2019

YouTestMe

YouTestMe GetCertified

API User Manual

youtestme

Table of Contents

1 Introduction

This document contains the information regarding YouTestMe GetCertified REST API. To access the API, you need to have an ***admin*** account. The authentication that is used in the API is a combination of Basic Auth and JWT. Username and password should be sent to **/auth/login** using Basic Auth and HTTP GET Request to obtain the token (that the response consists of). After that, every request (except those that are sent to /auth/login) should contain that token in the headers, the Header should be **Authorization,** and the value should be **Bearer {token}**, while {token} should be the actual token. **Every request sent to the API should be sent over HTTPS.**

Every POST or PUT request must contain the header **Content-Type** whose value is **application/json**.

2 User Service

2.1 List Users

List user accounts with their information (userId, username, firstName, lastName, email, userRole, userStatus), using the HTTP GET method. The format of the response is JSON.

Provided	Uri	Comment
List of all users	/users	
User by username	/users?username=XXXXX	Uses query parameter, an actual username needs to be appended to the Uri

2.2 Get user

Get the user account information for the specified userId, using the HTTP GET method. If the user account does not exist, an error response is returned. The format of the response is JSON.

Provided	Uri	Comment
User by userId	/users/{userId}	The type of userId is integer.

2.3 Create user

Create a new user account with the following properties: username, email, firstName, lastName, userRole, and userStatus specified in JSON format using an HTTP POST method. If any of the attributes doesn't contain a value, an error will be returned in the response. The userRole attribute should contain value ATT, ADM, or INS, if that user represents student, administrator, or instructor, respectively. The userStatus attribute should contain value ACT, DEL if the user's status is active or inactive, respectively. If userStatus is not specified, the status ACT will be assigned to the user. The response is the content of the new user along with HTTP Status 201 - Created.

Provided	Uri	Comment
User creation	/users	

2.4 Update user

Update user account properties: email, firstName, lastName, userRole, and userStatus specified in JSON format using an HTTP PUT method. The response is the new content of the user, along with HTTP Status 200 - OK. If the user with specified userId does not exist, an exception will be returned in the response.

Provided	Uri	Comment

User update	/users/{userId}	Example (if you want to change userRole of some user to ADM (admin), request in JSON format should look like this):
		{
		"userRole": "ADM"
		}

2.5 Delete user

Delete user specified with userId using the HTTP DELETE method. The response contains HTTP Status 204 - No Content. If the user with specified userId does not exist, an error will be returned in the response.

Provided	Uri	Comment
User deletion	/users/{userId}	

3 Retrieving test results

3.1 List test results

List the results of all tests specified by query parameters, and if parameters are not specified, listing the results of all tests. The basic Uri for this functionality (without query parameters) is **/testresults.** HTTP GET method is used.

Query parameters:

Query parameters	Type	Comment
username	String	

userId, testId	integer	**NOTE**: CourseId represents the Id of the group user belongs. The rule can be applied to the whole document.
start, size	integer	
after	Timestamp	Example: How to forward Timestamp after=2018-14-01 15:00:00

The combinations of query parameters are listed below:

Provided	**Query parameters**
List of the results of all tests	/
List of the results of all tests taken by a particular user, specified by userId	userId
List of the results of all tests taken by a particular user, specified by username	username
List of the results of the particular test, specified by testId	testId
List of the results of all tests taken after a particular moment in time, specified by parameter after	after
List of the results of all tests taken before the particular moment in time, specified by parameter before	before
List of the results of all tests paginated, specified by start and size	start, size
List of the results of all tests taken between moments specified by after and before	after, before
List of the results of the particular test, specified by testId, taken after a particular moment in time, specified by parameter after	testId, after
List of the results of the particular test, specified by testId, taken before the particular moment in time, specified by parameter before	testId, before
List of the results of the particular test, specified by testId, taken between moments specified by parameters after and before	testId, after, before
List of the results of all tests in the particular group, specified by courseId, taken between moments specified by parameters after and before	courseId, after, before

3.2 Get test result for a user

Get the result of a user on a test specified by userId and testId, respectively. If there is more than one attempt, the most recent attempt is returned. The type of testId and userId is BigDecimal. Also, a link is included in the response for all answers by that user on that test.
HTTP GET method is used.
If the user with those testId and userId does not exist, an error will be returned in the response.

Provided	Uri
Result of a particular user on a particular test	/testresults/{testId}_{userId}

3.3 List user answers on the test

List all answers to a particular test taken by a particular user. If there is more than one attempt of that user on that test, answers returned are those from the most recent attempt.
HTTP GET method is used.

Provided	Uri
All answers of a particular user on a particular test	/testresults/{testId}_{userId}/answers

4 Creating a question

4.1 List questions

List all questions, or all questions of a specific type, specified by the query parameter **type**. HTTP GET method is used.

Provided	Query parameters	Uri
List of all questions	/	/questions
List of all questions of particular type	type	/questions?type=XXXXX

Question types:

- SNC - Single Choice Question
- MLC - Multiple Choice Question
- TFC - True False Question
- ESY - Essay Question
- ORD - Ordering Question
- FBL - Fill in the Blanks Question
- MCH - Matching Question

Example Uri for listing all questions of the Single Choice Question type: /questions?type=SNC

4.2 Get question

Get a question specified by questionId. Uri for this functionality is /questions/{questionId}. , and the request used in this case is HTTP GET Request. If the requested question does not exist, an error will be returned in the response.

4.3 Create a question

Create a new question using an HTTP POST request. The response is the content of the new question, along with HTTP Status 201 - Created.

Provided	Uri	Comment
Question creation	/questions	HTTP POST Request should be used

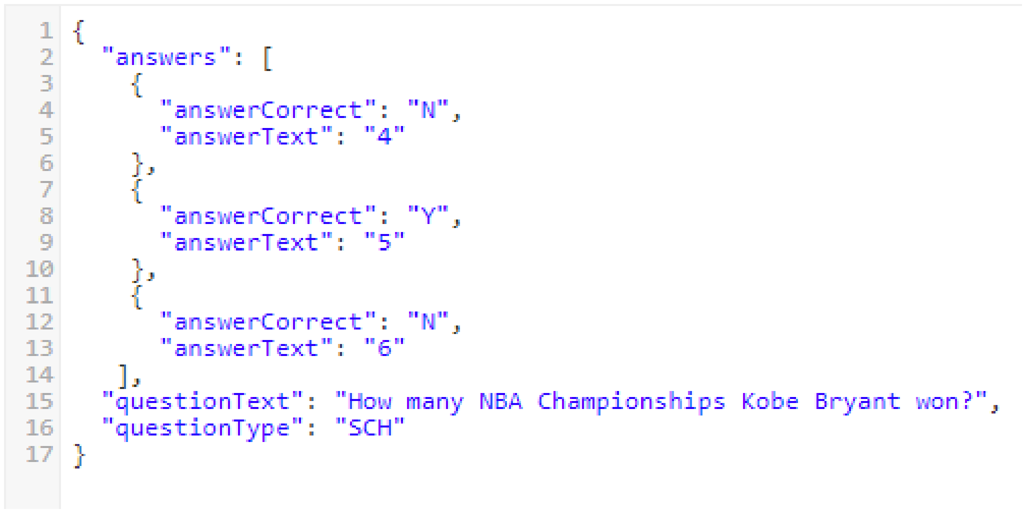
List of attributes of the question, and their types:

Attribute	Type	Comment
questionText	String	
questionType	String	• SNC - Single Choice Question • MLC - Multiple Choice Question • TFC - True False Question • ESY - Essay Question • ORD - Ordering Question • FBL - Fill in the Blanks Question • MCH - Matching Question
points	integer	
penalty	integer	
duration	integer	In seconds
difficultyCode	String	DHA - hard DME - medium DEA - easy
answers		

List of attributes of the answer, and their types:

Attribute	Type	Comment
answerText	String	The only required attribute of the answer
answerCorrect	String	Possible values: Y if the answer is correct, N if the answer is incorrect; If the question is an ordering question, the N value should be given for every answer.
answerWeight	BigDecimal	Weight of the answer
ordinalNumber	BigDecimal	If the type of the question is either matching or ordering question

Example:



```
{
  "answers": [
    {
      "answerCorrect": "N",
      "answerText": "4"
    },
    {
      "answerCorrect": "Y",
      "answerText": "5"
    },
    {
      "answerCorrect": "N",
      "answerText": "6"
    }
  ],
  "questionText": "How many NBA Championships Kobe Bryant won?",
  "questionType": "SCH"
}
```

4.4 Update question

Update question properties using the HTTP PUT method. The request should be sent to **/questions/{questionId}**. The response is the new content of the question, along with HTTP Status 200 - OK.
If the question specified with questionId does not exist, an error will be returned in the response.

4.5 Delete question

Delete a question specified with questionId using the HTTP DELETE method. The request should be sent to **/questions/{questionId}**. The response contains HTTP Status 204 - No Content. If the question specified with questionId does not exist, an error will be returned in the response.

4.6 List pools

List all question pools with their properties (poolId, parentPoolId, poolName, poolDescription, poolType), using HTTP GET method. The request should be sent to **/pools**.

4.7 Get pool

Get a pool, specified by poolId, using the HTTP GET method, which should be sent to **/pools/{poolId}**. If the pool with specified poolId does not exist, an error will be returned in the response.

4.8 Create a pool

Create a new question pool specified with the following properties: parentPoolId, poolName, poolDescription, poolType. An HTTP POST is used. The request should be sent to **/pools**. The response is the content of the new pool, along with HTTP Status 201 - Created.

4.9 Update a pool

Update properties of the pool specified with poolId using the HTTP PUT method. The request should be sent to **/pools/{poolId}**. The response is the new content of that pool, along with HTTP Status 200 - OK. If the pool with specified poolId does not exist, an error will be returned in the response.

4.10 Delete pool

Delete the pool specified with poolId, using the HTTP DELETE method. The request should be sent to **/pools/{poolId}**. The response contains HTTP Status 204 - No Content. If the pool with specified poolId does not exist, an error will be returned in the response.

4.11 List pool questions

List all pool questions with their information (poolId, questionId), specified by the query parameter: by poolId or questionId. If query parameters are not specified, all pool question records will be listed. The HTTP GET method is used. The request should be sent to **/poolquestions**.

4.12 Insert question into a pool

Insert the question specified with questionId into the pool specified with poolId using the HTTP POST method. The request should be sent to **/poolquestions**. The response is the content of the new pool question, along with HTTP Status 201 - Created.

4.13 Remove question from a pool

Remove the question specified with questionId from the pool specified with poolId using the HTTP DELETE request. The request should be sent to **/poolquestions/{poolId}_{questionId}**. The response contains HTTP Status 204 - No Content. If the question specified with questionId does not exist in the pool specified with poolId, an error will be returned in the response.

5 Assigning a test to a user

5.1 List quizzes

List all quizzes with their information (quizDefinitionId, quizName), using the HTTP GET method. The request should be sent to **/quizzes**.

5.2 Get quiz

Get the quiz specified with quizDefinitionId using the HTTP GET method, which should be sent to **/quizzes/{quizDefinitionId}**. If the quiz specified with specified quizDefinitionId does not exist, an error will be returned in the response.

5.3 List unique quizzes

List all unique quizzes with their information (uniqueQuizId, quizDefinitionId, uniqueQuizName, quizDefinitionName), using HTTP GET method. The request should be sent to **/tests**.

5.4 Get a unique quiz

Get the unique quiz specified with uniqueQuizId using the HTTP GET method, which should be sent to **/tests/{uniqueQuizId}**. If the quiz specified with specified quizDefinitionId does not exist, an error will be returned in the response.

5.5 List quiz instances

List all quiz instances, with their information (quizInstanceId, uniqueQuizId, userIdTested, quizInstanceStatus, enabledFrom, enabledTo, duration) specified by query parameter **quiz** (which represents uniqueQuizId) using HTTP GET method. If the query parameter is not specified, all quiz instances are returned. The request should be sent to **/quizinstances**.

Provided	Uri	Query parameter	Type of the query parameter
All quiz instances	/quizinstances	/	/
All quiz instances specified by quiz parameter (quiz represents uniqueQuizId)	/quizinstances?quiz=XXXXX	quiz	integer

5.6 Insert quiz instance

Create a new quiz instance, which assigns a test to a user. The uniqueQuizId, userIdTested, quizInstanceStatus, enabledFrom, enabledTo, duration should be sent in JSON format using HTTP POST request. The request should be sent to **/quizinstances**. The response is the content of the new quiz instance, along with HTTP Status 201 - Created.

There are following types of quizInstanceStatus: ATR (attempted with report), ATN (attempted without report), NAT (not attempted), QSS (suspended)

Property	Attribute type	Required	Comment
uniqueQuizId	BigDecimal	yes	
userIdTested	BigDecimal	yes	
quizInstanceStatus	String	no	If not specified, quizInstanceStatus will be NAT
enabledFrom	Timestamp	no	
enabledTo	Timestamp	no	
duration	BigDecimal	no	