Trivia API

Release 1.0

Scott R Smith

Nov 30, 2019

CONTENTS

CHAPTER

ONE

INSTALATION AND OVERVIEW

1.1 Introduction to the Full Stack Trivia API

The Trivia application is a web-based app that allows for the playing of trivia games.

The application:

1) Display questions - both all questions and by category. Questions show the question, category, and difficulty rating by default and can show/hide the answer.
2) Delete questions.
3) Add questions and require that they include a question and answer text.
4) Search for questions based on a text query string.
5) Play the quiz game, randomizing either all questions or within a specific category.

1.2 Getting Started - Backend

1.2.1 Installing Dependencies

Python 3.7

```
Follow instructions to install the latest version of python for your platform in the␣
↪docs.

https://docs.python.org/3/using/unix.html#getting-and-installing-the-latest-version-
↪of-python

PIP Dependencies
~~~~~~
```

Once you have your virtual environment setup and running, install dependencies by navigating to the root directory and running:

```
pip install -r requirements.txt
```

This will install all of the required packages we selected within the `requirements.txt` file.

1.2.1.1 Key Dependencies

- Flask is a lightweight backend microservices framework.
- SQLAlchemy is the Python SQL toolkit and ORM.
- Flask-CORS is the extension used to handle cross-origin requests from the frontend server.

1.3 Database Setup

With Postgres running, populate the database using the trivia.psql file provided. From the backend folder in terminal run:

```
psql trivia < trivia.psql
```

1.4 Running the server

From within the `backend` directory to run the server, execute:

```
export FLASK_APP=flaskr
export FLASK_ENV=development
flask run
```

1.5 Documentation

1.5.1 Opening the API Documentation

Documentation is generated with Sphinx.

HTML Documentation

```
From the root folder, open the index file in a browser

.. code-block:: bash

   ./docs/build/html/index.html

PDF Documentation
~~~~~~~~~~~~~~~~~

The PDF version of the documentation is located in the root project directory. Named␣
↪triciaapi.pdf

Generating documentation
^^^^^^^^^^^^^^^^^^^^^^^^

Documentation is generated with Sphinx.

Installing Sphinx and support tools
~~~~~~~~~~~~~~~~~
```

To install Sphinx, reference the documents at https://www.sphinx-doc.org/en/master/usage/installation.html

For example:

```
pip install -U sphinx
```

Install dependencies by navigating to the `root` project directory and running:

```
cd docs
pip install m2r
pip install recommonmark
pip install rinohtype

pip install -r requirements.txt
```

Generating the documentation

```
Generate the documentation with the following commands:

.. code-block:: bash

   # From the root project directory
   # Convert readme to rst to be included in generated docs
   m2r README.md README.rst --overwrite
   cp -R README.rst ./docs/source
   cd ./docs
   make html
   # Make pdf
   make latexpdf
   cd ..
   cp -R ./docs/build/latex/triviaapi.pdf .

API End Points
--------------

The following APIs are available. Detailed html documentation can be found in the
↪'docs' folder.


* Create or Search Questions. This API will create a new question or search questions.
* Retrieve Questions. This API will retrieve a page of 10 trivia questions sorted by␣
↪question_id.
* Delete Question. This API will delete a question from the database.
* Retrieve Category. This API will retrieve a list of trivia categories.
* Retrieve Category Questions. This API will retrieve a list of questions by category␣
↪Id.
* Play Quiz. This API will play the quiz. Presenting a random, non-repeated question␣
↪in the category.

Error Handling
--------------

Errors are returned as JSON objects in the following format:

.. code-block:: bash

   {
       "success": False,
```

(continues on next page)

(continued from previous page)

```
       "error": 400,
       "message": "Bad Request"
   }

The API will return three error types when requests fail:


* 400: Bad Request
* 404: Resource Not Found
* 405: Method Not Allowed
* 422: Not Processable
* 500: Internal Server Error

Testing
-------

Testing is done with UnitTest

From the trivia/backend folder. Run:

.. code-block:: bash

   . ./test.sh

Full Stack Trivia API Frontend
------------------------------

Installing Dependencies
^^^^^^^^^^^^^^^^^^^^^^^

Installing Node and NPM
~~~~~~~~~~~~~~~~~~~~~~~

This project depends on Nodejs and Node Package Manager (NPM). Before continuing, you␣
↪must download and install Node (the download includes NPM) from `https://nodejs.com/
↪en/download <https://nodejs.org/en/download/>`_.

Installing project dependencies
~~~
```

This project uses NPM to manage software dependencies. NPM Relies on the package.json file located in the `frontend` directory of this repository. After cloning, open your terminal and run:

```
npm install
```

1.6 Required Tasks

1.7 Running Your Frontend in Dev Mode

The frontend app was built using create-react-app. To run the app in development mode use `npm start`. You can change the script in the `package.json` file.

Open http://localhost:3000 to view it in the browser. The page will reload if you make edits.

CHAPTER

TWO

TRIVA API CONTROLLERS

2.1 Introduction

The Trivia API includes multiple REST methods. These methods are to create, list, and search questions as well as perform operations to list by category and search. The Quizzes API is for playing the game.

- Create or Search Questions. This API will create a new question or search questions.
- Retrieve Questions. This API will retrieve a page of 10 trivia questions sorted by question_id.
- Delete Question. This API will delete a question from the database.
- Retrieve Category. This API will retrieve a list of trivia categories.
- Retrieve Category Questions. This API will retrieve a list of questions by category Id.
- Play Quiz. This API will play the quiz. Presenting a random, non-repeated question in the category.

controller.**create_or_search_question**()

Create or Search Questions

This API will create a new question or search questions.

Create a new question

- Sample Call create question:

```
curl -X POST http://localhost:5000/questions
     -H 'content-type: application/json'
     -d '{"question":"One plus one is","answer":"2","difficulty":1,"category
↪":1}'
```

- Expected Success Response:

```
HTTP Status Code: 200

{
 "success": true
}
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
    "success": False,
    "error": 404,
    "message": "resource not found"
}
```

Search Questions

- Sample Call search question:

```
curl -X POST http://localhost:5000/questions
     -H 'content-type: application/json'
     -d '{"searchTerm":"Anne"}'
```

- Expected Success Response:

```
HTTP Status Code: 200
{
 "questions": [
    {
    "answer": "Tom Cruise",
    "category": 5,
    "difficulty": 4,
    "id": 4,
    "question": "What actor did author Anne Rice first denounce, then praise␣
↪in the role of her beloved Lestat?"
    }
 ],
 "success": true,
 "total_questions": 1
}
```

- Search not found:

```
HTTP Status Code: 200

{
 "questions": [],
 "success": true,
 "total_questions": 0
}
```

controller.**retrieve_questions**()

Retrieve API Questions

This API will retrieve a page of 10 trivia questions sorted by question_id. Pages are designated by the 'page' url parameter.

- Sample Call:

```
curl -X GET http://localhost:5000/questions?page=1
     -H 'content-type: application/json'
```

- Expected Success Response:

```
HTTP Status Code: 200

{
    "categories": [
        "Science",
        "Art",
        "Geography",
        "History",
        "Entertainment",
        "Sports"
```

(continues on next page)

(continued from previous page)

```
    ],
    "current_category": 5,
    "questions": [
        {
        "answer": "Apollo 13",
        "category": 5,
        "difficulty": 4,
        "id": 2,
        "question": "What movie earned Tom Hanks his third straight Oscar␣
↪nomination, in 1996?"
        },
        {
        "answer": "The Palace of Versailles",
        "category": 3,
        "difficulty": 3,
        "id": 14,
        "question": "In which royal palace would you find the Hall of Mirrors?
↪"
        }
    ],
    "success": true,
    "total_questions": 20
}
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
    "success": False,
    "error": 404,
    "message": "resource not found"
}
```

`controller.`**`delete_question`**`(`*question_id*`)`

Delete a Question from the database

This API will delete a question from the database.

- Sample Call:

```
curl -X DELETE http://localhost:5000/questions/2
     -H 'content-type: application/json'
```

- Expected Success Response:

```
HTTP Status Code: 200
{
    "deleted": 2,
    "success": true
}
```

- Expected Fail Response:

```
HTTP Status Code: 422
{
 "error": 422,
 "message": "unprocessable",
```

(continues on next page)

(continued from previous page)

```
 "success": false
}
```

controller.**retrieve_category**()

Retrieve Trivia Categories

This API will retrive a list of trivia categories.

- Sample Call:

```
curl -X GET http://localhost:5000/categories
     -H 'content-type: application/json'
```

- Expected Success Response:

```
HTTP Status Code: 200

{
    "categories": [
                    "Science",
                    "Art",
                    "Geography",
                    "History",
                    "Entertainment",
                    "Sports"
                    ],
    "success": true,
    "total_categories": 6
}
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
    "success": False,
    "error": 404,
    "message": "resource not found"
}
```

controller.**retrieve_category_questions**(*category_id*)

Retrieve Category Questions

This API will retrieve a list of questions by category Id.

- Sample Call:

```
curl -X GET http://localhost:5000/categories/1/questions
     -H 'content-type: application/json'
```

- Expected Success Response:

```
HTTP Status Code: 200

{
"current_category": 2,
"questions": [
    {
    "answer": "Escher",
```

(continues on next page)

(continued from previous page)

```
    "category": 2,
    "difficulty": 1,
    "id": 16,
    "question": "Which Dutch graphic artist-initials M C was a creator of␣
↪optical illusions?"
    },
    {
    "answer": "Jackson Pollock",
    "category": 2,
    "difficulty": 2,
    "id": 19,
    "question": "Which American artist was a pioneer of Abstract␣
↪Expressionism, and a leading exponent of action painting?"
    }
],
"success": true,
"total_questions": 24
}
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
    "success": False,
    "error": 404,
    "message": "resource not found"
}
```

`controller.quizzes()`

Play Quiz

This API will play the quiz. Presenting a random, non-repeated question in the category

- Sample Call:

```
curl -X POST http://localhost:5000/quizzes
     -H 'content-type: application/json'
     -d '{"previous_questions":[],"quiz_category":{"type":"Science","id":"0"}}
↪'
```

- Expected Success Response:

```
HTTP Status Code: 200

{
"question": {
    "answer": "The Liver",
    "category": 1,
    "difficulty": 4,
    "id": 20,
    "question": "What is the heaviest organ in the human body?"
},
"success": true
}
```

- Expected Fail Response:

```
HTTP Status Code: 404
{
    "success": False,
    "error": 404,
    "message": "resource not found"
}
```

CHAPTER

THREE

TRIVA API MODELS

Introduction

The Trivia app includes two sql Alchemy classes used manage the trivia questions and categories

- Questions: List of questions and anssers
- Categories: List of categories for the questions

class models.**Question**(*request_json*)

This class is the model for each question plus the support CRUD operations

id

id is the auto assigned primary key Type: Integer, Primary key. Required.

question

question is the question to be presented to the trivia players Type: String, Required.

answer

answer the answer to the trivia question Type: String, Required.

category

category is the question category Type: String, but respresented as an Integer and is the Forigen key of Category. Required.

difficulty

difficulty is the difficulty level of the question Type: Integer, Required.

insert()

Insert the Question object, as a row, into the trivia database

update()

Update the Question object with any changes

delete()

Delete the Question object from the trivia database

format()

Return the Question record object as a Python Dictionary

class models.**Category**(*type*)

This class is the model for the list of tricia categories

id

id is the auto assigned primary key Type: Integer, Primary key. Required.

type

type is the question category name Type: String. Required.

format()

Return the category record object formatted as a Python Dictionary

PYTHON MODULE INDEX

INDEX