Software Engineering at QUOINE

Nam Nguyen
This thesis is my diary note on what I was working on at QUOINE as a software engineer. QUOINE’s main product is the modern cryptocurrencies trading platform which is powered by blockchain technology. During the time of writing this thesis, my main focus was on the web app development especially on the client side. I participated in code review process, refactoring existing bad code and widespread the test-driven development process. The key things I would include while working here list as the followings:

- The UX/UI should be at the core of any product
- Having 2FA enabled for any serious application
- URL is part of the UI and it should be treated with care
- Software development process is utmost importance
- Coding convention are required upfront in order to make code reviewing process to work in practice
- Architecture is the foundation of any app
- Speed and CPU power are matter to web performance
- Accessibility should always be considered while developing any application
- Having different set of environments for developers to check up with QA is absolutely nice thing to have for large team

**Keywords**
Software engineer, web performance optimization, clean code architecture, enterprise communication, test driven development, data structure and algorithm analysis
Table of contents

1 Introduction ................................................................................................................... 1

2 The start situation.......................................................................................................... 2
   2.1 Analysis of your current job situation................................................................. 2
   2.2 Interest groups in the workplace ................................................................. 3
   2.3 Interaction skills in the workplace ............................................................... 4

3 Diary entries .................................................................................................................. 6
   3.1 Week 1 ................................................................................................................. 6
      3.1.1 Monday ...................................................................................................... 6
      3.1.2 Tuesday ..................................................................................................... 6
      3.1.3 Wednesday .............................................................................................. 6
      3.1.4 Thursday .................................................................................................. 6
      3.1.5 Friday ........................................................................................................ 7
      3.1.6 Week 1 Analysis ..................................................................................... 7
   3.2 Week 2 ................................................................................................................. 9
      3.2.1 Monday ...................................................................................................... 9
      3.2.2 Tuesday ..................................................................................................... 9
      3.2.3 Wednesday .............................................................................................. 9
      3.2.4 Thursday .................................................................................................. 9
      3.2.5 Friday ....................................................................................................... 10
      3.2.6 Week 2 analysis ..................................................................................... 10
   3.3 Week 3 ............................................................................................................... 11
      3.3.1 Monday .................................................................................................... 11
      3.3.2 Tuesday ................................................................................................... 11
      3.3.3 Wednesday .............................................................................................. 11
      3.3.4 Thursday .................................................................................................. 12
      3.3.5 Friday ....................................................................................................... 12
      3.3.6 Week 3 analysis ..................................................................................... 12
   3.4 Week 4 ............................................................................................................... 13
      3.4.1 Monday .................................................................................................... 13
      3.4.2 Tuesday ................................................................................................... 13
      3.4.3 Wednesday .............................................................................................. 13
      3.4.4 Thursday .................................................................................................. 14
      3.4.5 Friday ....................................................................................................... 14
      3.4.6 Week 4 analysis ..................................................................................... 14
   3.5 Week 5 ............................................................................................................... 16
      3.5.1 Monday .................................................................................................... 16
      3.5.2 Tuesday ................................................................................................... 16
3.5.3 Wednesday ................................................................. 16
3.5.4 Thursday ............................................................... 17
3.5.5 Friday ......................................................................... 17
3.5.6 Week 5 analysis ......................................................... 17

3.6 Week 6 ........................................................................... 19
3.6.1 Monday ...................................................................... 19
3.6.2 Tuesday .................................................................... 19
3.6.3 Wednesday ............................................................. 19
3.6.4 Thursday ................................................................. 20
3.6.5 Friday ........................................................................ 20
3.6.6 Week 6 analysis ....................................................... 21

3.7 Week 7 ......................................................................... 22
3.7.1 Monday ...................................................................... 22
3.7.2 Tuesday .................................................................... 22
3.7.3 Wednesday ............................................................. 23
3.7.4 Thursday ................................................................. 23
3.7.5 Friday ........................................................................ 24
3.7.6 Week 7 analysis ....................................................... 24

3.8 Week 8 ......................................................................... 26
3.8.1 Monday ...................................................................... 26
3.8.2 Tuesday .................................................................... 26
3.8.3 Wednesday ............................................................. 26
3.8.4 Thursday ................................................................. 26
3.8.5 Friday ........................................................................ 27
3.8.6 Week 8 analysis ....................................................... 27

3.9 Week 9 ......................................................................... 28
3.9.1 Monday ...................................................................... 28
3.9.2 Tuesday .................................................................... 28
3.9.3 Wednesday ............................................................. 29
3.9.4 Thursday ................................................................. 29
3.9.5 Friday ........................................................................ 29
3.9.6 Week 9 analysis ....................................................... 30

3.10 Week 10 ................................................................. 31
3.10.1 Monday ................................................................. 31
3.10.2 Tuesday .................................................................... 31
3.10.3 Wednesday ............................................................. 31
3.10.4 Thursday ................................................................. 32
3.10.5 Friday ........................................................................ 32
3.10.6 Week 10 analysis .................................................. 33
1 Introduction

The thesis is notetaking my work during the time I was working at QUOINE as software engineer. The starting point is on 13th September 2018 and finished by 7th December.

With regards to QUOINE, they are one of the leading fintech product companies that provides trading, exchange, and next generation financial services powered by blockchain technology. QUOINE’s headquarter is located in Japan, the other two offices are placed in Singapore and Vietnam. The company has approximately four-hundred employees which includes marketing, cyber security, accounting, product design, customer service, back-end and front-end teams.

The working environment is enterprise-class level, everyone is provided MacBook Pro 2017 version with 24-inch monitor. The cafeteria is located right in the office and isolated from working area. Teams are divided into different tables and sitting around. The communication is widely open and very modernized. We have more than 10 meetings rooms are fully equipped, nicely functional along with booking system provided.
2 The start situation

2.1 Analysis of your current job situation

My current responsibility involves the following assignments:

- Daily meet-up: We have a meeting around 15 minutes among all team member to discuss what has been done, what is blocking and what going to be done on the current day
- Resolve task: We use JIRA software management application for management our development process. The task is created by reporter, then assign to developer and that person needs to follow the unifying workflow that documented in JIRA’s confluence pages
- Cooperate with QA team: We are grouped by team instead of feature like traditional AGILE development. We are all communicating via JIRA tickets and responsible for the task throughout its lifecycle.
- Participate in code review: The code review process is conducted in GitHub as we are using its enterprise version. Every single pull request is run through CircleCI process to make sure nothing break in term of linting and test cases. The developers review code written by others and leave comments or request to change in case anything not correct. Once all developers related to the PR reach full agreement, the PR is merged to the corresponding branch and run though CD process to its matching environment.
- Train new member about software development process, technical capability and toolkit: At QUOINE, we are using fairly number of packages that comes from both open-source and commercial ones. I am responsible to onboard new members about our daily work flow, knowledge requirement, contacting points and toolkit that we are using.
- Detect issues in production environment if any and report back to chief head of product: Every employee at QUOINE is responsible for our production running in production. Whenever we detect a problem related to our application’s logics, we should report it as soon as we can.
- Identify application performance issue: Web performance optimization is a journey and that would involve all software engineers as we all share hands to make up the application. It consists of API design, application loading time, rendering power, responsive web design and caching techniques appliance.
2.2 Interest groups in the workplace

The followings list all the interest groups that are related to my work:

- **End-users**: they want to experience the best possible trading app that help them make more money
- **Product designers**: they want to see whether their design mock-up coded up in proper way or not
- **Inventors**: they want to see how our app doing in production. All the numbers and figures tell a lot what’s going on
- **Software engineers**: As we are working throughout software development process, we need to collaborate with each other to improve our codebase and ship new feature to production

![Figure 1: QUOINE’s Liquid application overview](image-url)
2.3 Interaction skills in the workplace

At QUOINE, we are grouped and sit along with team. We rent an office in the 22nd floor Saigon centre. Instead of having several regions for different team, we have an open workplace where everybody can see each other and know what they are doing. The current size of company is around 150 people. There are in total as listed below:

- Front-End Engineer
- Customer Service
- Data Engineer
- Back-End Engineer
- Human Resource
- DevOps Engineer
- Marketing
- Accounting
- Quality Assurance Engineer

Figure 2: QUOINE’s development hub in Vietnam
The current situation provides me one big challenge in software development process. As we move fast, having more and more features to develop. We are not doing well in term of communication and separation of specific job concern. We often have a intention to cross functional and that’s not a good thing. The optimal solution is to have team group by domain. The team should include front-end, back-end, project manager and tester to get things transition from requirement to production.
3 Diary entries

3.1 Week 1

3.1.1 Monday

Today we started setting up my machine, install all the dependencies needed and clone the repository to develop in local environment. The team was very supportive, every newcomer is provided a MacBook PRO 2017. We were walking to places in our company and introduced to all teams in the company. There is not much things to do in the first day with respect to development except from open up company mail account, request access permission to the codebase and set up machine.

3.1.2 Tuesday

Today we kick-offed the onboarding process for all new members. Our team leader walked us through the codebase, giving all necessary links to documentation, invitation to Slack communication channels and point out the problems we were currently facing. The codebase is quite messy, there are a ton of problems that need to be addressed and provide quick fix as soon as possible. We started collecting them and put up a whole presentation so that the whole team can understand where they are and how they should move forward. The document is somewhat okay and there are a lot of things could be better explained. The terminology, development workflow, setup instruction and content translation setup.

3.1.3 Wednesday

Today we started cleaning up the codebase as we had fairly amount of dead code has been written and not in use anymore. We did profile our web application using Google's Chrome DevTool audit tab and Webpagetest to provide deep insight about how our app’s current performance. The result showed that we did not doing well in production since the initial page load takes more than 15 seconds to finish and the repeated process take approximately 10 seconds. (Google Chrome DevTool 2018 and Webpage Test 2018)

3.1.4 Thursday

Today we were working on revamping our codebase structure, specifically the linting and build processes. We completely enforce battle-tested Airbnb’s standard linting rules for
our codebase by leveraging Eslint latest feature. Additionally, we put a lot of effort on customizing our current build system to match the latest development on master branch of create-react-app which is the underlaying piece of technology that power our application development process (Airbnb Coding Standard 2018)

### 3.1.5 Friday

Today we gathered up all the problems we found along the way while reading the codebase and put up a whole presentation. All team member went to the meeting room where we were discussing about how we should address all problems one by one by start prioritizing them.

### 3.1.6 Week 1 Analysis

The first week went pretty smoothly for us as we were very welcome by the team. We got to know the codebase, setup machine, understand how enterprise software development process work, coding convention, the workflow and the core product of the company which is the LIQUID trading platform.

There were many things that could be improved as we keep expanding our platform to ship more and more features to our clients. The git branching models we currently applied could not fit us very well. We typically used git-flow branching model to power our daily workflow with some incorrectness. The biggest mistake was to have feature deployed straight up from release branches instead of the master one that lead to a lot of confusion. The code review process is not very good and it’s very hard to verify whether this feature fulfil the requirement completely by just reading through the code without tests written. Last but never be the least, the codebase was heavily influenced by React’s famous coding pattern so-called Container-Presentation. Unfortunately, the team was doing it all wrong and now technical debt was piling up lead the very fragile codebase.

The solution we would like to apply is to first educate the team toward Behaviour Driven Development process. This is going to help a lot in code review process and better off code quality by remove the factor of manual testing that could not give us confidence shipping our code to production. The git branching model should be fixed as soon as possible to remove confusion for new member entered the codebase. The codebase should be rewritten for two critical reasons. The first one is to have a clean code architect applied to the codebase, and second is to leverage new industry de-factor API solution so-called
GraphQL. The current application is using REST API, the biggest problems we are facing is when dealing with a lot of deep nested data fetching that needed to make several round trips to the server instead of making one like GraphQL. Furthermore, as we provide API for web, android and iOS platforms – we often found ourselves over-fetching data that is absolute not in need. GraphQL is the big step for our trading platform app with regard to technical advancement in the near future.
3.2 Week 2

3.2.1 Monday

Today we kick-offed performance optimization that rooted in API level. The current situation was that there were too many duplicated API calls that leads to more and more pressure on server. The most prominent one is unauthorized API calls made before user logged-in. This results in error response by server and being ignored. The quick fix solution we came up with was to simulate an unauthorized API call error right at the client which helps to mitigate the heavy load for our back-end server. Due to unauthorized API calls has been made all over places, there was no better way. Instead of touching every single file that possibly lead to absolutely server error return, we decided to quick fix for now and pay more attention on the next version of our app.

3.2.2 Tuesday

Today we had a meeting about error handling in case server return message “500 – Internal server error”. We had a long discussion about this as back-end guy insisted that all 5xx error code return from the server should be treated identically by displaying maintenance page. As the result turned out, we could not reach final agreement and there was a request for this feature sent to our CTO for final decision.

3.2.3 Wednesday

Today we received a decision from our CTO that we should treat all server the same regardless it is 500 or 503. Initially, the front-end app would start making API called once all static resources loaded (HTML, CSS and JavaScript). In the case server return 500 error code, our front-end app would resend API call few more times until it reached maximum amount – 4 times in this case, the front-end would display maintenance page. In the end of the day, we deployed new release included this feature. (HTTP Status Codes 2018)

3.2.4 Thursday

Today we got a quick call from our business team in Singapore about maintenance page displayed without any plan. Due to the fact that server internal error, this leads to our front-end app display maintenance page which was totally misleading. The maintainance page should only displayed when there was a decision from business team in Singapore. The back-end team was working hard to fix these issues as they decided to scale more
resources in Amazon Web Service. Nonetheless, the issue was still remained due to the fact that anything bad happened with server would cause front-end app to display maintenance page.

### 3.2.5 Friday

Today we had a traditional team sharing event in which all of us could bring anything on the table for discussion. We had a fairly big argument against the way we were architecting React components. We decided to apply some advanced React pattern namely render-prop and higher-order-component. And we received good news that our company decided to sign a contract with JavaScript consultancy company so-called Formidable. They are well-versed and expert in full-stack JavaScript application and has a vast experience working on different kind of projects, their well-known customers include Microsoft, Start-Buck, et cetera.

### 3.2.6 Week 2 analysis

The second showed us a lot of issues rooted in our back-end API system. Prior our working periods, the front-end team had done a tremendous amount of work that should never be part of front-end day to day job. These includes application logic, deployment system, continuous integration, continuous delivery and access token generation. The more calculation places on front-end app, the worse user experience is going to be. We took full commitment to stop letting that happens by prevent all low-quality code involved application business logic that wearing out our app’s speed. Additionally, web performance optimization was taken really seriously from this point on. We decided to be in partnership with Formidable JavaScript consultancy company to build the next major version of our application. The next version of QUOINE’s front-end application will be built from the group up with the support from JavaScript’s expert. It seemed to be very exciting news for all QUOINE’s stakeholders.
3.3 Week 3

3.3.1 Monday

Today we started sprint planning as usual, the whole team were looking at the backlog and decide what to includes based on relevance and priority. We decided to tackle on high priority bugs and implement some new features request from the business product team. There was a weird bug happened on the new iPhone X just came out of the smartphone market. The feature could not work in harmony with React.js as anytime the user clicks on dismiss button which leads to the full-page reload while routing happened on the same single-page application and there was no call that signal a full page reload. We decided to debug remotely by connecting Mac to couple of iPhone devices. As it turned out, the weird bug just happened on new iPhone X only as it is now fully stable and there were some bugs remain in the current OS.

3.3.2 Tuesday

Today we got tasked to work on responsiveness of our web app on mobile devices from displaying perspectives. The introduction to the app did not function very well on mobile devices as it could on desktop devices due to small screen sizes. We teamed up to do pair programming on this task as it did not involve much logic but rather visual problems. We had a quite long discuss about should we implement this introduction feature on mobile devices as there were no way we could fit all content that meant for desktop devices to fit on such small screen sizes. Eventually, we reached out to the business product team and asked them to rewrite the content specifically for mobile devices in order to make it fit on mobile devices.

3.3.3 Wednesday

Today we focused on listing out unnecessary calculations that happened on the front-end. As the fact that JavaScript language is single-threaded which means it could not do more than one thing at a time. The main responsibility of JavaScript should be on handling user events such as input, mouse scrolling, keyboard actions, et cetera. Currently, the front-end was tasked doing heavy calculation that choking the main thread leads to unresponsiveness of the app on some screens. There were several obvious things that front-end should never take of such as turning snake-case JSON data’s property to camelCase in
order to match JavaScript coding conventions, calculating rates of market exchange instead of letting back-end does the job, store user’s setting on browser’s storages instead of server-side ones that leads to many confusing bugs

3.3.4 Thursday

Today we continued the task that was unfinished yesterday. As the codebase was so big and quite messy, the unneeded logics are scattered all over places that made it very hard to do a complete clean-up. Additionally, there were no tests being written which made the refactor becomes very cumbersome. Eventually, we came up with the not-yet-complete list of unnecessary front-end logic issues that should be addressed on the next day.

3.3.5 Friday

Today we started to be working on the front-end logic’s issues. The logics that were so tight to the application’s logic as a whole, we had to work very closely with the back-end team so as to migrate logics from the front-end side to the back-end one. The logics should be implemented and deployed from the back-end sides in advance so that front-end side could remove those logics away from the codebase.

3.3.6 Week 3 analysis

The third week gave us a view over many opportunities that we can grab in order to optimize our app’s performance, reduce bundle sizes and provide user better and silky-smooth user experience.

Our vision was to have the front-end clients regardless of platform (web, iOS, android, television, watchOS, et cetera) to be fully data driven applications. In another term, all the data and business logics should all come from the back-end side to provide these following benefits: remove duplication logics located at the client side, horizontal scalability as we only need to scale our server machines in order to match up high requests from the end-user, et cetera. For now, what we really need to focus on is to move logics from the front-end web app to the back-end side in order to avoid duplicated codes. Furthermore, we would push our back-end team to provide a well-written and up-to-date documentation. Because documentation is first-class citizen in any kind of project, we tried to address these as our vision was to support our clients in various platform ranging from web, mobile and even IoT devices.
3.4 Week 4

3.4.1 Monday

Today we had a discussion about how we would prepare an onboarding process for Formidable consulting team that was about to arrive at our office early in the first week of December 2018. Additionally, we got to fix some messy things from JIRA’s SCRUM boards as many tasks were accidentally moved to the backlog. Every single front-end team member had to go back to the backlog and hunt down their previous tasks in order to move it back to the active sprint. We were working very hard on preparing the onboard documentation and put it up to JIRA’s confluence where all related working files were located.

3.4.2 Tuesday

Today we had a tasked of fixing undefined display on the web UI. As the data model was built by data returned from the server. The problem was that when some React children components tried to access missing properties in the built model passed down by parent components, this result in undefined value and the previous front-end engineer decided to let it be as displaying on the UI. We decided to fix that by switching displaying undefined by rendering nothing on that spot by putting one conditional rendering right at React’s render method.

3.4.3 Wednesday

Today we received a task of adding Google translation service to our web app. As requirement originated from the business product team few weeks ago about adding fifteen more languages to our current web app. Currently, our web app’s technical stack of technology could not let us do that since we bundle up all translation content into various chunk of JavaScript files and dynamically rendering per page right at the client. In the future, as our upcoming major release version is going to be completely rewritten by applying server-side rendering technology and provide progressive code splitting up on initial load together with PRPL pattern popularized by Google web software engineer. We had to config our web app together with different environment settings so that the benefit of Google Analytic remain intact. The Google translation service was capable of translating on-the-spot of over one hundred languages and the translated content were acceptable. All in all, this is a quick fix while we were trying to resolve the technical debt with heavy bundling issues at the current front-end development stack. (The PRPL Pattern 2018)
3.4.4 Thursday

Today we had many meetings and there were no tasks involved. The first one was held by internal Vietnamese front-end to discuss about inefficient leadership currently going on. The current team leader was temporary, and he was not very capable at what he was doing due to his background and specialization. He was recruited to do front-end development in the first place but for any reason leading front-end team. As the fact was that the team could not release many features as they should due to bottleneck in communication between front-end team and back-end team handled by this leader. The second meeting was held for a whole company, there was a special visit of all board member included QUOINE’s chairman, CEO, CFO, COO, CIO and CPO. There were several speak given by the board’s members to show the roadmap of QUOINE’s product in the near future and how we were doing in the market compared to other big competitor like Binance, Coinbase, Bitfinex, OKCOIN and Kraken. The figures showed that we were very competitive in the crypto trading industry due to the fact that we were always ahead of the game. QUOINE is the first company got regulated by Japanese government and soon will be done by Singaporean government also. The future of QUOINE looked really prosperous and fascinating by all means.

3.4.5 Friday

Today we are finalizing the task of implementing Google translation service to our current web app. The initial idea came from one of our biggest competitors namely Binance. We borrowed the idea of having small Google translation service icon on the top right corner of the app. The user story was “Given user is at any screen in our app, when user hovers on google translation service icon at navbar, Then the modal popup will appear with dropdown button for users to select specific language, and when users select their preferred language, the whole application will be translated into that language”. We as front-end developer had to also take care of other cases like network error, mouse-click outside of modal, loading states of the application, et cetera. Additionally, we had to conduct integration testing in order to resemble the way our user interact with our application and strengthen our confidence before turning our build code to QA and ready to ship into production environment if passed.

3.4.6 Week 4 analysis

The fourth week showed that we did run into a lot of problems from both technical perspectives and management’s one. On the technical side, we noticed how important it is for
us to restructure our application technical stack in order to make our app fully internationalization-supported. As of now, the web application is capable of handling languages thanks to Google translation service. Unfortunately, it is not going to last in the long run due to that fact that Google translation service is a third-party tool and not their translated content is totally out of our control. We have run into some translation issues in the past with Google translation service due to the fact that it translates the content very much word by word instead of contextual translation. Our product team were working so hard on getting languages up and running for all intentionally supported eighteen languages. Besides, the tech team started working hard on revamp our codebase structure to make the process of code-splitting become more progressively. Our final goal of code-splitting is to have our app fully component-based splitting instead of just route-based splitting.

On the management side, we figured out that we were not running SCRUM properly. The daily meeting supposed to be contributed equally by all team member. Instead, we had one project manager monitor everything and he could not control the backlogs, meeting schedules and retrospective review efficiently. We decided to propose our solution by asking for a qualified project manager instead of letting any experienced programmer to take this role so that the team would not be slowed down anymore due to communication bottleneck between project manager and other teams. Additionally, as the team size was quite large – twelve members, this lead to many problems in term of human scalability perspective. We had a plan to split the team into functional modules per team. Each team was going to have QA engineer, Back-End engineer, Front-End engineer, DevOps engineer and project manager. This was going to have a big impact on how communication happened at software development process. We were striving for the better process, faster time to market, efficient time management and productivity in we way we do software management.
3.5 Week 5

3.5.1 Monday

Today we had a new front-end team lead to take over the temporary one. We were helping him to go through onboarding process on the spot as mentors. The process took a week to get him up and running with the current software development process of the company. On this first day, we were mainly helping him get equipped with the coding environments, local project setup, company’s email configuration setup, access to various projects’ repositories under QUOINE’s GitHub organization. At the end of the day, we successfully got him from zero to get the projects started from the command line, open web application and connect to the back-end team’s API nicely and smoothly without any trouble shooting steps needed.

3.5.2 Tuesday

Today we continued our mentor programming with new front-end team member. We introduced to start reading through our cumulated documentation so-called JIRA’s confluence. All team members have put more and more effort on improve our long living projects document in order for better off our onboarding process. The document included list of environment setup and configuration, the process of release, getting-started procedure, business logic requirement, different set of security protocols, SEO and web performance optimization, difference between marketing page and web application of LIQUID, all retrospectives we have had overtime and API documentation. Additionally, we started to discuss Google translation service’s capability as it does not work very on live changing data and click events of our web app. The final conclusion was putting the task back to JIRA’s backlog and waiting for further confirmation from business product team.

3.5.3 Wednesday

Today we received two new tasks about forms validation. The first one was about deposit bank account status which involves user’s input such as bank name, branch, SWIFT number, account number and holder name. The problems we ran into was about bank’s statement attachment file. The business requirement only allowed the file to be less than twenty megabytes, width and height should be less than six hundred pixels respectively. Additionally, the file format allowance was limited to “pdf, jpg, jpeg, png, bmp, tif, tiff”. Our solution was leveraging the browser’s HTML5 capability to limit users' input by setting accept attribute to the file format allowance list. The second form was almost identical to the first one except we had to do one more check for duplicated files names. As the form was
about profile verification to mark user as authorised users instead of just random one trying to be poking around with our trading platform application. We abstract the logic to one utility file and share it across two forms. In the end of the day, we finalized the ticket and ship it over to QA team for testing in staging environment. (Browser Image Format Support 2018)

3.5.4 Thursday

Today we got back to work on Google translation service. This task was canceled few weeks ago due to misunderstanding among business product team and software development one. It is now considered acceptable that Google would not be able to translate all dynamic content on user navigating around our web app. We noticed some UI's break in the app when the positions of difference layers overlapped that lead to display weirdly. The root cause was because we were not setting z-index correctly that lead to the composite process did not process the way we expected. We did a quick fix on that by re-layering our components to make UI display properly. After that, we deploy our application to staging environment for testing.

3.5.5 Friday

Today we received a task with regards to automation testing. The QA engineer who were responsible for testing our current balance's screen reported that the current DOM displayed in the web app showing many duplicated id attributes on element and some empty ones. This led to very unstable test runs in automation test. We decided to look through the source code to figure out what was the root cause of the bug. As it turned out, since the back-end team did not return the unique id for every single record they sent over to the client. Accordingly, front-end team had to do monkey-patching our application as React require each element to have a unique id in the loop call, by generating correspondingy unique id for each element. Unfortunately, there were several cases missing in action. We decided to go through all sections of the app that causing that problems by adding unique ids for them. After then, we did all necessary test cases and deliver it over to our CI/CD deployment pipeline and ready for testing.

3.5.6 Week 5 analysis

The fifth week went through smoothly and we resolved a great amount of problems. The current workflow was replaced by well-known git branching model so-called git-flow. All front-end and QA team members were really excited about this new workflow. This is going to resolve the most prominent problem resided in our software development process for months. The business requirement came out from business product team, everyone
involved started to understand and kick off the programming development. Unfortunately, the bottleneck had always been laying at the communication where we were trying to figure which environments to deploy for testing. As the application not only include first party codes from back-end and front-end teams but also third parties one such as Google Analytic and bug tracking system Sentry.

Besides, we made one big mistake this week as the release came out with fairly amount of in-progress and in-test tasks. The communication between release manager and the front-end team was very weak. Fortunately, the issue was found and get resolved right on-the-spot. We concluded that we would need to work more closely to make sure the same mistake will not happen again.

The team had a breakthrough when we decided to enforce as much automatic process as possible. The CI/CD process has been being built up from the get-go to reach the goal of completely remove all manually tedious configuration for deployment, testing, code reviewing and linting processes. It is one of the hugest wins for QUOINE ever since the company was established.

Last but never be the least, we were well-known for cloning features from our competitors and enhance them for the better. This creates new controversial conversion among all stakeholders, there were some believed that was the right thing to do while others did not. On the one hand, we believed that by cloning those features and make it better will speed things up and get us competitive quickly in the market. On the other hands, there were opinions believed that we should make our own product unique and not be like anything in the market. Subjectively, we believe that there was no single right conclusion for this conversation as everybody has their own belief. We were going to see how it turned out soon from the strategic management.
3.6 Week 6

3.6.1 Monday

Today we had a planning session hosted our new front-end team's project manager. He spent about one hour and a half to explain some terminology in professional AGILE software management system model. We started to apply SCRUM point to evaluate the complexity of the ticket. This leads to many nice effects on the way our team functions. As SCRUM point is the team’s unit that would be developed and understood among the team. It is the best unit to measure the complexity of the project as a whole instead of other unit like day, hour or week. The way SCRUM point task estimation is very simple. Firstly, the business product owner is going to explain the ticket’s requirement to the whole team. Hereafter, the team will have around few minutes to think about the task’s requirement before voting for the final decision. The sprint planning took place in a day to make sure everyone absolutely understands the project’s requirement and draw out a final estimation for that respective SPRINT. (Sitepoint 2018)

3.6.2 Tuesday

Today we received some tasks from automation test team and manual QA team. The first task was about giving each element in the list a unique ID. The current situation was that only those elements that have content in it can be rendered. Otherwise, it would render empty string without any unique ID attached. The way we solved this problem is by giving every element a unique ID regardless of how much content it has. The second task was about long name could not fit nicely inside a table cell. The recommended solution from business product team was to make text truncated and overflow on long text. We did implement this feature by applying CSS’s property overflow with a value of hidden. After that, we prepared dedicated environments on Amazon’s S3 buckets and ship the tasks to according team for testing.

3.6.3 Wednesday

Today we got some tasks about responsive web design on mobile devices. The current situation was that our introduction section for new user first time visit our page is not very effective on small screen sizes. The reason was because we never meant for our web application to work responsively on mobile devices in the first place. We decided to turn into business product team and request them to change the text content of introduction modal in order to make them fit nicely on mobile devices. The business product responded quickly in less than an hour as they came up with more precise and informative text on
small screen. Not only it helped our front-end team to make responsiveness of our web-site easier but also transfer a more succinct messages to our new users. In the day of the day, after finish testing on local environment, we made a Pull Request on GitHub and ask our front-end team members to do code review.

3.6.4 Thursday

Today we got some tasks about fixing memory leak of real-time display the book of order from all traders. The root cause of problem was that there were many WebSocket connections opened without closing when not in use anymore. This leads to vast amount of memory allocation without releasing on subscribe and unsubscribe events of WebSocket. We were trying to detect the related files that was causing the problems. This process took a great amount of time in our effort to figure out what was really causing the problems in our codebase. We used a Pusher as our third-party service in dealing with real-time data. The reason why we chose Pusher because we could get over the implementation detail of specific back-end languages since Pusher provided support for all major back-end languages includes Java, C#, JavaScript, Golang and Python. The only disadvantage of this approach was that we would be stuck in something called “vendor lock-in”. Since we chose to use Pusher as our real-time third-party service, it would be very hard if one day we decided to migrate away from it.

3.6.5 Friday

Today we received task about fixing some miscalculation in the front-end. We did a round of investigation to figure out the root cause of the bug. As it turned out, the front-end code in that section did a wrong calculation. The correct way should be done via data return via the back-end API service. We decided to resolve that by draw out the property of data returned by back-end API service and store it in local data storage. Whenever user request for data generated as maximum option, we extract the data from local state of React and hand it over to rendering process to update the UI.

The second task we did today was quite simple. We needed to fix some text translation as the current content was very confusing to our trader. Instead of having percentage display incrementally as 25%, 50% and 100%. We replaced 100% by the word “MAX”. This leads to a big confusion for some customer coming from other trading platform such as Binance that already familiar with the old terminology.
3.6.6 Week 6 analysis

The sixth week went by with a lot of improvement in our web application. Our software development process became much better thanks to new approach of applying SCRUM points and proper sprint planning. All together, we have fixed more than fifty bugs in total and release the minor version of our application into production. The git-flow branching model was shining in our application software development. The testing process was smoothened and streamlined so that our speed to the market was boosted rapidly. The QA team and Front-End team were very satisfied with the new process we achieved after all as we could collaborate much better.

Unfortunately, we have another problem that needed to be addressed as quickly as possible. The new way of management was very cumbersome since the new front-end lead guy having a very bossy working style. He has made a lot of rushing decision to release features to production without caring much about the foundational architecture of the application. The reason behind was that the calls came out of the CXO office that pushing us to be as much competitive as possible in the market. On the one hand, from the business point of view, people only think that having that feature running in product to keep current users happier and attract more user is the topmost important thing. On the other hand, with respect to technical standpoint, the team think that it’s very risky to do thing not in a proper way. We had to trade security and scalability against speed to the market. The current slogan of our head people was “better be done than perfect”. It was totally against our philosophy “it does not matter how many things we have done, what really matter is how well we have done it”. The current situation turned out to be fairly heated among all front-end team members. Even though we just got our software development process became better, we started to face bigger problem by having the working culture and philosophy towards getting thing done very diverged. This is definitely something we would all join our hands to solve it in the upcoming week.
3.7  Week 7

3.7.1  Monday

As usual, we had sprint planning on Monday. Since we have done a great deal amount of work on improving our software development process previously. From this point on, our strong focus will be placed on tackling performance issues that we currently have in our app. We were listing out the accumulated lists that had been accumulated over years based on previous engineers’ documentation and current ones. We decided to put strong focus on the architecture of our current API REST’s services system and foundational develop, test and build steps.

3.7.2  Tuesday

Today we started revamping our current development’s scripts which was rejected one year ago from the original Facebook’s create-react-app tooling. This was one of the biggest mistakes we have ever made ever since the development of front-end team started. Instead of forking the original tooling and make some tweaks as necessary, we rejected instead and went on our own path of configuration which has always been nightmare for most of developers.

![GitHub workflow eject and fork strategies](Nam_Nguyen_presentation_2018)

Figure 3: GitHub workflow eject and fork strategies (Nam Nguyen presentation 2018)
As the attachment show above, our front-end team at QUOINE decided to go on our own path of maintaining the ejected version of original create-react-app's react-script. The biggest drawback we can see from the figure above is that we ceased ourselves from receiving any updates from the mainstream branch of Facebook's team. What we should have done is to fork our own version of create-react-app and adjust it as we see it fits our needs while manually rebasing the forking version to the original version to receive updates. This approach is a bit cumbersome but doable and become well-known among React's community.(FE codebase QUOINE 2018)

3.7.3 Wednesday

Today we kick-offed our process of researching how to get back the original create-react-app’s react-scripts tooling. If we could succeed with this, our developer experience would be boosted dramatically since the current create-react-app’s tooling were equipped by the most known and up-to-date versions of critical piece of technology out there such as Webpack version 4, Eslint version 5, Babel version 7, Jest test runner version 23, TypeScript supported, CSS modules supported and SVG file as React's component by default. The current version of our webpack was behind the latest one 2 major upgrade and the build time was really slow. The community have showed many figures to illustrate how fast new version of webpack is. The numbers never lie, and it proved that it’s at least two times faster than the previous versions.

3.7.4 Thursday

Today we continued with the task of investigating our current rejected version of create-react-app tooling and figure out whether we can get back the original version of create-react-app. Since this is such a big task and require a lot of man hours of investigating and coding, we decided to break down the task into three subtasks that involve dev, test and build stages. We would tackle the test step first since this step is almost isolated from the other two because testing stage is Jest's responsibility. We forked the original version of Facebook’s create-react-app and named it “react-scripts-wankara” to test out and fix it along the way as we see it fits our current situation. Surely, there was a lot of incompatibility issues with the mismatch between our app’s current configuration and the original ones from Facebook's team. We have to look through and compare side-by-side these and adjust to get it fit our current situation. Eventually, we figured out that the original version of create-react-app was designed to accept only one single app in order to bootstrap it at runtime while we have up to three of them running in production at the moment. We need to figure out the way how to abstract the testing script to make it as generic as possible so
that wherever we run the script test it will look into the right test folders and executed on demand.

3.7.5 Friday

Today we continued working on fixing the testing stage from our forking version of create-react-app. Since our app was architected as mono repos in which there were three applications located under one umbrella project – chart, accounts and trade. The original create-react-app expected to run under only one application. Due to this, we have to re-configure our applications’ paths to prevent duplication among all three projects while minimizing the trade-off we have to make. We extracted out test configuration to have it located under the root directory of our project. And we placed the script test in each application that at start-up always look for the configuration from the root folder. We also had to configure a bunch of mapping modules to have it matched the current pathing resolver of our app. We did a whole lot of aliases to have the import look nicer at the cost of harder to maintain. Instead of having import like so “import component from ‘../../../src/relativePath’ “, we leveraging aliases to have it nicely imported as absolute path like so “import component from ‘core/components/app’” which made things looked nicer at the cost of harder to jump around between files. We had to do extra step to re-config both Eslint and our code editors to make it understand those aliases. Eventually, we committed our changes to source version control and finished out working on adjusting our testing step of forking create-react-app tooling

3.7.6 Week 7 analysis

As the seventh week went by, we started put a strong focus on performance tasks and digging into the root cause of problems to see how we could improve the current situation. Thanks to the strengthened software development process and new front-end leaders. We received a huge boost in our productivity. We started to release much more often and the remove the duplication of testing rounds in staging environments before shipping it out into production to reach our end users. We spotted the opportunities to upgrade our current foundational development system and started investing man hours on it. The upgrade process might take weeks and even month if necessary, as usual in IT industry. There is no single piece of technology could be withstanding against times as big players in software engineering such as Google, Facebook and Microsoft keep investing on it. At QUOINE, we were in no exception. We were looking into GraphQL API as replacement for current RESTful system to optimize our fetching resources mechanism and minimize our cloud cost on Amazon web service. The roadmap of revamping our front-end codebase
started from the foundation and bubble it up to the implementation detail level of each features accordingly.

The attachment below showed how we successfully done the first step if our process of re-architecting our build system. As the figure on the right-hand side shows that the testing process executed on the whole codebase. Comparatively, the new testing system only run relative files which have effects on testing files. This reduce the testing time dramatically as the test runner never have to go through the entire codebase and execute every single test which very resources consuming. In the upcoming weeks, we would continue working on finalizing the development and build steps since they play as significant role as testing step. Our goals were to reduce the reload time of web browsers on files’ content changes to less than five seconds instead of ten second at the moment of writing. And the build step was around twelve minutes instead of several ones. We strived to reduce this number in order to make quick releases or hot fixes as the changes need to be up and running as quickly as possible in production. (Nam Nguyen blog 2018)
3.8 Week 8

3.8.1 Monday

This week we continued working on what was left in the active sprint. We decided to tackle on the translation problems we have always have. Back in the day, we only supported three languages which are English, Japanese and Korean. We could not foresee the future of our application. All translation contents were bundled up into JavaScript files and ship directly to the user. At the time it wasn’t a big deal for us as our content did not take up a whole lot of bytes we sent over the Internet, and the translation step on the client happened very quickly since everything was packaged in chunk. As our application evolves, we started to support more languages and business content also increased dramatically. Our build bundle sizes just kept going higher and higher. The problem was that not all of our users going to need instant translation on the client as per analytics. Thus, sending bunch of languages content would not bring much benefit to the end-users. We decided to take a leap by dynamically translate those business content as per user’s preference. This would take a lot of work to do, the estimation was approximately one week by a pair of developers.

3.8.2 Tuesday

Today we continued what we left off yesterday by identify all necessary dependencies needed to make the translations happened. The job was quite tedious and time consuming. We had to split single translation filed so-called translate.js into separated files according to supported languages. At the end of the day, we finished migrating around twenty percent of the business content.

3.8.3 Wednesday

Today we kicked off the day by fixing one bug which is date format specific. Instead of having date format content identified in translation file, someone among us left the US format (YY/MM/DD) instead of provides proper formats corresponding to various languages supported by our application. Besides, we continued with migrating translation file into separate ones.

3.8.4 Thursday

Today we continue working on copy and paste content from translation file into separate ones as per languages which included Korean, English, Japanese, Chinese and simplified Chinese.
3.8.5 Friday

Today we finalized our migration of business content from one big package into separate files as per user's preference. The up side of this approach was to minimize the content we send over the network to client machine, this results in much faster time to first meaning content on user's devices. The down side was about extract complexity for development as we will have to input business content manually into multiple places instead of just on centralized file as before.

3.8.6 Week 8 analysis

This week did not involve a lot of changes in our product as we are just fixing a few minor bugs and big migration happened on business translation contents. From user’s perspectives, they would receive a whole lot of page speed improvement since their devices will not have to download a bunch of unnecessary data before could get to the main information that matter the most to them. The goal of us for next week would be placing strong focus on moving all calculation logics away from client, processing raw data, improve test coverage and bring down the memory consumption due to many web socket opened by multiple tabs.
3.9 Week 9

3.9.1 Monday

Today we had a sprint planning session in the morning. We discussed which features we should include in the upcoming two weeks. As the fact that we were about to release shiny new feature so-called “Liquid Infinity” with a completely new user interface. There were a lot of work need to be done. Other than that, we wanted to level up the security of our application by place authentication over our real-time trading system. At the time, there was no authentication on any WebSocket connections so that those are all available to the market. In the future, this should be avoided at any cost since someone might just grab the socket’s URL from developer tool and consume it. At QUOINE, we had a dedicated team working on the API level of our application. They wrote a custom client library for our front-end team to authenticate WebSocket. We tried to use this library in our application without affecting the rest of codebase. Unfortunately, it is not as smooth as expectation since there are couple of abstraction already in place that prevent us from integrating easily. We stuck in the process and had to re-check with the API team about how the client library should be written in order to make it work in harmony with the current front-end application.

3.9.2 Tuesday

Today, we had to fixed two bugs in development branch of front-end codebase. The first problem happened due to user’s data is not available upon successful login. This lead to serious issue in navigation bar of our application as sign-up and sign-in buttons are still available aftermath. The root cause of problem was that we were in the process of merging three application hosted in different domains into a single one. Whenever the redirection happened on the client, the full page reload was needed. Later when the routing mechanism changed, the routing is the responsibility of one single domain code only. The solution for this was to load user info after user provide correct username and password. Accordingly, the user’s info would be available in the next routing. The second problem was about layout issue we had with order book. As someone in our front-end team unintentionally changed one line of CSS that leak the style into global scope that lead to inconsistent styles across several components. We provided a fix for this by scoping our CSS into separate module to prevent the style being leaked out. In the end of the day, we had a brief discussion with API team about how authentication should be done over the WebSocket, we requited them to change how they built the bundle for our front-end application since they used the old approach of loading JavaScript at the client which is via script tag instead of web module.
3.9.3 Wednesday

Today we received an important task from our backlog. Our current application work flow always required full-page reload after user’s successful authentication. The reason was because back in the day we have multiple applications running in production instead of one nowadays. We had always used “window.location.href” to redirect our user to different app which lead to unpleasant experience. As of now, we unified two routing mechanism into one which provided by well-known react-router package. The fix requires fairly amount of work because two routing systems had always worked hand in hand. The merge was not very straight forward as we needed to make sure the data stored in memory remained intact. In other words, we migrated from multiple applications to single page application. The mental workflow was shifting accordingly. In the day of the day, we finished fixing the log-in issue for new users, change related test files and clean up some deprecated components along the way.

3.9.4 Thursday

Today we continued replacing full-page reload by react-router in the browser. Along the process of figuring out all the lines of code needed to make it happened, we bumped into several problems. The first one was checking user’s condition within rendering method which violate a Single Responsibility pattern in programming. This made the unit testing become very cumbersome as we had to mock the unnecessary dependencies. The fix was to separate them into completely isolated methods so-called render and helper. The second was naming problem in JavaScript, as our team members previously decided to name one URL’s query param as continue. This leads to subtle problem as continue is a reserved keyword in JavaScript as well as many programming languages. This also prevents us from using object destructing. We had to renaming all of these to redirect with the help of refactoring tool. The last but also biggest problem found was about authenticated routes. Currently, users can still navigate to those routes without being authenticated first. Surely, the user interface will load infinitely which is bad for our application’s reputation. We discussed with some of team member to draw out a plan to fix it as there should be a complete list of protected routes according to business requirement.

3.9.5 Friday

Today we continue to work on improving our current routing system of QUOINE’s front-end trading application. The biggest pain point hit us was regarding to the state management changed during client-side routing. As the front-end application periodically save huge amount of user preference on the server instead of arbitrary action performed by the user. This leads of one of the must-avoid error of programming so-called “race-condition”.

As the user navigates around our application, the state changed locally, and it will hit the point where it’s saved to the server. The initial state was bootstrapped on initial load of each application before they got merged, this required full-page reload. As of now, there is only one single application to work on, so does the bootstrapping process. We were working on the way to make it happened. Another problem we needed to solve is big chunk of data delivered over the network unnecessarily on saving user’s preference. This chunk was measured around 80 Kilobytes on each save, and this happened very frequently. Currently, it’s store as a string of JSON data. The client needs to parse them before it can actually be used. This is a fairly expensive process for browser to do and should be avoided at all cost.

3.9.6 Week 9 analysis

This week showed many areas of our applications that should be improved in order to deliver better user experience. It was ranging from our application’s routing system, state management and API call frequency. We started diving deeper into the legacy code written ten years ago embedded deeply in our front-end codebase. There was a lot of bricks have been put down in the first place, any replacement should be a perfect match to prevent catastrophe from happening. We were trying to work hard on balancing the amount of code added and removed ones. As our front-end application performance was not that great compared to our competitors like Binance, Bitflyer, Bitbank, et cetera. Our front-end team decided to put more effort on the tooling and rethinking best practices toward our front-end development cycles.

In term of management, the first thing to do was to split the team into smaller manageable unit. As of now, the team size is way too big compared to AGILE’s software development best practice. We were the team of twelve front-end developers working under several independent domains. The upcoming weeks would involve having multiple teams working on isolated unit such as landing page, margin-trading, spot-trading, lending and newest feature so-called Infinity. Not only the front-end team but also the company was really looking forward to this change to happened and it would work fantastically for us.
3.10 Week 10

3.10.1 Monday

Today we had a training session about functional reactive programming. The library we examined was Rxjs which is Reactive extension implementation in JavaScript. The presenter was our front-end leader. The workshop was kick-offed by a brief introduction about what reactive functional programming is and what kind of problems it is designed to solve. In short, functional reactive programming is about treating asynchronous data as a stream and provide reactive actions so-called observers. The data stream is identified as an observable that is a collection of future values come from any data sources. The job of observer is purely a call-back function that wait for values being pushed by observable. The workshop turned out to be very interactive as we used Visual Studio Code live share feature that allow up to maximum of twenty participant to join a single session. The whole team went through a couple of exercises and together we put Rxjs into practice by implementing a small caching feature for our product page. (RxJS Overview 2018)

3.10.2 Tuesday

Today we finalized the process of customize our testing process. Due to the nature of our application architecture and the testing configuration, the tests always run through every single test in our codebase which was about five hundred-unit tests. This led to a very severe problem that whenever the developer hit save, instead of running tests on related files, the test runner just took a holistic approach which cause a long time to finish. The solution for this was to reconfigure the testing framework we were using which is Jest from Facebook. After many trials and errors, we got that working by the end of the day and published new PR for the whole team to review.

3.10.3 Wednesday

Today we were having a guest workshop from product team. The goal for this workshop was to give everybody in the team a big picture about what we were working on, where were we in the market and what we were going after. The sessions were very interesting as we got to know more and more about our product. As we were working on it day by day, it was a good knowledge to have as we never got such a chance to deep dive into the business logic behind all those magic numbers, we were handling every day from user’s input and server API data return. The workshop was wrapped up by having a list of cryptocurrencies trading resources for the whole team to read later to consolidate our know-how in the trading industry. We realized that we needed more training session like...
this to share our knowledge to better off not only the individual but also the team which played the vital part at QUOINE.

3.10.4 Thursday

Today we had to prepare the minor release that deliver the new face for our front-end application as we merged all applications into one. The staging environment was prepared for QA to test. Since this is just a refactoring task so that features should not be broken. We spent time to support QA understand the reason behind why this such refactoring matter to our company as we were trying to deliver a faster and smoother user experience to our clients. The smoke tests were executed to make sure all core functionalities were still functioning properly. There were two key aspects that we all wanted to address heavily in our release. The first one was how our application looked in term of User Interface. As refactoring task is meant to make things either cleaner or faster, everything should remain intact after all. The second aspect was how our application behaved. The same user experience should be delivered safe and sound without any breaking changes happened. After a day of testing and fixing minor bugs, we could finally get our optimized version of the application out of the door.

3.10.5 Friday

Today we had to deliver some hot fixes in our development branch. The first one was about user could still see the login screen after logging in. This definitely did not make up any sense to our end user. As we might have a scenario where users share an URL to their friend to the log in screen. As logged in user, he can still see the modal showed that email and password are required to continue with the trading application. The solution was to add event listener for the moment before user enter the screen and our client code started rendering new screen. This event listener should check whether user already logged in to redirect him back to home page screen. The second one was about letting user’s login in again after he decided to end sessions on all logged in devices. The old logic was to check whether data was stored in the browser’s cookie. This is a bad approach as our application code would check whether user logged in or not by having token available in the cookie without checking the validity. The better approach was to make the API call to the server and get user’s data info. Accordingly, we can check whether token is still valid or not implicitly by data return from the server. As user continued using our application, the browser will get the updated token after log in and replace the staled version in the cookie.
3.10.6 Week 10 analysis

This week turned out to be very successful for us as we could be able to merge three applications in one single domain. In term of codebase architecture, the shape of codebase looked much nicer since there would not be any separated folders namely trade, core, token, chart, et cetera. Instead, there should be only one single folder called src and everything living under this directory. We finally got into the habit of having featured deliver with tests baked in. The tests should not be many but efficient. During the code review process, the thing we cared the most is whether the use cases got covered by written tests. This is something we could have done long time ago but not. This became part of our daily job and sync into habit of every developer in front-end team at QUOINE.

After a big refactoring on replacing all full-page redirect which due to the nature of how applications functioned at QUOINE. The single page application architecture was fully achieved by then. The user experience was smoothened deeply as resources were cached in memory and ready to render our applications whenever the Enter button got hit by user.

Last but never be the least, we started doing more and more workshops and live-sharing sessions among our team members. This created a motivation for us to learn more and share more as teaching others is the best way to retain our knowledge according to science. Moreover, the sharing could be anything that were ranged from technical skills, people skills, health or happiness. We wanted to prove that the happiest workplace can only be achieved if we kept sharing and working together toward some similar goals, not just by receiving pay checks and get jobs done.
4 Discussion and conclusion

4.1 Progress and development

During my working time at QUOINE while writing this thesis, there was many major changes happened to our team structure and the front-end codebase. I was able to apply my knowledge to the professional working environment like Liquid app at QUOINE. The tasks I have done cover all aspects of front-end development ranging from responsive web design, state management for single page application, project architecture to web performance optimization using modern web technologies.

I feel that the tasks given to me was really beneficial. As the problems I received become bigger and bigger, so did my working attitude. I did strive for the success of problem solving with the efficiency and effectiveness always bear in mind.

4.2 New approaches or methods for work

During the thesis I have noticed several things that could be improved in order for me to deliver better work. First and foremost, it is the meeting always took reasonable amount of my time since I found many of them were very ineffective and wasteful. The better approach would be having everyone aware of the tasks they are doing and act on-demand instead of having everything reported on the table which most of the times are not relevant to the rest of the team.

4.3 Things learned during the thesis

The things that I valued the most while working at QUOINE are people skills rather than technical skills. As the team size grew bigger and bigger, from couple of team members and went up to dozens of developers. As we conducted many code review processes, meetings, workshop, team building and knowledge sharing. I had to figure the way to communicate as efficient as possible since time is the most valuable thing at work. Besides, technical skills are quite valuable too as we had to research couple of new technologies along the way to solve our scaling problems in term of developers and code sharing within single repository.

4.4 Future development ideas

The things I wish I could have done is to have our team members divided into smaller and manageable team to work toward some specific feature instead of having many of us working across different domain. This came to live at the end of my thesis writing, and I
was able to benefit from it more or less. It really worked for me as the communication bottleneck was solved. I did not have to come over BE or QA guys table to communicate on some tasks or messaging over Slack and wait for response up to few hours.

4.5 Advantages of work analysis

The work of analysis turned out to be very beneficial for me. As the fact was that in the beginning whenever I was working some task, I have a tendency to focus on it without taking a broader look at other alternative approaches. This definitely limited my capability to do good work as I often got stuck into the situation that I tried to do the work in a bad way. It definitely would not pass code preview process of my talented co-workers. The work analysis gave me an opportunity to review everything I have done during the day in a totally fresh way. This is the moments I have often found many mistakes of me trying to solve some problems without planning things carefully before started implementing feature or fixing any bugs.
## 5 Appendices

### 5.1 Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2FA</td>
<td>Two Factor Authentication</td>
</tr>
<tr>
<td>BE</td>
<td>Back-End</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>CEO</td>
<td>Chief Executive Officer</td>
</tr>
<tr>
<td>CFO</td>
<td>Chief Financial Officer</td>
</tr>
<tr>
<td>COO</td>
<td>Chief Operation Officer</td>
</tr>
<tr>
<td>CIO</td>
<td>Chief Information Officer</td>
</tr>
<tr>
<td>CPO</td>
<td>Chief Product Officer</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>CD</td>
<td>Continuous Delivery</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>FE</td>
<td>Front-End</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Assurance</td>
</tr>
<tr>
<td>QC</td>
<td>Quality Control</td>
</tr>
<tr>
<td>SEO</td>
<td>Search Engine Optimization</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
</tbody>
</table>
6 References


