Problem Statement for candidates

Application overview
Create a price alert application that triggers an email when the user's target price is achieved.

Say, the current price of BTC is 28,000$, a user sets an alert for BTC at a price of 33,000$. The application should send an email to the user when the price of BTC reaches 33,000$.

Similarly, say, the currency price of BTC is 35,000$, a user sets an alert for BTC at a price of 33,000$. The application should send an email when the price of BTC reaches 33,000$.

Things to do for the assignment

- Create a rest API endpoint for the user's to create an alert `alerts/create/`
- Create a rest API endpoint for the user's to delete an alert `alerts/delete/`
- Create a rest API endpoint to fetch all the alerts that the user has created.
    - The response should also include the status of the alerts (created/deleted/triggered/.. or any other status you feel needs to be included)
    - Paginate the response.
    - Include filter options based on the status of the alerts. Eg: if the user wanted only the alerts that were triggered, then the endpoint should provide just that)
- Add user authentication to the endpoints. Use JWT tokens.
- There is no need to add tests.
- Write a script that monitors the price of the cryptocurrency
- You can use this endpoint to fetch the latest price of the cryptocurrency: https://api.coingecko.com/api/v3/coins/markets?vs_currency=USD&order=market_cap_desc&per_page=100&page=1&sparkline=false
- When the price of the coin reaches the price specified by the users, send an email to all the users that set the alert at that price. (send mail using Gmail SMTP, SendGrid, etc)
- You should set up a background worker(eg: celery/python-script/go-script) to send the email. Use Rabbit MQ/Redis as a message broker.)

Bonus:

- Add a caching layer for the "fetch all alerts" endpoints.
- Everything in the bonus section is optional and can be ignored. However, just think about how you'd go about designing such a system with the aforementioned functionality.

Requirements

- You can use Python/ Go / Ruby.
- Use Postgres to store data (or any DB you feel that gets the job done)
- Need to use Rabbit MQ / Redis as a message broker for the task to send emails.
- Bundle everything inside a docker-compose file. Otherwise include setup instructions in the README.md file.

- Please fill this form after you complete the task: https://forms.gle/cXqAMk9TsWdsLEiK6

Candidates will be evaluated on

- The overall design of the system (keeping the bonus section in mind)
- Database design.
- The functionality of the web APIs.

Feel free to shoot us a message if you have any doubts or if you feel something needs to be flushed out. We'd be more than happy to help.

You can join this discord server to post your queries: https://discord.gg/qZmeSuWz

The deadline for the task is 10 PM, 10th September 2021.

Instructions

- You **must** make a GitHub repository for your project. Make sure that the repository is **publicly accessible** before you submit the form.
- You must not make any commits/changes to the GitHub repository after **10.00 PM on 10th September 2021**. All submissions violating this will be **disqualified immediately**.
- All plagiarised submissions will be disqualified. It is advised that you make your repository public **immediately after the submission deadline** so as to prevent people from copying your code. Please note that evaluations will commence right after the submission deadline, so you must ensure that the repository is public before evaluations begin.
- The Github repository **must** have a file called "**README.md**" which contains information about how to install and run your project. If you have deployed your application, you may include a link in the README.