# Trivia Quiz Website

**Project Overview**

The goal of this project is to develop a Trivia Quiz API that allows users to retrieve trivia questions, answer them, and receive feedback on their answers. The API will serve as the backend for a quiz application, enabling users to engage in a fun and interactive trivia experience.

**Backend**

* First Started with the creation of the API.
* Requirements of the API were as follows:
* Question Retrieval:
* Users can request a set number of trivia questions from various categories.
* Questions will be accompanied by multiple-choice answers.
* Answer Submission:
* Users can submit their answers to the questions.
* The API will check the submitted answers against the correct answers and return feedback.
* Score Calculation:
* After answering a set of questions, the API will calculate the user's score and provide a summary.
* API Endpoints

1. GET /questions: Retrieve a specified number of trivia questions from a random selection.

* Query Parameters:
* count: Number of questions to retrieve (e.g., /questions?count=5).

1. POST /answer: Submit an answer for a specific question.

* Request Body:json

{

"question\_id": "123",

"user\_answer": "A"

}

* Response:
* Returns whether the answer is correct and the correct answer.

1. GET /score: Retrieve the user's score after completing a set of questions.

* Technologies used:
* Framework: Flask for API development.
* Testing: Postman to test the API endpoints.
* Went slowly by adding each feature at a time and finished it with the successful linking with the database for questions.
* If using POSTMAN for testing POST request should enter the below in body->raw->JSON:

{

"answers": [

{

"id": 2,

"answer": "Mark Twain"

},

{

"id": 6,

"answer": "4"

},

{

"id": 1,

"answer": "paris"

}

]

}

* If using curl for testing should use below command line:

curl -X POST http://127.0.0.1:5000/api/answer \

-H "Content-Type: application/json" \

-d '{"answers":[{"id":1,"answer":"Orwell"}]}'

**Frontend**

* Used a template from <https://themewagon.com/>
* After downloading the required template modified it to serve my requirements and outlook.

**Database**

* Database Schema
* Questions Table:
* Id: Primary Key
* Question Text: The text of the trivia question.
* Correct Answer: The correct answer for the question.
* Options: A list of multiple-choice options (e.g., A, B, C, D).
* Used POSTGRESQL for creating the database
* For updation of data:

UPDATE questions

SET difficulty=’easy’

WHERE difficulty=’easy’

* For insertion of questions:

INSERT INTO questions (question, options, answer, difficulty)

VALUES

('How many colors are there in the rainbow?', '["7", "8", "6", "5"]', '7', 'easy'),

('What’s a shape with five sides called?', '["Octagon", "Pentagon", "Hexagon", "Nonagon"]', 'Pentagon', 'easy')

* Connected it with the API by providing the necessary credentials in the API code.

**DOCKER**

* Created a Dockerfile for API, Frontend and Database.
* Created a docker-compose.yml and dockerignore file.
* Then ran the following command lines in terminal of VS Code after successful checking of its function:
* docker-compose up –build
* docker tag trivia-api sairam76/trivia-api:latest
* docker tag trivia-db sairam76/trivia-db:latest
* docker tag trivia-frontend sairam76/trivia-frontend:latest
* docker push sairam76/trivia-api:latest
* docker push sairam76/trivia-db:latest
* docker push sairam76/trivia-frontend:latest
* This creates docker images and uploads them into the docker hub.

**AWS**

* Launched an EC2 instance using the following data:
* Name: Trivia
* Application and OS Images: Amazon Linux 2023 AMI
* Instance Type: t2.micro
* Key Pair: quizz.pem
* Network settings: launch-wizard-2 (As it has inbound rules for port 8080, 5000, 5432)
* After successful launching, open local terminal and run the following commands to connect to the EC2 instance:
* ssh -I Downloads/Trivia\_Quiz/quizz.pem ec2-user@Public-IPv4-address (if the security key pair is available in same location no need to mention Downloads/Trivia\_Quiz)
* After connecting to the EC2 instance run the below:
* sudo su
* yum update -y
* yum install docker -y
* systemctl start docker
* systemctl enable docker
* curl -L "https://github.com/docker/compose/releases/download/$(curl -s https://api.github.com/repos/docker/compose/releases/latest | grep tag\_name | cut -d '"' -f 4)/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
* chmod +x /usr/local/bin/docker-compose
* docker-compose --version
* docker pull sairam76/trivia-frontend:latest
* docker pull sairam76/trivia-api:latest
* docker pull sairam76/trivia-db:latest
* docker run -d --name trivia-db -e POSTGRES\_DB=trivia\_db -e POSTGRES\_USER=postgres -e POSTGRES\_PASSWORD=password -p 5432:5432 sairam76/trivia-db
* docker run -d --name trivia-api --link trivia-db:db -p 5000:5000 sairam76/trivia-api
* docker run -d -p 8080:8080 sairam76/trivia-frontend

**Important Details**

* Create requirements.txt using below line:
* pip freeze > requirements.txt
* To create the .sql file:
* pg\_dump -U your\_db\_user -h your\_db\_host -p your\_db\_port -d your\_db\_name > trivia\_backup.sql
* To create a Dockerfile in the same folder as another Dockerfile, name the second Dockerfile with Dockerfile-feature without any suffix. Then enter your Dockerfile contents and save. It saves itself as a Dockerfile.

**Link for GitHub with the project**

* <https://github.com/PlayerX5/Trivia_Quiz>