Haaga-Helia
University of Applied Sciences

Working as a software developer in a FinTech company.

Diary Thesis

Tram Nguyen

Bachelor's Thesis
Degree Programme in
Business Information
Technology
2022

Haaga-Helia
University of Applied Sciences

Abstract

Date 03.05.2022

Author(s)

Tram Nguyen

Degree programme

Business Information Technology

Report/thesis title

Working as a software developer in a FinTech company.

Number of pages 80

People now live in a world where digitalization has integrated into all aspects of a business. Organizations across various industries can benefit from digital solutions as it allows them to upgrade legacy processes, streamline operations, and boost profitability. Software developers and engineers are the significant impacts that create digital solutions apply different cutting edge technologies. They require considerable skills to develop a career path.

As a software developer stepping up in the IT field, I understand the challenges and opportunities. In this paper, I attempt to record and assess my learning progress, encourage and reflect on self-development, support and improve personal competencies and weaknesses. It can also be used to clarify concepts, issues, and theories. My thesis is composed of four themed chapters during ten weeks of developing front-end skills in a Sass company.

This thesis is also beneficial for those who want to start a career path as a software engineer in a fast-growth start-up environment.

Keywords

Front-end, Vue, Typescript, UI library, transfer pricing.

Table of contents

1 Introduction

This section gives a brief overview of the company and its products and clarifies technical terms in a field of knowledge. It also provides insights related to my scope of work, required skills, and working environment.

Before writing the thesis, I had a meeting with CTO and COO to agree that credential information should not be mentioned, including sniped from the codebase, detail of solutions and customer information. I will mainly report my daily task and analysis a specific topic that is beneficial for me at work to meet the requirement and expectations of the team.

1.1 Company introduction

Aibidia was founded in 2014 and has employed around 40 experts, including tax lawyers, economists, data scientists, and software developers. The office is in the heart of Helsinki, Finland (Aibidia n.d.). Aibidia is the world leader in digital transfer pricing management and provides multinationals, professional services firms, and tax authorities a modern way to execute and manage cross-border transactions and business activities on data and intelligence. The system includes a full range of digital transfer pricing solutions and applications covering worldwide transfer pricing compliance and operational requirements. After eight years, the company successfully researched and developed four applications and has created an ecosystem to digitalize international tax. (Aibidia n.d.)

1.2 Key term in domain knowledge

While various definitions of the term transfer pricing have been suggested, this paper will use the definition suggested by Gilbert, Mcmillan and Walters (2013). He saw it as an accounting practice that illustrates establishing a price for the transaction between two divisions of the same organization. Transfer price is based on the market price when charging another division, subsidiary or parent company for services rendered. In many cases, companies take advantage of transfer prices to reduce their tax burden through tax avoidance and tax evasion activities. They circumvent the law by charging a higher price to divisions in high-tax nations to cut down profit to pay less tax. In contrast, firms set a lower price for divisions in low-tax countries to raise profit to optimize taxation (Investopedia, 2021).

From the business perspective, transfer pricing is a complicated topic related to revenue, profit, agreements, policies, laws, and rules. On the other hand, the whole process against the crime related to tax avoidance and tax evasion is a nightmare for tax authorities and

involves legal parties. Therefore, Aibidia creates digital advantages for transfer pricing by providing different solutions applying various cutting-edge technologies. (Aibidia, n.d.).

1.3 Products

There are four applications:

- TpDoc - Transfer Pricing Management
- CBC - Country by Country report
- VCA- Value Chain Analytics
- OTP- Operative Transfer Pricing Solution.

TpDoc solution aims to help global companies avoid needless alteration and error that arises in tax control carried out by the tax authorities in different countries. It also eliminates the risks of double international taxation. Another highlight is that the solution will benefit organizations to interpret intra-group transactions - the transactions between companies of one group or parent company- in a better manner. Intra-group transactions are. If these transactions are not appropriately documented, the parent organization will have inflation of the net income. (Aibidia, n.d.).

Unfortunately, the Country-by-Country report remains a poorly defined term. According to a definition provided by Deloitte (2016), the Country-by-Country report is first described in the OECD's Base Erosion and Profit Shifting Action Plan 13. Delloite, (2016). It declared that large multinational corporations must submit an annual return, known as the CBC report, covering significant parts of the jurisdiction's financial statements. Revenue, income, tax paid and accumulated, employment, capital, retained earnings, tangible assets, and activities are all visible in a CBC report to local tax authorities. Our CBC delivers a digital solution to produce required documentation. (Aibidia, n.d.).

The tax authorities always look for a holistic view of the tax profiles of companies and organizations. The transfer pricing is not audit-proof until one can show its accounts for value-creating within the entities. VCA is an optimal solution in this situation. (OECD Countries, 2009).

OTP is the management of transfer pricing data, processes and governance using the software. The OTP solution aligns transfer pricing requirements with company goals, segments Profit and Loss statements (PNL), promoting compliance and reducing complexity. (KPMG, n.d.)

In addition, the firm built its UI library to synchronize its look throughout four applications called AibidiaLayout. As a SaaS, software as a service company, all products are on a mission of digitalizing transfer pricing. (Aibidia, n.d.).

1.4 Working environment and scope of work

The solution team has around 20 highly skilled and experienced software engineers. I join the company in the middle of November 2021 as a junior Front-end developer. I help optimize web presentation and contribute to client-side development of TpDoc, VCA and AibidiaLayout using front-end framework Vue, along with typescript and javascript in the core. My thesis is composed of four chapters conducted in 10 weeks starting from 10.1.2022. I have maintained and developed front-end features and functionalities to meet diverse requests from over 20 customers.

The research information and data in this thesis are drawn from assignments assigned by the team lead and supervised by my Chief Technology Officer of the company. The company is using the SCRUM methodology to develop and work continuously. During the Sprint planning, the team will define the Sprint goals, which can be done within three weeks. (SCRUM Org, n.d). Every day we have a daily stand-up meeting to keep updated. The remote policy at the company allows developers to work from different places, many of which are even abroad. Work from home become a company culture even before the pandemic. Therefore, it is crucial to keep communication clear to avoid bottleneck situations when working remotely.

1.5 Required skills

Technical experiences are essential for a wide range of job duties regarding skills. My competency is a solid background in programming language and modern web applications. I have knowledge of JavaScript with React on top to process the front-end side and NodeJS to handle the backend, while company tech stacks are Vue, another front-end library, Typescript and .Net. The fact that I need to learn another framework and technology in a short time leading to my onboarding process, is quite challenging. It is indeed a tremendous opportunity to learn new techs despite the learning curve. It is not required, but understanding the section that the Aibidia is working on is recommended. Hence, I also learned the core concept of transfer pricing.

Soft skills are a common condition that has a considerable impact on developing a successful career. Soft skills include communication, planning, approaching, and solving a problem, etc. Most of these skills are needed daily and hopefully gradually improve throughout my work at Aibidia.

1.6 Key professional concepts

Azure Devops	Version control, reporting, requirements management, project management, automated builds, testing, and release management are some of the features of this Microsoft application. It allows DevOps capabilities and covers the complete application lifecycle. (Microsoft, 2019).
CMD	Command-line, text interface for computer. The command line is a blank line on the screen with a cursor that allows the user to write commands for instant execution. Almost every central operating system (Windows, Mac, Unix, Linux, etc.) (Janssens, J., 2021)
GIT	Git is a version control system that allows you to organize and trace the history of your source code. (Git, 2022)
JavaScript	A scripting language, interpreted language, used on the Web (JavaScript | MDN, 2022)
Jest	JavaScript testing framework (Jest, 2022)
NPM	Node package manager is the Node JavaScript platform's package management. It installs modules so that node can discover them and intelligently resolves dependency issues. (npm, 2022)
PR – Pull Request	Pull requests (or Merge requests in some terminologies) - later referenced as PRs - are a common practice for managing the flow of feature branches into the main development branches. At Aibidia, PRs are the only way to merge your work into our main development branches (master/main). Merging or pushing directly to the main branches is prevented. It is also preferred to use PRs when working on a task that is a piece of a more extensive feature still under development. Meaning the target branch of the PR will be the feature branch and not the main branch. (Lenarduzzi, Nikkola, Saarimäki. and Taibi, 2021)
SASS	Syntactically Awesome Style Sheets. Sass is a scripting language for preprocessors that is interpreted or compiled into Cascading Style Sheets.(SASS, n.d)
Sprint	Sprints provide predictability by guaranteeing that progress toward a Product Goal is monitored and adjusted over a set period of time. In Aibidia, a Sprint last 3 weeks. (SCRUM Org, n.d).

Typescript	A superset of JavaScript
Vue	A progressive framework for building user interfaces. Vue is client-side framework. (Introduction — Vue.js, 2022)

2 Framework

This section describes the procedures and methods used in investigate various aspect of my current work.

2.1 Analysis of your current work

I have worked in Aibidia since the middle of November 2021. The first month is considered getting on board when many meetings are conducted with different solution team members and other departments. The communication channels are Slack and Microsoft Teams. The material is partly found in the video bank and the Wiki of Azure DevOps. I need to set up my local machine during the first week and grant permission to access the codebase. The first ticket I got is refactoring the old code in JavaScript to Typescript.

I also learn about how the workflow goes in Aibidia. The solution team applies the SCRUM methodology to define the working process with dynamic changes to suit the company's fast growth. The term SCRUM methodology is used here to refer to a development process. As we understand, the systems development process is a complex, unpredictable process that can only be defined in broad strokes as an overall trajectory. (Doğan. and Tüzün, 2022, 1-28). SCRUM illustrates the systems development process as a loose collection of activities that combine well-known, valuable tools and procedures with the best that a development team can develop to build systems (SCRUM Org, n.d). Controls to regulate the process and inherent risk are used since these activities are loose. SCRUM is an iterative/incremental object-oriented development cycle that is commonly utilized (Schwaber, 1997, 117-134). Sprint is an essential role in SCRUM, considered as the duration throughout a development effort. The figure below illustrates in detail this agile approach.

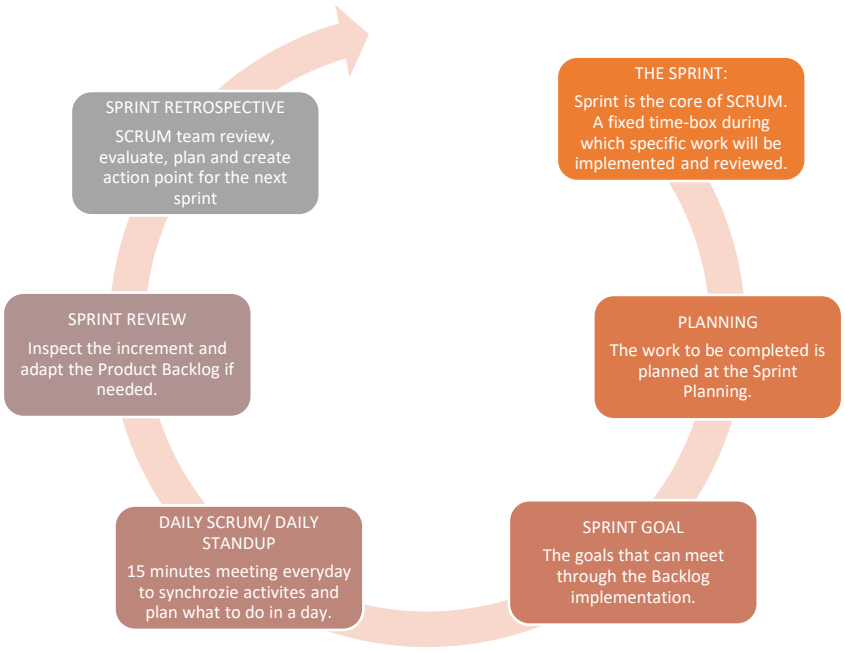

Figure 1. SCRUM process (SCRUM Org, n.d).

My main work tasks for 10 weeks working with four different projects include:

- Integrate new technology to current codebase.
- Investigate and implement the new features.
- Fixing bugs and refactor code base.
- Review and test pull request from other developers.

2.1.1 Integrate new technology current codebase

Transforming and updating new tech contains a massive workload, so we gradually integrate the current codebase from old to new. The new tech stacks make developers' lives easier due benefits that new tech offer. For example, there is no way that software engineers can predict a specific data that function in JavaScript will return. In contrast, developers can precisely know what will be produced even more than that by using Typescript. It was one of my first tickets when I joined the company.

One of the most significant tasks I have done so far is refactoring AibidiaLayout, an Aibidia UI library using Vue2. As we may understand, Vue 2 is not fully supported by Typescript. Therefore, I need to work around refactoring all Aibidia Layout to Typescript. Unlike Javascript, Typescript requires typing every data, props, function, method, parameter, etc. Accordingly, I need to go to solution applications and debug to understand their types. It is a complicated process, but the final work is done nicely, saving time for those who continue with AL.

2.1.2 Investigate and implement the new features

The projects' improvements and specifications are constantly evolving to satisfy and fulfill the needs of end-users. Therefore, many new features are implemented to the products. This process is happening in sprint planning or stakeholders' meetings.

I need to learn about the projects and new technologies that can be utilized to implement new features as a new developer on the team. Interestingly, I spent more time studying than coding during the development process. When a ticket is assigned to me, I first investigate the description before coding and trying to find a solution to understand the requirements and expectations. The resources usually come from documentation of technologies on their official website, Q&A platforms such as Stack Overflow, blogs and medium articles. After researching, I usually can find the solution by myself, but it is not always the case. In that situation, I ask for help from seniors to succeed in my tasks.

Whenever I finish a ticket, I create a pull request and ask other developers to check and leave comments on my work. After resolving all comments, the gatekeeper will approve and merge my branch into the master/main branch. The last thing to do is return to Azure DevOps and mark the ticket as done.

2.1.3 Detecting and fixing bugs

It is normal to have bugs in developing software even though the company conducts a strict testing process using the unit test, integration testing and CI/CD pipeline. A bug is an issue, error, fault, or failure in a computer program or system leading to the application not working as required. Fixing a bug is finding a solution for that problem. Fixing bugs and refactoring the codebase always come together.

Moreover, I need to perform a refactoring task when I add something new to the existing application and maintain a clean and clear code. This effort saves time for the next developer to join the team, as well as time for me to explain each line of code. Experience is the most important qualification for this position. All the recommended practices for writing clean code were taught to me at my previous job.

A well-defined coding convention for the entire development team is necessary. That is why the company occasionally has sharing sessions to synchronize good practice and update conventions for everyone in the team. I find this is an excellent chance to learn from peers.

2.1.4 Review and test pull request from other developers.

In addition to establishing my own assignment at Aibidia, I am also responsible for reviewing and evaluating the work of other developers. There is a tester specialized in testing and maintaining high-quality code. He is the gatekeeper to ensure that the new implementation cannot break the current functionalities. However, developers still need to check code for each other. There are several reasons for that. First, it will reduce unnecessary work for the tester; second, we make sure that the code is double-checked and as clean as possible.

I have worked in a team before, but it is very different from the group work at school. Reviewing and leaving suggestions on coding logic for other developers is more complicated than I thought. When I do this job, after fetching the branch that others are working on to my local machine, I first mistake regarding styles and convention, spot Eslint error. After that, I run test cases to make sure that they are passed, understand the logic and the solution used to solve the problem. I also play myself as the end-user to evaluate if the solution is getting the same result as expected.

2.2 Evaluation and development plans

I have a meeting to get feedback on my performance from my CTO, a senior front-end engineer, and a colleague. They are all working closely with me. The discussion occurs during my third month at the company. I have been received positive opinions on the tickets that I responsible for. I was complimented for my active attitude and ability to solve a problem independently. Some comments for improvement include how to be more actively contribute to retrospectives, and they advise me to be more balance between work and life after seeing my commits at midnight. I appreciate an environment where everyone respects and cares about individual development and mental health.

I do not have much working experience since Aibidia is my first company. Even though I still make clumsy mistakes and need detailed descriptions to understand the task, my capacity to write clean code is improved a lot when I get help from such a supportive environment.

JavaScript was my first programming language when I learned coding skills and continually explored them in university. Since then, it has changed a lot; now, my level of using this programming language is up by using Typescript. I spent most of my time learning a popular front-end library, React, but did not have a chance to use it at work.

However, React and Vue shares many key concepts, so I did not waste anything from learning React.

I hope to learn and work with backend tasks in the future, as my goal is to become a full-stack developer. Moreover, I am interested in testing and DevOps. I will start to get fundamental knowledge by learning open courses on MOOCs provided by the University of Helsinki. My development plan is to find domain knowledge as I want to use coding skills to benefit different sections.

2.3 Interest group at work

My company size is medium with around 42 employees and will continuously grow next year, which is ideal for several reasons. There is a general agreement about the advantage of working in a midsized company. Firstly, the proposals and action points are quickly approved or rejected due to the low hierarchy. In addition, there is much room for growth as the company has seniors to mentor juniors, and when the company expands, there are more chances for career promotion. Most importantly, your work is significantly contributed to the company's development in different ways. Here is the list of groups in which my works are affected directly or indirectly.

Internal groups:

- Solution team includes developers in front-end, back-end, DevOps, tester.
- Managers and Product owners.
- Other departments such as sales teams, marketing team, analyst team, customer success team.

External groups:

- Tax authorities
- Customers
- Partners

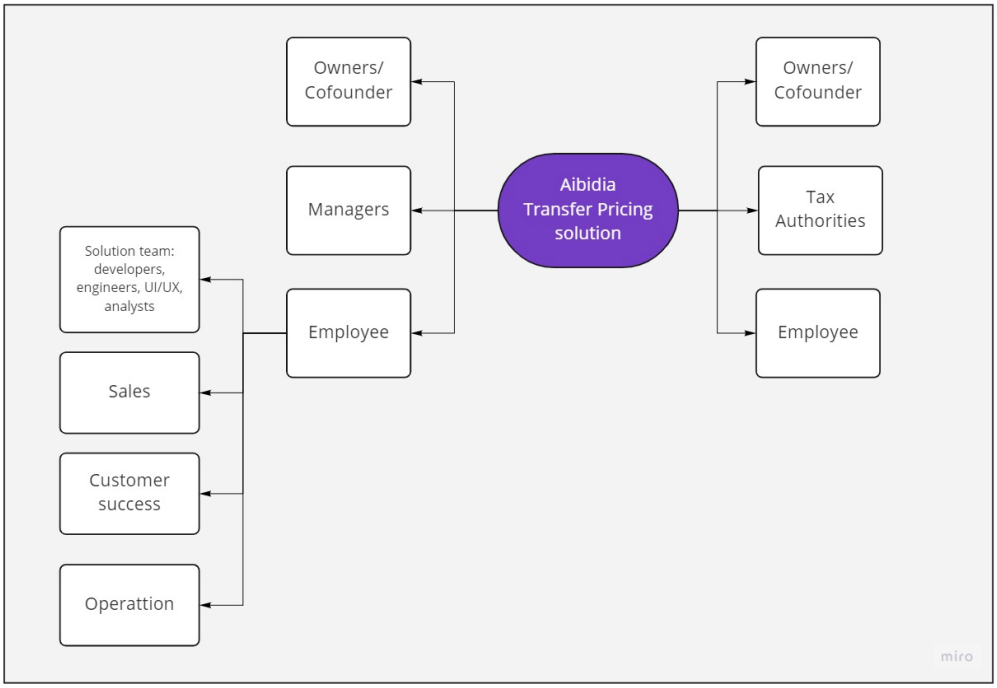

Figure 2. Internal and External Groups

I receive several types of information from our company's internal interest groups. However, the most pertinent is the feedback from our customer success team concerning the transfer pricing solution. These data and interactions pertain to solution performance and a list of mistakes discovered, the need for technical assistance, suggestions for development, and new ideas. They assist me in identifying unnoticed flaws, resolving unresolved faults, and optimizing the performance of our product. Business logic is carried out with external interest groups using online interfaces, web Application Programming Interfaces (APIs), and file transfer protocols. We share information regarding settings, API documentation, document format, and security, which are necessary for the business to get up and run.

2.4 Interaction skills at work

As previously stated, working from home has been a company culture because of the versatile remote policy. As a result, it's critical to keep communication lines open when working remotely to avoid misunderstandings and congestion in implementation.

The solution team conducts daily meetings that last for 15 minutes. We are finding the best way to hold the daily meeting, and the current format work quite well. Details, we go to workstream every Tuesday and Thursday to update the Sprint goals; the rest of the week, we apply the individual update where everyone reports what they have done and the plan for a new day. A stand-up meeting is also an excellent time to raise questions or concerns that should be discussed and clarified by another meeting later if needed. We

use Slack as a company channel of communication. Other than that, the Microsoft package includes Teams for meetings and Outlook for email. Tasks, pull requests, and documentations are stored in Azure DevOps.

I don't deal with clients regularly. I usually get customer requests or other information about the product via the Product Owner, or the team leads. The uncomfortable nature of online communication is, I feel, the most apparent obstacle I've experienced. People have more freedom in how they begin their days now that they are not required to go to work, as long as they finish their obligations. Consequently, it's difficult for me to contact people if I need to talk about something because people aren't always online at work or react to messages promptly.

3 Diary entries

3.1 Observation week 1 (January 10- January 14, 2022)

Monday 10th January 2022

I have been doing my job at Aibidia for one and half months. In my first days of working, I was provided a laptop and its accessories. My laptop is a Lenovo ThinkPad t14s 16Gb ram with a Windows Operation system. Since I am a Mac user and familiar with Mac OS, it takes some time to get acquainted with the command line in Windows. In my opinion, Mac is more intuitive and developer friendly as it gives more control over a terminal. Nevertheless, the company uses .Net to build its backend, so Windows provides more power to develop code on the server-side. As a junior developer, I am curious about many things; in those situations, I always look for onboarding materials and ask for suggestions from my peer advisor. He is a senior front-end developer with ten years of experience in big corporations and start-up environments. With the help of my colleagues, I can set up the environment, config projects and be ready to work. My current assignments are front-end technology, so I used Visual Studio Code as my code editor.
On Friday last week, I was working on refactoring Aibidia Layout from JavaScript to TypeScript, so my plan today was to continue working on that. While testing new features writing on TypeScript, I found a bug in Tpdoc; I informed the tester to create a ticket on that bug and fix it. I finished my day by resolving comments on my pull requests from other developers.

Tuesday 11th January 2022

I finished all tickets related to refactoring Aibidia Layout, created PR for those and let my colleague review my works. I approached the bug yesterday by starting to investigate similar functionality in the application to understand what the reason is leading to unexpected behavior. Then I find the solution. I reuse available methods already defined in the codebase to solve the problem instead of coming up with a new idea. I believe this approach is more efficient and maintains the consistency of coding styles. I am also fixing the linting error involving the file so that it would make the code clear and clean for later usage.

Today, we have a daily meeting using the workstream format, in which we go through sprint goals and update how we progress with tasks to achieve these at the end of the sprint as we planned. From that, the leader of these streams can have a big picture to adjust the workflow or set the priority and motivate other team members to do their job actively. I have been working on the customer requests stream in this sprint. Those tasks are quite challenging and require experiences related to the product and back-end with a

strict deadline, so the team leader decided not to assign me many tasks to implement customer requests. Instead, I was assigned to fix the persistency state from Local Storage in Tp doc. At first glance, this should be straightforward to approach.

Wednesday 12th January 2022

I do not have the daily meeting with the dev team today as everybody will attend sprint review, where we give presentations to demo the new features for other departments. In this meeting, people are open to asking questions and clarifying some problems. It is interesting to discuss another topic than just technology in the sprint meeting. The managers also give recommendations and business insight for engineers to build products that are beneficial for current customers and the company's future development.

After the meeting with the whole company, the solution team had another meeting called Sprint Retrospective. As clarified in the SCRUM process, this is an occasion for the solution team to reflect on what was going well and what was not during the last three weeks. Today retro is the perfect event welcoming members from the holiday spirit to return to a working mood. Different questions before the meeting are provided to collect ideas on how people think about the result. Action points are also mentioned to improve the quality of work. Everyone is given a chance to elaborate on ideas and suggestions of how the team approaches a problem. There is no unreasonable request; when different people have different opinions, we openly discuss and give the pros and cons of each concept. Even though the company's focus is continually developing, this is an essential pause for continuously improving and continuously learning.

On Slack, my CTO listed an urgent task related to Implementing support for actual async generation of Local and Master File. I was interested in that and spent some time digging into that. In addition, I carry on fixing the persistency state from local storage. I found out that the problem is not from the Tpdoc but the Aibidia layout indeed. I investigate this for the rest of my date.

Thursday 13th January 2022

Continuing the Sprint review and retrospective yesterday, we hold a Sprint planning today to plan for the following sprint goals. I assign to work on different projects but mainly working on the Rich Text work stream during the next three weeks as I have developed a feature related to this before. Besides that, I have been working on VCA, TPdoc and CBC to fix bugs and implement new features. I have another meeting related to the async generation of Local and Master File. There are CTO, frontend developers and backend developers in the talk. After the discussion, B was responsible for this as he has more

experience using message broker services, so he can solve the problem quickly to ensure a short deadline.

During the rest of my day, I adjusted to enable the sorting functionality in tables. Create a PR for that at the end of my working day.

Friday 14th January 2022

There is a bug on the backlog from last year that is still not resolved; I take that ticket and work on that. That bug is causing a small error when rendering the sidebar menu; it is not trivial as it doesn't affect the application's performance but only its appearance. Therefore, it remains in the backlog for such a long time. Regarding the bug, when there are many layers, the higher the z-index, the higher the order of that layer. I fix that bug by changing the z-index in style. Moreover, I create a function to parameterize the Z-Index values to CSS variables and have a single file where those are defined to control the order easier.

The tester returns from his vacation today and comments on the pull requests that I created this week. I need to solve them to merge my branch into master and main.

I finish my day by reviewing pull requests from others. I spend the least of my time doing this task as it is difficult for me. There are two types of pull requests: developing a new feature or fixing bugs. For the pull request to develop a new feature, I first read the product backlog item description to understand the new function, expectations, and requirements. Then, I pull the branch and run the test to ensure that all tests are passed. The next thing to do is to break down and observe if the functions work fine under different actions.
On the other hand, I try to reproduce the bug in a test environment and pull the branch and run it in my machine for the pull request of fixing a bug. Now, I struggle to understand others' logic when they solve a problem, so I give comments about the convention. My goal is to learn how to read code faster and understand the error throughout to give proper logic.

Week 1 analysis

This week closed the holiday season and preparation for the next release in January. Every department is excited but also hustles for the big event. I have been working to fix minor bugs on the current code base and implement new features in TPdoc.

The team lead assigned me to fix an error related to using local storage to persist data that allows users to sort the table's result in either ascending or descending order. He

wants me to finish this as soon as possible as it is an important task that would affect user experiences. I allocated the problem, proposed a solution to fix it, and everything worked fine without any matter. However, when the tester reviews the work, he is not happy with my answer. The diagram below illustrates how the current logic runs now. I personally have not found any problem with this approach yet. I will spend more time researching and asking for help if needed to provide a more optimal solution.

```
TypeScript
  WORKS NOW:

  mount (load the state from local storage)

  load from localStorage and store internally in data the SEARCH state
  load from localStorage and store internally in data the FILTER state
  check if columns exists:
      load from localStorage and store internally in data the SORT state
      update the HOT sort state

  start loading data

  FREEZE the tabledata

  ...

  ALL THE API CALLS


  finish loading data
  ...

  UNTHAW the tabledata

  trigger the the watchers for columns and for tabledata

  columns watcher:
      updates from the internal FILTER state the column filters
      update HOT filters
      update internalColumnState
```

Figure 3: Current logic of persist data from Local Storage

This week is full of meetings; as a result, this week's analysis attempt to review skills on attending a meeting. In the book, Effective meeting skills (1988), attending is defined by Haynes and Crips as devoting all of your physical attention to another person. Whether you know it or not, attending has a significant influence on the quality of communication between two individuals. It's critical to understand the meeting's purpose. I've discovered that we have meetings for action planning rather than conversation. Important decisions are even made ahead of time. A meeting's ambiguous purpose of discussing a specific topic does not create effective outcomes. Make a meeting agenda and make sure it is in

the hands of those attending at least the day before the meeting. Everyone who attends a meeting should know why they're there and what they're supposed to accomplish.

Furthermore, I have taken down some other concerns when attending company meetings based on my personal experiences. First, it is essential to understand the business insight. In fact, I assumed that technical skills are overweight other skills when doing software engineering. The practice proved me wrong: when I attended the sprint review, I realized that it is essential for developers to understand the business insight provided by other teams, such as sales and customer success, because the critical factor is the end-user, not the product. Secondly, I need to ask questions actively and express personal thoughts. Suppose the right questions are not asked at the right time. In that case, other people will implicitly think that I have thoroughly understood the task, leading to an expectation that I could finish the job quickly. Sharing personal opinions is beneficial in brainstorming new software development options. Expressing my thoughts also allows me to hear other people's perspectives. (Haynes and Crips, 1988)

3.2 Observation week 2 (January 17- January 21, 2022)

Monday 17th January 2022

Last week, I worked to persist data that allows users to sort the table's results in ascending or descending order. My solution was not optimizing as it did not solve the root problem but only hacking to function temporarily. The logic and flow of code are set out in Figure 3. After investigating, however, I found out the reason causing the bug.

First, we need to clarify some key concepts. I was curious about the lifecycle in Vue. Lifecycle refers to stages when methods work behind the scenes to build Vue components and enable different actions to trigger at other times in different interfaces. There are the four stages of a life cycle of a Vue Component.

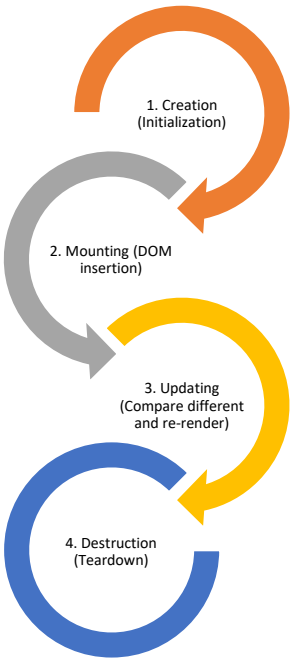

Figure 4: Four main category of Lifecycle in Vue (Lifecycle Hooks, n.d.)

We can define actions that run before the component has ever been added to the DOM in the creation phase. Nevertheless, it is impossible to access a DOM or target mounting element inside creation hooks. `beforeCreate()` and `created()` are hooks implemented during this phase. Whereas `beforeCreate()` run even before Vue components are non-reactive, events during this component's life cycle have not been set up. `created()` hook is called set up event and data observation, which means templates have not yet been mounted or rendered, but active events and access to reactive data are enabled. Next, `mounting` is a perfect time to access the component once before and after it is rendered for the first time. It is helpful to handle data and modify the DOM of elements immediately before or after the initial render. After template has been complied initial DOM and updated virtual DOM, `beforeMount()` is called. Whenever the native

DOM is updated, the `mounted()` hook will be given full access to reactive components, template, and DOM. It is suitable now to fetch and collect data so that manipulate the DOM. (Lifecycle Hooks, n.d.)

Observe figure 3 again and pay attention to the mount phase. After component initial and render, the function should only load data from `localStorage`. Any checking conditions here are redundant, and hence, it is eliminated.

```
TypeScript
  WORKS NOW:

  mount (load the state from local storage)

  load from localStorage and store internally in data the SEARCH state
  load from localStorage and store internally in data the FILTER state
  check if columns exists:
      load from localStorage and store internally in data the SORT state
      update the HOT sort state
```

```
TypeScript
  HOW IT SHOULD WORK

  mount (load the state from local storage)

  load from localStorage and store internally in data the SEARCH state
  load from localStorage and store internally in data the FILTER state
  load from localStorage and store internally in data the SORT state

  start loading data
```

Figure 5: Fix the logic inside mounting lifecycle.

Now, I can find another problem in the current approach.

```
  UNTHAW the tabledata

  trigger the the watchers for columns and for tabledata

  columns watcher:
      updates from the internal FILTER state the column filters
      update HOT filters
      update internalColumnState
```

In the existing solution, the watcher is used to track changes in data here; we forgot to check if any columns exist, and therefore `internalColumnState` is invalid, and the `sort` state is not ready to be present from `localStorage`. This is how it should work after minor modifications but massive logic changes.

```
TypeScript
HOW IT SHOULD WORK

trigger the the watchers for columns and for tabledata

columns watcher:
    updates from the internal FILTER state the column filters
    update HOT filters
    sanity check if column exists for sort
    update the HOT sort state

    update internalColumnState
```

Figure 6: Fix the logic inside updating lifecycle.

Overall, the lifecycle in Vue is an essential concept when working with Vue. More detail and intuitive description can be illustrated through diagrams from its official website.

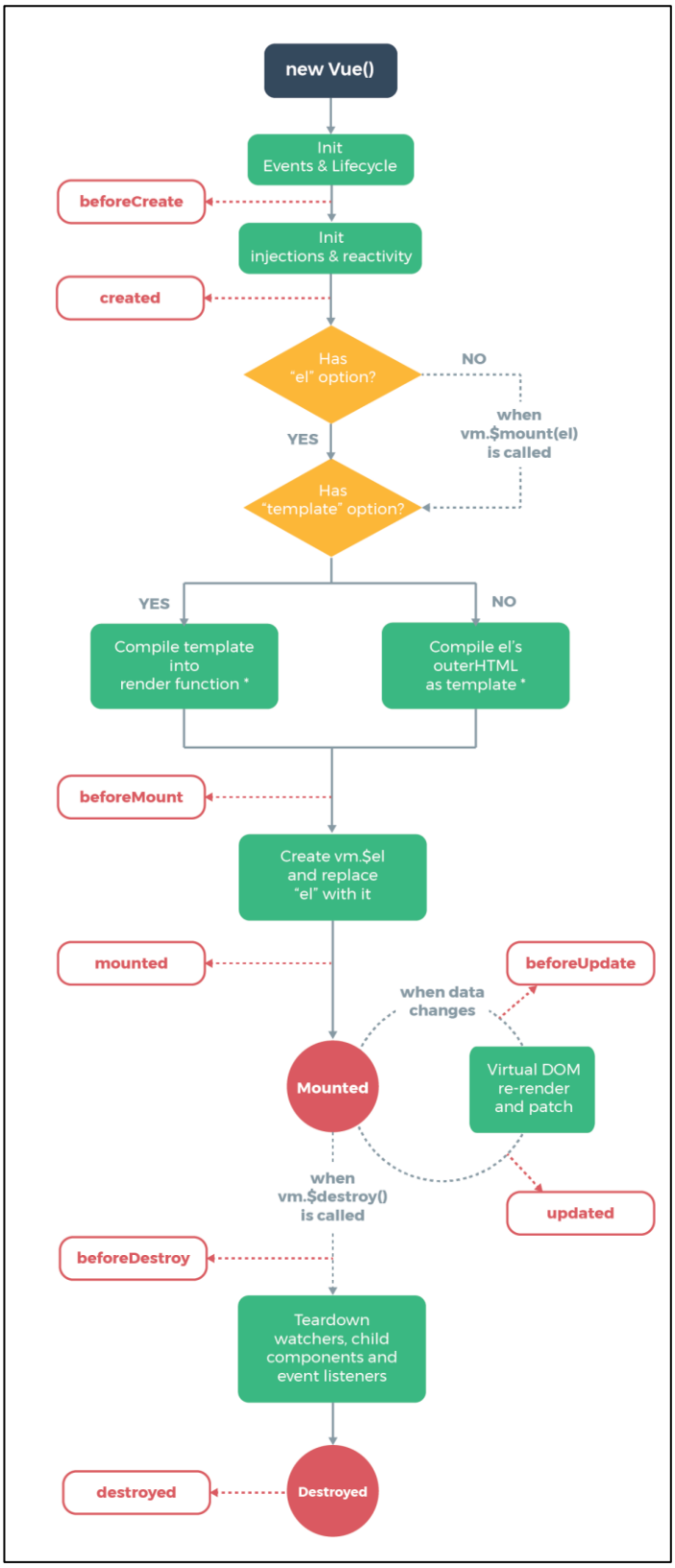

Figure 7: Vue lifecycle (Vue.js documentation, n.d.)

Today, I can understand deeply about the lifecycle in Vue and propose a proper solution. I create a pull request for this ticket in the end of the day and ask the tester again to review it. This time, he is happy with my approach.

Tuesday 18th January 2022

I start my day by resolving comments on PR and creating some minor adjustments on pull requests to refactoring Aibidia Layout tasks. Then, I come back with my primary mission this Sprint: to improve Rich Text's functionality to benefit customer needs. The rich text includes format setting and compares to standard plain text; it is more engaging. Text formatting such as bold, italics, and underlining, as well as different fonts, font sizes, colored text, etc., are all supported and available in rich text documents. The solution has rich text to format paragraphs in bullet points or order lists, but it needs improvement. The goal is to level up the rich text features to enable users to format text in bold, italic, and underline. The rich text should replace the current text area and expand into modal if needed; we should care about sanitizing user input and allowing users to copy from different fields into the information. We build the rich text based on an open-source library called TipTap. TipTap is renderers WYSIWYG, built-in 2017, the latest version is 2.11, but we are using tiptap version: "^1.32.1", and its extensions "tiptap-extensions": "^1.35.1". Tip tap allows transforming input to HTML/XML for storing in the backend. HTML and XML are useful for developers, but they make no sense for the customer. Thus, it is necessary to format the text before rendering so that the user interface should only contain strings without HTML tags. I store the function in the helper folder and apply it whenever formatted text property is needed. I have a meeting with UX/UI designer in the afternoon to agree on how rich text should appear.

Wednesday 19th January 2022

I start my day by looking at PR comments and trying to resolve simple ones. One of the tickets related to refactoring AibidiaLayout is having a problem running the build test in Azure Pipeline, I investigated, and the reason is quickly found out. Because of the new update in the web pack, typescript strictly checks typing of the data. The solution is straightforward, and I define a type for those untying data.

Next, we have a 15-minute daily meeting; people usually report individual tasks. My feature-rich text is progressing alright. I wrote a test case using cypress for the function `formatHTMLbeforeRendering` created yesterday. In the end, I have nine issues in total that cover different scenarios of user input. All of them are passed as expected, and I will create more when I encounter a new situation.

Here is the list of my test cover

- `formatted simple paragraph`
- `formatted two paragraphs`
- `formatted paragraph - dash list - paragraph`
- `formatted bullet list`

- formatted two bullet lists
- formatted ordered list
- formatted two ordered lists
- formatted complicated text
- formatted text contains special character < > &', ()

```
test('formatted ordered list', () => {
    /**
     * Ordered list is
     * 1. item
     * 2. item 2
     */
    match('<ol><li><p>order 1</p></li><li><p>order 2</p></li></ol>>', 'order 1 order 2')
})
```

Figure 8: Test case to format HTML string to string.

Thursday 18th January 2022

I continue working on the rich text, and today's goal is to manage the state to hide or show the rich text menu. In detail, some views have multiple rich text fields; it would not be nice if the button were present all over the UI. It needs to hide until the user clicks to editor and typing. Tiptap provides a focused property; its type is Boolean, which returns true if the user triggers the editor to be active and false if the editor is not activated. The only problem is that the expand button to open rich text in modal trigger focused is valid and does not toggle to false when closing the button. This makes users confused if the editor is active or not and how they can hide or show the menu bar. This is affected tremendously to user experiences. The reason is expanding button is now under the scope of `focused`; that why it triggers a `focused` state.

```
<tiptap-menu
    v-slot="{ commands, isActive }"
    :editor="editor"
>
    <ai-controlbar class="menubar">
        <a-button
            icon="fa fa-bold"
            :class="{ 'is-active': isActive.bold() }"
            :title="$t('components.rich-text.bold.tooltip')"
            @click="commands.bold"
        >
        </a-button>

        <a-button
            icon="fa fa-italic"
            :class="{ 'is-active': isActive.italic() }"
            :title="$t('components.rich-text.italic.tooltip')"
            @click="commands.italic"
        >
        </a-button>

        <a-button
            icon="fa fa-underline"
            :class="{ 'is-active': isActive.underline() }"
            :title="$t('components.rich-text.underline.tooltip')"
            @click="commands.underline"
        >
        </a-button>

        <a-button
            icon="fa fa-list"
            :class="{ 'is-active': isActive.bullet_list() }"
            :title="$t('components.rich-text.bullet-list.tooltip')"
            @click="commands.bullet_list"
        >
        </a-button>

        <a-button
            icon="fa fa-list-ol"
            :class="{ 'is-active': isActive.ordered_list() }"
            :title="$t('components.rich-text.numbered-list.tooltip')"
            @click="commands.ordered_list"
        >
        </a-button>
        <a-button
            v-if="canOpenInModal"
            icon="fa fa-expand"
            :title="$t('components.rich-text.enlarge-to-modal.tooltip')"
            @click="handleOpenModal"
        >
            <span>{{ $t('components.rich-text.enlarge-to-modal.title') }}</span>
        </a-button>
    </ai-controlbar>
</tiptap-menu>
```

Figure 9: Demonstrate button enlarge in modal is under the tiptap-menu.

As we can see from the code snippet, the last button opens the rich text in modal. It is now placed under the tiptap menu; it can trigger the focused, leading to a bug. First, move the button outside and put it in the top corner right. Now, I can open and close the modal without trigger focus. Next, I write Sass for styling. Rich text appearance should behave exactly like this:

- Only show editor area and the menu bar is hidden when the editor is not active, or users are not typing.
- A button that allows the user to open rich text in modal should appear when the editor hovers.
- When modal open, `focused` should not be triggered

- When close modal, menu bar is hidden, and editor loses its focus.

I manage to achieve today's goal and meet all new rich text requirements.

Friday 18th January 2022

I have a sync meeting with my colleagues related to the tooltip and rich text as these topics interact. I will continue to work if there is any problem. Otherwise, I can review some requests or something else. We also discuss building and releasing the layout package in the Azure pipeline and install in the dev environment. I created two beta versions and saw my new feature available in the dev environment.

Week 2 analysis

This week I have been working with Rich Text. We need to use a third-party library TipTap, leading to extensive work scope with different problems. I have developed my skills in researching and reading documentation when I implement this library. The situation now is that Tip tap is no longer extended support version 1 we are using (TipTap, 2022), so I need to read the codebase to understand how it works behind the scenes and find a way to modify it to fit our problem.

I have many meetings during the week to clarify the proper way to execute and progress the task. Discussions are around expectations to optimize user experience and user interface. I have tried to do it multiple ways and finally compare the output to choose the best solutions. There are obstacles during the implementation; one of them is very interesting: sanitizing user input.

What is sanitizing user input, and why is it necessary? Input sanitization is a cybersecurity solution that comprises inspecting, cleaning, and filtering data inputs from users, APIs, and online services for unwanted characters and strings to prevent hazardous code from entering the system. Hackers utilize remote file inclusion (RFI) and injection attacks such as SQL injection (SQLi) and cross-site scripting to exploit the gap in the interface between the website and the server (XSS). They can encrypt special characters and other illegal behaviors that put security at risk. Input sanitization can help prevent these sorts of attacks. Unknown sources submit questions and requests to an application, putting the system at risk of harmful assaults. By deleting potentially dangerous characters, input sanitization ensures that supplied data complies with subsystem and security standards. (What is Input Sanitization? | Webopedia, 2022)

We should never trust user input as they are all dirty and need to be sanitized in both the front and back end for security reasons.
Hackers can update, modify, and manipulate the input supplied from the user's browser via GET, POST, and cookies to get access to the webserver.
Input data sanitization functions as a filter, screening encoded data passing through the webserver. (Edmunds, 2016).

In my situation, I need to validate and sanitize all user input through rich text. The current codebase has a helper function called `cleanWordPasteHTML.ts` explicit the benefit of library `sanitizeHtml` to ensure all user input, including data they type and paste from others, is sanitized not used harmful to the system.

`sanitizeHtml` is a straightforward HTML sanitizer program with a basic API. It's perfect for cleaning up HTML fragments leftover from rich text editors. Copying and pasting from Word is extremely useful for deleting unwanted CSS. We can use sanitize-html to specify which tags to allow and which attributes allow for each. The contents of a tag that is not permitted are not discarded. There are several exceptions to this.

Here is the list of what `sanitizeHtml` can help with.

- The syntax of `p` and `img` elements that are not adequately closed is cleaned up.
- The `href` property is checked to ensure it only contains `http, https, ftp`, and mail to URLs. It's also possible to use relative URLs.
- The same goes for the `src` attributes. They filter hostnames to allow specific urls as a source to an iframe tag.
- The HTML comments aren't saved.
- Sanitize-html also escapes ALL text content, which implies that ampersands, greater-than, and less-than signs are changed to HTML character references (& --> &, -->, and so on).
- Additionally, quotation marks are escaped in attribute values (" --> "). (sanitize-html, 2022).

The function `cleanWordPasteHTML.ts` is the place we use `sanitizeHtml.` It will initially contain a particular set of tags before removing all undesirable characters from user input, such as excess white spaces, tabs, and tags not on the list. Sanitizing also rejects incorrect data requests and eliminates inputs to avoid them from being misinterpreted as codes. After all, it'll provide you with clean data to render and save.

```typescript
allowedTags: [
    'b', // Bold
    'strong', // Bold
    'i', // Italics
    'em', // Italics
    'u', // Underline
    'p', // Paragraph
    'ul', // Bullet list
    'ol', // Ordered list
    'li', // List item
    'a-ol', // Aibidia XML orderer list
    'a-ul', // Aibidia XML bullet list
],
```

Figure 10: Allowed tags from user input

It is essential to create several unit tests to ensure the function work ideally. My colleague handles that task, and I also learn from reading his code lines. We compose different scenarios by ourselves, but most of them are from real user input, incidents, and a series of XXS attacks that can be used to get around some XSS defences from OWASP's cheat sheet (XSS Filter Evasion - OWASP Cheat Sheet Series, 2022).

The whole process of building Rich Text is quite exemplary, except that documentation for TipTap version 1 is not supported and updated. I am discussing this with my colleague, and we are considering upgrading to version 2 as the current TipTap is relatively small. We can prevent the technical debt when the TipTap becomes bigger by implementing that plan as soon as possible.

3.3 Observation week 3 (January 24- January 28, 2022)

Monday 24th January 2022

Last week I was working on the Rich Text editor. I created and managed to toggle the menu bar and enlarge the button to open rich text, edit the content, and send the context to the user. However, I observed that the corresponding data is not changed when I change to a different legal entity. It is a bug that I need to work on today.

At first glance, I think the reason is that the request does not return the correct data for rendering. I would need to check the function to handle demand in solution requests. But when I do debug, everything seems to work just fine. Therefore, the problem is from Aibidia Layout in the rich text editor. I continued investigating and discovered that the condition to render the data was wrong, so it never behaved as expected. The debugging consumes a day of work.

Tuesday 25th January 2022

I will continue fixing the bug that I investigate yesterday. With the help from a senior in the team, I find out the solution which is demonstrated the diagram bellow.

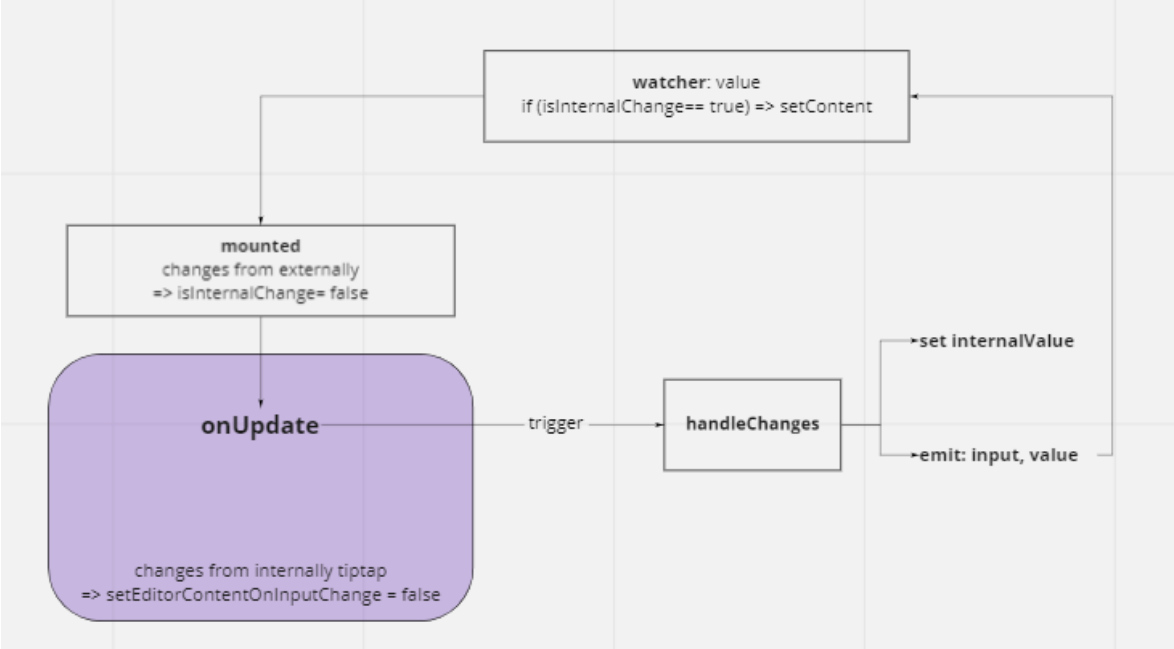

Figure 11: Solutions to fix internal changes.

Overall, the main problem is that the current mechanism does not recognize when changes are internal, which is user input from the editor or externally from GET requests. The essential change is inserting an additional data watcher called `isInternalChange`, which returns Boolean values. It will evaluate if the changes are from internal

`isInternalChange` will return true, which will trigger `setContent` methods to update the user input without sending request; otherwise, it will return false and accept change from GET requests.

Wednesday 26th January 2022

I first resolve comments from the tester and other developers on my PR on fixing the z-index. That takes a half day. Z-index does not need to be dynamic CSS variables since the future theme would not want to override these. Plus, having these also pollutes the main variable space. As a result, static SCSS variables are enough.

In the afternoon, I continue to implement the solution to fixing yesterday's bug. At the end of the day, I manage to finish it.

Thursday 27th January 2022

There is another problem that needs to be investigated related to Rich Text. This is not considering a bug but rather an unnecessary feature as it keeps sending the requests. In detail, we consider sending a request in only two cases. First, users click outside the editor when they are not in the modal. Second, when they press save inside the modal.

However, the TipTap sends requests every time user chooses format options. The detail is demonstrated in the flow below.

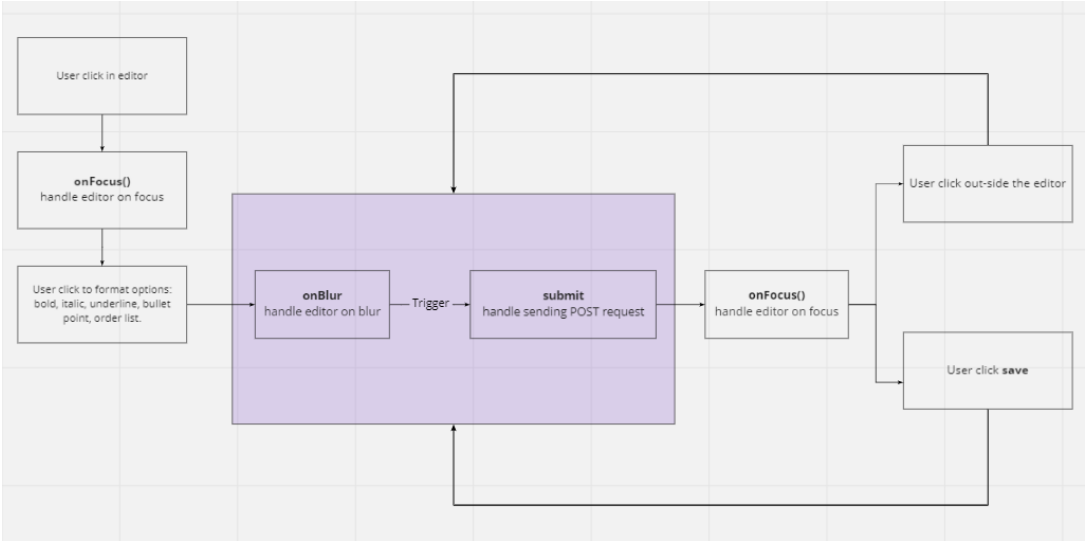

Figure 12: Tiptap sending unnecessary request.

It is seen from the figure that when the user clicks to format options: bold, italic, underline, bullet point, order list, it triggers onBlur and then autofocuses again. However, it should keep focusing without blurring and sending the submission. The fact that it sends the

request doesn't affect the user interface but causes an unoptimized application's performance.

My goal now is to find a solution to stop this and improve web performance. The first idea is to take advantage of the event property to see if I can set a condition to only handle blur when the related target is either outside of the editor or the inside the modal. This approach works well when the user clicks outside the editor, leading to nullifying the related target. Nevertheless, things get complicated in the other situation as the corresponding target is located deeply in another component. This solution does not work, and I need to find another way.

Friday 28th January 2022

Today's goals are to find a way to work around the current situation in which TipTap handles unnecessary requests.

My second attempt is to detect when the user clicks the formatting button, or inside the editor, it will not send the request; otherwise, it will send the request as usual. I manage to do it now by combining the function v-click-away to handle user click outside (vue-clickaway, 2022) with mouse-enter and mouse-leave to detect when the user is inside the editor area. While v-enter starting state for entering and added before an element is inserted, removed one frame after an element is inserted, v -leave starting state for leave, added immediately when a leaving transition is triggered, removed after one frame. (Enter/Leave & List Transitions — Vue.js, 2022)

Everything works just fine; my supervisor says the code is working but is not happy as the root problem is not solved. I explain that is the best solution as the TipTap version 1 has the problem. The issue will go in its recent release, which means if the team decides to upgrade TipTap to a newer version, that problem will be fixed. We decided to leave the code with unnecessary sending requests as it does not affect the user interface.

I made some minor changes in the modal after hearing feedback from the UX-UI designer.

- Rich text editor should have min-height if the content is empty and max-height in case content is long.
- When the user opens the modal, the format button should always appear.
- The output should return HTML strings wrapped in a single root tag <article>

Week 3 analysis

This week was comprised of small independent tasks related to rich text. I have a chance to work with people from the backend to sanitize user input and process current data that is not formatted in the new HTML string. Most of the tasks deal with existing code to add more functionality. Starting to dive deeper into components in Vue is also one of the exciting and essential tasks of this week. I learn about global and local components to pass props and manipulate the logic due to the reactivity of Vue (Component Registration — Vue.js, 2022).

According to Bo Andersen, a global component can be used everywhere in an application, including within other components. On the other hand, a local component is not registered globally and may thus only be used on parts where it is written. (Andersen, 2022).
There is a difference between my pet project and the company project. I prefer to register a global component in my project, while globally registration is rarely used in the company. Globally registering all components may cause an unnecessary amount of JavaScript code that users have to download since those files are still included in the final build even when components are not being used. In addition, for some components, it does not make sense to have them registered globally. Therefore, global registration is often not ideal when our project uses Webpack to build a system. The good practice is defining the components as JavaScript objects and registering them when we need to use them.

Global and local components are essential topics. There are some notes related to component names and slots that I took during the work in Abidia based on the literature. First, I was perplexed by component names in Vue primarily compared to naming in React since sometimes I see people use camelCase, and some other times, people use kebab-case. Vue's official website clearly explains how to name a component correctly. When using a component directly in the DOM (rather than in a string template or single-file component), the component's name should be in all-lowercase and contain a hyphen that avoids clashes with existing and future HTML elements. (Component Registration — Vue.js, 2022). Similar logic applied with props, props in the component is camel cased, but when using in-DOM templates, it needs to use its kebab cased. Two snippets of code are the real example of either using kebab or camel case. Props `canOpenInModal` is defined in `a-rich-text-editor` and passed to other component `a-rich-text-fields`. The code does not make more sense as I could not reveal all the context and components.

```
<a-rich-text-editor
  :value="internalValue"
  :can-open-in-modal="canOpenInModal"
>
</a-rich-text-editor>
```

Figure 133: Props canOpenInModal in camel case in a-rich-text-editor.

```
<a-rich-text-field
  v-model="internalValue"
  no-label-element
  :can-open-in-modal="false"
>
</a-rich-text-field>
```

Figure 14: The component a-rich-text-fields use can open in model in kebab case

Regarding slots in Vue, we can use slot props to make slots into reusable templates that can render varied content depending on input properties. This is especially handy for creating a reusable component that encapsulates data functionality while still allowing the consuming parent component to alter a portion of its look. (Slots — Vue.js, 2022).
I had faced the concept of slot scope when I implemented the function to add a button open rich text in modal before. I reencounter it in executing TipTap.

TypeScript Copy Caption

```
<template>
  <editor-menu-bar :editor="editor" v-slot="{ commands, isActive }">
    <div>
      <button :class="{ 'is-active': isActive.bold() }" @click="commands.bold">
        Bold
      </button>
      <button
        :class="{ 'is-active': isActive.heading({ level: 2 }) }"
        @click="commands.heading({ level: 2 })"
      >
        H2
      </button>
    </div>
  </editor-menu-bar>
</template>
```

Figure 14: TipTap use v-slot in renderless component. (Source: Files · TipTap - commands@1.10.5 · Adam Gerard / tiptap, 2022)

`<editor-menu-bar />` is a component defined behind the scenes of library tip-tap, which process different variables stored in an object. When users use this component, they can destructure properties inside: `commands, isActive, editable, ect`. I explicit the power of using a third-party library, so I did not dig into how commands and `isActive` value in tip-tap work. I studied the documentation to learn how to achieve the result without create all functions from scratch.

Table 1: Properties of <editor-menu-bar /> (A render less rich-text editor for Vue.js | BestofVue, 2022)

Property	Type	Description
commands	Array	A list of all commands.
isActive	Object	An object of functions to check if your selected text is a node or mark. `isActivenode
getMarkAttrs	Function	A function to get all mark attributes of your selection.
getNodeAttrs	Function	A function to get all node attributes of your selection.
focused	Boolean	Whether the editor is focused.
focus	Function	A function to focus the editor.

3.4 Observation week 4 (January 31- February 4, 2022)

A break week due to travel.

3.5 Observation week 5 (February 7- February 11, 2022)

Monday 7th February 2022

Today, I was catching up with the work after one week of vacation.
I checked the PR and saw that my co-worker helped me finish the remaining work of rich text while I was on vacation. I watch the sprint planning and retrospective record of catching up with the current workflow. I have asked my team lead earlier this morning to understand my task in this Sprint.

I have been working on the state in Rich Text Editor and having discussed with my college to Manage to keep the cursor still when new changes are from editing changes internally and handle new changes when they have come externally. More detail on figure 12. This was the final touch on improving rich text to minimize the POST action when formatting the text. After those significant changes are merged, other front-end developers and I move to the next stage, testing and finding bugs on this new feature.

Tuesday 8th February 2022

Today, I resolved the comment from the tester on PR in the rich text. I made changes based on this suggestion, including:

- Write a readable and clear description for the PR.
- Add JSDoc for variables, functions, methods, components, etc.
- Make sure all test cases are passed.
- Recheck the logic and ensure it works as expected.
- Including additional changes outside the scope of this PR: Fix an issue with conflicting tip-tap-content value changes with external value changes. External changes to value prop were not updating the internal TipTap state.

The PR is merged into the main branch at the end of the day.

Wednesday 9th February 2022

Today, I am writing a bug report for rich text. Here is the list of bugs we have encountered and collected when users copy and paste from MS Word to Aibidia's rich text systems:

1. Rich text content render incorrectly nested list: Create the nested list in MS Word, and after pasting it to rich text, the format for the nested list is gone.
2. Rich text is rendered incorrectly in cells with no <p> tags in the text.
3. Rich text is sending invalid tag: Convert RichTextEditor linebreak tags to be self-closing.

4. The rich text adds extra whitespace: When pasting from Word two or more paragraphs with an empty line between them, the editor adds whitespace on the empty line.
5. Rich text editor modal updates wrong row when sorting/filtering is enabled.
6. Resizing of bullet number - Rich text editor
7. Copying any plain string to a rich text table cell will throw a validation error.

I take responsibility for fixing bug numbers 2, 3, and 4. Those bugs are minor, and the effort to fix them is low; I have a good feeling that I could finish it within a day.

I also upgrade my environment. There was a problem similar to my previous problem when I upgraded to npm version 8.

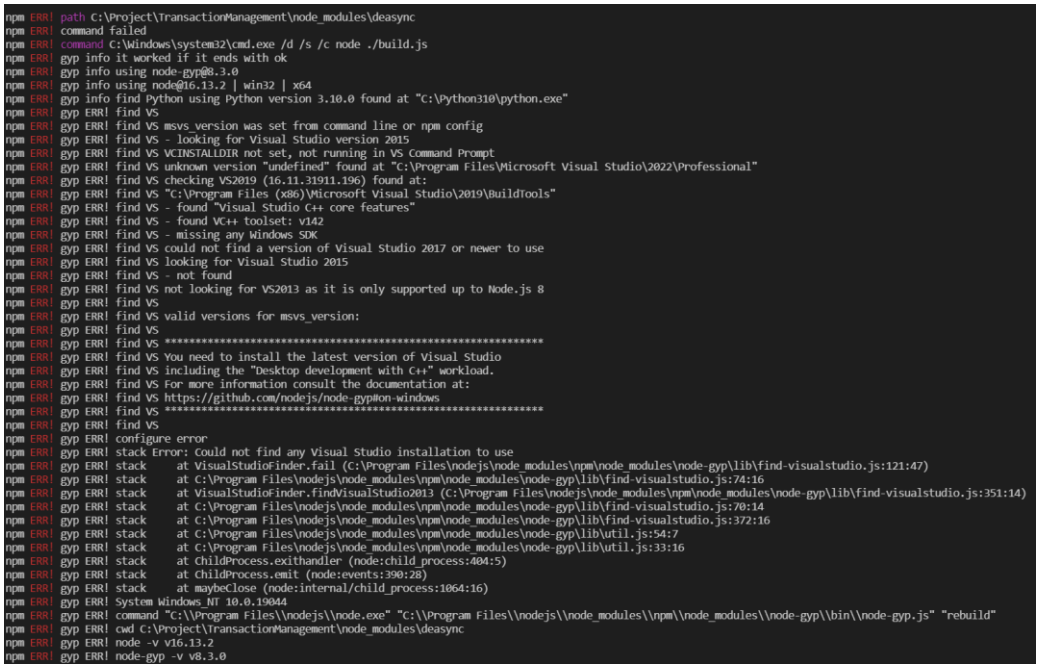

Figure 15: Screen shot of error when install node.

From the log, I understand that using a version of Node that does not have binaries for desync on NPM, node-gyp tries to build it, and the build fails because it depends on Python, so I reinstalled Node with chocolatey. Chocolatey packages encapsulate everything required to install Node (Chocolatey - The package manager for Windows, 2022). The log says that I am missing Python, but Python 3 was installed as part of chocolatey build tools (Python + VS build tools), so I will install Python again. Finally, I got it to work by removing the package.lock.

Thursday 10th February 2022

I have continued working on the tickets that I took yesterday. I will go through each ticket and explain my solution respectively.

First, the bug related to the rich text is rendered incorrectly in cells with no <p> tags in the text. From the figure below, if the content is empty, nothing will be wrapped around by the <p> tag. In addition, all tags need to be closed as API uses XML parsing to validate the rich-text content, and validation will fail if there are missing closing tags.

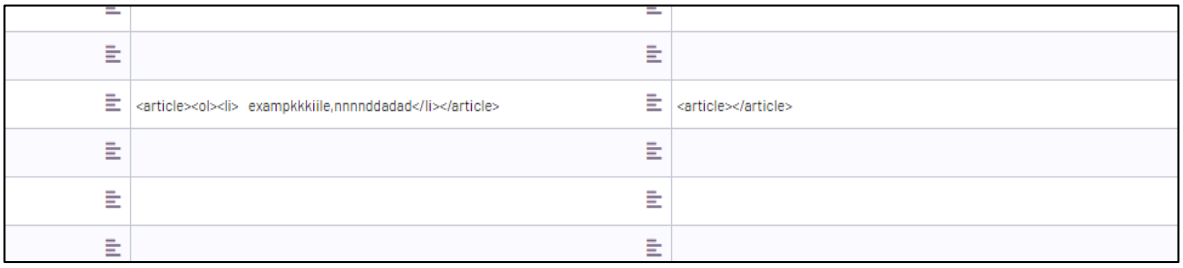

Figure 16: Rich text is rendered incorrectly in cells where there is no `<p>` tags in the text.

This problem will be easier to fix from backend. Therefore, I contact with backend developer to add a function to validate the content of rich text, it should convert all values into HTML even the empty strings.

With the second bug, when the user creates a line break on a `rich-text field` (using Shift+Enter) and tries to save the content, an "Unexpected error during validation" error is shown. The reason is that the `RichTextEditor` (and underlying Tiptap) use `<br>` to indicate a line break in the text content. Using `<br>` is valid for HTML, but not for XHTML (or XML) which requires closing tags. API uses XML parsing to validate the rich-text content, and validation will fail if there are missing closing tags. Solution is to convert
 tags to self-closing `<br/>` tags in `RichTextEditor` which ensures that the content is valid XHTML.

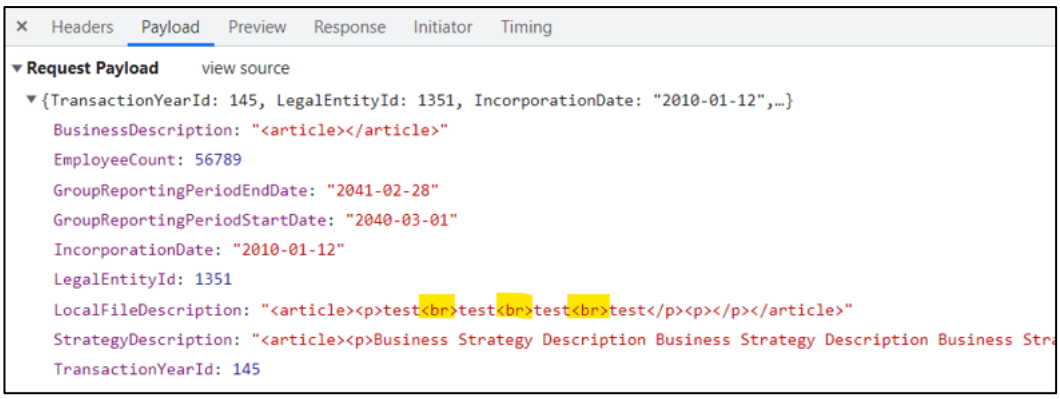

Figure 17: Request sending invalid
 tag.

The last error happened when pasting from MS Word two or more paragraphs with empty lines between. The editor adds whitespace on the empty line, which produces a validation error 'Unexpected error during validation.' After investigating, I understand that the request replace the new line `<p><p> by <p> </p>.` ` ` is short for non-breaking space that does not break into a new line. A non-breaking gap between two words will cause them to stick together (not break into a new line). When breaking the words might be distracting, this is useful. (HTML Entities, 2022). To solve this problem, I replace ` ` with an empty string using regex. I also write several unit tests for the new changes.

Friday 11th February 2022

When the new tooltip mechanism is merged into the current system, it replaces the old one. In general, the upgrade version of the tooltip is easier to use and improves the user experiences and developer experiences. However, it doesn't work as expected with the rich text system I have developed. The problem is that rich text fields include HTML tags in the tooltip form.

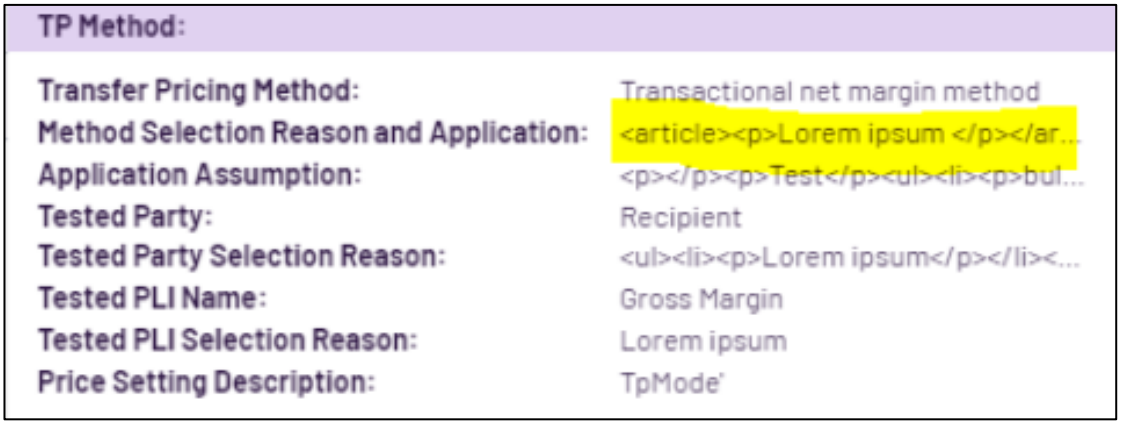

Figure 18: Tool tip show rich text content with HTML tags.

Further analysis showed that there is no function to format the HTML string of rich text. There are several methods to solve this. My approach is to add a new `displayTextValue` function to `baseField` which uses `formatHTMLbeforeRender` to convert HTML content to plain text.

Plus, when users hover to the content to see more, rich text fields also include HTML tags since the tooltip shows the default content of the p tag. I reuse `displayTextValue` to render the value of the field and its title in `LabelField`.

At the end of the day, I also review PR, fixing bugs that can not copy paste from rich text cells to normal cells from others.

Week 5 analysis

My abilities improved dramatically throughout the week. I became more acquainted with the product, better understood what I was doing, learned new tools, and gained knowledge of JavaScript or Git. It has been a productive week for me, and I am pleased with my performance and learning development. This week, I discovered a unique application that helps me better my Git experience while saving time at work.

Git is a popular piece of software that helps developers and engineers to keep track of file changes. It's usually used to coordinate work among programmers working on the same source code during software development. Git's notion is what I find most appealing. Git is a free and open-source distributed version control system that can easily handle everything from small to large projects to give you a quick overview (Git, 2022). Git is easy to learn and use, with a minimal footprint and lightning-fast speed. It beats SCM systems like Subversion, CVS, Perforce, and ClearCase with inexpensive local branching, accessible staging areas, and multiple processes. (Git, 2022). I first started learning and using Git when I did my first project to build my portfolio. At that time, I used my personal computer: a Macbook Pro 2015 with the operating system Mac OS. Using the terminal with developers and Mac users is very fast, effective, and convenient. I was doing the project alone, so I did not use anything so advanced. There is the list of command-line that I used the most.

- git clone: This command downloads the most recent version of a remote project and copies it to a specified location on your local machine.
- git fetch: This Git command will get all updates, including new branches, from the remote repository.
- git checkout: You may use the checkout command to change the branch you're working on right now.
- git init: Use this command if you wish to establish a new empty repository or reinitialize an existing one in the project root. It will generate a git directory and subdirectories inside it.
- git commit: This is the most used Git command. After making changes locally, you may "commit" them to save them. A commit is basically a local snapshot of the branch's present state, to which you can always return.
- git push: Git push is a command that pushes locally committed changes to a remote branch.

- git pull: You may pull all of the changes from the remote repository and merge them into the current local branch using git pull.
- git add: This is the command you'll need to use to stage files that have changed.
- git branch: Using git branch will display all the repository's branches.

(10 Common Git Commands Everyone Should Know - TestProject, 2022)

My experience using Git is limited as I do not have an opportunity to practice, and Git shows its perk when many different developers join the same project. When I joined Aibidia, my limited knowledge helped me understand the general idea, but I needed to learn more. For example, when I want to commit my code, I first need to use `Git add .` to add all the changes in the working directory to the staging area. I did not know that `Git add .` is not recommended as it will accidentally bring new files and modifications to stages without deleted files. Overall, Git is simple and complicated simultaneously, but it is an essential skill to become a good programmer.

An optimal solution, in this case, is using Git GUI, which allows programmers to communicate with Git using a graphical user interface rather than the command line or a web browser. My team leader suggests that I use the Git Hub desktop. GitHub Desktop is a program that helps teams work together using Git best practices. Most Git operations may be completed from my desktop with visual confirmation of GitHub Desktop changes. I can push to, pull from, and clone remote repositories with GitHub Desktop. (GitHub Desktop, 2022)

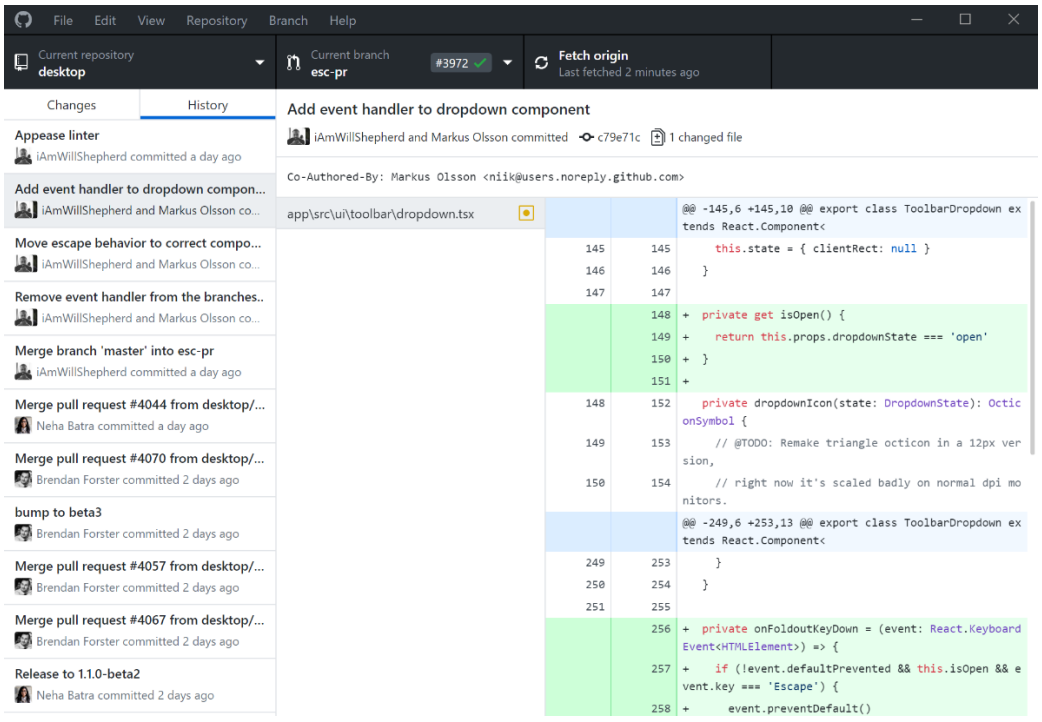

Figure 19: GitHub Desktop screenshot. (GitHub Desktop, 2022)

3.6 Observation week 6 (February 14- February 19, 2022)

Monday 14th February 2022

I fix the comment on my pull requests. Discuss with others about handling encoded HTML characters in rich text. The situation now is interaction with MS Word is okay. But with other editor software, for example, Google Docs, there are some problems. I quickly investigate to observe that rich text behaves differently in different web browsers. I report those behaviors.

When copying text from Google doc to Rich text, if the browser is:

- Chrome: GDoc does bring italic and bold over
- Firefox: GDoc does not bring italic and bold over
- Edge: GDoc does bring italic and bold over
- Safari: GDoc does bring italic and bold over

So rich text behaves the same in Chrome, Safari and Edge. The firebox is an exceptional case. We decided to report the bug but not work on that until the customer required us to do so.

Tuesday 15th February 2022

I work on improving user experiences when interacting with the date picker form. It is difficult/unclear for a user to select the year 1990 for the Legal Entity Incorporation date, as the date picker only displays years 2012 to 2032 (±10 years from the currently selected year) in the dropdown. I have to click the year, the select year 2002 first, then 1992 and then 1990 to set the year.

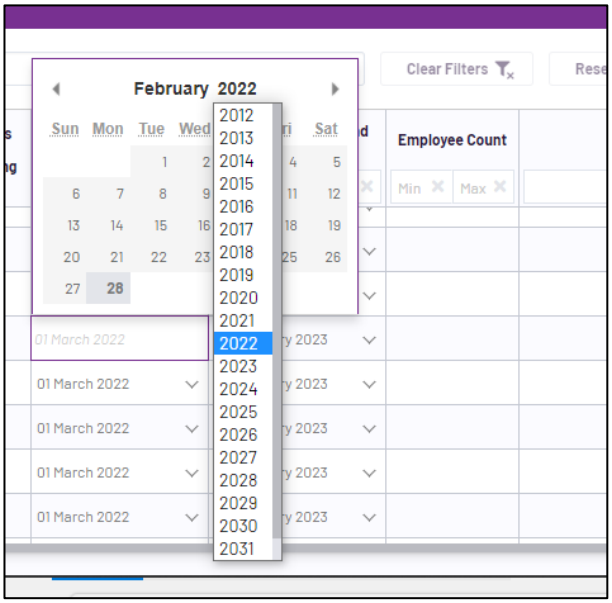

Figure 20: Date picker before applying UX improvements

The changes will generally improve the UX when the year range is now ±50 years from the selected year, so users can easily choose a year. I also insert arrows beside Month and Year Title to indicate that the user can open the dropdown. The background of the Month and Year Title is changed when hovering.

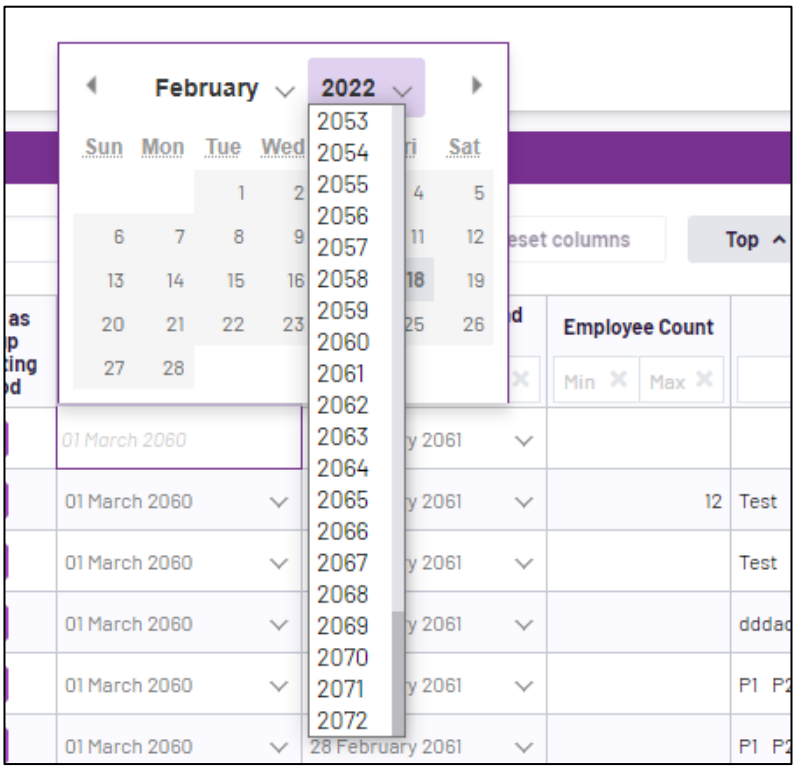

Figure 21: Date picker after applying UX improvements

Wednesday 16th February 2022

Today I fixed a bug in CBC Fix showing warning text in Summary views. I first reproduce the bug in the test environment.

- Go to Entity Data > Summary
- Try to delete a mandatory field such as City from an address

Notice that "Table has missing or invalid data" - warning appears and stays, even though City is not deleted since mandatory fields can't be deleted.

As usual, I try to debug and find out the root causing error. The reason is quite hidden, but it is easy to fix. The flow of logic is when `setActiveEntity`, `selectActiveEntityById` are executed, we need to execute `revalidateFields` which will report any existing error. The validation mechanism does not trigger to return `isValid` equal true, which indicates that the field is mandatory and can not be deleted.

When identifying the cause, things get simpler to fix. All I need to do now is to call `revalidateFields` when `selectActiveEntityById` is executed. In addition, the

mandatory fields can't be deleted, so the validation should always trigger to return `true`. In `revalidateFields`, emit an event to return `isValid = true`.

Thursday 17th February 2022

Today, I am handed over a task from my colleague while he is still on vacation. That task is an enhanced user experience feature. As a user, on pages with entity progress table, I'm now confused when I filter the legal entity list as Ultimate Owner is always included in the search. Even if it doesn't contain the search string, the active entity on the right never changes when filtering the entity progress table on the left.

After filtering the progress table, behavioural changes are that Ultimate Owner is no longer shown if it is not part of the filtered result. If the active entity is not found from the list, the first item from the list is set as an active entity. Implementation changes Do not return `ultimateOwner` as the first list item in `filteredLegalEntities`(). Refactor overlay implementation to match implementation used elsewhere. If the `searchedLegalEntities` list is empty, show `overlayContent`: In case the list is empty because progress table filters show. There are no entities with this progress status. In case the list is empty because searched string shows No search results. Remove the functionality that skips the ultimate owner from `searchedLegalEntities`.

Friday 18th February 2022

I added minor changes to improve user experiences in the date picker form task. I created a here is a PR for that. All my work is currently merged or in the PR state. Today, I will review the pull request from other people.

Week 6 analysis

This week has gone so quickly and efficiently for me. After three months on the job, I'm more confident, practical, and involved in the team. This week, I'd like to talk about Azure DevOps. I have been working with it and enjoy the advantages it offers.

Microsoft Azure DevOps is a software-as-a-service platform that gives users a complete DevOps toolchain for creating and delivering applications (Azure DevOps, 2022). DevOps is a combination between software development (Dev) and IT operations (Ops). The goal is to provide a continual stream of improved software that is both quick and error-free. Developers utilize DevOps to shorten the development process. It's enough to make anyone's head spin with the amount of DevOps solutions accessible to app developers (Azure DevOps, 2022). At Abidia, we're using Microsoft Azure DevOps.

Azure DevOps is not a single software, but rather a collection of services (Azure DevOps documentation, 2022):

- Azure Boards: Agile planning, work item tracking, visualization, and reporting capabilities are all covered by
- Azure Pipelines: This CI/CD platform supports containers and Kubernetes and is language, platform, and cloud neutral.
- Azure Repos: This service offers a cloud-hosted private Git repository with pull requests, complicated file management, and other capabilities.
- Azure Artifacts: The components in question provide integrated package management for developers, with support for public and private Maven, npm, Python, and NuGet package feeds.
- Azure Test Plans is an integrated, all-in-one scheduled and exploratory testing solution from Microsoft.
  (Azure DevOps documentation, 2022)

Azure DevOps provides a wide range of capabilities for development teams due to its features:

- Dashboard Control: You can rapidly browse to different project regions, build and manage dashboards, and set dashboard widgets using the DevOps dashboard functionality.
- Source control has been improved: Git and Team Foundation Version Control (TFVC) are two well-known methods of source control supported by Azure DevOps solutions. You may add and manage Azure Git tags, review, download, and modify files to observe change history.
- Plan and Execution: You may utilize Azure DevOps systems to track features, requirements, user stories, tasks, issues, and more using a variety of work items. You may use a variety of backlogs and boards to support the critical agile approaches, such as Scrum, Scrum ban, and Kanban, for planning reasons. You can add and edit essential work items, manage the product backlog, schedule sprints with sprint backlogs, and visualize the process with Kanban boards.
- Continuous Integration and Deployment (CI/CD) is a method of integrating and deploying software: Many developers use CI/CD, and Azure DevOps makes it easy. Developers may automate various design activities using Azure pipelines, including defining builds and associated steps, writing test instructions, and managing simultaneous releases.
- Manual and exploratory testing are supported: Manual, exploratory, and continuous testing are made more accessible with Azure DevOps' test capabilities,

including workflow customization, end-to-end traceability, criteria-based selection, and real-time visualizations that show test activities.

- Services for Integrated Collaboration: The capability that allows teams to work across the whole set of Azure DevOps services and functions is called: o Dashboards for teams o Discussion inside work item forms o project wiki o Traceability is supported by linking work items, commits, pull requests, and other artifacts. The ability to seek and manage feedback o Analytics service, analytic views, and Power BI reporting o The ability to manage alerts and change notifications per user, team, project, or organization
- Cloud-hosted services on Azure: Azure delivers cloud-hosted services to DevOps teams to help with application development and deployment. These services can be utilized independently or in tandem with Azure DevOps.
  (Azure DevOps documentation, 2022)

3.7 Observation week 7 (February 21- February 25, 2022)

Monday 21st February 2022

Following a daily meeting with the development team, I have a conversation with the supervisor about which ticket I should pick up next because all my work is either merged or in the Pull Request state. There aren't many tickets remaining in the queue, but they're all blocked by other issues that other developers are working on. My college has returned from vacation, and I am the one who is responsible for completing his chores. We've received a request to sync new modifications.

In the meantime, a developer has just completed his ticket; therefore, my responsibility is to examine and test it. This issue has precise specifications, so I can quickly evaluate and test it using the testing instructions. Also, because this engineer is used to writing clean code, there is no need to rework his ticket. He's also meticulous when providing extra dummy data to support test cases and writing unit tests for each function he's built.

Tuesday 22nd February 2022

I switched to a new ticket today. It isn't an urgent bug, but it would be beneficial if we could fix it. I moved on to the lower-priority task as I can't start working on other higher-priority tickets until other incomplete tickets do not block them. I need to correct when the user date input is not in a proper format.

.

Different legal entities worldwide use our solution, so the date format is changed based on geographical location. For example, in the US, the date format is "mm-dd-yyyy," while in Europe, the date format is "YYYY-MM-DD." In our solution, the bug can be reproduced like this:

- Go to any table with a Date column
- Insert value a user might think is valid value: 01.01.1995.
- After submitting, notice the date is changed to the current date

The reason is simply that DateEditor only stores internally Date objects, so invalid data objects are converted to default new date objects (current date). My solution for this is:

- Change DateEditor to store the current value as a Date object or string if a date is invalid.
- Emit invalid date value as string from DateEditor so Table and Field can show proper validation error.
- Add Validator to DateField.
- Render invalid date values in the fast renderer so the user can see the error.

When I have a clear direction to solve this problem, I can implement it efficiently.

Wednesday 23rd February 2022

Today, we have a sprint review where all companies will join in demoing new features and improvements of a released application from developers. We conduct Q&A sessions to explain and discuss the products' road map. There are five presentations of the demo. I was impressed by my colleague's presentation; he showed a proof of concept of a bold idea that will be a game-changer compared to the similar application from our opponents. Proof of Concept is a generic way of confirming that a concept is realistic, viable, and valuable in practice by verifying a specific assumption. In other words, it demonstrates if the software product or a particular feature of it is appropriate for solving a specific business challenge. (Lvivity, 2022)

Next, we have 1.5 hours set up for a retro meeting with all the developers. The Sprint Retrospective occurs after the Sprint Review and before Sprint Planning. It is essentially an "improvement" meeting where everyone is present - the product owner, scrum master, and development team members– to discover ways to spot possible trap mistakes and devise new ways to prevent such problems.

I think a general feeling is that we have good momentum. We did pretty everything that we had planned. That's one of the things that we should keep doing in the future. I still wish for more collaborative work with others. I had some spare time when I finished all my work, so maybe we can estimate the work more accurately. Two new members will join the management team. They can help us achieve goals within constrained scope and time set, which task is priority and kinds of stuff.

Thursday 24th February 2022

I start my day with miscellaneous tasks as we will have a sprint planning meeting later today where we discuss sprint goals and allocate people to a suitable workstream. A sprint, according to Atlassian, is a set period during which all work in a scrum is done. However, it would help if you first prepared for the race before taking action. You must choose the duration of the time restriction, the sprint objective, and the starting point. The sprint planning session establishes the sprint's plan and focus. It also creates an environment where the team is motivated, challenged, and capable of achieving success if done correctly. Bad sprint planning can cause a team to become disorganized by establishing unrealistic expectations. (Sprint Planning | Atlassian, 2022).

Our sprint planning last 1,5 hours. I will work on the front-end task as usual. The workstream that I will join in this sprint is to develop the supply chain views and complete customer requests for adding their logo to our application.

Friday 25th February 2022

I do not yet work on new tasks; instead, I spend time on other urgent tasks fixing bugs. I work on bugs related to RichText HTML that are seen in different places in TP doc and resolve comments. I also quick checkout on my colleague branch: set an active entity after filtering.

I investigate why the tooltip is not shown in Transactional level, Profit level indicator and Arm's Length Validations views. I discussed that with a co-worker, and we found the solution. I worked on the task that one senior assigned me yesterday: On the Legal Entities Tab, all the below views should only list TP Documentation Entities in the entity progress table, but now those show all legal entities. It means that one can add information about TP documentation for entities that will not be documented.

- Management Structures
- Main Customers & Competitors
- Financial Accounts
- Main Product Categories
- Geographical Markets
- Contributions & Business Activities

I debug and find out the reason leading to this bug. The default value of `onlyTpEntities` is `true`, but it is overridden as `false` in those views. Remove setting the `onlyTpEntities` value to `false` in the mentioned views. By default, only the TP Documentation Entities will be shown, not all Legal Entities.

Week 7 analysis

This week has passed smoothly and productively for me. I am satisfied with the result of my work. However, I have to keep fixing a bug related to rich text even though I finished it quite before. The reason is that we depend too much on a not-good third-party library to sanitize user input. That library is missing a lot of cases to sanitize user input, causing a bug in backend validation.

I and a senior working on rich text discuss finding another good one to replace the existing library. We understand that it is painful to discard the whole existing code and replace a new one, but it is essential to avoid technical debt later.

Don't reinvent the wheel has been one of the software engineering concepts. You should avoid wasting time and effort on problems that others have already handled. The most typical approach for a developer to put this notion into effect is to use third-party libraries. However, its application is a contentious topic in the IT world. Senior developers who are suspicious of them and refuse to work with them are not uncommon (Backes, Bugiel and Derr, 2016, 356-367). As a start-up, we prioritize development speed and sustainable development in our company situation. We combine using the library and building our library to get more control. Aibidia Layout is a highlight example of our custom UI library. It gives us much power to decide how to style solutions applications. We also use external libraries; while they offer advantages, it also brings disadvantages.

The most prominent benefit of using third-party libraries is that you save time by not having to create the library's functionality but focusing on the most critical aspects of your app's business logic. Of course, reading the library's documentation and learning how to use it will take some time, but it will be a fraction of the time to create and test its functionality from scratch (Pozrikidis, 2002). At Aibidia, we employ the power of external libraries to reduce the work for a developer. For instance, we use the pikaday library to handle all UI and logic of toggling the dropdown for the user to choose date and time while manipulating the user input in store.

Another benefit of adopting a library comes when we need 'proof of concept.' The library's user community will utilize it in various settings and scenarios to continuously report and fix bugs due to the feedback loop. As a result, employing a well-known third-party library increases the speed of development and should not jeopardize your application's overall quality and stability. (Pozrikidis, 2002).

Working with third-party libraries has another advantage: it pushes you to work with and build modular code. The library's code is kept distinct from the rest of your application's code and communication through well-defined boundaries (the library's API). If you want to build a feature yourself, it may not be easy to separate it from the app's code. A well-written library's writers prefer to work at a higher abstraction level, resulting in cleaner, more general code. Unfortunately, decoupled code isn't necessarily consistent with modular code. It leads us to the drawbacks of using third-party libraries. (Pozrikidis, 2002).

To begin with, using a third-party library binds your code to that library. (Brown, Malveau, McCormick, 1998). If you are compelled to switch libraries at some time, your code may need to undergo considerable modifications to adapt to the new library. Another possibility is that the author of the library will forsake it. The worst case is that the library will not be

supported and fix bugs. In Aibidia, we faced that situation once when RichText editor Tip tap stopped support for our version that we are using, causing the pain point of finding documentation while implementing new features. An excessive number of libraries may bloat your software, making it larger and consuming more memory. Your app's performance will suffer as a result. Last but not least, the usage of third-party libraries may present security risks since hackers are increasingly targeting weaknesses in open-source libraries. (Brown, Malveau, McCormick, 1998).

3.8 Observation week 8 (February 28- March 04, 2022)

Monday 28th February 2022

In this sprint, I work on the TP Doc Supply Chains update. There are several work items under this epic in both the front-end and backend. The current supply chain has only two tabs, one for add and display supply chains; the other is called market description.

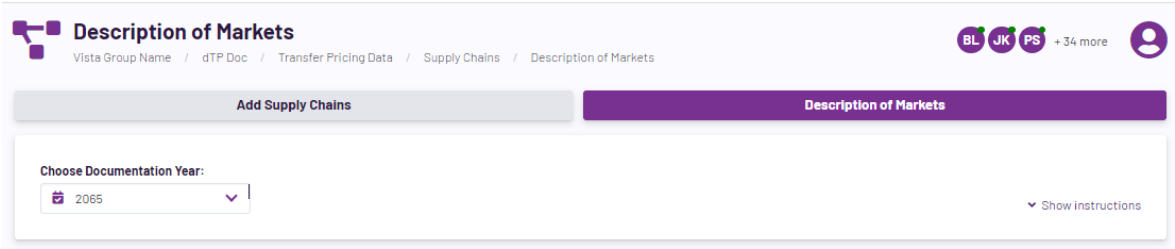

Figure 22: Two tabs of supply chains

At this point, we found out that the view of the market description is not helpful for customers. That is why we remove it from the UI but keep it under the hood to effortlessly revert if we want to use it in the future.

Another task under TP Doc Supply Chains update is to rename Supply Chains to Transactional Supply Chains. When we create a new feature, we always come from the user story to define if that feature is needed. Therefore, as a user, I want to work with Transactional Supply Chains, so I don't confuse them with Supply Chains (previous tab). Plus, I want to select Supply Chain from the dropdown and wish to remember my Supply Chain selection if I later come back to the view.

We clearly define the acceptance criteria and expectations for this task:

- SupplyChains View and Editor is renamed, and Localization tokens updated.
- Select Field is added to page filters
    - If no supply chain is selected, and there is supply chains. first supply chain is selected automatically.
    - Select field remembers active supply chain when navigating away from/back to the view.
- Definition Abbreviation column is moved after Reporting Level.
- Overlays cleaned.
    - Overlay should direct user to add supply chain if there are no supply chains to select.
    - The SupplyChainTransaction editor overlay should not direct user to rename SupplyChains.

- Possibility to create, navigate (select or prev/next), rename or delete **is removed** from SupplyChains editor.

I work on this during the first day of the week.

Tuesday 1st March 2022

I work on the supply chain: rename Supply Chains to Transactional Supply Chain. Most of the requirements were implemented, so I created a draft PR. I still need to save the supply chain to the local storage to remember the supply chain when users navigate away or back and forth.

I also resolved the PR comments about refactoring the hot setting store. Today, I continue to work on the supply chain, comment on PRs and resolve errors. Later in the afternoon, when everything is done, I start to move on to the next task in the Supply chain update epic: improve the UX in the transactional supply chain. Similarly, we use the user story technique to approach this problem.

- As a user, I want to see Reporting Level content as a tooltip on the transaction's editor
- As a user, I want to see the pointer cursor when selecting transactions to supply chains
- As a user, I want to see transaction Definition Abbreviation easier
- As a user, I want to see counter how many transactions there are
- As a user, I want to see counter how many supply chains there are
- As a user, I want to see all Configurable Columns in the Related Transactions Modal

The expectations corresponding are

- Reporting level shows the relevant tooltip
- Definition Abbreviation column is moved right side of Reporting Level
- When hovering over transaction rows, the cursor is a "pointer."
- Transactions Counters are added
- Supply chains Counters are added
- Column configurator in Related Transactions Modal shows all configurable columns

Wednesday 2nd March 2022

In the morning, I work on the task of improving user experience on transactional supply chain user experiences.

The transactional supply chain does not exist in the current system but was created on my rename branch, so to improve the user experience task, I need to create a new branch

based on the branch rename transactional supply chain. This first immensely confused me, so I asked the gatekeeper about this situation. His explanation is if I develop a branch based on another branch, I do everything as usual, except when I create a pull request, the target branch is the parent one, not the main branch.

Other than that, the work is progressing quite well. However, the scope of adding tooltips for reporting level column has stretched beyond the supply chain. So, in the sync meeting with other people who also work on the supply chain in the sprint, I spread this issue, and we finally decided to separate it into a different work item. I am happy with is action point because it makes my current work leaner and cleaner.

I will continue to work on that and finish it at the end of today. We also conduct a roadmap meeting at noon.

Thursday 3rd March 2022

I finish the user experiences task, and currently, it is in a PR state, waiting for review from others. I also resolve team members' comments and then review the pull request assigned.

In addition, I work on the task of adding tooltips. I have some problems defining which level a transaction is currently on. There are three different levels, including Transactional, Tp Model and Tp Function. I asked for help on the channel dev-front in Slack; people who have more experience with this situation actively give comments and suggestions.

From the open discussion, I find out that the transaction mapper is the place making the decision.

Figure 23: Transaction Mapper decide reporting level of a transaction.

A transaction can be either TpModel if it contains TpModelId, or TpFunction if it has RecipientTpFunctionId or DelivererTpFunctionId. And if a transaction doesn't include those IDs, it should be at the Transactional level.

I was suggested that I should bring the transaction mapper's logic to the store where the tooltip is implemented to create a generic solution. Reporting level columns are applied in many places in the solution so a generic setup will be helpful in this case.

Friday 4th March 2022

I was working on adding reporting level tooltip. It now shows whenever it has reporting level columns. There is some issue when row id is not in number format in PLI Values & Financial Data views. I have a call with my colleague to discuss this. We agree that the reporting tooltip should detect if it is in the PLI Values & Financial Data view or not. But overall, it is now working as expected. I create a draft PR for that and will finish the remaining work later.

Week 8 analysis

My abilities improved dramatically throughout the week. I became more acquainted with the product, worked closely with other developers, and became acquainted with the users. This week, I mainly focused on improving the user experience. It has been a productive week for me, and I am pleased with my performance and learning development.

The term "user experience" relates to a person's sentiments and attitudes about a product, system, or service. It encompasses all aspects of human-computer interaction and product ownership that are practical, experiential, emotional, meaningful, and valuable. "A user's view and feedback of a product, system, or service," says Wikipedia. All user emotions, beliefs, preferences, perceptions, physical and psychological input, behaviors, and achievements that occur before, during, and after usage are included in the user experience. (Law *et al.*, 2009, 719-728)

Three factors affect user experience: System, User, and Usage Context.

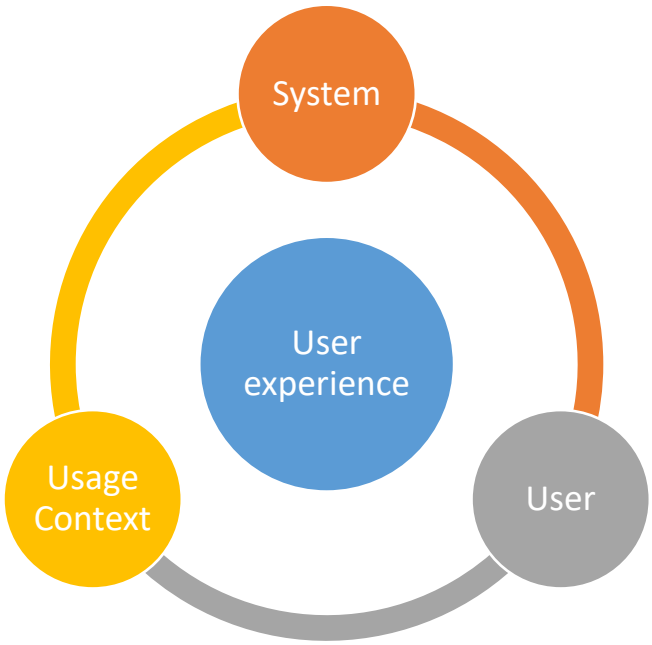

Figure 24: Factors affect user experience (Garrett, 2010)

Usability explains how easy a product is to use and is sometimes mistaken for user experience (UX). However, user experience has become more important than usability, as UX is critical to deciding if a product is a success or failure. It is essential to pay attention to all aspects of the user experience to successfully deliver the product to the market. (Law *et al.*, 2009, 719-728). At Aibida, we regularly listen to and collect user feedback to build and develop suitably with users' demands. For instance, the Description of market view is an available feature that we invented, but during the usage time, we observed that this feature is not valuable for a user; that is why we removed it from the UI.

Seven elements describe user experience, according to UX pioneer Peter Morville who has written many best-selling books and consulted for many companies on UX: (Pathuma, 2021)

- Useful

- Usable
- Findable
- Credible
- Desirable
- Accessible
- Valuable

At Aibidia, we apply the user Story idea to improve user experiences. A User Story is an introductory document that lists product needs from the user's perspective. Typically, the customer or a client representative writes the User Story. Nonetheless, with the Developers' involvement, the team and the client will have a better common knowledge of the product. User Stories are printed on little cards or stickers for organizations that utilize physical boards. At Aibidia, the team primarily works remotely, so we use Azure DevOps to create Product Backlog items.

User stories have the format:
As <specific user/role> I want to <do something> to <serve some purpose>

User stories should follow the 3C model:

- Card: Usually, User Story is written on a small card. That means it's usually short enough to write on a card. When writing on the Azure DevOps system, we keep it short.
- Conversation: Stories are stories between customers and Developers. Therefore, details of the User Story are clarified through conversations (preferably face-to-face) with customers. The content of the User Story will become more and more specific depending on its priority. If the priority is high, it needs to be done soon; its content is explicit. If the priority is low, it only contains general content.
- Confirmation: User Story has Acceptance Criteria, so customers think specifically about the requirement and Developers can understand the requirement better and confirm when the product is completed.

(Lucassen *et al.*, 2016, 205-222)

The Product Owner is the one who manages all the User Stories but is not the one who writes the entire User Story. Developers can all participate in writing User Stories. Developers play an important role in describing product features. At our company, in the ideal case, the actual users of the product would be involved in writing the User Story. In other cases, the Product Owner may represent the user but should always write the User

Story in the user's role, not the Product Owner's role. User Story writing takes place throughout project development, which means members can add new User Stories at any time.

3.9 Observation week 9 (March 07-March 11, 2022)

Monday 7th March 2022

Because the release is tomorrow, everyone worked hard to resolve all of the minor flaws and difficulties. Due to a request from the product owner, the daily team chose to exclude several Sprint Backlog Items that were for the next version in favor of addressing issues before release. The tester maintained providing issue fixes to the team throughout the day. A bug is coming from a rich text editor that I worked on a few weeks ago. This time customers report that bug, so it is quite urgent to fix. As I am the person who understands rich text implementation best, so I quickly grab this task, work on this and create pull requests.

Tuesday 8th March 2022

I do miscellaneous things today, including resolving comments and reviewing pull requests. I work on a bug in Tpdoc: Entity Description - Descriptions are editable when the Status is completed. I also helped my colleague with the front-end task of adding customer logo tasks.

Since I understand the user story concept, I love applying it whenever possible. The adding logo task is described:

- As a user, I want to upload an image in Group General view to have the logo appear in Master/Local Files.
- As a user, I want to clear the image I uploaded so it will no longer appear in Master/Local Files.
- As a user, I want to see a preview of the image I uploaded so I know what it looks like.
- As a user, I want to select if the image should appear in Master File, Local File or both.

Wednesday 9th March 2022

I install a new package to decode JavaScript-based HTML entities. I chose he since it is well-known and commonly used. He (short for "HTML entities") is a JavaScript-based HTML entity encoder/decoder. It supports all HTML-compliant named character references, handles ambiguous ampersands and other edge circumstances like a browser, has a robust test suite, and can handle astral Unicode characters, unlike many other JavaScript solutions.

I finished writing some more unit tests and responded to the resolution remark on my PR. I'm also attempting to put up a local backend to assist Mia with the front-end logo assignment. Also, investigate how scope is utilized in tpdoc. Today, in the sync meeting in

the supply chain, I will also ask Pauli about the date picker and ask if I grab some items from Azure DevOps.

Thursday 10th March 2022

After yesterday's sync meeting, some bugs were found in the supply chain improvement epic, and I grabbed two of them and worked on them.

The first bug reproduces like this:

- Go to the transactional supply chain, the select year 2000
- There are no results of the available supply chain; therefore, the dropdown shows "No result".
- Users can still select 'No result' as an option to clear the overlay.

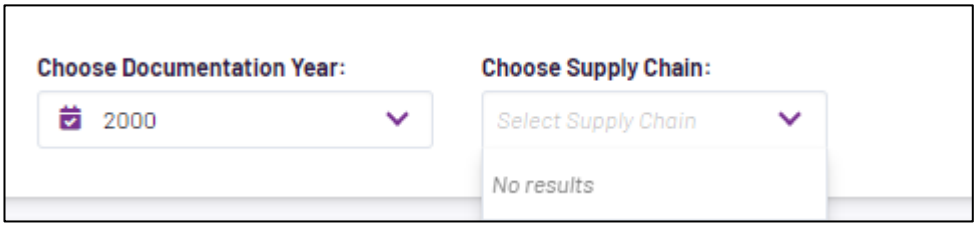

Figure 25: Screenshot reproducing a bug in supply chain dropdown selection.

This bug is done when the user can not select an item, and Overlay is displayed if there is nothing from the supply chain dropdown selection. As the requirement and expected behavior are clear, I will work on this without difficulty. I create a branch from the main branch of supply chain update and create pull requests.

Friday 11st March 2022

Today I worked on the remaining bug related to the unit test. When running unit tests, they fail with weird errors about an undefined endpoint. When testing action, we suppose to mock actions and create requests to API. However, there is some problem with the test causing this error. Fail unit test leads to break when running pipeline in Azure DevOps.

```
TypeError: this.endpoint(...).then is not a function

  174 |     private async constructConcurrentRequest(dto: T): Promise<unknown> {
  175 |         return this.endpoint(dto, this.emptyResponse)
> 176 |             .then(resp => {
      |              ^
```

Figure 26: Screen shot fail unit test message.

Week 9 analysis

Stress was the driving force behind this week's events. This week required a lot of hard work and attention, but in the end, it paid off since we had our release ready, with all the minor and significant concerns identified had been resolved. The sprint about came to an end this week as well. One important topic that I have worked on this week is software testing. I have had to write many tests from day one until now and realized that writing tests are as essential as producing code.

Software testing is a technique for determining if a software product meets the specified requirements. It uses software/system components to evaluate one or more properties of interest using human or automated approaches. In contrast to basic requirements, software testing tries to detect errors, gaps, and missing requirements. Program testing is essential because any flaws or errors in the program may be discovered early and fixed before the final product is delivered. A well-tested software solution ensures dependability, security, and high performance, saving time and money while enhancing customer satisfaction. Testing is essential since software flaws may be costly or even catastrophic. Software defects have a track record of causing financial and other damage. (Pal and Karakostas, 2021, 114-131)

In software engineering, there are a few fundamental techniques to consider:

- Unit testing: refers to the process of the programmer using this software testing fundamental technique to test the program's unit. It helps developers determine whether each precise unit of code is working correctly. (Mayeda and Andrews, 2021, 41-48,)
- Integration testing is involved with the software's structure and design. Each separate software module is combined and tested in groups. Integration testing occurs after unit testing and before validation testing. Integration testing takes unit-tested input modules, groups them into larger sets, applies the test cases defined in the integration test plan to that set, and provides output for the integrated system. (Mayeda and Andrews, 2021, 41-48.)

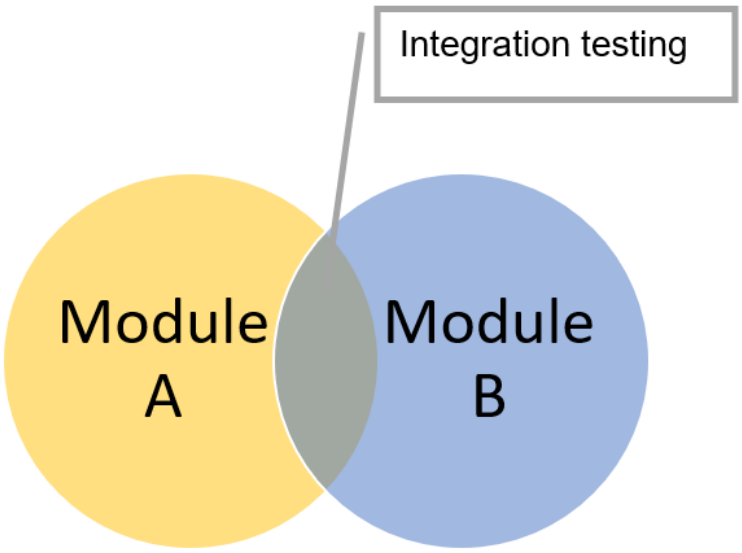

Figure 27:Two different modules 'Module A' and 'Module B', are integrated then integration testing is performed.

- System testing: This approach entails compiling and testing your program. This testing technique examines the software's functionality, security, and portability, among other things.

(Mayeda and Andrews, 2021, 41-48)

At Aibidia, we use Cypress and Jest's test frameworks to develop unit and integration tests. It is standard practice for developers at the company to do unit tests whenever they create a function. I've written a lot of unit tests but haven't had the chance to create any integration tests. To guarantee the accuracy of the JavaScript codebase, I utilize Jest. It enables development tests using a user-friendly, familiar, and feature-rich API that provides speedy results. Jest employs a custom resolver for imports in tests, making it easy to fake any object that isn't in the scope of your test.

In addition to functions, I also need to create a unit test for the store in Vuex. Test file for store need to follow this pattern: (Aibidia Wiki, 2021)

1. prepareSampleData: Prepare fresh data for each test case. Return the object of mock data, which is used for testing.
2. prepareStore: Prepare the store instance for the testing environment

3. prepareMockApi: Prepare mock Api methods, used in-store actions which will return an object of mock methods, used to check mock method call time, parameters.
4. initializeState: A helper function to prepare data in the state before a test.

Function beforeEach is created to help create new sample data and run before starting each test suite—similarly, afterEach clearing all mock functions' states before the end of a test case.

Then I create test cases for mutations, getters, and actions in Vuex to ensure they work as expected. We can choose to write the test for actions or not because actions are usually related to mocking API requests causing unnecessary checking. When testing actions, we suppose to mock actions, and the return is usually a Promise.

Writing a test is not a complicated task, but it consolidates a program and minimizes the risk of having bugs during the use process.

3.10 Observation week 10 (March 14- March 18, 2022)

Monday 14th March 2022

We have a daily as usual, and this time, we use a personal update, which means everyone will report what they have done and the plan for a new working date. I like this format because everyone has a chance to speak, and we also have a small talk after that.

Today, I work to fix unit tests in a supply chain store. Then I work on the date picker in pdoc. I also resolve the comment on my pull request.

Tuesday 15th March 2022

I continue to work on fixing a minor bug in validation in Tpdoc. Yesterday, my colleague gave feedback on supply chain improvement in the test environment prepared for tomorrow's demo day. I present a new feature, so I also prepare a presentation and try to merge all branches that other people developed.

Our new project manager collects opinions from the team and suggests a new way to conduct a retrospective. I will fill out the form and give some valuable insight to our PM.

Wednesday 16th March 2022

On the Sprint review, I present some updates in supply chains in Tpdoc. The first change is we remove the Description of Markets from the views. There are two tabs in the side menu: add supply chain and transactional supply chain. "Add supply chain" presented in a table that allows users to edit all data objects just like other solutions' views. It has three columns: Supply Chain Name, Description, Description of Markets. Move on to the `Transactional Supply Chain` tab. It is renamed from the old `Add transaction` views. But now, within this view, the user can not create, navigate, rename or delete supply chains because it is separated in the other view: `Add Supply Chains`.

The most important column in this view is the Definition Abbreviation column. It was previously located in the end and seemed hidden from the table, but now it is moved after Reporting Level columns that make more sense for the user. Users can choose the supply chain from the dropdown selection; then, when they navigate away or back and forth between different views, the solution still remembers the latest supply chain selected.

Some user experiences improvements were also added. For example - counters indicate the order of the selected supply chain out of the total supply chain. - The cursor is a

pointer indicating that the user can click into rows. - user can see all Configurable Columns in the Related Transactions Modal.

Lastly, the user can open the tooltip on Reporting Level column to find more information. It is not applied to the supply chain but anywhere in the solution with a placeholder column "Reporting Level."

Thursday 17th March 2022

After the sprint planning meeting, I started to work on the first work item assigned to me. The workstream that I develop is to create the possibility to notify logged-in users about a recent production update.

It is an important feature that we decided to include in this iteration because of several reasons:

- Users should be aware when using an old version of the platform and be informed about refreshing the browser to utilize the latest version.
- If the users keep using the old version, they might experience unexpected system behavior and possible data loss (e.g., the latest changes might not save properly).
- If the platform automatically informs the active users about production updates, the users have less need to contact customer service about events that seem like the platform is unstable or buggy.

We have four people to develop this feature. One developer takes responsibility for technical design; two front-end developers: one creates components in the UI library, and the other applies that Ui library to the solution; one back-end developer.

The goal is: after the production update, users running an old app version are shown the notification bar informing users about the new version. After the browser refresh, the latest version of the app is taken into use, and the notification will disappear.

Friday 18th March 2022

On Friday, I worked on creating a notification bar in Aibidia Layout, our internal UI library. The goal is to create a new notification bar component that other developers can pass markdown as content to the notification. Plus, there is an option for whoever uses it to pass Font Awesome icon to the notification as it is exported from AL as an extendable component. The component styles themselves must use theme parameters defined as CSS variables. The component interface is JSDocs documented to enable the reviewers and other developers to understand the problem the component is trying to solve. The most important thing is that components can potentially be inserted into AppLayout without additional CSS styling.

Later, I created a draft PR for that ticket and continued working until the end of the day. I managed to finish by today before my vacation next week.

Week 10 analysis

Reusable component

This week is another productive week. I encountered some challenges but managed to overcome them. I have worked on creating reusable components in the Aibidia Layout UI library and understand we should reuse components as much as we can as it will bring convenience and efficiency. When I dig deeper into this topic, I find out that reuse components have a long history.

Instead of designing software systems from the ground up, software reuse involves repurposing existing software. In 1968, it was launched as a simple yet potent idea. On the other hand, software reuse has yet to become a conventional software engineering technique. Researchers have rekindled their interest in software reuse and the challenges of putting it into practice (Krueger, 1992, 81). Adhoc code reuse has been around since the dawn of programming. Programmers have long reused pieces of code, templates, functions, and procedures. Code reuse saves time and resources. The critical concept behind reuse is that once generated, computer program components can (and should) be reused in later versions of the building. Code reuse may require setting up a separate instance of independently maintained reusable assets. While code is the most typical resource for reuse, other assets created throughout the development cycle, such as software components, test suites, design, and documentation, can also be reused. (Lombard Hill Group, 2022)

It's tough to reuse software. This is especially true for Aibidia as it has many product components and development teams spread around the globe. Here are three reasons why reusing software is complicated (Sametinger 1997, 17-18). Software reuse becomes more difficult as the number of projects and developers increases. It is not easy to effectively express the nuances and needs of code reuse. It is also challenging to provide adequate support and feedback on code reuse. Cataloging, archiving, and retrieving reusable content worldwide is complex.

Platforms like GitHub can help with this. However, creating a useable and scalable code repository requires time and work. Office politics is one barrier to reusing software at the corporate level. As business units seek autonomy — or compete with one another — they may try to prevent their assets from being reused by other units (Sametinger 1997, 16).

At the individual level, developers may see code repetition as suffocating their originality or indicating that their employer lacks faith in their technical ability. Some developers are resistant to initiatives to improve code reuse because of these misconceptions. Developing reusable code requires time and money commitments. Reusable code can be reused in new ways significantly different from the code's original design intent. (Harvie, 2018)

Development teams must devote more effort to developing documentation for their code to promote intended reuse. Testing processes are conducted more carefully. For developers working on tight schedules, this is a pain.

There are six types of reusable code in programming (Benmccormick, 2022):

- **Using the existing component in its current state** (Benmccormick, 2022): The benefit of this strategy is that it requires the least amount of work from developers. It also promotes uniformity and simplicity. In other circumstances, however, it is not permitted. It may be sacrificing user experience for developer convenience. The first choice is the most straightforward. Change the requirements if you have a component similar to what is required but does not quite meet them. This is useful when you have a UI element or control that you utilize across the system. Instead of designing a new UI element for the page, it could make sense to reuse the same element in a different location for consistency and user understanding. It, of course, necessitates a working environment that values developer contribution in the design and requirements process. However, developers should use caution in this area. It's a win-win situation if adopting an existing component reduces unneeded complexity from the design while also saving time. However, if the design is good, but consumers lose functionality due to developer "efficiency," the product may suffer. Worse, such judgments frequently result in a recurrence of the need that was ignored, as the need that was denied will frequently reappear in different contexts.

- **Copy and Paste**: This is a simple approach to implement (Benmccormick, 2022): In the near term, this strategy maintains the existing code stable. On the other side, this type of code may raise the maintenance work and make the code base less readable and understandable for other developers. It also copies existing code and partially alters it to fit the present scenario. It is similar to duplicating the User avatar code described above and merely altering it to make it bigger. It's worth noting that copying and pasting aren't necessarily terrible. There are often compelling reasons to leave existing code alone. It might be code you don't have control over (for example, forking an open-source project that is no longer actively maintained to add a feature you desire) or code from another project (or from a legacy portion of the current project that is handled as a different code base). However, DRY isn't insane advice, and if you're working with a single codebase that you control, this isn't always the best solution.

- **Inheritance in the classical sense** (Benmccormick, 2022): The advantages of this type are that it is easily understood by most programmers, particularly those with an academic background. It works well when you have numerous items that are variants of the same type, with more similarities than differences. The disadvantage is that it is ineffective in cases where the items are more dissimilar

than similar or when they are fundamentally two separate entities with expected behaviors. Inheritance binds things together by requiring them to share an API and implying that they may be used interchangeably. If the objects are not interchangeable, maintaining a standard API may require jumping through hoops, with a small base class stretched between two or more lengthy child classes. Interfaces especially tend to be wrong. Interfaces specifically tend to be deficient places to use inheritance. They tend to require minor tweaks and differences when used in different places that aren't friendly to be passed as configuration properties or over-ridden methods.

- **Configuration**: Everything is kept in one place using this strategy. Declarative configuration is often easier to comprehend and change than logic. However, it can get out of hand fast as use cases increase. It is also often unclear what defaults should be; bad defaults make everyday use cases verbose. We may specify individual actions for two identical items using a single customizable object. Because everything is in one location, it maximizes code reuse and offers an easy-to-read abstraction over the code for future updates. However, it works much better for similar objects with slight differences than for different objects with shared behaviours. Because it can lead to "configuration creep" if the differences expand over time, resulting in a monster configuration and spaghetti code that tries to handle all of the potential configuration possibilities. Finding suitable defaults is a significant design issue with configuration-based techniques. Setup-based objects and APIs may be a lot of fun to work with if they're built so that the default case requires little to no configuration, and only the most extreme edge situations require a lot of it. However, objects with poor or no defaults that force typical scenarios to provide lengthy settings eventually result in bloated, difficult-to-debug programs.

- **Mixins** (Benmccormick, 2022): The benefit is that particular behaviors or characteristics are shared quickly. The disadvantage is that it may fail when the behavior or property demands extensive knowledge about the object. Mixings work best when objects are considerably diverse yet have similar behaviors instead of inheritance and configuration. The approach of adding specific methods or attributes to an object without assuming any link between objects that utilize it is known as mixins. When used to provide a stand-alone behavior to various objects, mixins are fantastic. They prevent inheritance issues and don't bind an object to a specific API. They prevent inheritance issues and don't bind an object to a specific API. Building good mixins, on the other hand, may be challenging since many behaviors need knowledge about the object performing the activity, such as making assumptions about the structure of the object's data. This either requires the objects that implement the mixins to follow the rules about how they're structured, such as inheritance, or it involves offloading significant complexity into the mixins to allow it to handle different formats (potentially resulting in bugs), or it risks naming conflicts if the mixins try to implement data storage itself.

- **Composition** (Benmccormick, 2022): This approach is adaptable to a variety of subtle differences in behavior. It has the potential to increase complexity, making it difficult to perceive the entire system. It may only push them down instead of fixing code reuse issues, resulting in a slew of little, similar modules. When presented with two comparable objects or functions, this strategy is utilized. The ideal option is to construct more objects or functions. Micro-services and applications made out of tiny single-purpose modules are touted on the server-side. Instead of having one extensive, complicated system, it can be created a few more minor "things" that can be wrapped around and worked together. The issue is that it frequently shifts complexity from the components to the architecture. It was figuring out how components work together, building interfaces, and selecting what should go

where became the tricky part. Although a single line of code may be understandable and accessible, the system is more challenging to envision and comprehend. Composition, like mixins, works best when you have two or more "things" that share behavior. The composition mindset considers code similar to Lego blocks that can be snapped together in various configurations. It allows for beneficial abstraction when done correctly, allowing developers to design complicated components out of relevant sub-components without knowing all the intricacies of the subcomponents. It can be dangerous if done incorrectly. It might result in a disjointed project and has numerous undocumented dependencies between components when done incorrectly.

4 Discussion and Conclusion

When I re-read my thesis and synthesize all the knowledge I learned, I feel my practice has made tremendous progress. The content of what I learned throughout ten weeks covers different topics: From critical topics in software development, for example, Vue lifecycle, Vue global and local component, to generic concepts, for example, why do we need to sanitize user input, user experiences using user story concepts or promising practices in coding, for example reusable, software testing, using third-party lib. Writing a learning diary thesis is a valuable toolbox for my self-learn journey. At the end of each week, I conduct a weekly analysis where I read books, articles and literature sources to understand better the problem I have to deal with in that week.

I established three goals for myself at the start of the job:

- Growing professional knowledge and training
- Fast adapting to a new international working environment
- Contributing to all the company's products.

After ten weeks at the new firm, I've accomplished all my objectives. It is my first job to start my career as a software engineer. Compared to me as a tech student, I can notice a substantial improvement in myself compared to when I was a student. During the initial weeks of the onboarding process, I was terrified and doubtful of my abilities. I've always felt that my academic knowledge was insufficient to match the job's criteria. For me, every day at work had been tremendously stressful. I was scared to seek help from co-workers because I was frightened of bothering people, and I felt it was my job to solve the problems in my own time. However, the time has passed, and I've gained some valuable skills that will help me advance in my career.

Communication is an essential skill; the friendly atmosphere at the company has taught me that there are no stupid questions. I can freely ask if there is nothing explicit. Of course, that does not mean that I keep asking questions that I could easily find the answer to since it will reduce productivity. I am now confident in communicating with other co-workers to accelerate the development process. I am not scared to seek help since I was not expected to know everything, and I should not be embarrassed to ask questions. After all, it may help the company save time and money. While working, I acquired two more essential talents: researching abilities and learning quickly by doing. My time at university helped build up my research skills. I understand that all the answers are available in books, articles, and company documentation on the internet. The right way of researching will enable to shorten the implementation time.

I try to analyze my anxiety during the first few weeks and then contribute the input to HR and my supervisor when they ask how to improve the onboarding session. The learning curve is one thing I emphasize as each person has their own learning curve time. That is essential for newcomers to understand that there is no need to feel stressed if they can not code all the time, as once you get a problem, you spend 90% of your time thinking about how to implement it and only 10% coding.

Debugging skill is also very important. I always debug when I need to work on fixing bugs or adding new features. My debugging method was trying to print unclear things to console to understand the logic. Other developers in the team also teach me other techniques, including using the network tab in the developer tool to see the content of the request, asking the right questions, paying attention to error messages, trying to replicate the problem, etc. Thanks to all those things, I can debug quite quickly and save a lot of time.

Regarding my programming skills, consistently learning and working create daily improvements. Looking back to the day I started as a junior front-end developer, even though I learned several programming languages and practiced my skills with different projects, I could not help myself feel confused and overwhelmed by how an actual software is built. My first task was refactoring the legacy code, and then I was assigned to fix bugs before developing new features. That procedure is a perfect path for me to complete my technical skills gradually. I also realized that learning and working with other languages or frameworks will be a breeze if I have good foundational programming abilities in any language. I've grown more excited to get started on a project that involves technology I've never mastered or worked with before. This is a fantastic opportunity to expand my knowledge and flexibility, making me more valuable at work.

Related to technical skills, I learned how to use several tools to improve my working experiences, including GitHub desktop, Azure DevOps, and testing library Jest.

I already learned from university skills to use the Git version control system, but I can only understand and use them properly when working in a significant development team. My pet projects are mostly solo coding, so I use git with basic functionality, including creating a git branch, sending a pull request, pushing new commits, and cheery picking a commit from the past. There is no conflict in the code. In the firm, though, I worked on a project with 20 other developers, and the front-end team included seven employees who often edited the same file. After accidentally deleting other people's code several times, I've learned a lot about resolving conflicts in git. I used to use git in the terminal; then, a

coworker taught me how to utilize the Github desktop to automate tasks with only a few clicks. Git is a concept, not a problem. I learned how to utilize git rebase instead of git merge in my present job, and I'm more confident in my ability to use git reset when necessary.

The testing skill I mention here includes writing unit tests. I have not written coverage tests when I work in the company. I created a good habit along is write clean code. It is crucial as the code base is large and results from many developers. It is a high priority to keep the code maintainable, easy to read and understand, and at a high-security level. Clean code enables writing test experiences to be smoother and more efficient. Working with senior developers, who can examine my code and provide critical criticism, has given me those experiences and abilities. I was focused on absorbing these vital nuggets of knowledge with an open mind and a desire to learn. I write unit tests using the Jest library for JavaScript code. Jest requires trim configuration and provides benefits to ensure the correctness of writing tests with an approachable, familiar, and feature-rich API that gives you quick results.

Moreover, I set specific disciplines to adapt to the company's remote working culture quickly. The first is completing my tasks as fast as possible to avoid procrastination when I work alone. The backlog items are always full of problems to go to pick up a new thing to work on when I finish my tasks. Second, if I spend more than a day solving a problem, I should seek help from other people to avoid being a bottleneck and slow down the development process. Lastly, I motivate myself with little things. Due to the covid 19, I have experience studying online at my university. During that time, the huge problem was losing motivation when I had to handle everything alone without interaction from my classmates and teacher. Therefore, when I work from home, I understand that I should find a way to keep myself motivated. I find joy when my work is merged into the main branch, and that feature is available on the product that will be delivered to the customer.

Writing a thesis in a journal has proven to be a rewarding experience that has provided me with several benefits. Because being a programmer or a developer is a long and continual learning process, I've been keeping track of everything I've learned from my first day at work in a journal. It is an excellent opportunity for me to take notes on the situation and dig further into it. I've had the opportunity to reflect on how I've grown psychologically from week one to week ten.

After reading the whole diary in 10 weeks, the critical thing I realize is that work isn't always productive, but it motivates me to keep going. I sometimes feel like I was wasting

my time at work and accomplishing nothing, yet the experience was invaluable. Every time I failed, I learned something new. And it was via those teachings that I developed into a skilled developer.

I am glad and grateful that the team recognized my expectations and provided me with additional opportunities to learn, grow, and develop my front-end skills. It was difficult at first, and it was full of hurdles, but it was how I grew. I plan to concentrate on being a full-stack developer in the future.

References

Aibidia. (n.d.). Aibidia. [online. URL: https://aibidia.com/about-us/>[Accessed: January 18, 2022. [Accessed: 6 February 2022.

Andersen, B. 2022. Global & Local Components. Coding Explained. URL: https://codingexplained.com/coding/front-end/vue-js/global-local-components#:~:text=A%20global%20component%20is%20a,components%20where%20it%20is%20registered.. Accessed: 6 February 2022.

Atlassian 2022. Sprint Planning | Atlassian. URL: https://www.atlassian.com/agile/scrum/sprint-planning#:~:text=Sprint%20planning%20is%20an%20event%20in%20scrum%20that%20kicks%20off,with%20the%20whole%20scrum%20team.. Accessed: 24 February 2022.

Backes, M., Bugiel, S. and Derr, E. 2016. Reliable third-party library detection in android and its security applications. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.

Benmccormick.org 2022. URL: https://benmccormick.org/2016/01/08/reusable-code-patterns. Accessed: 20 March 2022.

Bestofvue.com 2022. A renderless rich-text editor for Vue.js | BestofVue. URL: https://bestofvue.com/repo/heyscrumpy-tiptap-vuejs-rich-text-editing. Accessed: 14 February 2022.

Brown, W.H., Malveau, R.C., McCormick, H.W.S. and Mowbray, T.J 1998. AntiPatterns: refactoring software, architectures, and projects in crisis. John Wiley & Sons, Inc..

Cheatsheetseries.owasp.org 2022. XSS Filter Evasion - OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html. Accessed: 4 February 2022.

Chocolatey Software 2022. Chocolatey - The package manager for Windows. URL: https://chocolatey.org/. Accessed: 9 February 2022.

Delloite 2016 ‘Country-by-Country reporting The FAQs’, Delloite, September. URL: https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Tax/dttl-tax-country-by-country-reporting-faqs.pdf (Accessed: 15 January 2022).

Developer.mozilla.org 2022. JavaScript | MDN. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed: 4 February 2022.

Docs.microsoft.com 2022. Azure DevOps documentation. URL: https://docs.microsoft.com/en-us/azure/devops/?view=azure-devops. Accessed: 19 February 2022.

Doğan, D. and Tüzün, H. 2022. Modeling of an instructional design process based on the problem-based learning approach in three-dimensional multi-user virtual environments. Education and Information Technologies,

Edmunds, B. 2016. Never Trust Your Users. Sanitize ALL Input!. In Securing PHP Apps (pp. 1-8). Apress, Berkeley, CA.

Garrett, J.J. 2010. The elements of user experience: user-centered design for the web and beyond. Pearson Education.

Gilbert, Jahlesha and McMillan, Henry and Walters, K., Transfer Pricing 2013. URL: https://ssrn.com/abstract=2215015 or http://dx.doi.org/10.2139/ssrn.2215015. Accessed: 6 April 2020

GitHub Desktop 2022. GitHub Desktop. URL: https://desktop.github.com/. Accessed: 13 February 2022.

GitLab 2022. Files · tiptap-commands@1.10.5 · Adam Gerard / tiptap. URL: https://gitlab.umich.edu/aglane/tiptap/-/tree/tiptap-commands@1.10.5. Accessed: 6 February 2022.

Git-scm.com 2022. Git. URL: https://git-scm.com/. Accessed: 4 February 2022.

Haynes, M.E. and Crisp, M.G. 1988. Effective meeting skills. Kogan Page.

Harvie L. 2018. 4 Ways to Make Your Code More Reusable. Medium. URL: https://medium.com/@lanceharvieruntime/4-ways-to-make-your-code-more-reusable-bc20889c1e4. Accessed March 23, 2022.

Janssens, J. 2021. Data Science at the Command Line. " O'Reilly Media, Inc.".

Krueger C. W., 1992. Software reuse. ACM Computing Surveys,

KPMG. (n.d.). Operational transfer pricing. Operational Transfer Pricing. Retrieved January 18, 2022, from https://home.kpmg/xx/en/home/insights/2020/03/operational-transferpricing.html#:~:text=OTP%20is%20the%20management%20of,and%20driving%20better%20bus. Accessed: 18 January 2022

Law, E.L.C., Roto, V., Hassenzahl, M., Vermeeren, A.P. and Kort, J. 2009, April. Understanding, scoping and defining user experience: a survey approach. In Proceedings of the SIGCHI conference on human factors in computing systems.

Lenarduzzi, V., Nikkola, V., Saarimäki, N. and Taibi, D. 2021. Does code quality affect pull request acceptance? An empirical study. Journal of Systems and Software, 171, p.110806.

Lombardhill.com. 2022. Lombard Hill Group. URL: http://www.lombardhill.com/#:~:text=The%20Lombard%20Hill%20Group%20empowers,and%20shortened%20time%2Dto%2Dmarket. Accessed: 4 April 2022.

Lucassen, G., Dalpiaz, F., van der Werf, J.M.E. and Brinkkemper, S. 2016, March. The use and effectiveness of user stories in practice. In International working conference on requirements engineering: Foundation for software quality.

Lvivity 2022. What is a Proof of Concept in Software Product Development?. URL: https://lvivity.com/proof-of-concept-meaning#:~:text=Proof%20of%20Concept%20is%20a,solving%20a%20particular%20business%20problem.. Accessed: 23 February 2022.

npm 2022. sanitize-html. URL: https://www.npmjs.com/package/sanitize-html. Accessed: 5 February 2022.

npm 2022. vue-clickaway. URL: https://www.npmjs.com/package/vue-clickaway. Accessed: 5 February 2022.

Mayeda, M. and Andrews, A. 2021. Evaluating Softwar aluating Software Testing T esting Techniques: A Systematic Mapping echniques: A Systematic Mapping Study.

Microsoft.com 2019. What is DevOps? DevOps Explained | Microsoft Azure. URL: https://azure.microsoft.com/en-us/overview/what-is-devops/. Accessed: 18 January 2022.

OECD Countries. 2009,. Forum on tax administration: compliance management of large business task group. OECD. URL: https://www.oecd.org/netherlands/43241144.pdf Accessed: 18 January 2022.

Pal, K. and Karakostas, B. 2021. Software Testing Under Agile, Scrum, and DevOps. In Agile Scrum Implementation and Its Long-Term Impact on Organizations.

Pathuma, G. 2021. 7 Factors of User Experience by Peter Morville. URL: https://medium.com/ascentic-technology/7-factors-of-user-experience-by-peter-morville-cb2fdfb45bcb. Accessed: 4 April 2022.

Perforce Software. 2022. What Is Code Reuse? Code Reuse Best Practices | Perforce Software. URL: https://www.perforce.com/blog/qac/what-code-reuse-code-reuse-best-practices#:~:text=Code%20reuse%20is%20the%20practice,these%20requirements%20is%20a%20challenge. Accessed: 4 April 2022.

Pozrikidis, C. 2002. A practical guide to boundary element methods with the software library BEMLIB. CRC Press.

Sametinger, J. 1997. Software engineering with reusable components. Springer Science & Business Media.

Schwaber, K. 1997. Scrum development process. In Business object design and implementation.

Scrum.org. (n.d) ‘The Scrum Development Process’, Scrum.org, n.d. URL: https://www.scrum.org/resources/scrum-development-process (Accessed: 15 January 2022.

Simplilearn 2021. Azure DevOps: The Next Big Thing in Application Lifecycle Management. URL: https://www.simplilearn.com/azure-devops-article. Accessed: 28 February 2022.

Shobhit, S 2021. ‘Transfer Pricing’, Investopedia. URL: https://www.investopedia.com/terms/t/transfer-pricing.asp#:~:text=Transfer%20pricing%20is%20an%20accounting,for%20goods%20and%20services%20provided.&text=Transfer%20pricing%20can%20lead%20to,authorities%20may%20contest%20their%20claims. Accessed: 14 January 2022.

Test Project. 2022. 10 Common Git Commands Everyone Should Know - TestProject. URL: https://blog.testproject.io/2021/03/22/git-commands-every-sdet-should-know/. Accessed: 13 February 2022.

V1.tiptap.dev. 2022. tiptap. URL: https://v1.tiptap.dev/. Accessed: 5 February 2022.

V2.vuejs.org. 2022. Slots — Vue.js. URL: https://v2.vuejs.org/v2/guide/components-slots.html?redirect=true#:~:text=Slot%20props%20allow%20us%20to,customize%20part%20of%20its%20layout. Accessed: 14 February 2022.

Vuejs.org. 2022. Component Registration — Vue.js. URL: https://vuejs.org/v2/guide/components-registration.html. Accessed: 6 February 2022.

Vuejs.org. 2022. Enter/Leave & List Transitions — Vue.js. URL: https://vuejs.org/v2/guide/transitions.html. Accessed: 5 February 2022.

Vuejs.org. 2022. Introduction — Vue.js. URL: https://vuejs.org/v2/guide/. Accessed: 4 February 2022.

W3schools.com. 2022. HTML Entities. URL: https://www.w3schools.com/html/html_entities.asp. Accessed: 10 February 2022.

Webopedia. 2022. What is Input Sanitization? | Webopedia. URL: https://www.webopedia.com/definitions/input-sanitization/. Accessed: 4 February 2022.