attempts to delete quiz using:
  - DELETE `/api/quizzes/${quiz_id}` first
  - if not available (404), tries DELETE `/quizzes/${quiz_id}`
- Logs request, response or error
- Ensures quiz deletion succeeds with HTTP 200