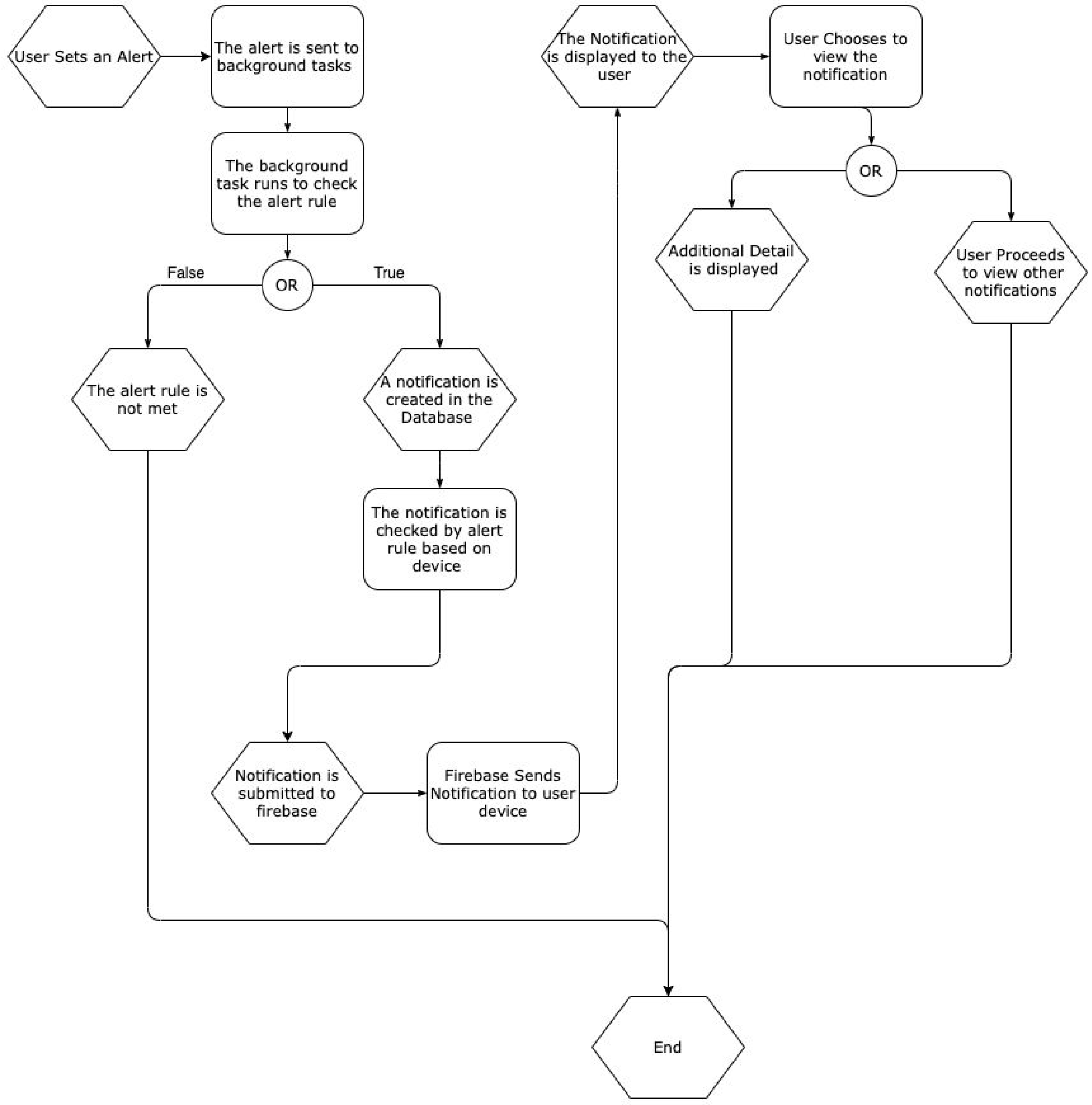
Focus Mobile Backend Take Home Challenge Question 3

The diagram below shows the flow of actions


User Sets an Alert
The alert is sent to background tasks
The background task runs to check the alert rule
OR
False
True
The alert rule is not met
A notification is created in the Database
The notification is checked by alert rule based on device
Notification is submitted to firebase
Firebase Sends Notification to user device
The Notification is displayed to the user
User Chooses to view the notification
OR
Additional Detail is displayed
User Proceeds to view other notifications
End

To get the best for a mobile application to track rises and falls in stock, the following are the requirements.

1. Google Play Store account
2. Firebase account
3. AWS account

The app will have a feature where a user can create an alert. The alert will have the details of which type of alert, the actions to be performed by the alert and when to make an action

The alert will be managed by a background task which continuously monitors the prices of stock and makes necessary alerts to different task queues. The task queues include AWS Lambda Functions, Linux CronJobs and async task queues.

The system will also maintain a web socket connection from the user app to the backend system to ensure all stock price changes are displayed to the user.

Whenever the background task is run, it will always be checking if there is an alert rule that has been met by the current stock price. If the alert rule is matched, a notification instance is created.

The notification created is then saved to the database. Immediately, the notification is sent to firebase where it is forwarded to a user's device.

On a user device, the notification is shown to the user who then chooses to view, dismiss or proceed with other actions.

The system will track current price by periodically checking the stock prices at https://fixer.io and https://currencylayer.com. This will be done by

1. Python asyncio
2. AWS Lambda Functions
3. Linux CronJob

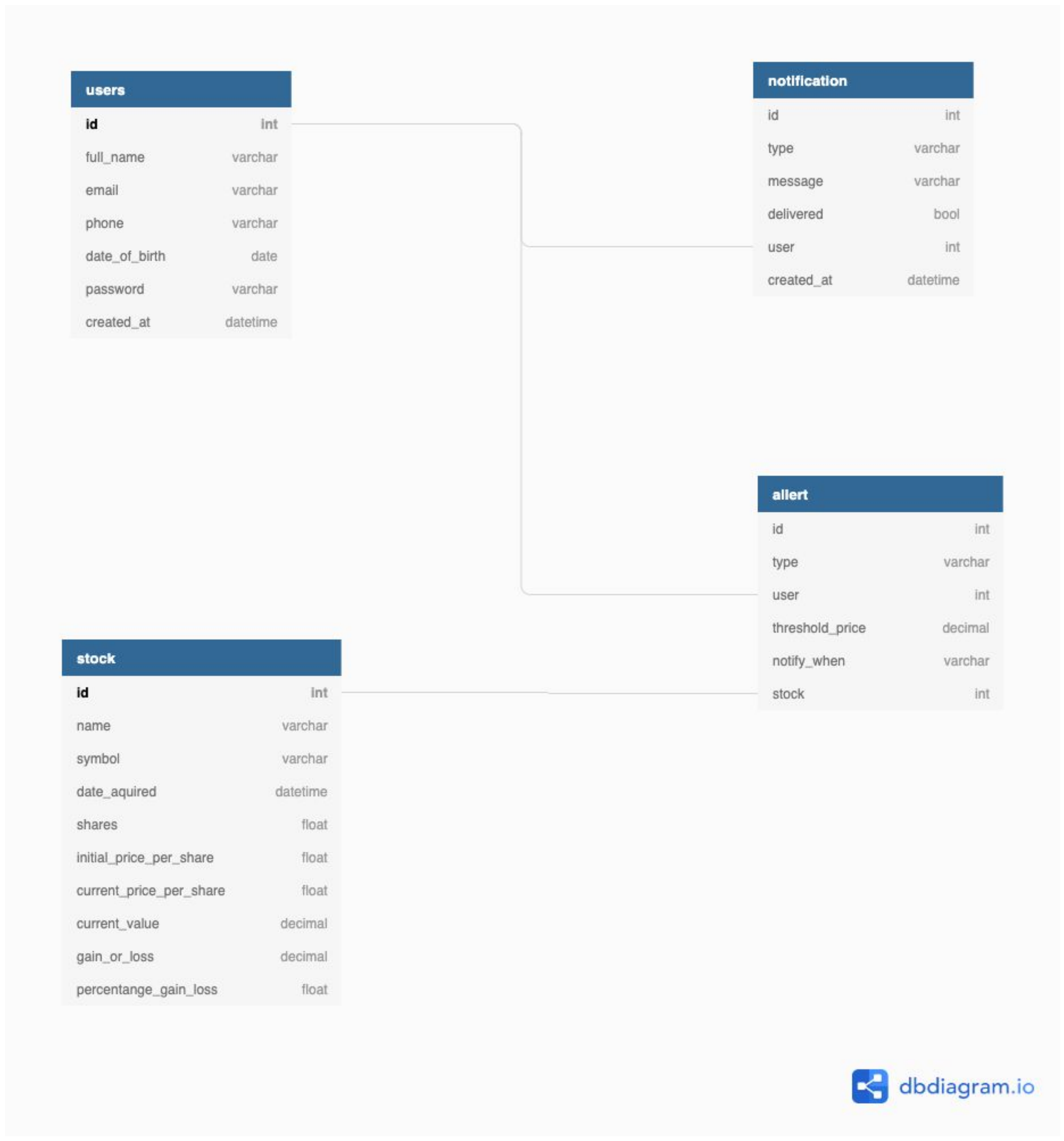
The system will know when to notify by checking the alert rules set by users. If the alert rule is valid, then a notification instance is created.

The system will send notification in two ways:

1. Google firebase
2. Web Socket connection

Google firebase offers a simple way of notifying a user even if the app is not running while web socket connections are used when the app is actively running.

The diagram below shows the simplest database schema.



When handling scallability of the system, the following are the means that will be applied in order to increase the user base without affecting the system response time:

1. Moving all or most of the tasks to be handled by APIs
2. Cache everything using Redis, Elasticsearch and RabbitMQ
3. Create a distributed system version whereby each microservice handles a specific task.
4. Moving most tasks to be asynchronous

The main datastores to be used are:

1. Postgress/Mysql database for storing persint data
2. Redis for caching