API Performance Monitor (APM) Design Document

Understanding requirements.

- A client could be having multiple software applications running on different cloud platforms.
- Although each hosting platform offers analytics data, traversing multiple platforms could be tedious.
- Furthermore, when working with teams, sharing credentials is not preferred by many.
- There is sometimes a need for a solution that offers insights into traffic while offering discrete credentials to each user.
- API Performance Monitor (APM) can address these.

User

User Role	Interactions
User	View logs from other applications.

Software needs interfaces for the above user roles.

1. Software must allow new users to register by themselves and access the platform.
2. Passwords must be encrypted.

Software Architecture

- APM is deployed as a Java Spring Boot application with MongoDB as the database.
- Users interact with the HTML-CSS page.
- Other applications can make POST requests to the REST API.

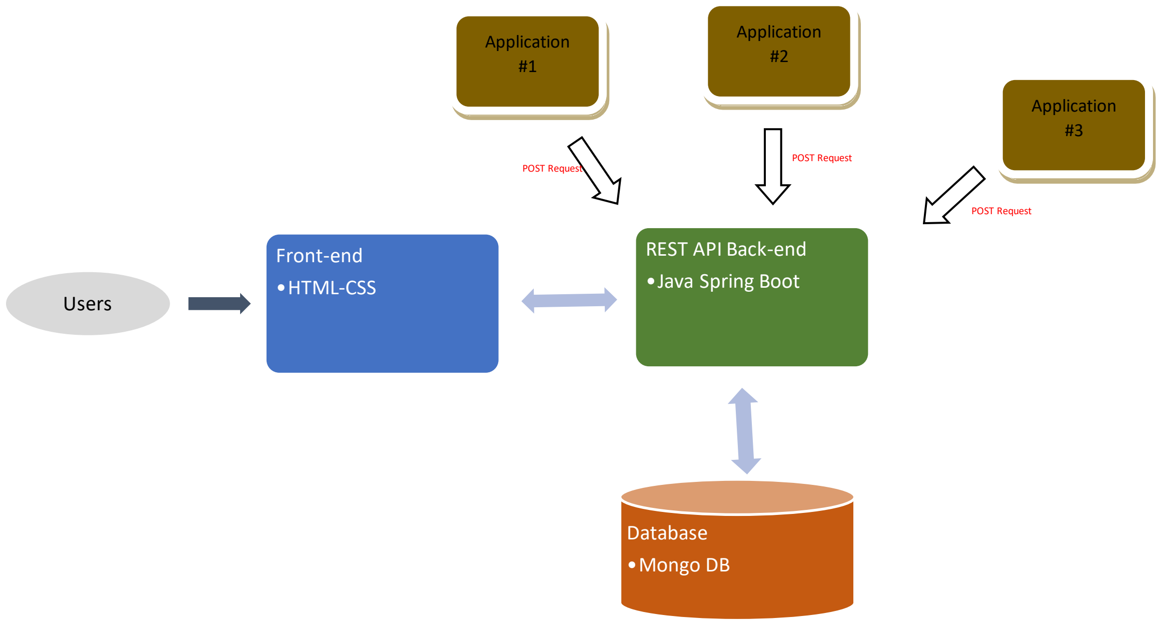

Figure 1: APM Architecture

Mongo DB Configurations:

api_calls

Stores logs from other applications.

Column	Datatype	Description
callId*	ObjectID	The value of this field is automatically added by MongoDB. This is the primary key of the collection.
callerMessage	String	Message from the application.
callerTimestampUTC	String	Timestamp at which the call was received. This is saved in the UTC time zone. This time is converted to local time of the user when viewed on the APM Dashboard.

apm_users

Stores information about the users.

Column	Datatype	Description
username*	String	User's username. This is the primary key.
emailID	String	User's email address.
password	String	Hash of user's password.
nameFirst	String	User's first name.
nameLast	String	User's last name.
timestampRegistration	String	Timestamp at which the user registered. This will be in the UTC time zone.
loginAttemptsFailed	Integer	This is to lock a user's account if they exceed a specific number of failed login attempts.
timestampAccountLocked	String	Timestamp at which the user's account was locked. Time is recorded with UTC as the time zone.

Frontend

The front is written in HTML with JavaScript scripting and CSS styling.

Backend - REST API

The Java Spring Boot REST API has an endpoint which other applications can hit with the logs.

JSON Web Tokens (JWT) is used to secure access to other endpoints.

MongoDB

To ensure fast operation, this project uses MongoDB as its database. Although, with Java Spring Boot other databases are also supported.

Docker

To support portability, I have also used Docker for running the project.

Trivia

For my college project I had to deploy 3 different applications on the Microsoft Azure cloud services.

Since I was on a tight budget, I used to frequently monitor the applications' usage to ensure I do not exceed the threshold. I found it tedious to look at the usage of each application.

I was also curious about their usage by their patrons. I then realized that if I could use a logging tool that other applications could log, I could easily monitor their usage. This led to the development of API Performance Monitor (APM).

Thank you very much.

Disclaimer

I have tried my best to thoroughly write about all aspects, however, there could be additional factors that I missed. Also, I have to say that APM is not fool proof as it was designed in a very tight schedule, there could be bugs that I missed. So, before you use it, I kindly encourage you to test it thoroughly before deploying.