Haaga-Helia

Working as a React Native developer

Nadezhda Zhuravleva

Haaga-Helia University of Applied Sciences

Bachelor's Thesis

2022

Bachelor of Business Administration

Abstract

Author
Nadezhda Zhuravleva

Degree
Bachelor of Business Administration

Thesis title
Working as a React Native developer

Number of pages
53

This report is a diary type thesis about the author's professional and personal development while working as a full-time Junior React Native developer at Steerpath Oy. The thesis consists of eight weeks of daily reporting, each week followed by an analysis. Total time spent for the report writing is ten weeks including the introductory and conclusions parts.

The main goal of the thesis is to follow on the author's development as a software specialist. The key parts of the report are weekly analyses. The main purpose of the weekly analyses is to find new information, models and best practices and technics in literature and apply the information so that it translates into professional development. The daily reporting, on the other hand, is just a starting point for reflections. The conclusions include author's reflections on own professional development, new solutions and methods discovered for work, points on how the thesis writing process was benefitial to the author, and goals for the future development.

This thesis would be useful for anybody who is interested to know what kind of tasks Junior React Native developer might be facing on a daily basis in the beginning of their career. The discussions in weekly analyses can help the reader to see what problems or obstacles the entry level specialist might have while performing their duties as well as possible solutions and analyses that could potentially help the reader to overcome their own struggles if they are facing similar issues in the future.

Keywords
React Native, TypeScript, JavaScript, frontend development, web development

Table of contents

1 Introduction 1
2 Framework 3
2.1 Analysis of my current work 3
2.2 Interest groups at work 4
2.3 Interaction skills at work 5
3 Diary entries 21 February 2022 – 14 April 2022 6
3.1 Observation week 1 6
3.2 Observation week 2 11
3.3 Observation week 3 16
3.4 Observation week 4 21
3.5 Observation week 5 27
3.6 Observation week 6 32
3.7 Observation week 7 37
3.8 Observation week 8 42
4 Discussions and conclusions 47
References 50

Abbreviations

adb	Android Debug Bridge
API	Application Programming Interface
APK	Android application package
CI	Continuous Integration
CLI	command-line interface
HTTP	Hypertext Transfer Protocol
ID	identifier
i18n	internationalization
KPI	key performance indicator
npm	Node Package Manager
OS	Operating System
PR	pull request
RNP	React Native Paper
R&D	research and development
UI	user interface
URL	Uniform Resource Locator

1 Introduction

I work as a Junior Software Developer at Steerpath Oy. The company was founded in 2013. The headquarters is in Espoo, and there are two branch offices – in the USA and Singapore. Currently there are several projects going on, but I am working mainly on Steerpath Smart Office application, developing both mobile and web versions using React Native.

Steerpath Smart Office app is helping employees to manage and plan their working days. The users can choose whether they are going to work from home or office and this information is available to their colleagues, also they could see where their team is located and in which area within the office. We are using this tool ourselves as it is helpful in planning the office use without the need to actively communicate to the teammates, therefore, it is relatively easy to synchronise the visits.

At some point I will also be working on the Spacehub application. Spacehub is a private work- and meeting space that can be accessed with a phone whenever needed. The Spacehub solution is a great fit both for companies and individual freelancers and entrepreneurs. Spacehub pods can be found in central locations close to key services and traffic hubs.

I will be observing my work for eight weeks. I work in a team of six people, - backend and frontend developers. As per requirements, the diary thesis consists of daily descriptions of work tasks and weekly analyses for forty working days. Overall time that is spent for thesis writing is ten weeks. One week is reserved for the introductory part which includes the description of my current working tasks and skills, and another week is dedicated to final conclusions and analysis.

To perform my duties at work I use the following technologies: React Native which is an open source, cross-platform framework that is used for mobile applications and web development, JavaScript – programming language that allows developers to create dynamic content, it is used for developing frontend and backend applications as well as mobile apps, TypeScript – programming language that is based on JavaScript. It has optional typing and compiles to plain JavaScript. Tools used at work include Visual Studio Code – integrated development environment, Slack – main communication tool, GitHub – web service used for IT-projects hosting and collaboration on development.

At the moment of writing, I believe React and React Native by Adam Boduch and Roy Derks is going to be a helpful resource for me because it explains most of the basic concepts necessary to know for successful use of React Native. It focuses on the latest developments

in the React ecosystem, such as modern Hook implementations, code splitting using lazy components and Suspense, user interface framework components using Material-UI, and Apollo (Boduch & Derks 2020, cover summary).

As well as I find Daniel Bugl‘s book called Learn React Hooks specifically necessary for me to read. The Hooks are covered partly in the above-mentioned source, but I would like to dive deeper as all of the projects that I am working on using Hooks and I have limited knowledge on this. Learn React Hooks does not just give the theoretical background, but the learning is done by implementing a project while reading it.

I think as much as it is important for me to get more technical knowledge to perform my everyday tasks well, I also need to work on my time management skills. For that reason, I find the article Top 21 effective time management tips for students and researchers by Dawid Hanak quite useful. He talks about the most relevant tips for students and researchers, but I am definitely planning to keep on following many of them in the future as I think it is going to make my life easier at work and in general as well.

2 Framework

This chapter includes analysis of my current work, skills necessary to perform my tasks and skills that the job already helped me to gain, my current level of expertise and my long-term goals, as well as the description and the diagram regarding interest groups at work and a discussion about my interactions while doing my job.

2.1 Analysis of my current work

As I have started working on this position three weeks ago at the moment of writing this part, my current work tasks are to implement the user interface (UI) for a new booking page. The goal is to do relatively easy things in the beginning while I am learning more about the project structure. Also, I perform debugging and manual testing on Android Operating System (OS) mobile device and on the web application.

To keep new code consistent, I am carefully looking through the current code base in case something similar was implemented for the booking screen that is in use right now. Depending on the outcome, I either research the concepts and methods used that I am not familiar with or look for the guidance from my colleagues. If I need to do anything from scratch, I found it helpful and more efficient to draw on paper first to plan the connections between different elements and how one would affect another. I ask about the design and performance requirements if something is unclear from Figma design tool. I start working on the implementation once I understand the design and performance requirements. I prefer to write the code block by block, testing and styling each element before coding the other. When all seems complete and is working as expected, I double check if I followed the design precisely and perform thorough manual testing. The last step is to push changes to the remote branch, open pull request (PR) and wait for comments in case something needs to be improved.

The following skills are needed in my work duties: organisational skills and ability to multitask, willingness to learn new technologies and develop in already known, creativity, problem-solving and analytical skills, ability to effectively collaborate with my teammates. Here are the skills that I believe I obtained so far while performing my daily tasks: improved logical thinking, seeing bigger picture from the beginning, ability to write cleaner code, no hesitation to ask any kind of questions for help, better day planning. The best and most used sources of information for me at work are official React Native documentation, my teammates experience and StackOverflow community-based space.

I believe my skills are suitable for the requirements at my workplace. In the near future, I am going to be responsible for the app releases. This task requires the ability to follow the

instructions step by step at which I am good. I might be lacking creativity and fast thinking for my code writing for now due to not having enough experience with the technologies I use but this is going to be overcome as I am growing professionally and learning more every day. I am able to multitask, which would help me to be working on several projects simultaneously. I enjoy solving problems, tackling issues, and working in a team.

My stage of professional development is a beginning-stage developer working towards becoming a skilful programmer. I am able to work independently but often to complete the work tasks I might need to consult with my colleagues or google, I am flexible at completing my tasks, meaning it is not an issue for me to be working on more than one ticket at a time. It takes extra time to finish my work assignments. I am not skilled enough yet to be commenting on my teammates' PRs unless I notice something with which I am already familiar.

My focus for the future is to become more independent, learn more complex concepts for JavaScript, TypeScript, React.js and React Native, make programming my free-time hobby, become more efficient and get things done faster, take more responsibilities, be able to help others, learn to read and understand other people's code quicker.

2.2 Interest groups at work

Business customers mostly use Steerpath Smart Office service while Spacehub is a service targeted for both external groups. The concept for the Spacehub is to provide safe and convenient working environment be that for a professional or free time use.

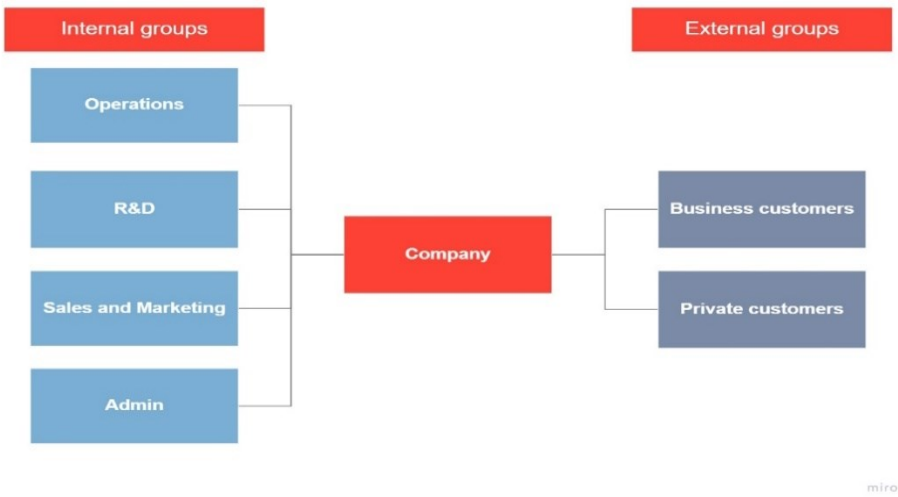

Diagram 1. Interest groups

As displayed on the Diagram 1, there are internal and external interest groups at the company. Internal groups are represented by Operations, Research and Development

(R&D), Sales and Marketing and Admin departments inside the organization, external groups consist of the business and private customers.

Internal group's goal is to work smoothly in collaboration with each other to follow and reach the objectives and goals set for the customers' and company best interests. I perform my duties in R&D department. We get valuable feedback from the customers and cooperate with other departments to fulfil organization's mission.

2.3 Interaction skills at work

I am closely collaborating with my teammates and colleagues. Every day we have daily where everyone shares what they did yesterday, what they will do today and what kind of obstacles there were if any. There are two experienced frontend developers who I could call my mentors. They are guiding me and helping whenever I have a question or need some advice. We can discuss design requirements, what kind of approach is better, what library or concept I should be using to successfully complete the task.

The communications take place both face-to-face and via Slack. Also, we have a meeting bi-weekly where all departments get together and share latest news and achievements on their side, and goals for the next sprint. After bi-weekly, my R&D department would have Sprint start Slack huddle where we would go through all the tickets for the closest release, discuss their importance, assign them among ourselves.

For now, I am not interacting with customers in any way. In the future one of my responsibilities could be online customer support. In general, the company follows some of the principles of agile development. There are not too many meetings so that there is a lot of time to focus on the direct work tasks.

3 Diary entries 21 February 2022 – 14 April 2022

I will be reporting my work daily and analyzing it on a weekly basis for the next eight weeks. The main goal is to observe my professional development. To achieve that, I will be describing what I have done at work. Possible concerns and issues might be addressed in daily entries which would be discussed in more details in weekly analysis. As well the concepts, technics and strategies needed to succeed in my duties will be explained in weekly. By the end of this two months period, I hope I will be more independent, be able to take more responsibilities, improve my expertise in React Native and my soft skills such as multitasking, communication and time management.

3.1 Observation week 1

This week my goals are to finalize last tab and new booking screen UI overall, start and complete new tickets, get the iOS development environment set up, figure out how the buildings data get hooked up to the components, take initiative and get my UI components working with real data.

Monday

I have been working on a new booking screen for our Smart Office app for three weeks. Today I am aiming to continue working on the Spacehub tab section. During the previous week I just finalized the duration picker and two out of three tab views: one is for the meeting booking and the other one is for the desk booking.

Today I have been working on the last tab view which required me to add a picture that I would use to the assets of the project, creating a button that would open our Spacehub app if it is installed on user's mobile device or the particular link for web version, or would direct the user to the store app depending on their platform.

I needed to hide the duration picker when the third tab is opened but I could not achieve that today. I got assigned new tickets that would be included to the next app release. I believe I have progressed today on the set goal – working on the Spacehub tab section – which allows me to mark it as achieved. I still need to finish the Spacehub tab but overall, it is almost ready.

Tuesday

On the second workday of this week, I am planning to finalize Spacehub tab and start new tickets for the upcoming release. Today my day consisted of quite many meetings, apart from having regular fifteen minutes daily, there were bi-weekly meeting and R&D Sprint start discussion with my team.

I finished Spacehub tab for the new booking flow, and added it to the existing booking screen PR. I have done small fixes to the tickets assigned to me the previous day, such as changed the button title wording as it was too long and would break the button's layout, fixed button alignment with adding styles to the card component and added top margin to the skip button for one of the screens as it was interfering with symbols at the top of the screen on some phones.

Since I was able to complete Spacehub tab as I planned and finish several new tickets, I would say that I managed to achieve the goals of the day. My biggest accomplishment was being able to dive deeper into deep linking and learning how to create *URLSheme* to use it in deep links. However, at the end of the day I ended up applying a simpler solution.

Wednesday

Today I need to set up iOS developer environment as one of the tickets assigned to me is a platform specific bug fixing. I still have two more tickets to complete for the upcoming release. I hope I will be able to finalize both of them today, but I aim to finish at least one of the tickets and start working on the next. I started with setting up the Xcode environment with the help from my colleague, and it took most of my working day. There were several errors preventing from finishing *pod install* command. It turned out I needed to install CocoaPods command-line interface (CLI) at first, and then run *pod install* with repo update.

After the app build was finally successful, I kept on getting the error "Unable to prepare iPhone for development". I was advised to check if I have my phone iOS version supported and if not – add the folder from the GitHub repository to make sure my device is supported by Xcode. It did not help, so we decided to come back to this issue tomorrow, so that I could do some coding today. I completed the ticket regarding adding improved language instructions for iOS users.

As I encountered numerous problems while setting up the iOS developer environment and we were not able to figure them all out today, therefore, the set up will continue tomorrow. I

still managed to complete one ticket today but did not start the last one yet, so I believe set goals are half achieved.

Thursday

I need to figure out why my phone is not working with Xcode – maybe the problem is with the Apple Watch being paired to it, also I am planning to start the last assigned ticket. So today I started from looking for a solution why my iPhone would not get set up as expected with Xcode. I tried many offered solutions after thorough research on StackOverflow and other similar platforms, such as rebooting phone or MacBook, connecting and disconnecting the cable, unpairing my Apple Watch and other manipulations. I started getting a feeling that the problem might be with my phone but then I decided to follow all the set up steps from the beginning, and I chose to do something differently – I added the 15.4 iOS version to the supported devices folder for Xcode even though I have 15.3. After a few more reboots, my phone was ready to be used for the development.

I also started working on the last ticket that I had pending. I needed to create a new building selector that would open buildings list on click in the form of a popup list instead of current one that would not allow to see the whole list at once. I chose to use dialog component from React Native Paper (RNP), which would include flat list from React Native and list item from RNP, but I had an issue such as the portal element from RNP would refuse to render on top of other elements which were new building selector relatives.

Friday

Today I aim to figure out why the dialog is not behaving as expected and continue on the development of the new building selector. After spending some time to get my first implementation idea to work, I decided to keep an open mind and check the project existing components again in case I missed something that I can reuse. I managed to find some modal component, which was used in several places in the project, so there were a few examples how it was applied that I could refer to.

I decided I would change the implementation to avoid creating something similar to what is already there. I had to modify the simple modal component, for example, I needed to get the option to get rid of the closing icon, so I added a prop closeIcon which would be a boolean, and it would not render closing icon if the prop had a value false. At the end the amount of styling that needed to be applied to the component was a bit too much and it was losing the point of reusable components. So, I decided to create my very own modal component for building selector specifically.

I ended up being able to easily customize it as required per design. At the moment the functionality of the building selector is not perfect, I will fix it and do the rest next week.

To sum up, today I changed the implementation to the building selector popup to keep the code consistent as I found the modal component which I could use as a reference but did not figure out why the dialog from RNP would refuse to work properly, I kept on developing the building selector using new approach so I could say the goals of the day are partially achieved.

Weekly analysis

During the first week I learnt how to hide elements when the specific tab is opened. I required an assistance from my colleague to solve this issue. It turned out I just needed to get tab index using *onChangeIndex* prop. It taught me to look more thoroughly through the source code of the components I use to understand how they work and what kind of props could be helpful.

My troubleshooting skills were also improved. While looking for an answer how to fix errors appearing while setting up the Xcode environment, I realized to look back and carefully reflect on the steps performed which led me to identifying the problem and successfully fixing it. The communication and collaboration skills as well as time management were further developed because I started feeling comfortable with asking any kind of questions instead of focusing on solving as much as possible all by myself.

When I was figuring out the way how to open another app or the app store page when the button is pressed in other app, I found the expo documentation incredibly helpful. It explained platform specific deep linking, how to set up the *URLSheme*, and even when it is better not to use the deep linking (Expo documentation s.a. a). I actually ended up using different approach as it happened so that the app that needed to be opened did not have URLSheme set up, so I was instructed to use simple Uniform Resource Locator (URL) instead which would either open a popup asking to open the app if it is installed or would open the browser page with a popup to app store.

Deep linking is mainly used when there is a need to provide an URL to direct the user to a specific point in the app on load. In a nutshell, linking provides an Application Programming Interface (API) that would allow us to listen for the incoming linked URL, which we would handle with the event listener. It lets us to set up the navigation to specific app page (Dabit 2017).

While working on the components and small fixes, I would need to find a source for quite a few errors and warnings. For example, at some point I noticed a warning ‘Require cycles are allowed but can result in uninitialized values. Consider refactoring to remove the need for a cycle’ when running the app on mobile phone. To understand it better, I turned to the StackOverflow post where I got to read very full and well written answer in the first comment. Basically, the parent component was importing child component to use in its render method, and the child component was importing constants from parent component, which was what caused it. There were unnecessary imports, so I was able to break the cycle simply by removing one of the imports (Lehnhardt 2019).

I came to realize how helpful the usage of the flexbox layout is when it comes to React Native. I recapped many flexbox properties on several occasions during this week. Especially useful for my case were *flex* that is defining how the items are going to occupy the available space along main axis; *flexDirection* which is great to specify desired elements direction; *justifyContent* – the one that lets to choose how to align the children within the main axis of their container; – which is working nicely in combination with *alignItems* that is also about children alignment but along the cross axis; *alignSelf* was useful when I needed to change the alignment of a single child. I also remembered how to use *absolute* and *relative* positioning when I needed to position the popup building selector relatively to the button that would trigger its opening (React Native documentation 2022 a).

Looking back at the goals list set up at the beginning of this week, I managed to finalize the last tab and the new booking screen UI for the booking screen, I progressed well on the tickets for the new release that were assigned to me, the iOS development environment was set up and I came to a better understanding how the component is being populated with real data. As I was needed to help prepare the app for the next release, I had to switch my focus to more important tasks at the current moment rather than learning how to access real data instead of using placeholder data in improved booking flow.

3.2 Observation week 2

The main goal of this week is to implement the new building selector that I started working on the week before as it is a desired new feature for the next Smart Office app release. Also, I would hope I manage to get all the fixes and code improvements done on time after my PR will be reviewed, and I would help my colleagues with the release process.

Monday

Today I planned to change the implementation for the building selector modal to include dividers and buttons, to make the building to be chosen after the confirmation button was pressed, to memorize, and to show the last building selection even after app was closed or reloaded. After giving it a thought, I decided to not include the dividers and buttons into building modal component as I felt it is better to keep it simple and pass desired elements as building modal children.

I added the translations to the button labels and other text because we are not passing it straight to the prop but rather keep in files in corresponding languages and pass it using i18next framework for the internationalization. I was able to get the chosen building to be shown on refresh using local storage concept to get and store the selected identifier (ID) but while manually testing the app I encountered the problem that when I would sign out and sign back in, it would display an empty string as a button title. I have been trying to solve this issue but unsuccessfully.

Tuesday

Today I aim to continue implementing the building selector. I need to solve the issue from the previous day, move styles to *StyleSheet* and make code cleaner, I plan to open the PR for this component at the end of the workday. After daily my R&D department had sprint discussion we would say where we stand at with ticket progress and if we feel we will be able to merge them today or at least push changes for review.

Then I went straight to debugging. I used the React Developer Tools extension in the Google Chrome browser for this purpose. I started by checking what values are returned for selected ID when the user choses item on the popup list, if the correct array item index is being assigned on click, and if the title state is getting set according to the choices. I noticed that meanwhile the selected ID was saved on refresh, and the title would change on confirmation button press correctly, the title value was not saved on reload. I decided to

store the title value in the local storage, and it worked but for some reason I was getting wrong name but correct ID and index for the first building option.

As the working day was coming to its end, I cleaned the code and pushed my current implementation which I felt was a bit too complicated so that more experienced teammates would get the chance to review it and I would have time to fix things tomorrow. Overall, I had eighteen comments in my PR, most of which were about wrapping titles into custom typography component since the font weight was showing too bold without it, and my teammate also noticed that the implementation could be done simpler even though the current UI is fine, and the component works as expected.

Wednesday

I need to implement the requested changes from PR comments for my building selector, get it approved and merge the branch with base branch. As it was important to get my component done as soon as possible, my teammate made most of the code improvements, but I had the opportunity to go through them and study why it is better this way and compare with my own implementation.

I did a few style-related changes, requested review again and was able to get it approved and merge to the base branch. There was no work left for me to do for the release anymore due to my experience level at the moment, so I was instructed to go back to working on improved booking flow milestone, and that I might be needed when the test version of the app is out to help manually test it.

While working on the building selector and getting more and more familiar with the project, I got a few ideas how to improve my code for the booking screen UI implementation that was waiting for the review. I started from removing some not necessarily variable declarations such as *const color = colors.blue*. It makes more sense just to use the value directly in places where needed.

Thursday

Today I plan to continue improving my code for the booking screen. When I had just started working at Steerpath, I would run the app on the web, and then run it on Android phone to do final tests. But I learned the hard way that what works on web, it will not necessarily work or look the same way on mobile device so now I run the app on both platforms at the same time.

I started getting the error in the terminal after executing a command to run the app on Android. The error meaning was that it could not get the Android application package (APK) installed. I fixed it by installing the APK with Android Debug Bridge (adb) command-line tool. That helped, and the project was built without further errors.

I noticed I had missed some styles from Figma which I added today. Also, I refactored pin location button component which is essentially a text button with the icon either on the left or on the right. To improve naming, and enable reusability, I renamed it as icon button and allowed to pass icon name, icon position, text, icon, and whole component styles via props. I disabled arrow buttons for duration picker and time picker components when certain conditions are met, and it allowed me to get rid of some blocks of code that would control *onPress* behaviour. I committed my changes and pushed the commit to opened PR.

Friday

I aim to start new ticket related to improved booking flow and help my teammates to manually test the app for the release. I started working on a new ticket, which was partly done by my colleague, they implemented new booking section to home screen for me to start working on new booking screen UI when I just joined Steerpath. I needed only to add translations for titles and accessibility hints and implement new team section.

I was able to complete the translations and then I moved to the testing of the app for the release. For distributing iOS builds for testing, we use TestFlight. I downloaded it on my device, was added to the test group and started testing following the list of points what should be done to make sure there are no bugs, for instance, in booking the meeting space via instant booking view or directly from map and so on.

After that the build for Android testing was ready. The set up did not go as smooth as for iOS. I was not able to find the app from play store even though I was added to the test group. At the end of the day, I was granted the access to the Google Play Console, where I managed to find the APK file to download.

Weekly analysis

I believe I improved my multitasking skills. During the second week, I switched between different tasks more often than the week before. Also, I was trying to take more responsibilities by offering my help with the release, I think it helped me to improve my communication skills and working relationship with my teammates. I assume it is important

to let colleagues know that you are willing to ease their workload even if you are only a junior developer, but it shows the motivation and desire to grow and learn.

I learnt the concept of internationalization. Internationalization simplifies the translation of the React Native app into multiple languages. Basically, internationalization (i18n) eliminates hard-coded text in the code, maintenance is easier as all text is located in one place, the time to localize the app in other languages gets reduced, and overall, use of i18n leads to customer satisfaction. It was written specifically for JavaScript (Wickramasinghe 2020). To get the *t* function and i18n instance inside the component we use *useTranslation* hook. As stated in React i18next documentation (2021), the *t* function is needed to translate the content and the i18n instance is used to change the language.

During this week I figured out one of my weaknesses – I tend to overthink my code and end up making the implementation too complicated when it could be done with fewer lines of code. I think it is because I do something one way, and if something does not work the way I expect – I keep on trying to make it work, instead of going back and reflect what could be done to make it easier to add more functionality. As an example, I was working on the building selector most of this week. It was already saving the value for the selected ID in the *useMemo* hook, I just needed to get the building title updated each time the confirmation button gets pressed, and it would remember the title accordingly when the selected ID value would get sent to the child component. Instead, I chose to find a workaround using local storage to get and store the value of the selected ID and to set title state in a parent component. I think it is mainly due to my lack of solid knowledge about React hooks.

```
25  useCallback(() => {
26    someFunction(x, y);
27  }, [x, y]);
28
29  useEffect(() => {
30    const timer = setTimeout(() => {
31      doSomething(a);
32    }, 1000)
33    return () => clearTimeout(timer);
34  }, []);
35
36  useMemo(() => {
37    someFunction(a);
38  }, [a])
39
```

Figure 1. Syntax for *useCallback, useEffect* and *useMemo*

The *useMemo* hook is a build-in React hook which takes two arguments – a function compute that computes the result and the dependencies array. During the initial rendering,

useMemo calls compute, saves the calculation result, and returns it to the component. If the dependencies do not change during the next render, *useMemo* does not call compute but rather returns the memorized value. In case the dependencies got changes during re-rendering, *useMemo* calls compute, memorizes new value, and returns it (Pavlutin 2021).

Now, that the use of *useMemo* hook is clearer to me, I would like to discuss the differences between *useCallback*, *useEffect* and *useMemo* hooks (the syntax for them is displayed in Figure 1). The *useCallback* is a React hook that returns a memorized callback when passed a function and a list of dependencies as parameters. It is greatly beneficial while a component is passing a callback to its child component to prevent the rendering of the child component. It modifies the callback only while one of its dependencies receives changes.

The *useMemo* is much like *useCallback* hook because it accepts a function and the dependencies array however it returns the memorized value returned by the function that is passed. It is beneficial to avoid recalculations on each render in case when the returned value is not going to change. The *useEffect* hook helps to perform mutations, subscriptions, timers, and different side effects after all the components were rendered. The *useEffect* hook takes an imperative nature function and dependencies array. When its dependencies change it executes the passed function (GeeksforGeeks 2021). The *useEffect* hook can be especially useful for cleaning up, like unsubscribing subscriptions etc., as the function that is passed to *useEffect* can return a function, and that function is called before the component is destroyed.

As for my weekly goals, I managed to finalize and merge the building selector I was working on, and it is going to be included in soon-to-come release for some of the Smart Office customers. I also got the chance to participate a little bit in the release process by taking part in manual testing on mobile devices. I feel this week was the busiest so far but nevertheless, it helped me to learn a lot of new concepts and understand how to become a better developer.

3.3 Observation week 3

This week I am going to continue working on new improved booking flow. There are many open issues for which I could do a UI, as well as try to hook up the screens, views, and components with real data. My help is still might be needed to perform more manual testing for the release and get a sense of what my team should focus on during the next sprint.

Monday

Today I plan to continue working on the improved booking flow. I picked up a ticket where I needed to create a new team section for the home screen that I partially completed last Friday. It was the easy one, but I struggled with the way how I should implement the background icon. For the booking section, there was used the section title component where I could set up a title, “show more” icon button, description, and the background icon.

I decided to use the section title component. I needed to make a few additions, such as a prop *iconName* which would make the component more customizable since I needed to have simple plus icon instead of “show more” text with the right arrow icon. Since originally the icon background was implemented out of the section title component, I changed it by adding the icon name to the *backgroundIcon* prop.

The chosen solution was not ideal, I did not have time to look into it any longer because the Smart Office build was ready for testing, so I ended up spending most of the day testing the app and reporting found bugs. Some of them were already known issues, others would be fixed in the next release, so the bugs discovered were not having high level of importance and they would not block the release.

Tuesday

Today we were having the team photoshoot at the office, so it was quite a hectic day trying to keep up the focus on work. Nevertheless, my teammate and I found time to try and figure out why I could not see correct versions of the apps opened for testing in play store. Somehow even though I was on the list for the internal testing, I could not see the correct version of the app. But once the release was moved to the beta phase, I could easily find it.

I have done some more testing for the release. In the process I found a bug that would throw an error on log out when logged in with the email and password option. The app would just quit instead of opening the initial screen. I reported the issue to my teammate, and it was decided it is important to get it fixed before the release is in production.

I was also able to continue working on the team section. Yesterday I moved the background icon implementation to the section title component, as it did not require any additional styling from me, and seemed as a good idea at a time, I ended up spending the rest of my day trying to make it work which stopped to seem possible as I needed the background item and the description to disappear both from the home screen and a team screen once the teammates were added to the list.

Wednesday

I started this day by doing more testing for the release version with the latest fixes. I learnt how to switch between two environments inside one project. This time I did not spot any bugs or design imperfections, the main goal was to make sure that the main functionality is working and that the fixes that were done by my colleagues are actually doing their job.

After I was done with testing and had a bi-weekly meeting, I continued figuring out the best way to implement the background icon. I have struggled some more trying to get the *backgroundIcon* from the section title component to work. I needed to hide the title and the “show more” button if the team screen is opened, hide the text and the background icon once the list of teammates is not empty anymore.

After my solution did not seem to make any sense anymore as it required way too much complicated customization for this specific case, I decided to go back to the original implementation and try to make it work. There was not much time left so I started by rolling back to the state before I changed the icon implementation and scheduled to finish the rest tomorrow.

Thursday

Today I finalized the new team section for the improved booking flow. The problem was that the original implementation for the background icon would not allow me to position it in the way as it was designed. But with the use of the section title component, it was really easy. But it was ultimately the best option to keep the original implementation because the title and “show more” icon needed to be displayed only on the home screen while the description and the background icon needed to be shown both on the home and on the team screens if the teammates list is empty and they need to disappear if there is at least one teammate added.

Once I finalized the team section, I moved to another milestone. I continue working on the Smart Office app during this sprint, so I moved to making small improvements for the next release. There are cards for each bookable meeting or resource space that contain the basic info such as resource name, capacity, description etc. There was a bug that when there is no resource description available, the message “No description” would always show in English even if the language changed to Swedish, Finnish, or Norwegian.

That was because the value was hardcoded in the helper function. As I could not use the hook for internationalization there, I was able to find another place where the description was rendered but it was not obvious because the hardcoded value would not allow it to ever be rendered. Afterwards, I moved to changing the old building selector to the new one in the resource screen as it was the desired change from the customer. Earlier it was replaced only in the instant booking view on the home screen.

Friday

I continued working on the replacement of the old building selector today. As there was no need to memorize any previously chosen values like it needed to be for the instant booking view, I managed to complete this task relatively quickly. The most difficult part was to re-write the *handleChange* function.

Next, I started a new ticket related to the button that had some unwanted shadow border around it on Android OS. That was also an easy fix as soon as I figured out that the button container was at fault. I set the elevation to zero and it was the solution. Another workaround could be to set the shadow color to transparent which was my first intent but after googling on the topic, I realized the elevation is a better choice.

After finishing the button shadow ticket, I needed to fix the bug when the title value on the resource or meeting card would not get updated when the location is changed. If the user does not specify the title, it is supposed to be the same as the location name. It was easy to fix for the resource card as there was no option for the user to type the title by themselves. But the meeting card fix was a bit trickier. I was not able to finish this ticket today and decided to come back to this next Monday.

Weekly analysis

This week it was clear to me that I started to feel more confident with Smart Office codebase. The time spent trying to find where the source of the problem is, was reduced a lot. I can already feel how much faster I am reading through the code I did not write and the most

important is that I have to google or ask my teammates less to understand what is what and what it does.

I learnt a lot during this week. For example, when I needed to remove the shadow border of the button container, at first, I just simply changed the color of the shadow to transparent but then I found out there is a property elevation exists. It works only for Android (that is precisely the platform where the issue occurred). Elevation basically adds a drop of shadow to the item and affects z-order for overlapping views (React Native documentation 2022 b). When I set it to zero, the problem was solved.

Also, there was another misunderstanding I had about style property *zIndex* and how styling works in React Native. As a person working mostly with CSS for styling, I did not have much knowledge how styling in React Native works. Initially I thought what we use to style elements in React Native was CSS with the difference of how we write the name of the properties but after I started learning React Native, it turned out that what we use is a *StyleSheet* abstraction that is only similar to CSS *StyleSheets*.

Regarding *zIndex*, I learnt there is a difference with CSS when it comes to it. The property itself allows us to control the order in which the elements are displayed over one another. In CSS, *z-index* will not work unless the absolute positioning is set for the element. In React Native, you can use it without position property. The property *zIndex* is really powerful, with it you can position the element wherever you want using properties *top*, *right*, *bottom* and *left* (Expo documentation s.a. b).

The biggest discovery I had during this week was the changing the environment for different versions of the app in one codebase. We have different customers who have various versions of the Smart Office app, for some reason I was thinking I would need to download another project to run them but all that was necessary to do is to switch between them with one command – *cat env.<environment name> -> env*. Turns out it was configured with react-native-config. This has several benefits such as the environment variables can be used in native code, not just in React Native code; the variables can be stored in different files related to different development environments, and the logic how to switch between the environment variables is relatively simple (Kostin 2021).

As I mentioned before, Hooks are a particularly important concept for me to understand. In the previous weeks I learnt the differences between them, but I did not learn the rules of how to use it. That was a loss, as I spent extra time while trying to put it where it was not supposed to be. For example, I needed to add the internationalization to the helper function, so I tried to use the Hooks inside it, but I should not have because the Hooks must be called

only from function components or from custom Hooks. This is important to ensure that the stateful logic in a component is clearly visible from its source code (React s.a.).

This week I was able to work a little bit on the improved booking flow, but I did not have the opportunity to start hooking the UI that I created to real data. That happened because we were having a release for the Smart Office app in progress since last week, and I spent most of my working hours on testing the release versions.

3.4 Observation week 4

This week I am going to continue working on the tickets for the next Office App release. I still need to make the title to update when the location is changed, and no title was specified by the user for the meeting. Otherwise, I do not have any specific things planned to do at this point. I will just see what tickets are free and pick up whatever I feel I can do, or I might also have something assigned to me by my supervisor or more experienced colleagues.

Monday

Today I need to figure out the rest of the fix for the title update in event details after the booked space is edited when the location is changed, and the user did not add the title for the meeting. It should work this way: the title takes the location name if the title input is empty, if the location was updated to the different one – the title changes accordingly if new title is not updated by the user, and if at any point the user specifies the title – it is supposed to stay the same even if the location changes, unless the user changes it or the title field is empty.

Described above actions are happening when the “edit” or “reserved” button is pressed. I located where the submit event happens and added the assignment for the title to be equal to the location name if the edited event title is the same as the edited event location. This way I was able to achieve the desired result but only partly. As edited event values are the previous values that the title and the location fields had, it was not considering the cases when the user would decide to give own title to the booking, it would take own title only on the second edit, and once the location would be changed – the title would become new location’s name.

I fixed it by adding an extra condition using logical and operator. Now, for the title to have the same name as the location, two conditions must be met. The edited event title and location and the submitted values of the title and location should be equal to one another. That fixed the problem I described in the previous paragraph. I tested my solution and sent the PR for the review. I also started working on the new ticket which was created a few months ago. As I spent almost the whole day for the title issue, I only managed to set up the environment for my next task and decided to continue with this tomorrow.

Tuesday

Today I continue working on the new ticket. It was about some booking filtering issue, where if you book the space for the specific date and time in the future, and then choose this date

and time again, when you see the list of the available spaces, you will still see the space you booked the first time even though it is actually unavailable and should not be showing. I tried to replicate the issue, but it seemed to work just fine with the latest app updates, so I closed the ticket and moved to the next one.

I needed to fix the issue with the emails being queried from our backend when the user wants to add other participants to the meeting for one of the Smart Office app implementations. It was happening because the guest list would be populated with the result of the concatenation of the user's own contacts and the data from out backend. I solved it by using the environment name, and if it is the one running, then the list would only populate with the contacts of the user.

It was a valid solution, but I did not consider the performance. It would be better to specify the environment in the function where the guests array gets data from the backend. This way if the specified environment is in use, the array would not get any data at all and would just return an empty array. With my first choice for the solution, the array would be populated with data even though it is not used.

Wednesday

I got a comment to my previously submitted PR because I did not realise to test one corner case. When the user books a meeting, and then chooses not to include any resources – the location name should say "No location provided" and the title should be "No title". In this case the location would change correctly but the title would show the last value it had. If create a meeting without a resource initially, it would behave as expected.

At first, I thought I broke it with my code. As it was just a few lines I added, I decided to roll back to the state before I made any changes and test the behaviour then. It turned out that problem was always there, so I did not cause it. I brought back my changes and started to figure out the source of the issue. The solution was to simply add the assignment for the title to be "No title" string when the event is a meeting, and no space is chosen.

Next, I got another ticket assigned which was probably the biggest I worked on so far and relatively difficult as I was told, so I accepted it gladly as a chance to challenge myself. When the space gets booked but the room is not accepting the booking, or the booking is being made outside the working hours, it would all trigger the same error message "Something went wrong. Please try again". Also, if the space declined the booking, it would still appear in the schedule. I needed to check that the room accepts the invitation, if yes, the confirmation animation would be triggered, if the room has declined or not confirmed –

the popup dialog would show up with three buttons that would direct the user to edit screen, cancel the booking or remind about the problem later.

Thursday

Today I started from figuring out the ways how to trigger the room status check. After researching the code where the confirmation animation was triggered, my idea was to use the asynchronous function with *await*, and *setInterval*, as the check is supposed to be done up to four times with five hundred milliseconds interval. It took some time for me to find the way to do this as I was not familiar much with asynchronous functions, *awaits* and had some doubts if I should rather use *setInterval* or *setTimeout*.

I ended up by using *setInterval* which would be converted into Promise; and resolving the Promise and clearing the interval once it was executed four times. I would then perform testing, and it seemed to work as it should until I tried to replicate the situation when the room would decline the booking. My solution would still return the room status as accepted even when it is not.

That is when I understood I took the wrong values for the check but on the other hand, the whole structure of the code block that would check the room status for four times before doing anything seemed correct. I was trying to find the part where the room status is being checked but there were no similar cases in the code base, then I made the decision to consult with my teammates tomorrow.

Friday

I have still tried to find out how to get the actual room status on my own for a little while today. But then I asked for my teammates help. It turned out I would need to pull the actual refreshed room status from the server and for that, I need to write the calendar API query of my own for Outlook, Google calendars and Timeworks reservation manager.

As I had little to none experience in the matter, my supervisor took time to sit with me and walk me through the possible solution. It gave me a good idea of what should be done and, at first, I tried to make the offered solution to work but it would not get the space ID as should. Afterwards, I looked into how the functions to create and update events were made and adjusted the solution so that it would pull the ID from the server, and then it would be used to check if the chosen room and the server data would have the same ID before the room status is retrieved.

As I was able to get the room ID and status check to work correctly closer to the end of the working day, I decided to test the solution more and continue the implementation next week. There are a lot of corner cases to be considered and, also, I need to confirm what kind of stuff should be happening when the user is trying to book the space outside the working hours as so far the popup is only supposed to cover the cases when the room was declined or not confirmed.

Weekly analysis

I have not overcome my bad habit of overfocusing on trying to get one solution work just yet. For instance, this Friday I was able to get great hints on how to proceed with checking the room status with data pulled from the calendars API but instead of thinking how to adapt it best, I focused on making it to work with minimal changes. It makes me less efficient, and it is something I should definitely improve. I believe it is great I am able to keep focused on the problem, but I should also learn how to look at it from different angles at the same time so that I can be more productive and flexible.

```
25  const someFunction = async () => {
26    await anotherFunction();
27  }
```

Figure 2. *Async / await* syntax

The new concept I used during this week was *async / await* which was not completely unknown for me, but it was something I needed to read about more. The basic syntax for the *async / await* displayed on the Figure 2. *Async* keyword is used to define the function for the *await* task use. If *await* is needed for a wait task, then it is necessary to define the *async* function first. The *await* keyword stops the execution till task response, and there can be multiple *await* calls in one *async* function according to the Infinitbility (2021).

The *async / await* is a good option to return response in *async* function as it simplifies the code and gives a feel of synchronous coding (Mittal 2021). On the other hand, the *async / await* can make programs non-linear; it is necessary to think about the workflows asynchronously, then try to write code synchronously with await; the error handling might be tricky with *async / await,* and to use it properly, the complete understanding of promises is essential since the *async / await* is nothing more than syntactic sugar, and the underlying technique is still promises (Li 2018).

A promise is a proxy for a value not necessarily known when the promise is created. It allows to associate handlers with an asynchronous action’s eventual success value or

failure reason. Therefore, the asynchronous methods return values just like synchronous ones. Meaning, instead of returning the final value right away, the asynchronous method returns a promise to supply the value at some point in the future (MDN web docs 2022).

```
35  const waitUntilFunction = async (someCondition) => {
36    await new Promise((resolve) => {
37      const interval = setInterval(() => {
38        if (someCondition) {
39          resolve('resolved');
40          clearInterval(interval);
41        }
42      }, 1000);
43    });
44  }
45
46  // call later with
47  const someVar = waitUntilFunction(someCondition);
```

Figure 3. A way to use *setInterval* with *await*

That is tricky to use the *setInterval* with *await*, as basically the promise can be resolved only once but, in my case, I needed it to be resolved multiple times. There is the best possible way of using *await* with *setInterval* displayed in Figure 3. There are many opinions on this matter. Some say that it is possible to achieve, others believe it is better to use a workaround by passing the function as a callback and invoking it as many times as necessary. Also, there is an opinion that it is not safe to resolve promise multiple times as it is basically a bug that is hard to catch because it is hard to reproduce.

After diving deeper into promises, I understood that resolving a promise multiple times is a no go. It made me to reconsider my current approach and taught me to research more before implementation as it is something that would reduce the time that I spend for the tasks I do. Of course, it is also an important practice for me. Next time I will be able to plan the implementation better since I gained a good understanding on the topic.

```
25  const someFunction = async () => {
26    try {
27      await anotherFunction();
28    } catch (err) {
29      console.error(err);
30    } finally {
31      console.log('I will be printed no matter what');
32    }
33  }
```

Figure 4. How to catch errors with *async / await*

Another interesting concept is the error handling which is crucial when using the *async / await*. There are several statements that are in use in such cases (see Figure 4 for the

visualisation): *try* statement is defining a code block to run, *catch* statement defines a code block to handle any error, *finally* statement defines a code block that is run regardless of the result and *throw* statement defines a custom error (W3Schools s.a.). *Try-catch* only works for imperative code.

Generally, there are two types of errors in React Native: JS Exceptions which are errors that are produced by JavaScript code including React, and Native Exceptions which are errors produced by Native Modules. To handle errors in React components which are declarative, the error boundaries were introduced. Error boundaries are React components that catch JavaScript errors anywhere in the child component tree. To handle Native Exceptions, the react-native-exception-handler library can be used as well as it works for JS Exceptions handling (El Azizi 2020).

I think this week I was able to be productive even though I take my time for completing the assigned tickets. I need to remind myself more often that I am a Junior Developer for now and as much as I am expected to grow professionally and handle things quicker, nobody expects it from me right away. Every developer was at the same position that I am at, all it takes is practice and time. I can already see how much I am a better developer now than I was a month ago.

3.5 Observation week 5

This week I am going to continue working on the implementation for the dialog informing the user if the room was declined or not confirmed. As well as I am planning to implement the error dialog for the Timeworks reservation system in case when the reservation cannot be made when the space is being booked outside the working hours. I am hoping I will get to review some of the PRs of my teammates and find out if my current ticket would also solve the problem that one of the customers reported about not being informed when the space is booked for the time when it is not available.

Monday

Last week I ended up successfully pulling a single event from Azure calendar. Today I will implement getting a single event for the other calendars and reservation systems such as Google, our own calendar that is based on Google calendar and Timeworks. That is relatively easy to do with a *.get* request if such API call exists.

Afterwards, I was able to get the final response status of the space that is being booked and added the functionality for the editing and the cancelation of the meeting. On edit, the dialog form should disappear, and the user is redirected to the edit reservation screen for the event using React Navigation. On cancel, I decided to use helper function for deleting the event by passing the event ID. The reservation can be made from instant booking view on the main page, reservation or edit screens and from the resource card, I implemented the booking polling and the dialog popping up for each of them.

Tuesday

Yesterday I noticed that the room status check was working as expected only with Google calendar. I started my day from looking into the reason why this happens. I was introduced with the new way for debugging using the network tab while inspecting the browser page. It turned out the Azure was returning proper room status, but it was not sent properly into my function. After some time, I figured that was happening because the properties names used for the conversion of the room statuses were written with a capital letter, once I changed them to start from lower case – my solution worked fine with Azure calendar.

While debugging Azure calendar, I noticed that the first few checks would return *needsAction* status before stating if the room accepted or declined the booking. It made me think that it was not right that with the Google calendar I would get *accept* as the first response. After investigating this, I found out the *accept* status was hardcoded in one of the

functions that would receive the event from the API. I fixed it by assigning the interface for response statuses and then I started to get the correct response on the booking polling.

Then I performed more testing for Timeworks, Azure, and Google calendars. The latter two were working as should with one exception for Azure but the Timeworks was not behaving as I would expect it to. I noticed it takes minimum eight seconds for the room to update its status with Azure calendar. That complicates the user experience, but it was decided I keep on developing the solution as it is and once it is completed, we would re-think when exactly the dialog should be triggered.

Wednesday

Today I needed to consult with my teammates about our own calendar that is based on Google calendar. It seemed I had the correct API call implementation for getting a single event, but it did not work at all. When I tried to book anything, I would keep on getting network errors and nothing would happen. It turned out the path was missing for the API call. I happened to test the Hypertext Transfer Protocol (HTTP) requests with Postman for the first time, the errors that the request would return were helpful for my teammate to figure out the issue.

I was also advised to test the Timeworks behaviour when the room has declined before my solution was implemented, and it happened so that the Timeworks does not allow double booking anyway. Though, it would throw the same error message no matter if the room would decline or if the booking is done outside working hours.

I decided to leave the issue described above for tomorrow and started looking into dialog design. We decided to use similar design to the one for the Spacehub app, so I started from looking into how it was implemented there to either copy the files into common resources submodule or adapt the implementation for my case. I also implemented the functionality for when the user clicks “Remind me later” button, which would basically close the dialog and make it reappear again after specific interval if there are any declined or not confirmed spaces.

Thursday

When I had some question during the previous days, one of my teammates noticed that my implementation is alright while I am testing but not an ideal solution as it would only resolve the promise once. It was decided I would use the original suggestion with a *for* loop polling the event four times with the *getEvent* function. That is what I implemented today. As we

are using ESLint to find the problems in our code, and the *for* loop would contain two *awaits*, I would get a warning *no-await-in-loop*. After I investigated the problem, I decided to ignore it as in my case I did not want the promises to be resolved at the same time but rather one after another.

Then I implemented the check for the rooms' statuses of the booked events. It was done with the *useEffect* hook and a *setInterval* so that the room status would be checked in a specific period of time and once the room status is declined or not confirmed the dialog would be opened, if all are accepted spaces, then nothing would happen until the next check and the interval would get cleared each time.

I did more testing today as well. One of the rooms was configured in a way that the room would release the booking if all, but one user has declined the meeting. That did not work as expected as it would return the final room status as both undefined and accepted for some time so the dialog would get opened even if two users accepted the meeting. If all but one declined the invitation, the room status was still accepted. I think it might be the issue is coming from the configurations.

Friday

Today I performed more testing. Another meeting room was configured so that it would not accept the invitation automatically, the user should accept it first. The user has three options: reply "yes", "no" or "maybe". "Yes" corresponds to accepted room status, "no" is declined and "maybe" is tentative. This test case seems to work as should with my solution.

I added the translations for the text values of the dialog as that is needed for the internationalization. Our Smart Office app supports four languages at the moment – English, Finnish, Swedish and Norwegian. Usually, I use a translator or google search if the translation consists of just one word, or it is a simple sentence. In this case I copy pasted the translations in English to the other languages files without modifying it and added a comment that it is something that should be done as I am not proficient enough in any of the other languages but in English and the phrases are not that simple to trust translator.

For the rest of the day, I have been trying to figure out how to make the Timeworks work, how to trigger different dialogs when specific conditions are met since it does not return the room statuses. I started the investigation from finding out where the opening hours restrictions are coming from. After struggling with it for a while, I asked for help. I was advised to get the error message from the API and return more specific error dialogs based on that.

Weekly analysis

This week I improved my organizational skills. As for the whole week I was working on the same ticket that would include many steps to complete, I learnt to plan the next day more thoroughly instead of jumping chaotically from one thing to another. In most cases the plan would come from issues that were discovered during the previous day.

```
25  const Stack = createNativeStackNavigator();
26
27  function App() {
28    return (
29      <NavigationContainer>
30        <Stack.Navigator>
31          <Stack.Screen name="Home" component={HomeScreen} />
32        </Stack.Navigator>
33      </NavigationContainer>
34    );
35  }
36
```

Figure 5. Basic React Navigation elements

I have been waiting when I finally get to start working with the React Navigation. This week I was able to dive deeper into it. We use React Navigation because React Native does not have a build-in idea of a global history stack like a web browser does. It allows the app to transition between the screens and manage navigation history. The basic React Navigation elements displayed in Figure 5. To create a native stack navigator, the function *createNativeStackNavigator* is needed (it returns an object of two properties – Screen and Navigator). Navigator contains Screen elements as its children to define the routes configuration. Screens inside the Navigator are called routes. Each Screen component takes a name prop (refers to the route name) and a component prop (specifies the component to render for the route). *NavigationContainer* is a component that manages the navigation tree, and it must wrap all navigators structure, it is something that should be rendered at the root of the app (React Navigation s.a.).

It is possible to pass data to routes by passing params to a route when putting them in an object as a second parameter to the *navigation.navigate* function. Then the params can be read in the screen component with *route.params*. The params can also be updated by the screens with *navigation.setParams* method. As a rule, params should contain minimal data required to show a screen and nothing more than that (Marks 2020). I used params to pass the event details when the user would click the edit button on the dialog in cases when the booking was declined or not confirmed.

I noticed that *useNavigation* hook was used a lot across the Smart Office app. This hook gives access to navigation object when the navigation prop cannot be passed into the component directly or if you wish to avoid passing it in case of a deeply nested child. The *useNavigation* returns the navigation prop of the screen it is inside (Bugl 2019).

While working with React Navigation, I started wondering about the differences between React Navigation and React Native Navigation. The former uses a combination of libraries to provide a near-native experience, and the latter utilizes native fragments for each screen. As React Native Navigation treats the screens as individual React applications, it makes the app more performant by nature. React Navigation has built-in hooks while React Native Navigation has a community library which contains a set of React hooks. Using multiple routers is relatively straightforward for both solutions. It is easy to integrate React Navigation with any third-party libraries and existing apps compared to React Native Navigation (Sajjad 2021).

As for my weekly goals, I managed to implement most of the functionality required for the ticket I was working on since last week. I managed to discover several bugs while testing my solution at different stages of development, tested and reviewed the PR of my colleague for the first time, and got a good idea of how to proceed with my task next week.

3.6 Observation week 6

This week I am aiming to finish the ticket related to informing the user when the room was declined or not confirmed. I need to make a modal with more specific error message for Timeworks when the user is trying to reserve the space outside working hours, or the space is already busy. I am going to clean my code overall, make sure all variables and function names make sense, and open a PR. I will also be responsible for the Smart Office app release process for the first time, so I would need to start the release at some point this week.

Monday

Today I investigated Timeworks case as there it would not make sense to use the dialog for informing the user about the room availability. Timeworks do not allow double booking by default, and it is impossible to reserve anything outside of opening hours. Therefore, I decided to get the error message from the API response to show different error dialog in these cases.

At first, I needed to figure out how to check the network traffic when the app is running on the actual mobile device to see what exactly the error is that the API returns in its response when the user attempts to book the space that is busy or if the booking is done outside opening hours. That did not go as smoothly as expected.

I did not know how to do it so initially I turned to Google. There I found several options how I could achieve it. First, I tried to set up a proxy server with Postman. This did not work, and after I tried to follow the steps from the beginning, I started getting an error message when I was choosing to start the proxy. Then I found Fiddler tool, which is like Postman. I started following the instruction steps for this tool to see if the proxy server set up would be easier and more reliable there. At the end, I was able to see the network traffic on web, but not when I used the app on the mobile phone.

Tuesday

Today I started from checking variables and function names for my ticket, I cleaned the code a little bit and opened a draft PR as it would be good if my teammates could already start checking my code and the functionality since my ticket is quite big. I got a few comments on how to improve my code, and I continued focusing on that. It included removing *async* where I did not have *awaits* anymore, I also had a few duplicates for the function calls here and there.

It was suggested to change simple error dialog to the new style dialog that I made for informing the user about the space status. I also got help with checking the network request when there is no web version of the app. It was easy to do with Android Studio network inspection. I also had a discussion with my teammates regarding my ticket. It was decided the space status dialog should be more specific, so I need to add a meeting room name and date and time to the dialog message. After this I continued with smaller fixes for my PR.

Wednesday

Today I added a room title, date, and time to the space status dialog. It was achieved by adding a property to the translations in the format *%{propertyName}*, and then assigning desired value to the properties where the space dialog is rendered. Since my teammates suggested to reuse the space status dialog for other error messages, I renamed the component to have more generalized name. Also, I made it more customizable by allowing the developer to add own styles for buttons, choose if they want to have the icon next to the button and if they need the text to be uppercase or not.

Then I changed simple error modals to the new style dialog everywhere where those were used and deleted the simple error modal file. After that I started working on Timeworks case. As in the future there will be more specific error messages in the API responses to make it easier for frontend developers to handle each case specifically, I decided to add more generalized but yet more specific text to the error modal for now. To handle such error, I needed to access the API response error message and if it would contain specific error text – new dialog would be triggered. Otherwise, the “Something went wrong” error dialog would appear.

Afterwards, I needed to check if my solution would allow me to close one more ticket. There the customer noted that when they book the space, and then try to book it for a different time – the space would just change to a different one without notifying the user. In that case Azure calendar was used. I tested it with my solution, and it happened that after pressing reserve, the loading spinner on the button would not reset as it should once the booking is confirmed but it just goes on infinitely. After debugging I figured it was happening because I could not get the final space status as the space email would start from the uppercase letter and it would not pass the condition check in the function. I added *toLowerCase()* method to the email, and the performance was as expected then, therefore, my solution solved the problem that the customer reported.

Thursday

From time to time, I pull the changes from the base branch and merge my local branch with the base one to keep it up to date. I did so as well at the end of the working day yesterday, but the merge did not happen automatically. So, today I started from solving the conflicts to have a successful merge. After that I cleaned my code and performed more testing. Then I pushed my changes to the remote branch and re-requested the review.

Also, I reviewed a few PRs of my teammates. Currently, they add me as a reviewer only to those PRs that they feel are corresponding to my skills level, so it was relatively easy for me to test their solutions and check their code. I had release process handover with one of my colleagues. They have prepared the file with the instructions. We went through it together, and they explained some points for me that I did not understand. We also started the release process with creating the release branch out of the base branch, writing change logs and the comments with the tasks to perform before the release goes to production.

After we were done with the initial steps for the release, I checked if I got any more comments to my PR. As the space status dialog was used in four different files, I was suggested to create a reusable component for that dialog, which would allow to shorten the space that the code implementation would occupy before.

Friday

During some weeks we have Friday huddles. That is when someone who found out something interesting or solved some problem, they believe it might be useful to talk about it to other teammates, and we all get together if we are interested, and go through the problem and solution or technics used together. Today we had a Friday huddle about modifying packages, resolutions, and *yarn audit* command.

After that there were more suggestions on how to improve my code for informing the user if the space declined the reservation. For example, the polling time could be reduced for when the booking accepted or declined the reservation if the loop is exited once the space status gets those values, or the polling needed to be done every 500 milliseconds for up to 16 times instead of every 2000 milliseconds for up to 4 times.

While performing final testing, I found a bug for when the booking would be done without including any space. The fix was to perform the polling only if the booking has the space, otherwise just accept the reservation. Also, there was a bug related to one of the previous tickets I have done. The title and the location of the booking would not update properly on

edit for the Azure calendar. My colleague and I were looking into it, he found the solution, but it did not cover some corner cases, so this issue needs to be investigated further next week.

Weekly analysis

This week I noted to myself that I should perform more thorough testing for my tickets. For example, this Friday while testing different thing with one of the customer testing accounts, I found a bug that I did not encounter before because I only tested Azure with our organization's test credentials which has only one bookable resource so I could not choose the other resource to see how the solution performed. Then I had no idea that it could not work with some of the calendars because I did not work on them much during previous few weeks.

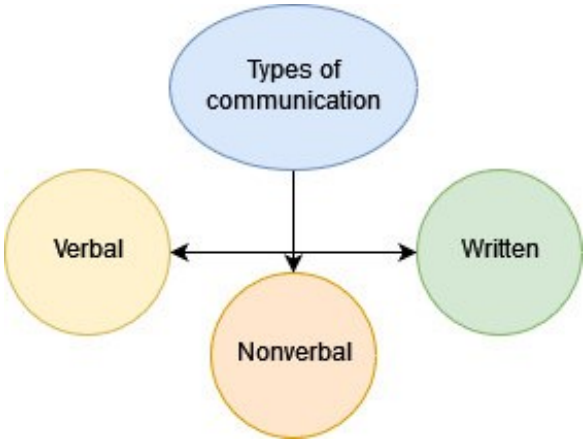

Diagram 2. Communication types

As there were a lot of new things I worked on this week, and I had to ask more questions than usually, I believe my communication skills improved more. Communicating clearly, effectively, and efficiently is a whole new skill that must be practiced, even though we communicate every day on a different level and in different situations. There are three types of the communication as displayed on Diagram 2: verbal communication type – it is the best to use a confident speaking voice with an appropriate tone and volume for the environment, filter words ("like", "well", "umm" etc.) should be avoided to improve conciseness and clarity of the message; nonverbal communication type which includes body language and face expressions, it is necessary to be aware of different actions that you do, such as eye contact, smiling, posture, the position and the movement of arms and legs; and written communication type – to make it effective it has to include sentence and paragraph structure, grammar, spelling, word choice, punctuation and tone (Hunter 2021).

During Friday huddle that I had this week, we were discussing the topic about modifying packages. If you need to modify or customize the React Native libraries while developing your application, you can do so in the node module. However, if you want to share these with your team members, you need to compress and share the library files. Otherwise, all libraries will be updated during the *npm install* (Node Package Manager) or *yarn install*, and changes may be lost. Patch-package is used to solve node module dependency issues. It helps the authors to make and keep the npm dependency fixes. Patch-package can change the code of the *node_modules* directory in the form of *patches* (Thomas 2020). To use patch-package, make direct changes to the package code of *node_modules* and run patch-package. A *patches* folder will be created containing the diffs needed to update the installed package version with your changes (Burger 2020).

There was a new command discussed during Friday huddle that I never used before – *yarn audit*. It is a good idea to keep track of vulnerabilities in *yarn*, and *yarn audit* is the right tool for that. *Yarn audit* is an integrated Yarn tool that checks for known vulnerabilities in package dependencies. It uses the official node.js and npm vulnerability databases like *npm audit*. However, unlike the corresponding npm, there is no *npm audit fix* feature. Dependency vulnerabilities are still visible, but the full upgrade path is not computed based on those dependencies (Debricked Editorial Team 2021). Luckily, there are several workarounds for yarn. The very straight forward option is to use *yarn-audit-fix* package to compose *npm audit fix* with lockfile converter (npm 2022).

Overall, this week I understood the project better, learnt to write cleaner code and improved my soft skills. As for my weekly goals, I finished the ticket related to informing the user when the room was declined or not confirmed. It was merged to the base branch at the end of this week. All the necessary and planned changes and code improvements were done, and the release process was started.

3.7 Observation week 7

This week I am going to do my first Smart Office app release for Android, iOS, and web. Last week there were some bugs discovered so I need to fix them. Then I am going to build and distribute the iOS and Android apps for testing using GitHub Actions functionality. I suppose this week will include a lot of manual testing, probably more bugs will be discovered, and more fixes will be implemented.

Monday

Today I investigated title and location updates issue. How should it work? When the user creates a meeting – if there is no user input on submit, the name of the selected space would be passed as a meeting title, otherwise, whatever the user has typed will be a title for the meeting. When the selected resource changes – the location name and automatically generated title should also change accordingly.

I decided to work with conditions. For example, if the title's length is zero, the location should be equal to the selected space name. I tried to manipulate what gets assigned by adding more conditions, but it never seemed enough. When I felt it was working as should, there would always happen some corner case that would make me to add, remove or edit the conditions.

I also did a lot of testing as basically after each new condition, it was necessary to make sure that it did not break the previous logic. In total, today I tested three different possible solutions: manipulating the output with more conditions; setting a state that would only get value if the user inputs anything, and then assign the user input to the submitted location if previously the user input was undefined but not anymore, and change the location or title if it is undefined again. After that did not work and seemed overcomplicated, I asked my colleague for some advice. Their idea was to try with a boolean type variable that will become true if the user inputs anything, and on submit – to check the value of this variable to decide what title or location should be.

Tuesday

Today I started from implementing solution that my teammate offered. It did not work as we thought it would. At some point I managed to find the perfect solution with the additional condition that my teammate advised me to add. As there are only two possible options what the title and location could be – either the user input or chosen space name – I figured we do not even need to detect if the user inputs anything or not.

It is enough to just find out if the submitted title or location is in the array of resource names. If it is not, then it is the user's input that we keep all throughout, even if the user changes the meeting space, and if it is in the array of the space names and the edited title or location is not equal to the chosen space name, then we change it to be the new space name. New solution reduced the lines of code significantly and simplified the logic.

I also had a conversation with my teammates about improving the UX for the space status dialog implemented previously. We decided it would make sense to add a better description to the space status dialog. As well there was a bi-weekly meeting where latest company news in different department is shared. Usually, different departments report what happened during previous two weeks sprint and what is the plan for the next one.

Wednesday

Today I added more description to the space status dialog to let the user know that the booking was added to their calendar but the space they tried to book has declined or not confirmed the reservation. The reason behind this was to improve the understanding of what exactly is being done if the user chooses "Remind me later" option.

The change regarding the space status dialog was done in the same PR where I implemented the solution for title and location updates. That is not the best practise if the fixes are absolutely not related to each other, but if it is something small, then it is fine to proceed this way. After this I read through the release process document to get familiar with the steps beforehand.

I started with pulling the latest changes from the base PR and merging it to the previously created release branch. Then I continued by building and distributing the iOS and Android builds with GitHub Actions. The Android build failed after fifteen minutes running, and it took about forty-five minutes for the iOS build to be completed. I re-run the build for Android and waited when the iOS Smart Office app will become available for testing in TestFlight app. The Android app build failed again, I had to re-run it multiple times while investigating the error that I found in the logs.

Thursday

Today I continued with the release process. I re-run the Android build again and it was successful this time. As for iOS, the app still did not appear in the TestFlight. I needed to consult with my teammate, and it turned out even though the build was successful, if I would

have opened the logs – I could see that there was an error related to some issues with the connection.

To solve that, I needed to log in to the App Center and re-connect. After that I re-run the iOS build and there were no errors. The app was available for testing in TestFlight after approximately fifteen minutes. That was a good thing that I happened to have this issue with iOS build because we were able to document it in the release process document so in the future, it will be easier to fix it for somebody running into the same problem.

I decided I would test the app myself first before informing the colleagues that the next release version is available for testing. I think it is better in case there is some easily discoverable and noticeable bug that I could fix, re-built the apps, and distribute them for testing again. So that my colleagues would be able to do more comprehensive testing with all necessary functionality working as it is expected.

Friday

I discovered a bug during yesterday's testing on iOS. When I would try to edit or delete the booking logged in with Google email, I get an error "Something went wrong". At first, I panicked because I thought my code broke everything as my previous fixes included improvements for the booking. Then I decided to test on Android and web because I figured it is possible but unlikely in this case that something that worked fine before would suddenly stop working now.

That was the right thinking. I was able to successfully edit or delete calendar events on Android and web. I started thinking what if the problem is related to the API calls and decided to check HTTP transactions. When the user edits the booking, the *updateEvent* method is being called, it includes a *.patch* call, and when the user deletes the event, *deleteEvent* method is called with the *.delete* request.

It took me some time to find out how to check HTTP requests on iOS. My idea was to use Xcode just like I used Android Studio when I was debugging another issue. I ended up using Instruments developer tool in Xcode. While inspecting the HTTP requests, I noticed that *.patch* and *.delete* requests would return a precondition failed error in their response body. I investigated the network traffic with Android and web to see what happens then, and also tested different calendars. Interestingly, this problem only occurs on iOS with Google calendar. I tried to solve it but did not succeed today, I will continue with it next week.

Weekly analysis

This week I mastered my ability to follow the instructions step-by-step while I was proceeding with the first part of the release process. I also believe my troubleshooting skills were improved further since I realised quite quickly that the root of the iOS error is in the API request responses while figuring out why there was the error when trying to edit or delete the booking. I have also been asking a lot of questions to clarify things, and I suppose every time I get to communicate with people, my communication skills get improved. I think due to my personality, I sometimes tend to give way too many details than necessary as well as because English is not my native language – I feel I can struggle to find the best words from time to time. But nevertheless, I notice I am able to express my thoughts in a more concise and clear way day by day.

As I mentioned at some point, we have bi-weekly meetings in the company. Those are basically status update meetings from every department about what was done and achieved during previous two weeks and what is planned next. The status update meetings are critical for the company success. They motivate the teams and help to make sure that the goals are on point, for instance, for estimated key performance indicators (KPIs) for the departments in the company I am working at (Avery Products 2021).

KPI stands for key performance indicator, and it is a quantifiable indicator of long-term performance for a particular goal. According to Wishart (2022), the KPIs are important for monitoring company health, measuring the progress, adjusting, and staying on track, solving problems, and tackling opportunities and analyzing patterns over time. To sum up, KPIs are essential for the health and success of any business, in my opinion.

While handling the app builds for iOS and Android, I came to realise I have never released a single mobile app before, and it was interesting for me to use the GitHub Actions. As Corti (2020) explains, the GitHub Actions allow to automate the tasks of the development lifecycle such as building, testing, analyzing, bundling, releasing, and deploying. Visual Studio App Center is another tool used for similar purpose. Both can be classified as Continuous Integration (CI) tools. App Center is easy to configure and set up, but it does not offer as many customization options as GitHub Actions. For example, one cannot change the target configuration to build an iOS app bundle. On the other hand, the GitHub Actions ecosystem is relatively young, so it may not have as many out-of-the-box or templated actions or workflows as other platforms (Varma 2021).

When I was looking for a way to debug the HTTP requests for the app running on the iOS mobile device, I ended up using Instruments Xcode developer tool. It is unfortunately

working only with Xcode 13 version and requires iOS 15 and a real device as a target. Nevertheless, I think it is a great tool considering I use Xcode to run the app on the iOS device. And with Instruments, I do not need to look for another solution to use on top of the Xcode. Harrison (2021) finds the Instruments interface hard to use but in my opinion, it is simple once you configure it the way you want. It just takes time to get used to it.

This week I planned to do my first Smart Office app release for Android, iOS, and web. I was only able to build Android and iOS apps so far and distribute them for testing. I managed to fix the bug with title and location update that was discovered last week. I have done a lot of testing, specifically for reserving a space, converting a space booking to a meeting, editing, and deleting events. As I suspected, more bugs were found during testing. There is no fix just yet, I believe it is going to be sorted out next week.

3.8 Observation week 8

This week I will need to fix the iOS issue for when attempting to update or delete the event. The app would have to be re-build both for iOS and Android. I will need to inform my colleagues that the app is available for testing on both platforms. I am planning to perform more testing myself. If any other bug is discovered, I would need to fix it, re-build the app, re-test it. After the app passes the review, I will release it to the App Store and Play Store. Then I will test and release the web version of the Smart Office.

Monday

Today I asked my teammate to help me figure out iOS issue with Google login when try to modify or delete the reservation. At first, I needed to explain what I gathered and tried so far while investigated the problem last Friday. I noticed a pattern that I can modify or delete earlier created event if I restart the app. But if I try to edit or delete the reservation again, I keep on getting 412 error “Precondition failed” which means that the *etag* supplied in the *If-match* header no longer corresponds to the current *etag* of the resource. That was very strange because after thoroughly inspecting the HTTP transactions – I noticed that the *etag* always matches with the *etag* that was assigned to the event once the booking was successful when sending *.patch* or *.delete* request.

My teammate offered to try to decode or encode the etags as they noticed the format looked different, for example, “22983287328” and “\”22983287328\””. I pointed out that this started to happen after I added a function to get a single event and implemented polling of the space to know if it accepts or declines or not confirms the reservation. My teammate mentioned that it was worth to investigate it as well.

I tried to change the order how things happen when the event gets booked by creating the event only after the event has accepted the reservation, or if the user chose “Remind me later” option from the space status dialog. But then I realized it does not make any sense because the dialog allows the user to either keep the reservation even if the space has declined or did not confirm the booking, to edit and to cancel the meeting. It is impossible to edit or cancel non existing event, therefore, *createEvent* should be called in the beginning as it happens originally. Then we tried to store the original *etag* and assign it to the event once the room status is known and vice versa. Also, we tried to add a timeout to fetch the events as that is what happens when the app restarts, tried to add *If-match:* * header and value but it did not work. The working day was coming to its end, and it was decided to continue tomorrow.

Tuesday

Today my teammate and I investigated further the iOS issue with Google login when try to modify or delete the reservation. We chose to go with learning more about the request headers since that is the root of the problem. We noticed there was no *If-match* header related to event requests but there was *If-none-match*. We decided to try and overwrite the *If-none-match* value. After completing some reading to get a better understanding what *If-none-match* is for and if it is necessarily to have in relation to the events, I decided to try and assign the event *etag* to the *If-none-match* in updating and deleting the event methods. It did not work.

Then I remembered that my teammate pointed out to the *etag* format differences earlier, and instead of just reassigning the *etag* in its original form, I did it like this *If-none-match: "\"event.etag\""*. That worked which means the problem was with the correct *etag* provided to the event when it is created but then when the .*patch* or .*delete* requests are sent, it has the correct *etag* in request headers but in a wrong format.

After the fix, I tested the app with Google login on other platforms (Android and web) to make sure the fix did not break anything. I also noticed a few other things that could be improved before I re-build the app. I adjusted the style for the duration picker: set the height for the bottom buttons to improve the look and deleted the font family assigned to the title because we were not using that font anymore. I fixed the link overflow on the resource card that would happen when the event was booked from the Google or Outlook calendar and the link would be extra-long. I noticed this when added my thesis meeting to my calendar so it appeared in the app, and the link that I put there as a meeting location would overflow the resource card border.

Wednesday

Today I noticed that some improvements to email list filtering implemented for adding teammates or meeting guest search did not work as expected, and it was missing in two other screens that were using the modal for adding a team member. I asked my teammate if they would want to fix the functionality and add the solution to other screens themselves or if I should do it as this ticket was originally done by them. They decided to fix it themselves.

Since it would not make sense for me to build the app until all fixes are merged to the release branch, while waiting, I tested the app more on Android with Google and Azure logins. I noticed the text was overflowing on the connect to the calendar button when I would

choose the language other than English, so I fixed that by adding more margin to the end. Also, I added missing translations and fixed scrolling in week planner details screen.

My teammate was done with their fix, I pulled lates changes to the release branch and started the new builds for Android and iOS. Android build failed on the first try with some Gradle error, in this case I was advised to run the build again. iOS build also had to be re-run even though it was successful, because some certificates has expired in App Center, and the build was not send to the TestFlight for testing.

Thursday

Today I informed my colleagues that Smart Office app was available for testing. I tested the app builds more myself. I finished testing with all login methods on Android device. It turned out that not all colleagues who wanted to help us with testing, had the access to the TestFlight because they were not added as testers to the TestFlight internal testing before, and now there is some bug going on with App Store Connect that it is impossible to add new testers due to the error that says that the email is in a wrong format (which is not the case).

To allow everyone to test the app on iOS, I need to be able to get the link instead, since I cannot add anyone new to the test group. For that the app needs to be moved to the TestFlight external testing. That is an easy task, but the thing is that before the app would be ready for the external testing, it has to pass the review from Apple. The review can take up to 48 hours in general.

Once I added the app for the external testing, I continued to test the app myself, and discovered a few bugs. The first bug: on profile tab, at first, I could change the status, status visibility and first and last names but after the app is opened for a few minutes (no matter what tab), status starts behaving weird on change, I cannot modify name and visibility anymore. I am only able to change the avatar but if I try to put something from phone gallery as profile picture, the avatar changes to the one selected before everything stopped working properly. The second bug: if I have two bookings – one ongoing and one in the future (next day). If I cancel the future one or the ongoing one, the resource card does not disappear. Retrying cancelling it prompts an error. If I cancel the remaining one, only then the view updates and the deleted previously booking disappears, or if I go back to the home screen and open reservations list again. I reported them on the channel related to Smart Office product. One of the backend developers started to investigate the first bug, and I started investigating the second one.

Weekly analysis

This week I realized the importance of the teamwork in software development. For example, while my teammate and I were solving the iOS related issue with modifying or deleting events, I noticed how much easier it is to tackle the problem as a team. My teammate would suggest something, then it directs me to reflect on the problem in a different way and come up with the solution that I would not think of if I would decide to continue figuring out the fix on my own.

There are many benefits that the teamwork brings, such as: teamwork helps to improve developer skills since each teammate has their own set of skills and experience, and when working together – that knowledge is being shared; teamwork increases code quality; and it also boosts productivity in a sense that the teammates get to know each other's strengths and weaknesses, they know who to ask for assistance with particular issues and how to approach them best. Teamwork inspires creativity and innovation, increases the efficiency, ensures the transparency, and increases business potential because improved developer skills inevitably lead to increased coding quality, efficiency, productivity, and breadth and depth of development skills. These allow teams or organizations to perform more work and efficiently handle a wider variety of software projects. With improved coding quality and cycle time, customers are happier and more successful. Profitable development teams have more opportunities for progress than non-profitable teams (Gitential Team 2022).

Communication is the key to successful teamwork, and it is as important as technical knowledge in software development industry. Good communication between teammates leads to better and healthier workplace. For example, when reviewing a PR, reviewers have two options how to formulate their comment – as a question or as a command. The command style suggests that the reviewers already know what the author has tried or has not try to do, they do not suggest a better course of action, and do not provide a solution based on their knowledge. While with the question style, the reviewer asks for clarifications without assuming they know all about the author's code and offer a solution without rejecting alternative arguments (Buomprisco 2019).

While working on the platform specific issue with my teammate during this week, I got to learn more about how to handle API errors in Google Calendar. Specifically, the error 412: Precondition failed. On the Google Calendar for Developers documentation (2022), the suggested action is to re-fetch the event and re-apply the changes. This error means that the resource was overwritten since it got its *etag*, and it is called conditional modification. If you want to update or delete a resource only if it has not changed since it was last fetched, you can provide an *If-Match* header that contains the value of the *etag* from the previous

fetch. This is very helpful in preventing lost changes on resources. The client has the option to re-fetch the resource and re-apply the changes. If the event, and its *etag*, has not changed since the last fetch, the change is successful and a new version of the resource with the new *etag* is returned. Otherwise, you will receive the error 412: Precondition failed (Google Calendar for Developers 2022). Even though that was not the exact problem why the modifying of the event would fail in my case, it was still helpful to learn the best practices on how to handle conditional modification.

The essential concept for solving the described above issue was the understanding of the *If-None-Match* request header. That is how the *If-None-Match* header works: the client requests a resource from the server; the server creates an *etag*, assigns it to the resource, and returns 200: OK response to the resource; the first *etag* created is "response_version_1"; resource is cached with the *etag* value "response_version_1"; when the client requests the same resource from the server again, “response_version_1” is sent to the server with the *If-None-Match* response header; after receiving the “If-None-Match: response_version_1”, the server compares the valid *etag* value to the “response_version_1”; if the current *etag* value is the same as the “response_version_1”, then the status code will be 304: Not Modified, and the server sends an empty response body (it tells the requestor to use the cached resource); if the current *etag* value is different, the server sends the resource again, and the browser updates the cached resource with the 200: OK status code (Gübür 2021).

To conclude this week goals achievement, I planned to fix the iOS issue for when attempting to update or delete the event, and that was accomplished. I re-built the app both for iOS and Android and informed my colleagues that the app is available for testing on both platforms. I did more testing myself and discovered more bugs. I was not able to fix them this week, therefore, the app release was re-scheduled to the next week, together with web version testing and release.

4 Discussions and conclusions

Based on the previously written analysis of my current work, I would say I was a complete newbie in the programming. I had a very basic knowledge overall, and I needed extra time to complete relatively simple tasks. I had no idea how to debug the applications other way than using console or React Native developer tools browser extension. I started by performing the tasks related to the new feature that would not be included into release for some time so that I would get a feel of the project. It could take a while for me to navigate through the project to find the root of the issue and as much time to understand the code before I could add my fix or additional functionality.

I believe I am still a beginner in software development. But nevertheless, I notice huge positive changes both in my hard and soft skills. For instance, I learnt to navigate faster through the project, it takes much less time for me to understand the code that I did not write. I am capable of reviewing my teammates code by testing their solution, checking the code logic, and commenting if necessary. I am taking more responsibilities, for example, I am now being in charge of the release process for one of the company products. I feel I am able to communicate to my teammates in a more concise and clear manner, it is easier for me to explain the problem using technical terms. I got better at working in a team and learnt to appreciate the teamwork even more. Currently I am capable of completing medium difficulty tasks that can provide the customers with the new desired features and functionalities. My communication skills significantly improved. I am being more creative with my solutions.

In my opinion, the courses I completed during my studies played a major role in my further learning process as well as they helped me to land this job. For example, while interviewing for this position, I was able to answer most of the technical questions by remembering what I learnt during Programming and, especially, Front End Development courses, also I have used the knowledge gained while studying to perform my work duties to some extent.

One of the new methods I found for my work is checking network traffic on different platforms. I learnt how to use Xcode developer tool called Instruments for iOS, Android Studio app inspection for Android, and network tab when inspecting a browser page. That helps to understand what kind of requests are being send and what response the API calls give. Also, I got a habit of testing the app on different platforms at the same time to ensure that everything works as should on all of them. Even though React Native claims to be a cross-platform framework, I came to realize the code might have to be changed for the feature to perform as expected on a specific platform.

During the past two weeks I implemented a new feature that many customers wanted to have. It would prevent the double space booking to happen without the user being informed. Or also notify the user when the space would not confirm the reservation and let them to decide if they want to keep the booking anyway, choose another space or cancel the reservation. There is also a check that happens after specific interval that goes through user's bookings in the schedule, checks if there are any spaces that has declined or not confirmed the reservation, and then the space status dialog is being prompted again. That was a new feature and the biggest solution I got to work on during the past two months.

I have learnt a lot during the diary thesis writing process. It helped me to improve my academic writing and learn how to manage my time in a more efficient way. I also feel it is a good habit I gained to write detailed notes on what I did and what I plan to do every day. Especially since that is the information I need to share during daily meetings as well as it is beneficial in a sense that I can always go back and check what I did and how it was done in case if I need to brush my memory if I have to implement something similar in the future.

Weekly analyses were also beneficial for me. It taught me to pay attention not only to technicalities I learnt while working but also to my soft skills and how they developed with the time. I think it is a good practice for every developer to have a journal that they fill in a similar to diary thesis way where they analyze their hard and soft skills development week by week. It was a great way to boost my motivation when I read through the whole eight weeks of reporting and analyses and noticed how much I changed and grew as a professional. As a Junior Software Developer, I personally enjoyed the most diving deeper into technical topics as there are so many interesting concepts and technics in programming that I am yet to learn.

In the analysis of my work in the second chapter of the thesis, I mentioned that my goals for the future were to become more independent, learn more complex concepts for JavaScript, TypeScript, React.js and React Native, make programming my free-time hobby, become more efficient and get things done faster, take more responsibilities, be able to help others, learn to read and understand other people's code quicker.

I already feel that I require less and less assistance from my colleagues. I believe the more experience I get, the less things I need to clarify. But also, I really enjoy collaborating with my teammates. Their input is valuable for me, and it also allows them to share their experience through active collaboration. I am learning every day how to communicate effectively and concisely in a professional setting.

There were many technical concepts and methods I have learnt and recapped since I joined Steerpath. Of course, there is still a lot to learn. That is exactly why I like working as a Software Developer. The technologies are changing rapidly, so many new ways, languages, frameworks would most probably come along. There is no guarantee I would still develop mobile apps using React Native in five years from now, or even if it would stay in use, I am positive there will be some changes and additions introduced to React Native, for example, the way the Hooks came into use not that long ago.

I also mentioned I was hoping to make programming my free-time hobby in a sense that I would work more on projects to learn something new about different technologies. That was not achieved so far. To be honest, I have not written a single line of code outside working hours. Throughout studies I used to feel bad about this but now that I am an entry-level specialist and I practice coding five days a week for over seven hours, I learnt to take it easier on myself. Sometimes it is good to push yourself to do things but most often it would only stress one out if they would not succeed in following their plan. I decided I need to tackle this goal little by little. Currently I try to read more or watch a video or two about the technologies I am interested in for about twenty minutes a day. I would also be interested in reading technical books where I could learn the best practices. And I am planning to gradually replace the theoretical approach with practical one.

Without doubt, I became more efficient, and I can get my tasks done faster. I started to take more responsibilities, and I can help my teammates, for example, while reviewing their code. I can understand other's code quicker as I am a more experienced developer now than I was two months ago. To sum up, my professional growth was significant compared to where I started from. I am learning and growing as a specialist and a person every day.

I enjoy working at Steerpath because of the company values, great company culture, international environment, and, most importantly, I also like the projects I am working on. My hopes for the future are to keep contributing to company success by gaining more experience and knowledge as a Software Developer and a team player. I hope to continue following my goals in the future to become indispensable employee, and to keep my passion in software development.

References

Avery Products. 2021. Status Update Meetings: How to Lead Better. URL: https://www.avery.com/blog/status-update-meetings-how-to-better-lead-your-team-and-projects/. Accessed: 10 April 2022.

Boduch, A. & Derks, R. 2020. React and React Native. 3rd ed. Birmingham: Packt Publishing.

Bugl, D. 2019. Learn React Hooks. Packt Publishing. E-book. Accessed: 24 March 2022.

Buomprisco, G. 2019. Tips for good communication in Software Development teams. URL: https://blog.bitsrc.io/tips-for-good-communication-in-software-development-teams-9797a7a70736. Accessed: 16 April 2022.

Burger, E. 2020. Super Easy NPM Package Patching with 'patch-package'. URL: https://javascript.plainenglish.io/live-npm-package-patching-with-patch-package-b8f30d47779c. Accessed: 3 April 2022.

Corti, N. 2020. How-to Github Actions: Building your Android App. URL: https://ncorti.com/blog/howto-github-actions-building-android. Accessed: 10 April 2022.

Dabit, N. 2017. Deep linking your React Native app. URL: https://medium.com/react-native-training/deep-linking-your-react-native-app-d87c39a1ad5e. Accessed: 22 February 2022.

Debricked Editorial Team. 2021. How to Fix Security Vulnerabilities in Yarn. URL: https://debricked.com/blog/fixing-security-vulnerabilities-yarn/. Accessed: 3 April 2022.

El Azizi, Y. 2020. Handling Errors in React Native: A Complete Guide. URL: https://elazizi.com/handling-errors-in-react-native-a-complete-guide. Accessed: 20 March 2020.

Expo documentation s.a. a. Linking. URL: https://docs.expo.dev/guides/linking/. Accessed: 21 February 2022.

Expo documentation s.a. b. Stacking overlapping views with zIndex in Expo and React Native apps. URL: https://docs.expo.dev/ui-programming/z-index/. Accessed: 8 March 2022.

GeeksforGeeks. 2021. When to use useCallback, useMemo and useEffect. URL: https://www.geeksforgeeks.org/when-to-use-usecallback-usememo-and-useeffect/. Accessed: 2 March 2022.

Gitential Team. 2022. Teamwork in Software Development. 8 benefits of building up a collaborative team. URL: https://gitential.com/teamwork-in-software-development-8-benefits-of-building-up-a-collaborative-team/. Accessed: 16 April 2022.

Google Calendar for Developers. 2022. Get specific versions of resources. URL: https://developers.google.com/calendar/api/guides/version-resources. Accessed: 16 April 2022.

Gübür, K. T. 2021. What is If None Match HTTP Request Header? URL: https://www.holisticseo.digital/pagespeed/if-none-match/. Accessed: 16 April 2022.

Harrison, K. 2021. Inspecting HTTP Traffic with Instruments. URL: https://useyourloaf.com/blog/inspecting-http-traffic-with-instruments/. Accessed: 10 April 2022.

Hunter, M. 2021. 5 soft skills to help you excel as a Junior Software Developer. URL: https://betterprogramming.pub/5-soft-skills-to-help-you-excel-as-a-junior-software-developer-1866795f2f18. Accessed: 3 April 2022.

Infinitbility. 2021. How to use async / await in React Native. URL: https://infinitbility.com/how-to-use-async-await-in-react-native. Accessed: 17 March 2022.

Kostin, Y. 2021. Configure Environment Variables with React Native Config for iOS and Android. URL: https://medium.com/armenotech/configure-environment-variables-with-react-native-config-for-ios-and-android-7079c0842d8b. Accessed: 9 March 2022.

Lehnhardt, P. 2019. StackOverflow post reply. URL: https://stackoverflow.com/questions/55664673/require-cycles-are-allowed-but-can-result-in-uninitialized-values-consider-ref. Accessed: 22 February 2022.

Li, C. 2018. JavaScript async/await: The Good Part, Pitfalls and How to Use. URL: https://hackernoon.com/javascript-async-await-the-good-part-pitfalls-and-how-to-use-9b759ca21cda. Accessed: 18 March 2022.

Marks, K. 2020. Passing params with React Navigation. URL: https://medium.com/swlh/passing-params-with-react-navigation-dd86c5de024c. Accessed: 24 March 2022.

MDN web docs. 2022. Promise. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise. Accessed: 16 March 2022.

Mittal, A. 2021. React JS – How to return response in Async function? URL: https://akashmittal.com/react-return-response-async-function/. Accessed: 17 March 2022.

npm. 2022. yarn-audit-fix. URL: https://www.npmjs.com/package/yarn-audit-fix#digest. Accessed: 3 April 2022.

Pavlutin, D. 2021. How to Memoize with React.useMemo(). URL: https://dmitripavlutin.com/react-usememo-hook/. Accessed: 1 March 2022.

React s.a. Rules of Hooks. URL: https://reactjs.org/docs/hooks-rules.html. Accessed: 10 March 2021.

React i18next documentation. 2021. useTranslation (hook). URL: https://react.i18next.com/latest/usetranslation-hook. Accessed: 3 March 2022.

React Native documentation. 2022 a. Layout with Flexbox. URL: https://reactnative.dev/docs/flexbox. Accessed: 24 February 2022.

React Native documentation. 2022 b. View Style Props. URL: https://reactnative.dev/docs/view-style-props. Accessed: 11 March 2022.

React Navigation s.a. Introduction. URL: https://reactnavigation.org/docs/hello-react-navigation. Accessed: 24 March 2022.

Sajjad, Z. 2021. React Navigation vs. React Native Navigation: Which is right for you? URL: https://blog.logrocket.com/react-navigation-vs-react-native-navigation/. Accessed: 27 March 2022.

Thomas, A. 2020. Patch-package in React Native. URL: https://anu-thomas.medium.com/patch-package-in-react-native-c7786a15e279. Accessed: 3 April 2022.

Varma, S. 2021. Comparing the top 10 mobile CI/CD providers. URL: https://www.runway.team/blog/comparing-the-top-10-mobile-ci-cd-providers. Accessed: 10 April 2022.

Wickramasinghe, S. 2020. How to internationalize a React application using i18next. URL: https://lokalise.com/blog/how-to-internationalize-react-application-using-i18next/#Adding_new_languages. Accessed: 28 February 2022.

Wishart, J. 2022. Why are KPIs Important? The Importance of KPIs and how to choose them. URL: https://www.rhythmsystems.com/blog/5-reasons-why-you-need-kpis-infographic. Accessed: 10 April 2022.

W3Schools s.a. JavaScript Errors. URL: https://www.w3schools.com/js/js_errors.asp. Accessed: 18 March 2022.