Group 6 12th December 2018
Functional Programming Coursework

App using HTTP requests and database

Introduction

The aim for this project was to create a Haskell application which retrieves data from an online source and uses a database to store and manipulate the data. For our project, we decided to use the Yahoo finance website to extract historical data for Bitcoin.

Source - https://uk.finance.yahoo.com/quote/BTC-GBP/history?p=BTC-GBP&guccounter=1

Steps to Run Program

1) Navigate to destination folder
2) Run *stack build*
3) Run *stack exec bitcoin-exe*
4) Prompt "Which currency are you interested in?" – Respond with one of the options
5) Prompt "What type of information would you like to see?" – Choose Database, Graph or Stats
6) After Choosing Database
   - Data downloaded and inserted into database
7) After choosing Stats
   - Average Price for Bitcoin is shown to user
   - Currency details displayed
   - Prompt "Select a date to view the closing price for. (YYYY-MM-DD)" – Choose a date
8) After Choosing Graph
   - Prompt "What type of graph would you like to see?" – Choose High, Low, Open, Close, Adj or Volume

Extracting Data

Our program downloads a .csv file from an HTML webpage with the use of a HTTP Request. For this we imported 2 modules (Network.HTTP.Conduit and Network.HTTP.types.status) into our Crypto.hs module. To download, the http address and cookies for our source were parsed to our module and the cookie value was stored it as a variable in our bespoke Cookie data type.

Inserting Data

Insertion of data is done through a batchInsert function that initialises the database and inserts data into the two tables used in this program; currencies and bitcoin. Bitcoin table contains data downloaded from the website. The first column of this table is the iso code of the currency and is a foreign key to the iso code in the currency table. The currencies table contains details about each currency.

Selecting Data

It was important to consider usability when designing the app. The user would want to know the key information about Bitcoin, the open and close values. We defined a function called 'query', which would take the database (defined type 'Database') and a list of currencies ('String'). The query function requires an input to declare the currency and feeds the result into other more detailed query

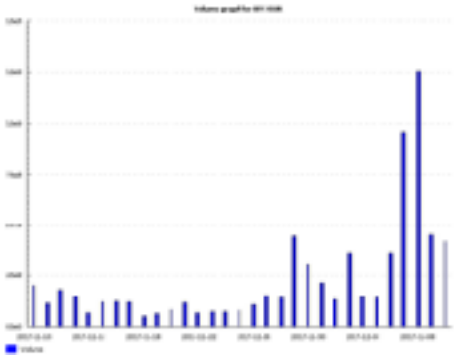
functions. Functions to query the average closing, average opening and the open & close values for a select date were included. Once the open and close values for a select date were extracted from the database, the difference was calculated, and a statement was outputted identifying the inflation or deflation of bitcoin for a given date. These functions take a database ('Database') and a date (String) and then perform an I/O action which will return a double (open/close value).

Deleting Data

With the delete query, consideration was given to the purpose for deletion. To delete entries from the database, maybe to remove old data, it would be ideal for the user to be able to specify the date they wanted deleted as oppose to deleting for a specified open/close value. Deleting is achieved through 2 delete queries. The delete queries allow the user to delete the database or remove specific rows. The query takes the database ('Database') and a date (String) and performs some I/O to delete data from the database.

Extra Features

- **Graph Function**
  ⇒ This makes use of Haskel's 2D charting library chart-cairo to render graphs that show trends in the data. For example;



- **Average Price**
  ⇒ In a separate module, we created a function to calculate the average price of Bitcoin for the whole dataset. Database and String values are read into the function and an IO double is returned after calculation. A SELECT query is used to retrieve to value of the close field for every row in the Bitcoin table. All the values are added to an array which is used to calculate the average.

Haddock Documentation

Haddock is an useful tool for documentation and is incredibly valuable for either further maintenance on the code or for an external reader. Locate the file in the terminal and run 'stack haddock' to implement. Haddock uses a pipe within the comments of Haskell (- -|some comment). This helps indicate that the comment is a special type of documentation. Haddock will only document the exported items in the code. So, it was necessary to check that the Haddock comments were for exported functions. For some functions, it was useful to use a different indentation (- -^some comment) where further documentation is required.