Cryptocurrency Monitoring Project Design

Overview

Our cryptocurrency monitoring website is a web-app dedicated to monitoring the prices of mainstream cryptocurrencies such as Bitcoin, Ethereum and Litecoin. It obtains the real-time prices of each cryptocurrency via the Coinbase API and displays the data in a line chart which shows the price for that currency for the last 30 days. Each of our supported currencies (currently 3) will have their own chart on the main page. Visitors can also add or remove charts on their page if they want (the changes will not persist). In addition, users can also subscribe for an email notification if the price of a particular currency has passed over or below a given threshold that they will provide.

User Stories

“As a visitor (not logged in), I can see the price charts for different cryptocurrencies in past 30 days on the main page”
- Alex visits the cryptocurrency monitoring website. The website displays a line chart showing the prices for each of the cryptocurrencies from the Coinbase API.

“As a visitor, I can customize which cryptocurrencies appear on the home page”
- Carlo is investing in Bitcoin and does not care too much about the price of other cryptocurrencies - so he clicks the X mark at the top right of each price chart he does not want to see and that price chart disappears from the page.

“As a new user, I can reigster and create an account”
- Bella wants to use more features of the cryptocurrency monitoring website, so she clicks the “Register” button at the top right corner and the website pops a form letting her enter her new account info.

“As a registered user, I can log in”
- Benny wants to sign into his account that he created so he clicks the Sign In button on the top right which allows him to enter his credentials and log in.

“As a normal user, I can subscribe for an email notification when the price of a certain cryptocurrency crosses some threshold”
- Deckard wants to get an alert if the price of Bitcoin goes above $6000. So he goes to the dropdown panel under the Bitcoin price chart, enters the threshold, and clicks the “add notification” button. He will receive an email when the price passes $6000 and that notification will never be triggered again after that.

Data Design

Overview - We will be using Postgres as our data store. The main things we need to track are currencies, users and subscribed notifications. All of our data will be either created by users or manually seeded by us.

Currency

- id (int)
- name (string)

The Currency resource keeps track of what currencies we currently track. It is mainly used as a foreign key reference in Notification (the resource that keeps track of subscribed notifications - see below). There is no need to store the prices since we will retrieve them from the API in realtime. This table's data will likely be manually seeded since it will just be 3 records for each of our currencies - Bitcoin, Ethereum, and Litecoin.

Notification

- user_id (foreign key to the User table)
- currency id (foreign key to the Currency table)
- threshold_price (int)
- above? (boolean)

The Notification resource allows us to let users subscribe for email notifications for when a currency of their choice passes above or below a certain threshold. It stores the user id for who we should email and the currency id that we are tracking. It also keeps track of the 'above?' flag which tells us if the notification should be triggered when the price goes above the threshold or below. We will have a background task that periodically polls the Coinbase API and checks through our Notification records to see if any notification threshold has been met - if it has, then we send the email to that user and then delete the record so that we don't re-notify later. For example, if the threshold is set to trigger once the price surpasses $3000 and we are tracking BTC, once BTC goes above 3000 - a email will be sent to the user saying "Notification triggered! The price of BTC has gone above $3000".

User

- email (string)
- password (string - which is a hash)

The User resource has the email and password for each of our registered users. The email will also allow us to send notifications to our users when they subscribe for them.

App Interface

Most interactions in our app would take place on the main page. By default, the main page displays a list of charts for the prices of all cryptocurrencies. Any visitor can add or delete charts on the main page. It also has a navigation bar to guide the user to register/login. A dropdown panel will also appear on each chart. It contains not only all the notifications that the current user has subscribed to for this cryptocurrency, but also a form to create a new notification. For each user, there is also a profile page where the user can edit or delete subscribed notifications and other user information.

Experiments

Hitting the CoinbaseAPI

We will be using the Coinbase API to get price data for each of the cryptocurrencies that we will be tracking. This experiment demonstrates getting the current price of each of the currencies we support - as well as getting the price data for the last 7 days for each one. This happens in an elixir script and it just prints the price data (after extracting it from the payload).

Running automatic background scheduled tasks

This experiment demonstrates our understanding of the Quantum elixir library. We will be using Quantum to schedule an automatic task that polls the Coinbase API every 5 (?) minutes so that we may send any email notifications if we need to. It is important that this happens in a background task independent of serving web requests since we want to send notifications even when users are not sending us requests at that particular moment.

The experiment just shows that we can schedule some task to be executed every 1 second. Currently, it just prints to stdout when the task runs.

Sending emails via Mailgun

Sending email notifications is a key part of our cryptocurrency monitoring website. We manage to send an email by using the Mailgun API. Recipients have to pass the verification from Mailgun before they can receive emails. As a result, when a user registers for an account, they will receive an email from Mailgun to verify their email. In addition, we are going to set up our domain to send the emails instead of using the default domain in Mailgun to send emails.

Project Status

We have sat down and thought through the flow of our entire application. We have drawn out what each of our aforementioned views looks like and what the intended flow for a normal user or visitor should be. Based on this, we thought of 3 experiments that highlight the toughest/most unfamiliar parts of the application. We feel fairly confident that we are on track to finish the project.

As of now, we just have a plain phoenix application that will be the starting point for our project. We will likely be able to re-use a lot of the code we wrote in our experiments since it is directly useful in our real application.

Thankfully, we have not run into any significant issues so far. The only hiccup is that we need users to verify their email with our email sending API (MailGun) in order for notifications to work. To get around this easily, we will just make users verify via MailGun when they register for an account.
Also, initially we were planning on storing the cryptocurrency price data ourselves but we realized that it was redundant since it would be simpler to just poll the Coinbase API in realtime.