Take home challenge PHP/SQL

Introduction

Thanks for applying to join Learnosity's development team. To help us understand your skills and experience, please complete this technical task. Read the whole scenario carefully, and **pay special attention to all instructions and submission criteria**. Take your time and submit a solution you are proud off!

The whole challenge should take you about 4-6 hours, which you can spread over 7 days from receiving the challenge. If you have any questions, please feel free to ask us for clarification.

Scenario

You work on an eLearning solution that allows students to complete online assessment activities. Activities are presented as a paged experience, where the student answers one question at a time and uses a "next" button to navigate through each question.

You are implementing a new feature for analysing how students interact with their online exams. Another team has already implemented a detailed event stream that is sent from the front-end assessment application in real time, and now you need to implement the backend solution to receive, persist and analyse the events. Your team is particularly interested in understanding how long students spend on their exam questions.

You are tasked with building an API that has an endpoint that receives events and stores them in an SQL database. You will also have to build other endpoints for querying the data.

The `*/receive*` endpoint should accept a JSON payload with an `*events*` array containing a set of event objects:

```
{
    "events": [
        {
            "name": "start",          // "start" | "next" | "stop"
            "timestamp": 1545188359,  // Unix timestamp of the event
            "user_id": "user134",     // User who triggered the event
            "activity_id": "Math-1"   // Activity that triggered the event
        }
    ]
}
```

The package contains a generator for sample events, that you can use to verify your approach.

Tasks

Part A:
Implement a REST app with a `*/receive*` endpoint that will accept payloads of events data in the format described above.
Hint: this part is the least important. It's ok to just link your code to the existing index.php. Feel free to use a framework to replace it, but don't spend too much time on this part - the existing index.php is intentionally simple.

Part B:
Design a storage model and persist the incoming data in the supplied SQL database.
We need to be able to reproduce your storage model locally. Please make sure you include the necessary SQL files and/or scripts in your submission that would allow us to do so.

Part C:

1. Implement an endpoint that returns how long each student took for each question in an activity. The endpoint should accept an `*activity_id*` and an array of `*user_ids*`, and return the breakdown of time spent by each student on each question in the activity.

2. Implement an endpoint for finding which activity takes the longest overall time (on average). The endpoint should return the `*activity_id*` and the average time students took to complete it.

Part D:
Pick one of the endpoints you implemented in Part C and cover it with unit and integration tests, or alternatively, use TDD and write your tests first ;-). To save time, you don't have to cover the other parts of the app.

Assumptions

- The other team has implemented the eventing as follows:
    - A `start` event is triggered when the student views the first question of the activity.
    - Subsequent navigations through the questions trigger `next` events.
    - A final `stop` event is triggered when the student submits their answer to the last question.
- The events data is sent in real time over the internet to your app from a variety of devices and remote locations. Safe and ordered delivery of each event is not guaranteed.

The challenge package

In the supplied package, you will find an existing docker-composer setup. You need to install docker and then run `**docker-compose up**` inside the package.

You can get docker for mac or windows here: https://www.docker.com/products/docker-desktop
If you run linux, please follow these instructions to install docker and docker-compose: https://docs.docker.com/install

If you use homebrew or any other third party installation method on Mac, you might run into trouble with port forwarding. We do not recommend using this sort of setup unless you are already familiar with it.

Before you start, make sure you can start the docker compose setup with `**docker-compose up**` and reach the dummy index.php under http://localhost:8321.

You are free to modify the setup, but it is not required to complete the challenge.

Folder:

- /app
  - **Please put your code inside the app directory**
  - Contains your application.
  - The app folder is mounted into the php-fpm and nginx containers. Any change to your files here will be instantly active. You don't need to rebuild the docker containers.
  - Entry point is app/html/index.php, app/html is the public folder. If you need to change this, either symlink the html folder or change the conf in nginx-backend (in that case you need to rebuild with `**docker-compose up --build**`)
- /events
  - Folder containing sample events to test your app. Run sendevents.php (after docker-compose up) to send the sample events to your app
- /nginx-backend
  - Docker container for nginx. You don't need to modify it, except you want to change the URL rewrite settings. It is already configured to rewrite all requests to index.php
- /php-fpm
  - The php-fpm process container. The code is not copied into this container!
  - You reach it from your local machine via http://localhost:8321
  - The following environment variables are provided for accessing the DB
    - MYSQL_DB_HOST: sql-db:3306
    - MYSQL_DB_USER: codingchallenge
    - MYSQL_DB_PASSWORD: codingchallenge
    - MYSQL_DATABASE: codingchallenge
    - DATABASE_URL: mysql://codingchallenge:codingchallenge@sql-db:3306/codingchallenge
- /sql
  - The SQL database container. This container runs MariaDB 10.4

    - You can reach this container from your local machine via mysql://codingchallenge:codingchallenge@localhost:8322/codingchallenge
    - An empty DB codingchallenge is ready for use in the challenge

Constraints

- Your solution should be as performant, elegant and robust as possible within the time constraints.
- Please think about scale
- Your main persistence is the SQL database.
- Your solution runs within the provided package.
- You are free to use any framework. However, please make sure we can clearly identify *your* work and how you solved the challenge. **Please do not use an ORM for Part C**, so we can assess your SQL skills.
- Please send us your solution as an archive (tar.gz/zip/7z) of the package folder.
- **Do not upload your solution to the internet. If you have problems delivering the package via email (because of size restrictions), we can provide means to deliver it.**

Good Luck!