Quiz Platform API Documentation

Backend Team

May 2025

Contents

1 Introduction

The Quiz Platform API enables a web application to manage users, quizzes, and statistics for Students, Instructors, and Admins. This document is tailored for frontend developers to integrate with the API.

- **Base URL**: `http://localhost:5000`
- **Authentication**: JWT tokens in `Authorization: Bearer <token>`
- **Content Type**: `application/json`

2 Authentication

Most endpoints require a JWT token obtained via registration and login.

2.1 Register User

POST `/api/auth/register`

```
{
  "email": "student@example.com",
  "password": "password123",
  "role": "Student"
}
```

Response (201):

```
{
  "userId": "671234567890123456789012",
  "email": "student@example.com",
  "role": "Student"
}
```

2.2 Login User

POST `/api/auth/login`

```
{
  "email": "student@example.com",
  "password": "password123"
}
```

Response (200):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "671234567890123456789012",
    "email": "student@example.com",
    "role": "Student"
  }
}
```

3 Information Flow

- **Student**: Register → Login → Create Profile → List Quizzes → Submit Attempt → View Results.
- **Instructor**: Register → Admin Approval → Create/Update Quizzes → View Statistics.
- **Admin**: Login → Approve Instructors → Manage Users → View Stats.

4 Key Endpoints

4.1 Update Profile (Student)

PUT `/api/users/profile`

Headers: `Authorization: Bearer <token>`

```
{
  "firstName": "John",
  "lastName": "Doe",
  "yearOfStudy": 1,
  "department": "Computer Science",
  "rollNumber": "CS001"
}
```

Response (200):

```
{
  "profile": {
    "_id": "671234567890123456789014",
    "userId": "671234567890123456789012",
    "firstName": "John",
    "lastName": "Doe",
    "yearOfStudy": 1,
    "department": "Computer Science",
    "rollNumber": "CS001"
  }
}
```

4.2 Get Assigned Quizzes (Student)

GET `/api/quizzes?status=active`

Headers: `Authorization: Bearer <token>`

Response (200):

```
[
  {
    "_id": "671234567890123456789015",
    "title": "Sample Quiz",
    "yearOfStudy": 1,
    "startTime": "2025-05-12T12:00:00Z",
    "endTime": "2025-05-12T14:00:00Z",
    "duration": 60
  }
]
```

4.3 Submit Quiz Attempt (Student)

POST `/api/quizzes/:quizId/attempt`

Headers: `Authorization: Bearer <token>`

```
{
  "answers": [
    {
      "questionId": "abc123",
      "selectedOptionIds": ["opt2"]
    }
  ]
}
```

Response (201):

```
{
  "attemptId": "671234567890123456789018",
  "quizId": "671234567890123456789015",
  "totalScore": 30,
  "isScored": true
}
```

4.4 Create Quiz (Instructor)

POST `/api/quizzes`

Headers: `Authorization: Bearer <token>`

```
{
  "title": "Math Quiz",
  "yearOfStudy": 1,
  "startTime": "2025-05-13T10:00:00Z",
  "endTime": "2025-05-13T12:00:00Z",
  "duration": 90,
  "questions": [
    {
      "text": "What is 3+3?",
      "options": [
        { "text": "5", "isCorrect": false },
        { "text": "6", "isCorrect": true }
      ],
      "score": 15
    }
  ]
}
```

Response (201):

```
{
  "quizId": "671234567890123456789017",
  "title": "Math Quiz",
  "yearOfStudy": 1
}
```

4.5 Get Quiz Statistics (Instructor)

GET `/api/quizzes/:quizId/statistics`

Headers: `Authorization: Bearer <token>`

Response (200):

```
{
  "quizId": "671234567890123456789015",
  "totalAttempts": 1,
  "averageScore": 30,
  "highestScore": 30,
  "lowestScore": 30,
  "attemptsByYear": [
    { "yearOfStudy": 1, "count": 1 }
  ]
}
```

4.6 Approve Instructor (Admin)

PUT `/api/admin/users/:userId/approve`

Headers: `Authorization: Bearer <token>`

```
{
  "isApproved": true
}
```

Response (200):

```
{
  "userId": "671234567890123456789013",
  "email": "instructor@example.com",
  "role": "Instructor",
  "isApproved": true
}
```

5 Error Handling

- **400 Bad Request**: Invalid input.
- **401 Unauthorized**: Missing/invalid token.
- **403 Forbidden**: Insufficient permissions.
- **404 Not Found**: Resource not found.
- **500 Internal Server Error**: Server issue.

6 Example Workflow

6.1 Student Workflow

1. Register: `POST /api/auth/register`
2. Login: `POST /api/auth/login`
3. Create Profile: `PUT /api/users/profile`

4. List Quizzes: `GET /api/quizzes?status=active`
5. Submit Attempt: `POST /api/quizzes/:quizId/attempt`
6. View Results: `GET /api/quizzes/:quizId/results`

7 Frontend Tips

- Store tokens securely (HttpOnly cookies recommended).
- Validate inputs client-side to reduce 400 errors.
- Poll for quiz results after `endTime`.
- Use Axios with interceptors for token management.