CRYPTOPI

Postmortem Report

University of Western Ontario

Written by: Brandon Mathew

Project Overview

CryptoPi is a software built for the Raspberry Pi. Written in C++, this web application displays and tracks cryptocurrency market data. It captures market data from the world's largest exchanges, similar to a stock price ticker. The Raspberry Pi functions as a server to capture real-time market data through existing C++ API's. It pulls data from multiple cryptocurrency exchanges and presents them through a web application that allows users to personalize their experience (via the CryptoPi Dashboard).

Various classes were created for each exchange to allow processing of currency pairs. Given that our application was real-time based and the volatility of the cryptocurrency market, performance was our utmost concern. The task of achieving a peak performing application was accomplished through dynamic memory allocation. We minimized memory usage and ensured there were no memory leaks to have better overall performance.

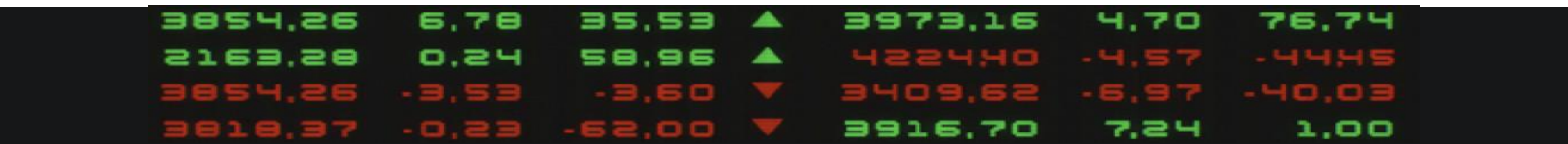
The data is organized and presented to users via a web application. Users have the ability to personalize their experience and organize their view presented through the CryptoPi Dashboard. More specifically, users can select which exchange to be displays and the currency pair to be displayed. In addition, users have the ability to manipulate the layout of the dashboard and display it according to their preferences.

Key Accomplishments

From a technical perspective, our group was successful in implementing the libraries we needed into the project. Foundationally, this was crucial since this was the source of that data. Building classes that used dynamically-created objects was another key accomplished because it impacts performance. Function calls to exchange API's were very efficient, especially towards the front end, and data was displayed instantly.

From a design perspective, multiple libraries were executed using a singular approach which facilitated towards a more congruent application in the end. Forcing each library to use the same format (i.e. BTC-USD) made compiling multiple strategies together easier. If we had chosen to implement each library based on its specifications, it would have resulted in more complications near the end of the project.

From a logistical perspective, organizing our team using various software methodologies assisted in keeping track of all our tasks along the way. We divided our team into two and had one work on the backend portion and one on the frontend. We found using certain resources proved to be very useful. In particular, we used the Agile-inspired project management tool JIRA to organize and track the progress of tasks assigned to each team member. Bitbucket allowed each member to work on their tasks remotely and push



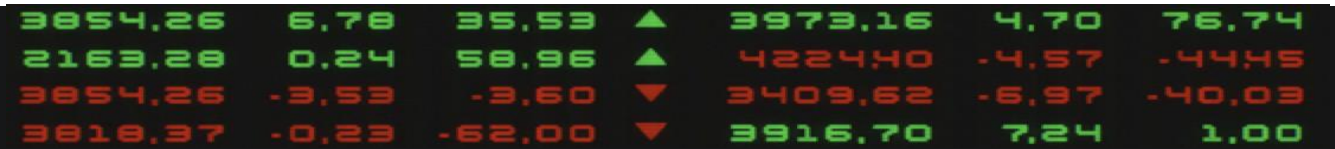
partitions of the code as needed. Furthermore, the use of instant messaging tool Slack allowed us to create separate channels to keep track of various parts of the project. For example, we had a channel that we kept just for files so any member could access certain files as necessary.

Key Problem Areas

Like many projects, there is a learning curve when working with new technologies. For example, there was an issue when we tried to combine the backend and frontend parts of the projects. We divided our group into two teams, so each team was implementing their section as necessary. Because each team was working separately there were some difference on how each of the tasks were implemented. To counteract these issues, we had to debug and alter the first iteration of backend classes, so it could meet the specifications required by the front end. Mostly, this was an issue of communication which could have been avoided by clearly specifying procedures earlier on. These technical issues affect the workflow and timeline that is originally set by the group members. As our workflow got interrupted and productivity began to be affected, it impacted our ability to get to more of the tasks that were optional and in turn the quality of our product.

From a design point of view, what we struggled with most was the lack of knowledge of design patterns and strategies that could have been useful. As a result, we had to tackle and fix problems as they came along not knowing whether it could cause problems in the future. For example, it was never considered that our software might require the need to structure multiple C++ libraries each making different API calls. In hindsight, an adapter pattern would have helped us make important decisions that would have been better in the long run.

Logistically, we had a few issues that arose during development. One of the things that made things a little difficult for us were cross functional dependencies. There were certain aspects of the project that could not be implemented until another part was completed. For example, the frontend team could not start working until the backend team had completed the part that was required for the frontend. This impacted efficiency as we didn't always have all team members working all the time. All the members of our group had different schedules. Different from a workplace environment where everyone works around the same hours, we had to deal with different schedules of team members and had to plan accordingly. Traditional methods of working as a team was not working for us. We noticed that setting a requirement to meet up was a hindrance. Too many team members had different schedules and waiting to meet to get organized was just a setback. Instead, we did most of our project remotely while communicating regularly on slack channels. This allowed us to work independently on our own time and complete tasks efficiently.



Lessons Learned

After finishing the project and looking back with the experience and knowledge of completing the project there are a few key lessons and takeaways from the project that we can decipher.

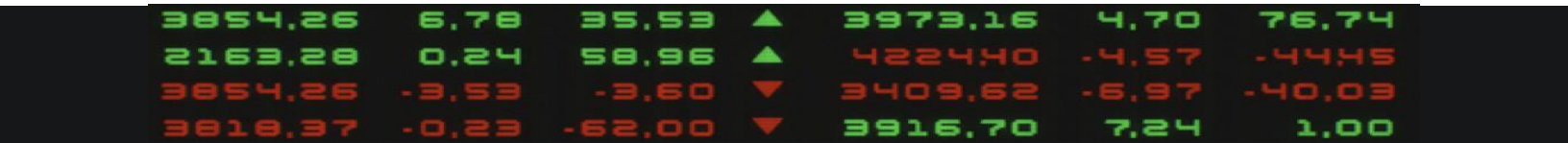
As our team had never worked on a project of this magnitude, there are many things that we simply did not know about. Shortcuts, ways to implement code efficiently or the best way to go about completing tasks are all part of knowledge that one gains only from experience. Following the completion of this project, there are certain things that we would change in retrospect. A better design patterns could help optimize and modularize the structure of our software. It would also have been wise to set up and test different libraries earlier on in the cycle. This allows us to select libraries that are compatible with the project that is being built. Technical issues are inevitable in the development cycle and sooner or later there is going a complication that will arise. While some technical errors cannot be foreseen, it is important to allocate extra time should an error arise.

Using the Agile methodology and resources were very helpful during the developmental cycle of the project. In the future, using Slack and JIRA software is imperative for most software projects. We also found that C++ is a very strong and efficient language. It works well with object-oriented programming and data structures and we would definitely consider it for future projects.

There were also a few things that we noticed worked fairly decently but could use some improvement. With regards to the physical features of the project, we would aim to design a better frontend for the project, so it is more user friendly. Perhaps even implement a phone application would be beneficial. Having the frontend team work with this as a goal in mind from the beginning would allow them to achieve this. Dividing our team into two was very beneficial because it categorized the tasks that needed to be performed. This helped cut everything down the middle, so each group knew what they needed to accomplish. However, this was not the most efficient because the frontend team would have to wait for backend to complete a task before they could work on their task or backend had to wait for feedback from frontend.

The project we worked on was fairly small and is very scalable. Those of us that are interested in the cryptocurrency market would be interested in working on a project similar to this or continuing to work on the same project. There are still many features that could be added and could even serve as a great subscription-based business model if one decides to go that route.

Key Lessons for the Workplace Environment



Some technical skills that we have gained from this project are C++, the use of software methodologies using agile development such as JIRA and experience with implementing external APIs and multiple libraries. These technical skills are very significant because they prepare us for the workplace. Having prior experience with such technologies puts us in a good place for future projects because now we know what to expect in the future.

Dividing our group into two sections with frontend and backend is also similar to what we might see in the workplace. Companies usually work on larger projects that have larger, cross-functional team. Interacting and working in a team that is divided into various groups is standard and having a similar structure for our project also gave us exposure to what a realistic project could look like.

Alongside these and perhaps just as important are the soft skills we gained from working on this project. In particular, working as part of a team, good communication skills, time management and problem-solving are all skills that are transferrable to a workplace. Gaining these skills are extremely important because these are skills that one needs to have and apply daily to succeed in the workplace.

Final Remarks

In conclusion, building CryptoPi was a beneficial experience for us all. It allowed us to work on a project that would be challenging yet rewarding. Our group was able to complete the tasks through various resources provided to us and accomplish some key aspects. Along the way, we experienced a few obstacles which we had to solve to go through. In the end, the project was a great tool for us to learn about software methodologies and design patterns and allowed us to gain experience that will be advantageous for us in the future.