Working Diary of a Junior Software engineer

Shannon Ronaldson

Haaga-Helia University of Applied Sciences
Haaga-Helia Bachelor’s Degree
2022

Abstract

Author
Shannon Ronaldson

Degree
Bachelor of Business Administration

Report/Thesis Title
Diary of a Working software engineer

Number of pages and appendix pages
72 + 3

This thesis was written using the Haaga Helia thesis guidelines for Diary thesis. Mendeley reference manager was used to manage and add citations, and Harvard's 'Cite them right' style was used in accordance with the Haaga Helia guidelines.

This thesis focuses on my experience as a software engineer for Hailer Oy, a software as a service (SaaS) company in Porvoo, Finland. I work as an intern/junior developer on various projects, including rebuilding the website in HubSpot, creating documents for customers and smaller development tasks.

This thesis focuses on my work in HubSpot while developing the new Hailer website; however, I undertake various tasks through the ten observation weeks documented in this thesis. Each observation week includes an analysis of the previous week and articles on topics broached during the week. After completing this thesis, I became a fully-fledged HubSpot developer and undertook the UI/UX styling of the Hailer application itself.

This thesis uses the programming languages and frameworks: HubSpot, HubL, CSS; SCSS, HTML, Git, BitBucket, JavaScript, jQuery, TypeScript, Angular and Node.js. The software platforms include Visual Studio Code, MongoDB, Docker, Sourcetree, Figma and the Hailer application. Methodologies used include Agile and Scrumban.

Keywords
Software engineer, junior developer, HubSpot developer

Contents

List of Abbreviations i
1 Introduction 1
2 Description of the initial situation 3
2.1 Analysis of your current work 3
2.2 Stakeholders 4
2.3 Interaction situations 6
3 Diary entries 7
3.1 Observation week 1 7
3.2 Observation week 2 13
3.3 Observation week 3 18
3.4 Observation week 4 24
3.5 Observation week 5 36
3.6 Observation week 6 42
3.7 Observation week 7 48
3.8 Observation week 8 55
3.9 Observation week 9 59
3.10 Observation week 10 65
4 Discussion 71
References 73
Appendices i

List of Abbreviations

Terms and abbreviations used in this thesis

Abbreviation	Definition	Explanation
B2B	Business to Business	Transactions that occur between companies (Chen, 2022)
B2C	Business to Consumer	Transactions between a business and its' customers/clients (Kenton, 2022)
CEO	Chief Executive Officer	The highest-ranking executive of a company (Bloomenthal, 2022; Hayes, 2022)
CMO	Chief Marketing Officer	Highest ranking executive in the marketing or sales roles (Bloomenthal, 2022)
CMS	Content management system	Software application that allows no or low code creation and management of a website (Fitzgerald, 2022)
CRUD	Create, Read, Update and Delete	The four major functions of a database applications (Techopedia authors, no date; MDN Contributors, 2022b).
CSS	Cascading Style Sheets	A programming language used to style and describe how elements should look. Generally used with HTML or XML. (MDN Contributors, 2022c)
CTA	Call to action	A marketing term used to refer to the next step that a marketer wants a customer to take(Kenton, 2020)
HTML	HyperText Markup Language	The basic built block language of the web, it lays out the structure of a web page (MDN Contributors, 2022d).
HubL	HubSpot Markup Language	The HubSpot templating language (HubSpot, 2022b)
JSON	JavaScript Object Notation	Lightweight data exchange format (JSON.org, no date). JSON is often used to refer to both the JSON object (which contains methods for parsing JSON (MDN Contributors, 2022f)) and the JSON language itself.

Abbreviation	Definition	Explanation
PR	Pull request	The initiation of merging (integrating) code (Garner, 2022)
QA	Quality assurance	Assuring that the output of the product or feature fulfils the goals and requirements outlined in the planning (ASQ, no date)
SCSS (SASS)	Syntactically Awesome Styling Sheets	A more advanced version of CSS (SASS-LANG, no date)
SVG	Scalable Vector Graphic	A text-based language used to describe 2 dimensional (2D) based vector graphs/diagrams (MDN Contributors, 2022h)
UI	User interface	The access point(s) that users use to interact with a product or design (Interaction Design Foundation, no date). These access points include the human-computer interactions and the way in which the user interacts with the product or application (User Testing authors, no date; Churchville, 2021)
URL	Uniform Resource Locator	The mechanism with which browsers retrieve any published information from the web (MDN Contributors, 2022i)
UX	User experience	The experience which the user undergoes while using the product, feature, or application (User Testing authors, no date)

1 Introduction

This thesis aims to evaluate and document my work as a full-stack engineer at Hailer Oy, Porvoo. The diary entries commence in the last week of April (25 April is the official start) and continue for 2-3 months. From 25 April until 20 May, the diary entries are for part-time work, with a maximum of 25 hours per week. During this period, I was primarily involved in developing document templates and redesigning the Hailer website. After 20 May, I began full-time work, during which time I worked on more projects, mainly in the frontend of the Hailer application.

Hailer Oy is a product company with one primary product supplied to businesses (business to business – B2B) and customers (business to customers – B2C). The product, hereon referred to as the 'Hailer app' or 'the app', is an all-in-one work management platform. Hailer aims to reduce clutter, increase organisation and allow our customers to customise their workflows and processes to suit their needs (Hailer Oy, 2022).

Hailer has three major teams – the development team, the projects team, and the sales team. I am part of the development team but work closely with the projects and sales teams.

The two major types of customers currently in the Hailer application are paid and unpaid customer groups. The unpaid customer group has limited features and cannot request custom workspace configurations. The paid customer group consists of most Hailer customers. These customers are assigned a salesperson, the primary contact person for the customer and create tasks, tickets and requests for the project team. A project team member is assigned to the customer as well.

The Hailer sales team comprises around six salespeople, all with multiple customers they manage. The projects team is expanding as Hailer's customer base grows and demand for customised workflows increases. Project team members manage multiple customer workspaces and collaborate closely with the sales team.

The development team at Hailer is also growing as the customer base increases, and we require more staff to meet the demands of major projects. The core Hailer team consists of two senior developers, one product owner/scrum master, three intermediate engineers and me. Several part-time software engineering students are contracted to do various tasks throughout the application. On top of these part-timer students, we also contract tasks to our consulting team.

Initially, I was hired as a part-time student, much like the other part-time students. I was primarily hired to create new document templates. Document templates are a client-facing feature, but I was hired as part of the development team. I am considered a 'special case' as I work in the projects and development teams. I will explain this detail further in section 2 – Initial work situation.

Document templates are PDF documents that clients print with generated and inserted data from their Hailer processes and activities. They are a vital custom component of Hailer, ranging from certificates and invoices to ordering templates. We utilise a specific software called PDFmake, which creates the PDF template for us.

PDFmake is a document generator and can be used server-side or client-side (pdfmake, no date). When creating a template, I utilise PDFmake's keywords to develop content variables of tables, columns, paragraphs and sections. We then create an array including the information from the client's activity and use the document generator to create a PDF.

After starting my work at Hailer, I was 'stolen' (as the team jokingly tends to say) by the marketing director to redesign the Hailer website. The Hailer website is created and managed in HubSpot. We have multiple integrations that rely on HubSpot and elected to use HubSpot to host and develop the website. This means that all website-related tasks are coded in HubSpot's' native coding language - HubL, as well as CSS, HTML and JavaScript (but not EC6) (HubSpot, 2022d). In general, I work on the website using HubSpot's Design Manager. The Design Manager is an online tool that allows me to create modules, templates, partials and layouts utilised throughout the website (HubSpot, 2022c).

HubSpot modules are reusable components and are the smallest building blocks (HubSpot, 2022d). Modules are then imported into templates or (template) partials. Partials include the footer and header sections used on every page of the Hailer website (HubSpot, 2022d). Templates are then imported/used in layouts. A layout is what the entire page will look like, including the partials and the specific templates only used on said page. These layouts are the finished products - but do not contain any specific content (HubSpot, 2022d). The content itself will be added later by a content creator.

I work closely with the UI/UX designer regarding the website, we are in constant communication, and I often need to confer with her regarding changes, big and small. She has created a working prototype for me in Figma, and I aim to recreate her designs in HubSpot. I am also in constant communication with the marketing director, who approves the designs from the UI/UX designer and any changes I need to make to the designs and my final website creations.

2 Description of the initial situation

The next chapter in the report describes the initial situation that analyses the current work, stakeholders, and workplace interaction situations.

2.1 Analysis of your current work

My current work before beginning this thesis mainly focussed on creating, editing and remodelling document templates. My knowledge base in the languages required for these tasks was rudimentary. While I have worked with JavaScript, CSS and HTML extensively on my own and in my internship – working with these languages in Angular proved to be a steep learning curve. I would cautiously say I was an intermediate developer regarding CSS and HTML, with novice to intermediate skills in JavaScript and React. I am a complete novice in Angular and will be learning how to code in Angular while working. I was also asked/volunteered to re-write the documentation for the entirety of Hailer, and on occasion, I still do some writeups depending on necessity. Documentation-wise I am an expert; I was a copywriter for many years and have created the documentation for various sites, as well as rewrote the documentation of my internship company last year.

Developing in HubSpot was entirely new for me; no one else had worked on HubSpot in the company before. Previously, the Hailer website was created by an external HubSpot contractor. However, the marketing director decided to do the new website in-house. This will be the focus of this thesis. I chose to do the HubSpot Academy videos before this thesis and managed to understand the basics of working in HubSpot; however, I will be learning as I go along.

Most of my technical skills have been gained from online web developer boot camps, studies at Haaga-Helia, and my ASP.NET internship in 2021. My overall coding ability would range from novice to intermediate. I am experienced in specific parts of web development, mainly frontend and styling, but I need to gain more knowledge in the backend frameworks.

I am more front-end inclined due to my previous content creator and copywriter work. I have a natural knack for creating and editing different user experiences and interfaces for websites. This experience and natural ability are one of the main reasons I was hired at Hailer.

Besides technical skills, time management is an integral part of this job. Firstly, I work most of the time remotely as Hailer is situated in Porvoo, and I live in Helsinki. Secondly, I am paid hourly. I need to ensure that I track my hours, or I will not be reimbursed for them.

On top of this, I have been working and studying simultaneously, which has required a lot of adjustment and changes in my lifestyle. It also means I must learn to set aside time for work,

school and personal time. One significant skill I have needed to know is to stop working when I have a specific number of hours. I was terrible at separating my work and personal life, and both suffered greatly.

These soft skills have been slowly but surely gained over work history. Starting as an administrator in South Africa helped me to be very admin minded. Working as a freelance content creator allowed me to develop my time management and writing skills.

The documentation aspect of my work focused around redoing the Confluence documentation site that was already started by other developers and updating all the documentation within the site. I had to create an easy-to-use flow in Confluence that anyone in the company could easily use. I took it upon myself to create an onboarding process in Confluence for all teams and have made it as simple and easy as possible for all members of the various teams to access and use Confluence.

2.2 Stakeholders

As I work on multiple projects, there are different stakeholders for each project and team. I work closely with the sales and project teams for document templates, with the end user being the customer(s). When working on the Hailer website in HubSpot, I work with the chief marketing director and the UI/UX designer. However, stakeholders for the website also include the sales team (for client references) and potential customers. The diagram below explains the very generalised layout of the stakeholders in each situation. The stakeholders related to each work type are further described below.

For website design. I, the Chief Marketing Officer (CMO) and the UI/UX designer are the three main entities with decision power in the redesign. The Sales team rely on the website to drive traffic and obtain leads, while the marketing team (and the CMO) rely on the website being updated with information and for use in the marketing campaigns. Customers use the website to find out more about Hailer, create accounts, log in to the web application or contact the Sales team for paid versions or with various questions.

The primary stakeholders for the document templates are the project and sales teams, who rely on me to create and edit templates for the respective clients. Therefore, the clients also depend on me to produce usable and practical documents they can use in their invoicing, accounting and other work processes.

I have the same stakeholders and interactions as all other developers for my work in the development team. As a developer, you rely on your senior developers to review and approve your

code, merge your pull requests (PRs), push your code to the pre-production site, and do releases with the newest features.

The product owner is the primary entity upon which all others rely. He determines what features will be developed, the requirements for the feature to be considered done (definition of done), and what the feature needs to do. He also determines when each feature will be developed. Therefore, the development team's work is highly dependent on the product owner to tell us what needs to be done and when, as well as deadlines. However, he also depends on the development team to follow these deadlines and create and implement the features as defined.

The development team relies on the quality assurance (QA) tester to feature test the code and ensure that there are no issues with the usability and that the feature performs as it should. The QA tester relies on the product owner to tell her what the feature is required to do and the anticipated uses of the feature. Similarly, the development team depends on the UI/UX designer to design the various features as defined by the product owner.

Finally, the rest of Hailer (sales team, marketing and project team) depend on the development team to create, update and implement any changes and fixes to the Hailer application. It depends on the development team to keep the application running as it should, allowing our customers to use it.

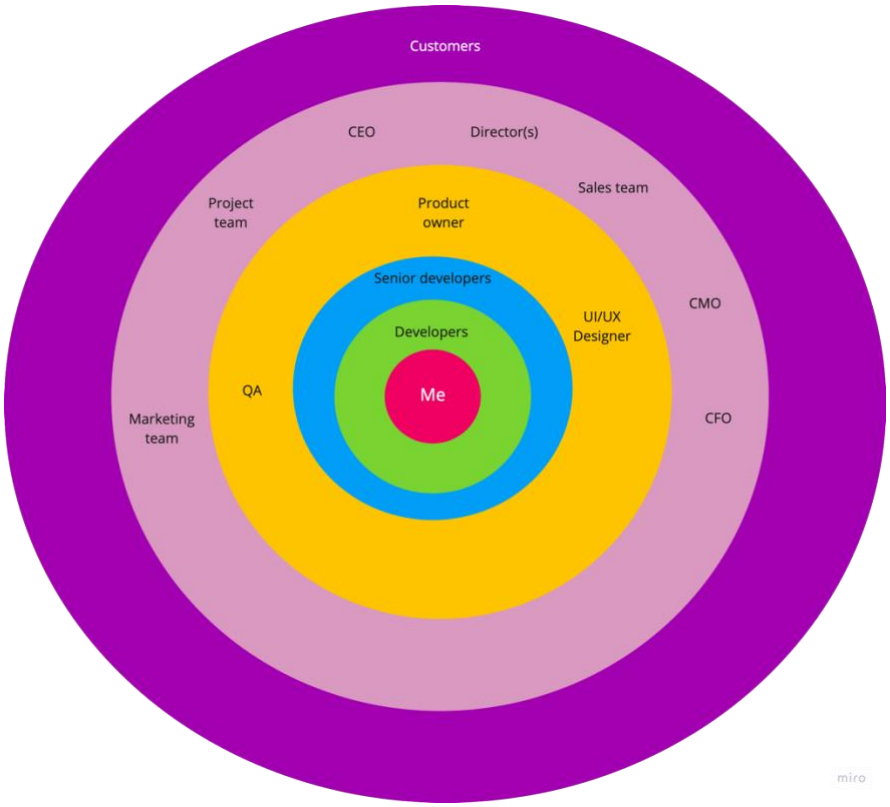


Figure 1. Stakeholders in Hailer

2.3 Interaction situations

My work in Hailer spans multiple teams, meaning that daily interactions and processes vary considerably depending on which task I complete. Document template tasks involve interactions with the sales team and project team; sometimes (in rare cases), I interact directly with customers. I receive a work brief from the respective project team member or salesperson and create the document template to these specifications. After the template is made, it is reviewed by the respective project member or salesperson and then sent to the client for approval. Once approved, the template is in production, and the only time I work on it again is if further changes are requested.

The Hailer website interactions are confined to the chief marketing officer (CMO) and the UI/UX designer. We discuss the changes on the website almost daily, and once I have completed a module in the HubSpot Design Tools, I will add a screenshot or video of the module to show the UI/UX designer, and we will discuss what needs to be changed.

In general, most of my daily interactions focus on Stand-up. While I am in contact with my team regarding specific changes – I am mostly left to my own devices and to figure out what works when, where, and how on my own. I prefer it this way, as I like to try and test out fixes before asking for help. However, I am fully aware that the team is just a message away, and if I need any help, I can message anyone on the team and receive their help quickly.

My most frequent daily interaction is with the UI/UX designer as we discuss the website changes almost daily and any styling changes. As this thesis period progressed, I began taking on more design-related tasks. Therefore, the designer and I are in almost constant contact.

3 Diary entries

3.1 Observation week 1

3.1.1 Monday 25 April 2022

This Monday was the start of our new sprint. Due to this factor, I started my thesis observation days, specifically on this day. Sprint planning occurs every three weeks in Hailer. We start the morning with the meeting, wherein each developer discusses what they will be doing for the next three weeks.

Currently, I am not working on any tasks related to the development team. This means that I have very little to contribute during these sessions for the time being. I spent the 2-hour meeting working on the Hailer website but listening to the meeting simultaneously. This is often how I spend these sessions unless I have something to discuss.

Once the sprint planning meeting was concluded, I focused entirely on my daily tasks. Today, I planned to get to know how to create components (or modules as named in HubSpot). I had completed some videos in the HubSpot Academy, and during those videos, I made modules that the videos were about. However, creating components from scratch is a good test of how well you know and understand a language and framework.

Therefore, I chose one component to work on for the day. I focused on creating a button module with various options for content managers to change. In the Hailer design system, we have two main button designs we use in Hailer, so I planned to create one button with a selection for content managers to choose which buttons they want. These are our ‘normal’ buttons in the Hailer application, the top button being a call-to-action (CTA) button. CTA buttons are designed to be eye-catching and entice customers to click them (Leaning, 2022).

My set goal for the day was to create these two main button types in HubSpot. However, before I could begin working on this – a critical bug was reported by a customer in our production document templates. This meant I had to switch focus immediately and fix this issue. The reported problem related to a table in a document template that was being printed as a blank table but had data in the Hailer activity.

This table is part of an invoice, and a lot of the logic of the table is in the document templates. It is supposed to print out a row for each linked activity, which are the work hours per employee and the sum of the hours worked as a total both in the last column (i.e., total pay per employee) and the calculated total of the table.

However, the table printed out the final sum of the rows, not the rows themselves. Before the bug was reported to me, the project team member in charge of this customer attempted to find the reason for the table not printing out the information. He read through the workspace's links, configuration, and function fields in question but could not find anything on his end. Therefore, he alerted me.

I had worked on optimising these templates the previous week, so I assumed there might be issues but expected much more minor problems such as misspellings. When I debug document templates, I always follow the same steps before moving to local development:

1. Attempt to print out the template myself (to ensure no error messages are missed).
2. Check that the information in the settings is correct

If the settings information is accurate, and I have all the error messages and the copy of the printed template, I then move on to local development.

I knew approximately where the issue was occurring – the table in question was coded in a specific function in the template, so I knew that I could check what data was being output by this function and see where the data was being 'lost'. I generally add console.log throughout the function to determine where the data is being lost. Using console.log, I quickly found the error (using the wrong data array). After correcting this issue, I created a pull request and informed my senior engineer, who fixed the problem immediately. This meant the client had the fix as soon as the frontend was updated. A few minutes later, the Hailer production frontend refreshed, and the client could print their document without any issues.

This process took 2.5 hours, including the time I spent setting out what I intended to do in HubSpot and sprint planning meant that my 5 hours of daily work was completed. Therefore, I ended my day.

3.1.2 Tuesday 26 April 2022

Today I planned to continue with the button module in HubSpot. I had the plan laid out from yesterday, and as far as I knew had no other tasks on my list.

I started with a blank module in HubSpot and built a primary link button using a div and an anchor tag. I tend to create CSS classes as soon as I start a new element – this is simply a habit of mine, and I have no reason for doing it. Sometimes, the class is not used – but I find that if the element is classed already, I add the CSS instead of also needing to add a class and ensuring it makes sense.

However, I ran into issues with this module immediately. I could not create a usable link because the link itself needed to be set in the content manager flow – and I could not achieve that with my current setup. Therefore, I looked back at the code for the default HubSpot button. In there, I saw that the default button has a link field. While I did not understand how this field worked, I added it to my button and copied the HubSpot-created snippet from the field. I pasted this snippet inside the button-wrapper div.

From here, my button looked exactly like the default HubSpot button. I planned to add padding, margins, and colour scheme conditionals to the button. I started with the more straightforward elements –padding and margins. HubSpot has a Style tab attached to each module, and within the Style tab, you can add various fields. One of the fields is ‘spacing’. Spacing includes the top and bottom margins and all-around padding of an element. I added a field for each element in my HTML (div and anchor tag).

I decided to create a conditional for the colour schemes. This decision is because there are only two button colour schemes in the Hailer Website design Figma file. In this way, I planned to prevent content managers from completely changing the colour of the buttons when they should not be able to.

If I were to use a colour field in the Styles tab – the content manager would be able to edit the colour of the button to whatever colour they wanted. This is not ideal, firstly because the website's design would be inconsistent – but also because Hailer has brand colours which we wish to maintain throughout the website and application.

Creating conditionals in HubSpot can be done in various ways. One way is to add an if statement within the require_css tags seen above and then have a choice field in the fields tab, allowing content managers to choose between various selections.

However, the way I opted to go is slightly different. Instead of using an if statement, I set the conditional choice values as CSS classes. These CSS classes were then set in the module.css of this module. I then further challenged myself to create a fully customisable button – including button colour and opacity, font colour and opacity, text alignment, margin and padding, border colour and opacity, border type, border size and border-radius.

I used the basic require_css tags to allow these changes and did not create any conditionals. This button is not meant to be used by content managers but will be there if there is any need for such a customisable button. The layout of the HTML of the button was the same as the 2-colour scheme button, but I added Style fields and linked the field to the CSS classes in the require_css tags.

3.1.3 Wednesday 27 April 2022

Technically this was a non-working day for me. However, there was a critical bug in the document templates for the Hailer sales team, and I needed to fix it. The template was not printing, and the error message informed me that there was no value found in one of the functions.

I ran my usual document template checks, started my local frontend, and immediately found the issue – a simple misspelling in which I told the generator what field to use. I spelt ‘feilds' instead of fields. Meaning when the generator was trying to create the template, no data was found. Once again, this was hot-fixed to production after the pull request was made.

3.1.4 Thursday 27 April 2022

Non-working day

3.1.5 Friday 29 April 2022

Non-working day.

3.1.6 Weekly analysis

This week was primarily focused on getting to grips with HubSpot development and fixing various document template bugs. Document template bugs are common and often only appear when the customer needs the template to work immediately. This usually occurs around the days when the company(s) invoice their customers or pay their employees. Therefore, it is critical that I can switch and immediately focus on these bugs.

The bugs this week were relatively minor. I had made similar changes so many times that the fixes were almost instinctual for me. I knew what I needed to fix; however, to ensure that I was fixing the correct bug and not creating new ones, I always debug and use excessive amounts of console.log throughout the template. Document templates-wise, I did not struggle much and did not do anything out of the ordinary in my day-to-day work. Therefore, I did not learn anything from these fixes, apart from not making these small mistakes again.

However, as this was my first week focussing on HubSpot development, it was a good thing that there were no major bugs or issues in the document templates. HubSpot development is much more complicated than I realised, especially when creating modules alone and without following how another developer codes it through videos.

I made it a challenge to entirely create the 2-colour button module from scratch and a button module with as many options as possible. While I mostly completed these modules, I will still need

some tweaks next week. I have submitted them to the marketing director for review. While it is possible to make the entire CSS of a module customisable for content creators, this is not recommended by HubSpot (HubSpot, 2022d) as it can lead to elements which should appear uniform being utterly different from one another. Therefore, the customisable button was more of a challenge for me to see how each style field works and to understand how I can make various changes.

This week was short for me, mainly because I was not meant to be working Wednesday – Friday. Therefore, I only had two half days of work. It is common for me to complete important document template fixes or changes on my non-working days simply because these are client-facing and client-requested features. I always put the client-facing features higher on my task list, and in this way, I ensure that the clients have very little to complain about regarding my work.

For such a short week, I learned a lot – primarily small things in HubSpot that allowed me to understand better how development in HubSpot works. Next week I intend to create even more complex and customisable modules.

Hailer implements an agile methodology known as Kanban Scrum. We use this methodology in the development team and the rest of Hailer. Agile is a project management methodology that implements iterative and incremental steps that ultimately build towards one major feature (Association for Project management, no date; Aston, no date). The idea of agile is that a specific feature or product is broken down into smaller pieces that are then developed independently. These pieces can be released as they are completed, increasing the speed at which development occurs (Ben Aston, 2017). The core values of agile according to the Agile Manifesto (Beck *et al.*, 2001) are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile has multiple variations in implementation; agile methodology is considered a mindset or framework (Association for Project Management, no date) and not a methodology. In Hailer, we use a combination of two of the most common agile methodologies – Kanban and Scrum – known as Scrumban.

Scrum is a method to manage and control the iterative phases in agile (Pratt and Torode, 2020). Scrum is typically used in cross-functional teams and is geared towards delivering products fast (Aston, 2017).

The key components of Scrum include sprints, ranging from 1 week to 3 weeks (The Agile Alliance, no date b; Aston, 2017). Within these sprints, tasks or features are planned to be worked on, and at the end of the sprint, a retrospective occurs in which the tasks and features are reviewed (Aston, 2017; Sutherland and Schwaber, 2020). This review does not necessarily mean that the feature must be completed within the sprint. At each retrospective, the feature is presented and reviewed; if necessary, changes are made to the feature to better align with the goals and requirements. Within the sprint, a daily scrum (or standup as we call it) occurs wherein the developers discuss their tasks for the day and any obstacles preventing or slowing their work (Aston, 2017; Sutherland and Schwaber, 2020).

The product owner creates a backlog of tasks/features (The Agile Alliance, no date b; Sutherland and Schwaber, 2020), which the development team will develop. These features are then organised into Sprint Backlogs, the features/tasks that will be worked on in a specific sprint (Sutherland and Schwaber, 2020).

Kanban is a workflow management tool that implements boards (Kanban) which are moved through various phases or steps to the end point or end phase (Kanbanize, no date; The Agile Alliance, no date a). The goal of Kanban is to make the workflow more visible and see where the tasks are in the development process (The Agile Alliance, no date a). As a Kanban is worked on, the Kanban is moved through different phases to denote which process or phase it is currently undergoing (Product Plan, no date).

Scrumban implements the workflow management of Kanban with the product management of Scrum (Product Plan, no date). By creating a visual representation of the sprint, sprint goal and backlog, and product backlog, it is easier to determine where the features/tasks are in the processes (Product Plan, no date; Ladas, 2008). It allows the product owner to organise the tasks and features to be developed and split the features into incremental pieces that specific developers can work on (Ladas, 2008). Specifically, Scrumban allows a visual representation of the coordination of labour (Ladas, 2008).

3.2 Observation week 2

3.2.1 Monday 2 May 2022

One of the significant document template tasks I often receive is remodelling an existing template. This is often due to the client's needs changing due to changes in their business structure or adding new items in their Hailer workspaces that they want in their documents.

Today I was tasked with changes for two clients. The first client only requested a quotation on how much it would take to remodel their three templates. I am often asked to give cost estimates, and how I estimate is doing the changes on one of the templates without testing or pushing to production. I added an extra hour or two for debugging and testing and 0.5-1 hour for my senior developer to review the code before it is pushed.

If the remodel is not a significant change, on average, it takes me 0.5 - 1 hours to remodel the template. Therefore, a safe estimate for a small change is 3 hours, as this ensures that if there are any issues, I have extra time to fix them and try other methods. Once this cost estimate was sent to the project team member, I moved on to different document templates.

Originally I planned to continue with the Hailer website today, but document templates have a higher priority. These remodels were requested last week during my non-working days – so they were the first tasks I needed to complete this week.

After the cost estimate for the remodel, I worked on the other client's remodel. This client requested minor changes to all 6 of their existing templates. While these are minor changes, they are styling changes and require much playing around to find the best look and layout on the template. These changes included making more space between the title and the subtitle and adding margins to bullets to make them look more like bullet points.

I did not originally code the existing templates; therefore, there were some differences in coding style. In general, I leave these differences alone; however, many of the values in this template were hard-coded instead of data pulled in from Hailer. We avoid this as much as possible in document templates, as it means that if there are misspellings or the client wishes to change something in the wording, the entire frontend needs to be updated each time we make any of these changes. Therefore, I removed these hard-coded values and converted them into the dataset versions I use in all the other templates.

After changing all these hard-coded values, I encountered many bugs ranging from output tables not being big enough to date fields outputting ‘invalid date’ instead of being blank.

Regarding the output tables, I changed the sizing values, which led to changing the document's margins and adding more space between the tables and other document elements. The client had requested that certain documents be contained on one page, which meant I had to do multiple test runs of the document to ensure that any changes I made did not make the page too long and pushed it onto another page. For the invalid date bug, I added an if statement that outputs an empty space (null) if no date was added to the dataset.

I sent the revised version to the salesperson whenever I completed a template. He then reviewed it and checked that the changes were consistent with the client's request. If he was happy, I did not have to make any more changes. However, he had a few more edits he requested regarding the spacing – either adding more space or less space. This remodel was for six templates, and all were styling changes – this took my entire day. After creating a pull request for all six templates, I was done for the day. While it was not what I had planned to do, I completed all the high-priority tasks for the day.

3.2.2 Tuesday 3 May 2022

My senior developer is currently working on moving the document generator from running in the frontend to the backend. The issue with the generator being in the frontend is that any time I make a change to any document template – the entire frontend must be updated. This is a pain point for clients for multiple reasons, the two main reasons being that the Hailer website reloads whenever we do a frontend push (meaning clients could lose work they were working on), and document templates tend to take a long time to get into production. This also means that any changes to a document template take a long time and can become frustrating for a client if the template is unusable. Therefore, moving the generator to the backend will make it easier to make changes, and the changes will be done faster.

This means that all the current templates need to be converted into forms that the backend can parse and use in the generator. Generally, this means converting the templates coded in TypeScript into JavaScript and adding some functions to parse dates correctly.

Before this thesis observation began, I had a call with my senior developer about converting these templates. He needed to hand over the conversion of the existing templates to me because he needed to work on other aspects of the Hailer application. I had completed the conversion of all these existing templates before this thesis began.

So, whenever I completed any changes to a template – I needed to update the converted templates. I spent most of today converting the templates I had been fixing both last week and the remodelled templates from yesterday.

Once I had completed this conversion, I had a Google meeting with the marketing and finance directors. This was not scheduled, but they are swamped, so when they had time today – they asked if I could speak to them about the documentation page I had created for Hailer the previous month. This call was not long, but we discussed changes they would want to be made on the page and ensured everyone in Hailer had access to this documentation page. I also explained how to add new sub-pages to the documentation page and link various other sub-pages together.

My goal for the day had been to convert the updated templates. I completed all the conversions today, with the bonus of having the long-needed call with the directors regarding the documentation page.

3.2.3 Wednesday 4 May 2022

Non-working day.

3.2.4 Thursday 5 May 2022

Technically a non-working day – but as usual, I get called in for something, and I cannot help but come when called. Luckily, this was short, and I was helping a co-worker with his onboarding. As I am one of the team's newer developers, I offered to help the latest developer with a few onboarding questions.

The issue ended up being that he needed to read through the onboarding documentation I had written on our internal site. So, I directed him to the website.

3.2.5 Friday 6 May 2022

Today was a team-building day. I travelled into Porvoo and spent the entire day with the team. We did a workshop on ways we think the application could be better, had lunch together and then went outdoors to compete in the first Hailer Olympics.

This team building was fun and important. It enabled us to get together, have fun, and get to know one another. Due to most of us being remote workers, it can be hard to put a face to the names of our co-workers. I enjoyed myself and got to know my co-workers well.

3.2.6 Weekly analysis

This week was dominated by template remodelling. It is a long and quite tedious process because most incremental style changes must be approved by the salesperson and the client. This means I often must go back and forth between different versions until the client is satisfied. Luckily, the remodelling for this client was relatively simple and required little effort. Unfortunately, I could not

work on the website this week – however, I am not behind schedule yet, as the release date is set for the end of June 2022.

I wanted to work more on the website this week, so I did not do as much as I would have liked. However, document templates have the highest priority on my task list – therefore, any other tasks are moved to the back until the templates are complete.

The best part of this week was the team workshop. It was great to meet all my co-workers. This was my first time meeting everyone, including the contractors and other part-time student employees. The fun and informal manner the workshop was conducted allowed everyone to chat and get to know each other. The Olympic games were silly but fun – and I think we all had fun doing the funny challenges the CEO and sales team set out.

This week was tedious, and nothing new was worked on or learnt. This is unfortunate – but it did allow me to spend more time on my schoolwork which is always a bonus. Whenever I have weeks like this, I tend to enjoy it simply because it means I can focus on other tasks.

This week was mostly spent working in TypeScript (frontend) and converting templates to JavaScript (backend). We are currently changing from producing the documents in the frontend to the backend because it requires a frontend production reload every time we make any template changes. This change is still ongoing, so when I am working on document templates, I not only have to make the changes in the frontend and follow a complete development task process, including waiting for my senior developer to have time to code review my work. I also need to convert the changes or the new template into the backend format we will use. Currently, the Hailer application is coded in two specific languages – JavaScript and TypeScript. The frontend of Hailer is coded in Typescript in Angular, while the backend is coded in JavaScript in Node.js.

JavaScript was initially developed as a client-side programming language to be used as a simple scripting language for browsers that was adapted and used as a server-side programming language (Pang, 2021). According to MDN Web Developer Docs (2022b) – JavaScript is '*a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else*' (MDN Contributors, 2022j). The power of JavaScript relies on flexibility and malleability. You can do almost anything with it, and multiple frameworks, libraries and functionalities are written on top of JavaScript (MDN Contributors, 2022j), such as application program interfaces (APIs), Node.js, React.js, jQuery and Angular.

TypeScript is one of the supersets built from JavaScript (TypeScript Contributors, 2022), which means that anything valid in JavaScript is also valid in TypeScript – however, what is valid in TypeScript is not necessarily valid in JavaScript. TypeScript is a strongly typed language

(TypeScript Contributors, 2022), meaning specific rules about the types can be used. When creating functions, it must state what types are expected and accepted. Programming types refer to the syntax and the type of the variable being used, for example – a string or number variable and a syntax error.

There are a few reasons for TypeScript being developed, specifically creating a strongly typed version of JavaScript (Shubel, 2022). JavaScript has many quirks, which revolve around compilation and errors due to JavaScript being a loosely typed language (Shubel, 2022; TypeScript Contributors, 2022). When programming with JavaScript, until the code is compiled, JavaScript is unaware of what a variable will be. This can lead to various errors in compilation when the variable is assumed to be one type but is a different type. The developer cannot know that these errors occur until the JavaScript program or function is fully compiled or breaks. Being a strongly typed language, TypeScript prevents these sorts of errors because it knows how to use the variables because their types are defined at the beginning (TypeScript Contributors, 2022).

Another advantage of TypeScript is that it can inform developers of errors as it compiles. When compiling JavaScript code, it is sometimes hard to determine precisely where the error occurs without fully debugging it. TypeScript, on the other hand, alerts developers to the errors during compilation and will generate compilation errors (Pang, 2021; Shubel, 2022).

At Hailer, it was decided to convert from JavaScript to TypeScript in the frontend to reduce the time taken to find and fix errors. However, the backend remains in JavaScript – I am not entirely sure why. As far as I am aware, there are no plans to change to TypeScript, and as a primarily frontend developer, I do not have any strong feelings about the backend. I prefer working in TypeScript because finding errors is far easier than in JavaScript. However, I am not very experienced in TypeScript – most of my experience is in JavaScript from my web developer boot camps and studies at Haaga-Helia.

Working in TypeScript is challenging if you are used to working in the more malleable JavaScript. Remembering to add types to everything is my main struggle. I can remember the types (or the basic ones, at the least), but as I am not used to using TypeScript, I still have the issue of skipping adding the types to new functions, leading to compilation errors.

This is frustrating because I know the issue, and I must keep reminding myself to add types. However, I feel this issue will occur less as I become more practised with the language.

3.3 Observation week 3

3.3.1 Monday 16 May 2022

I spent the previous week focusing on completing various university tasks and therefore missed the retrospective for the last sprint. However, this is not an issue, as most of my work revolves around non-development-related tasks.

Today was a sprint planning day; as usual, I did not have much to contribute beyond stating my goals for this sprint. My goal for this sprint is to complete at least 1 of the template layouts for the HubSpot website. I decided to focus on the global navigation bar.

However, part of the discussion also revolved around onboarding for new(er) team members, and I did ensure to put my opinions forward regarding this. Hailer is rapidly expanding the development team, and many onboarding processes need to be better defined. As someone who joined the team recently, I had written up documentation regarding this - however, the team felt that there needed to be a clearer procedure. So, our technical director arranged an onboarding session for all team members for Wednesday.

Once sprint planning had concluded, my goal was to set out the global navigation bar's layout and add the custom buttons I had created previously. This global navigation bar (hereon referred to as the navigation bar) has four main elements: the Hailer logo, menu bar, CTA buttons and the language switcher module. The design can be seen in Appendix 1.

Therefore, I created a new template layout named the header template. In this header template, I first laid out the four building blocks. These building blocks were, for now, divs with explanations of what is supposed to go in each block. From here, I knew I needed to create modules for the different elements. I specifically used HubSpot drag-and-drop areas in this template layout to allow content managers to make changes as necessary to the designs.

Once these major blocks had been completed, I decided to preview this header template in HubSpot. There are two ways to preview templates, and at first, I used the built-in preview feature of the Design Manager, which displays only the specific template. This was not helpful for me, as I wanted to be able to edit the modules as is typically done by content managers.

For me to be able to do this, I had to create a new webpage or landing page using my HubSpot theme. However, this was a learning curve as I still needed to do the HubSpot tutorial videos on this development section. I needed to create a demo webpage using the template to see the template used in the ‘live’ site. This ended up being very confusing for two main reasons – firstly, I had named the theme ‘test,’ which meant that there were multiple test themes, and I could not find

the theme I wanted by simply searching for the name. Secondly, when initially creating the theme, I followed the HubSpot tutorials and created a new theme using the built-in generic HubSpot template. This meant that any previews of the themes were the previews of the generic HubSpot template. It meant I could not find the theme I had just created. At this point, I chose to delete the newly created landing page as I had no idea what theme to use.

This was nearing the end of the workday, and I felt I had not accomplished my goals for the day. However, I learnt a lot about how HubSpot drag-and-drop areas work and how to add modules to drag-and-drop areas. As I had also realised that I needed to rename my theme and add preview images – I decided that would be the most important task the next day.

3.3.2 Tuesday 17 May 2022

Today started with a road mapping meeting. One of the major projects Hailer is currently involved in is the DigiOne project. DigiOne aims to replace the current school's technology system, making it easier for students to get the information they need and for parents and teachers to add and receive information (DigiOne Project, 2022).

The goal of this meeting was to bring the development team of Hailer up to speed on the feature requirements, needs and requests that will need to be implemented in the next two years. This meeting discussed the various high-level features, their essential requirements, and the project roadmap. This specifically focussed on what features need to be developed before others and which are more important than others and therefore are a high priority to implement. We were also shown the prospective roadmap proposed to the DigiOne team, which features they want to be released, and by when.

After the road mapping meeting, I planned to do some research on HubSpot theme settings, as well as start editing the settings. I began reading the theme settings documentation on HubSpot but realised that I first needed to map out the template layouts before I could grasp what was happening in the theme settings. This is how I tend to work and is not suggested by HubSpot documentation in any way. I like to have a high-level layout of the feature or component I am coding before I start doing 'nitty gritty' details like theme settings. The website currently has only generic template layouts, partials, and pages. My goal was to create the layouts and partials, with at least the divisions and modules I would need to make.

The header template partial was already structured, so I did not make any changes in that file. However, the footer template partial had not been coded, so I removed all the code in this file. Footers are not drag-and-drop areas, as they need to remain the same throughout the website. Therefore, I did not need to create a drag-and-drop area but a standard layout using divs.

I decided to use a grid layout for the footer, as there were four columns in the design, which can be seen in Appendix 1 and Appendix 2. I used a grid layout generator to create the layout and added various placeholder phrases in the footer file. I use grid.layitout.com as the generator. I like to use the generators because they visually represent the grid, which I often struggle to visualise. For the website pages (home, landing, blog, contact us and about us), I created a basic list of sections in each template and the initial modules and partials I will use. This took most of the day, but I will need to continue working on this tomorrow.

3.3.3 Wednesday 18 May 2022

Today I continued with the section template layouts. My goal is to complete them and get the theme settings completed. The rest of the section template layouts went quickly, as I had already worked out what I needed to do yesterday. I managed to finish them before our planned ways of working meeting, which was planned on Monday.

Once the meeting had concluded, I went back to working on the HubSpot website. I had finally completed the section layouts and could now focus on my original task of the week – the theme settings. Moving from HubL and HTML coding to pure JSON was a change for me. I have hardly ever worked with pure JSON strings and objects. I tend to use the data from these objects and strings but not make the objects myself. Therefore, I had to do a lot of reading on how to do the theme settings. I kept the documentation page open while doing everything and constantly referred to it.

Theme settings are set in the fields file in HubSpot. In this file, you create a main JSON object of a specific type. Only certain types are allowed according to HubSpot, including but not limited to global colours, global fonts, spacing, typography, buttons, forms, and tables. These are the objects I decided to focus on, as they will likely be used on the website, and I want the theme to be unified in design.

The generic field file comes with all these objects already filled in. Therefore, I did not need to create any of these. However, I did need to change the default values and add more. Within the JSON object, there are children objects which set the actual values of these objects. Within the children's array, there can be multiple objects. It is also possible to have inherited values which import the values from the theme settings in HubSpot and to have a default value. Default values are required as fallbacks.

Firstly, I renamed all the headings to match the Hailer design system naming, making it easier for me to choose the correct typography. I changed the values of these headings to match the design system and added the line-height property. After editing these headings, I added nine more child

fields to the typography field. These fields all corresponded to the Hailer design system typography and the specific Hailer website typography that the UI/UX designer had created for me.

These changes took the remainder of the day, and I felt I had finally completed something this week. After struggling with themes last week, I had finally completed the basic layout and settings for the website and was ready to work more on the actual website designs and modules.

3.3.4 Thursday 19 May 2022

Non-working day.

3.3.5 Friday 20 May 2022

Today my goal was to complete the header block, excluding the language switcher. The only element I still needed to complete was the menu and the associated dropdowns. I also needed to style the buttons but decided to work on the more complex part first.

To create the menu, I needed to implement a macro. Macros can be created inside a module; however, I decided to create a macro file so I can use the macro in any module, template, or section. The first macroblock was a simple for loop copied from the generic menu macro. This for loop would loop over the menu items in the HubSpot menu field in the module and create the list items for each menu field and its associated child dropdowns.

The second block was the barebones layout of the navigation menu. This block was designed to create one menu item (list item or li). This block would then be placed inside the first macroblock, which would be inside an unordered list (ul) which would be inside the module itself. As some of the list items would have children or dropdowns, I needed to create a separate for loop or iterator to go through the child dropdowns. For the child dropdowns, I knew that there would not be dropdowns for these as the design and layout of the website did not include them. Therefore, I could create an unordered list with a for loop inside it, creating a list item for each child item.

While coding these blocks, I realised that I was reusing the same code repeatedly – the URL styling code. Therefore, the fourth and final macro was created that styled the URL items. The main reason for creating this URL macro was due to the active branch or node linkage that HubSpot provides. These active branches or nodes enable Hailer to determine what page a customer was on when they decided to log in or register, as well as create a website heat map to see where there are pain points or problem areas in the site.

Once these blocks had been completed, I needed to add them to the header menu module I had created while doing the site layout earlier in the week. I created two different menu modules, one

for mobile and one for desktop, as the layouts of the mobile and desktop header menus were significantly different (Appendix 1 and Appendix 3).

At this point, I had spent the entire day on the macro and the styling. I did not need to do any JavaScript work on this module, as I used media queries to change from the desktop menu to the mobile menu. However, I will need to style the modules and enable the dropdown hover/click functions next week.

3.3.6 Weekly analysis

This week was primarily spent on creating modules and layouts of the website, as well as a lot of reading through the documentation on the theme and its properties and settings. While much work was done this week, I feel that not much was achieved simply because the work that was done was not very visible. Creating macros, changing settings, and creating the website layouts mean that these tasks are now done, and I can work on more visible changes, such as making the header module, which I will be using in the global navigation section. For the most part, I only completed my daily goals in the last two days of work. I do not think this is an issue simply because I have created much behind-the-scenes work I was unaware I even needed to do until I discovered it.

There is a benefit to all of this behind-the-scenes work, as I know what exactly needs to be done and can create the plan for the following modules and what needs to be done in those modules. I still need to do some finishing touches on the global navigation bar, but the changes are minor and should be able to be addressed on Monday without too much issue.

Developing in HubSpot is accomplished using the Design Tools (also known as Design Manager) GUI (HubSpot, 2022c). This GUI is found in the HubSpot CMS, aiming to create a more customisable experience for content managers (Juviler, 2021; HubSpot, 2022c). HubSpot is both a CRM and CMS, with various tools that allow marketers, sales, and content managers to create and edit various campaigns, website content and client interactions (Juviler, 2021; Pacewicz, 2021). The CRM portion of HubSpot focuses on the sales funnel, allowing the salespersons and marketing team to manage clients (Juviler, 2021). In contrast, the CRM portion enables content managers to create, edit and update the website's content, blogs, emails, and other digital assets (Sirk, no date; Juviler, 2021; Pacewicz, 2021).

The goal of using HubSpot as the Hailer website CMS/CRM is to allow our marketing and sales teams to have better access to client information and statistics and drive sales, as well as enable the marketing team to analyse and forecast. The other bonus of using the HubSpot CMS/CRM is

that the content manager(s) can change the data on the website whenever they want for whichever campaign they are currently running.

By allowing the content managers freer reign over the content and layout of the website, it reduces the amount of development necessary on a regular basis. The content managers can change the website's basic structure, but the theme and major blocks (header and footer) remain consistent throughout the website and various landing pages. While I am focussed on the development side of HubSpot, I need to keep the usage of the site in mind when creating new pages or modules. I am in constant contact with the CMO to ensure that all blocks, modules and pages are adaptable and suitable for the marketing team's uses.

For example, I can create a single module that can be used throughout the website, and the content manager(s) can select the module from the theme modules and drag and drop it wherever they wish in the page layout. I have designed the website mainly using HubSpot Drag and Drop areas. Essentially, I create a basic website layout and then allow the content manager(s) to add whatever extra information they wish. This enables managers to create different versions of the website pages for different clients or different campaigns. (HubSpot, 2022c, no date)

Another key feature of the HubSpot CMS is that I can create various fields or style fields for the content manager(s) to change a specific module's styles and fields (HubSpot, 2022c, no date). Within each module, I can set styles which are not changeable to ensure that the theme remains consistent, but there are various changes that I have allowed the content manager(s) to make. The main change that I have instituted throughout the website is the ability to change the colours. While Hailer currently has a set colour scheme, if it changes at any point, the ability to change these colours means that the website does not need to be reworked whenever we do a colour change. Style-wise I have not allowed many changes simply because the theme needs to remain consistent throughout, and if there are too many options, it can ruin the ‘feel’ of the website. However, I allowed the content manager(s) to change the spacing of the various elements to ensure that the overall layout was not too condensed and to create a cleaner look and feel.

However, I have ensured that each module's images, links, text fields, and backgrounds are entirely changeable. The other consistent changeable factor I have included is the ability to change the layout of the text/images. I used the grid layout to set where a text field and image field would be in the module, and then allowed the content manager to choose a side in which the image will be found (left side, centre, or right side) and then moved the text accordingly. I specifically used grid layouts in these sections to ensure that the size of the image and text columns were consistent, no matter the placement of the image. I chose this to allow the content manager(s) more freedom and flexibility when designing content.

3.4 Observation week 4

3.4.1 Monday 23 May 2022

This Monday begins the first week of full-time work for me. I plan to focus mainly on the website, with the goal of releasing the website at the end of June. Today I will focus on completing the global navigation module I began last week.

I still need to do some styling for the header menu and make the layout of the mobile and tablet versions look the same as in the design (Appendix 3).

I will start by ensuring the menu module looks as it should for desktop and mobile/tablet versions. Last week I completed the menu macro and did most of the desktop layout styling (Appendix 1). Dropdown children are only on the first and third menu items, and I also need to make the animation of the dropdowns smoother.

When creating this module, I used multiple online sources to figure out how to set it out. I often follow CodePen examples, as I can use the animations and layout of these designs and change them on the online source. This allows me to test various versions so that when I implement the code into HubSpot, it is already completed, and I do not need to do as many tests to ensure it works.

I had been using a design in CodePen to lay out the dropdowns and how to colour the backgrounds of the dropdowns and the parent menu item. My first primary task of the day was making the backgrounds of the dropdowns the same as the parent menu item when the parent item is hovered over. This included hiding and showing the dropdown on hover.

At first, when I hovered over the menu item, the dropdowns showed but did not have a background colour. This did not make sense, as I had set the background colour to be the same as the menu item and did this styling last week. I inspected the elements in Chrome developer tools and used the Elements tab. In the Elements tab, I looked at the computed styles and saw that while the background colour was being set correctly – there was a style that superseded this background style.

Unfortunately, finding where this style came from took a lot of work. There were two sources of styles; both termed main.min.css. This was an issue, as according to my HubSpot Design Tools, there was only one main CSS file, and according to that file – the background colour was set correctly. Therefore, I had to find out why there was a secondary main CSS file. I searched for similar issues, and while reading through the HubSpot developer forum issues – it was mentioned that it is possible to add a CSS file globally on every web page through the content settings.

An external company created the current Hailer website, so I was unsure how they set the files and global styling. I tried to follow the suggested styling according to HubSpot documentation, which says that themes should contain all global styles, and in general, one should not set a style page in content settings.

If a styles file is set in the content settings – no matter what the theme overrides or settings are, the website will follow the styles in the content settings file. This is not ideal. Especially when wanting to create multiple pages with different themes. While this is not what we plan to do with the website, having a globally set styles file is still not ideal.

When I went into content settings, I saw that there was indeed a global file set there. I was unsure if I could remove this file because if I removed it, it was possible that the current Hailer website styles would stop working. Therefore, I had to go into the theme of the existing Hailer website and add a link tag to the CSS file I found in the content settings. I then removed the file in content settings, refreshed the website, and hoped I had not broken the styles.

Luckily, adding the CSS file to the current website's header did the trick, and no style issues were seen or reported. When I reloaded my site to see if the fix had worked – the background colours were correct. However, removing the main style page caused the layout of the dropdowns to break. Instead of dropping down underneath one another, the child items were layered on top of the main menu item.

This was a simple fix as it involved the translation of the dropdown items. The menu layout was that of an unordered list with another unordered list inside of it (for elements with dropdowns). Initially, I had translated the child unordered list downwards by a few pixels. This meant the child list was sitting below the menu item instead of on top. Some of the sizings had changed when I removed the global CSS file. I am unsure what, but I needed to increase the translation downward by a few more pixels.

I now got to work on changing the style of the mobile/tablet version to be the same as the design (Appendix 3). The basic layout of the menu meant that the styling of the mobile/tablet menu was much more straightforward. All the items are visible, which I need to change. However, the basic layout will remain the same. I made the navigation tag for the desktop version using flexbox to create a horizontal version. However, this was not necessary for the mobile/tablet version.

To hide the dropdowns, I chose to use jQuery so that the items would only appear when the parent item was clicked on instead of the hover function of the desktop version. As I was using jQuery to set this, I removed the media query, which changed the display property of the desktop and mobile

menus. I had previously set it so that after 800px, the desktop menu would be visible while the mobile menu would be hidden. Before 800px, the opposite was true.

I removed this property from the media queries and used the addClass and removeClass functions from jQuery. These functions are built-in jQuery functions that allow one to add or remove classes from a specified element in the document. In this case, I wanted to add a hide class to the desktop version and remove the hide class from the mobile version.

I also added a hide function (which adds the display: none property to the style of the selected element) to the child item list. To create the onclick function, I used the built-in jQuery click function and the toggleClass function. The toggleClass function adds and removes a class from an element, depending on whether the element has the class already or not (jQuery API Documentation, no date b).

I did try using the toggleClass for hiding or showing the main menus. It, unfortunately, did not work as I wanted it to and would hide or show the wrong elements. I wanted more control over the hiding and showing of the main elements, so I chose not to use toggleClass.

However, toggleClass works perfectly for onclick functions because the hide or show class would be added only once the click has occurred. I added the toggleClass function to the arrow icon next to the menu item. I created a specific class that flipped this icon 180 degrees.

While working on these functions, I had the jQuery documentation open, as I do not often use jQuery and therefore do not remember the functions and their parameters or variables. I was reading through the toggleClass function because I wanted to add a duration or timeout to the dropdown. I found a toggle function allowing me to add both a duration and a built-in easing animation.

This toggle function works the same as the toggleClass function, but instead of me needing to create a CSS class to toggle, it has built-in easing animations that I can use instead. I, therefore, opted to use this function instead, as it would be far simpler to use the built-in animations.

Now that the layout, animations and overall styling of the header menu were completed – it was time to add this menu module to the header section and test it out on the live site. As the global template partials of the header and footer are used throughout the website and should not be changed in any way – I decided not to use drag-and-drop areas in these templates. Last week I created the partials using drag-and-drop areas; however, I chose to remove those from the header partial and instead use the traditional div layout.

I also created the mobile menu and associated overlay within the header module. At first, I tried making one header that changed the menu from desktop to mobile. However, this did not work because the mobile menu layout is quite different from the desktop menu, and it also needs to be able to slide in and out with the click of the hamburger menu.

Therefore, I first split the header into two different divs, one for desktop and one for mobile. I added all the associated modules and did the desktop header's basic styling layout. This was relatively simple, and I chose to use a grid layout to set out the four columns (logo, menu, buttons and language switcher). I still have done no work on the language switcher, as I decided to leave it until the entire webpage is completed.

The main reason I like using a grid layout for headers and footers is that I can set the sizing of these columns, and they do not move around as much when a page size changes as it does in flexbox. This is purely a personal preference, as both options achieve the same look. Grid typically requires more code than flexbox, but it has the advantage of the sizes being set, and no matter the screen size, the relative size will remain the same ratio.

Once I had completed the layout of the desktop header partial, I started to work on the mobile version. To style the desktop version, I had to remove the design of the mobile version, so I had to redo the layout completely.

The layout on mobile differs quite a bit from the desktop (Appendix 1, Appendix 2). Only the logo and the hamburger menu icon can be seen on the main page (Appendix 2). The mobile menu pops out when the hamburger icon is clicked (Appendix 3). This menu includes the logo, navigation menu (which layout was already coded in the menu module) and the two buttons (login and signup) (Appendix 3).

I first added media queries to hide the desktop layout when the screen size was smaller than 800px and then added the basic structure of the mobile layout. This layout included the logo and hamburger menu icon. I styled those until they looked like the design (Appendix 2), then added a separate div containing the pop-out menu. This div included another logo, the navigation menu, the two buttons, and the language switcher. I chose to use flexbox for this part, as the size and spacing needed to be very malleable because the mobile version covered not only all mobile devices but also tablets.

To create the pop-out functionality, I made two functions in the main.js file, wherein I added and removed classes using addClass() and removeClass(). If the menu was not visible, it could only be opened by clicking the hamburger menu item (therefore, the handleMenu() function ran), which

added a slide animation to display the menu and the overlay which blurred the background behind the menu.

Once the menu popover was out, the other function because usable. I named this function handleClose(), and it ran if the overlay was clicked or if the close button (X) was clicked. I tried using toggleClass or toggle functions at first. However, they did not work the way I wanted them to – instead of always opening or closing the menu, they would toggle the classes. If you clicked the buttons enough, it would eventually toggle them at the wrong times (open the menu when you click to close).

The classes I created for these functions have animations that slide the overlay and menu from the left, using translation animations and transitions with delays. The open menu is fixed, and you can scroll the website behind it but not click on anything except the overlay, close button or buttons in the menu.

I had previously created a hide function for the menu dropdowns for the mobile version. However, on testing this on the header template, I realised that some of these items were URLs; therefore, instead of opening the dropdown when you clicked on them, they opened a new webpage.

This did not work. So, I returned to the menu module and added a new menu field for the mobile version. This version did not have URLs in the main menu items and instead had more dropdowns that linked to the pages the original menu items linked to. This did not affect the header template, as it used the same menu module. The only necessary change was in the menu module.

The final thing I did today made the mobile header (menu closed) a sticky header when you scroll down. This was interesting, as I needed to do quite a lot of reading into window offsets and scroll functions in jQuery.

Window offset tells you how much is scrolled in the y-direction (i.e., vertically) (MDN Contributors, 2022k). Scroll functions are functions which only run when the window is scrolled. Therefore, the header becomes sticky whenever the window is scrolled past a specific point. I chose the edge of the header div to be the ‘sticky’ point.

Offset functions are jQuery functions, which return an element's top and left coordinates relative to the document (jQuery API Documentation, no date a)—finding an element's offset().top tells me how far the top of the element is from the top of the screen. Therefore, if the top is higher than the scroll – the header must become sticky.

After working on this header template, the entire day, I had finally completed all the necessary pieces and could tick off one block of the website from my ‘to-do’ list. Next on the list was the hero block. However, this was the goal for tomorrow.

3.4.2 Tuesday 24 May 2022

Today my goal was to complete the hero block of the new website. It is a relatively simple design of two rows. The first row consists of two columns, one with text and one with an image. The second row is a logo carousel of different partner companies.

Therefore, my goal for the day was to create this entire section and have it ready for testing on the home page. I created two modules, one for the logo carousal and one for the hero image block (hereon referred to as the hero block).

At first, I tried to create a logo carousel module myself. I looked on CodePen for creating carousals prototypes and saw many of them used Slick slider. Slick slider is a jQuery plugin that makes an automatic carousel of images (Wheeler, no date). To use Slick slider, you need to add the plugin script in the header and footer of the webpage.

As this website in HubSpot is split into various modules, I was unsure how to link these scripts. Therefore, I searched for ‘How to add Slick slider to HubSpot’. One of the first search items was from the HubSpot developer community forum, which linked me to an article by HubSpot. In this article, there was a ready-made carousal in use.

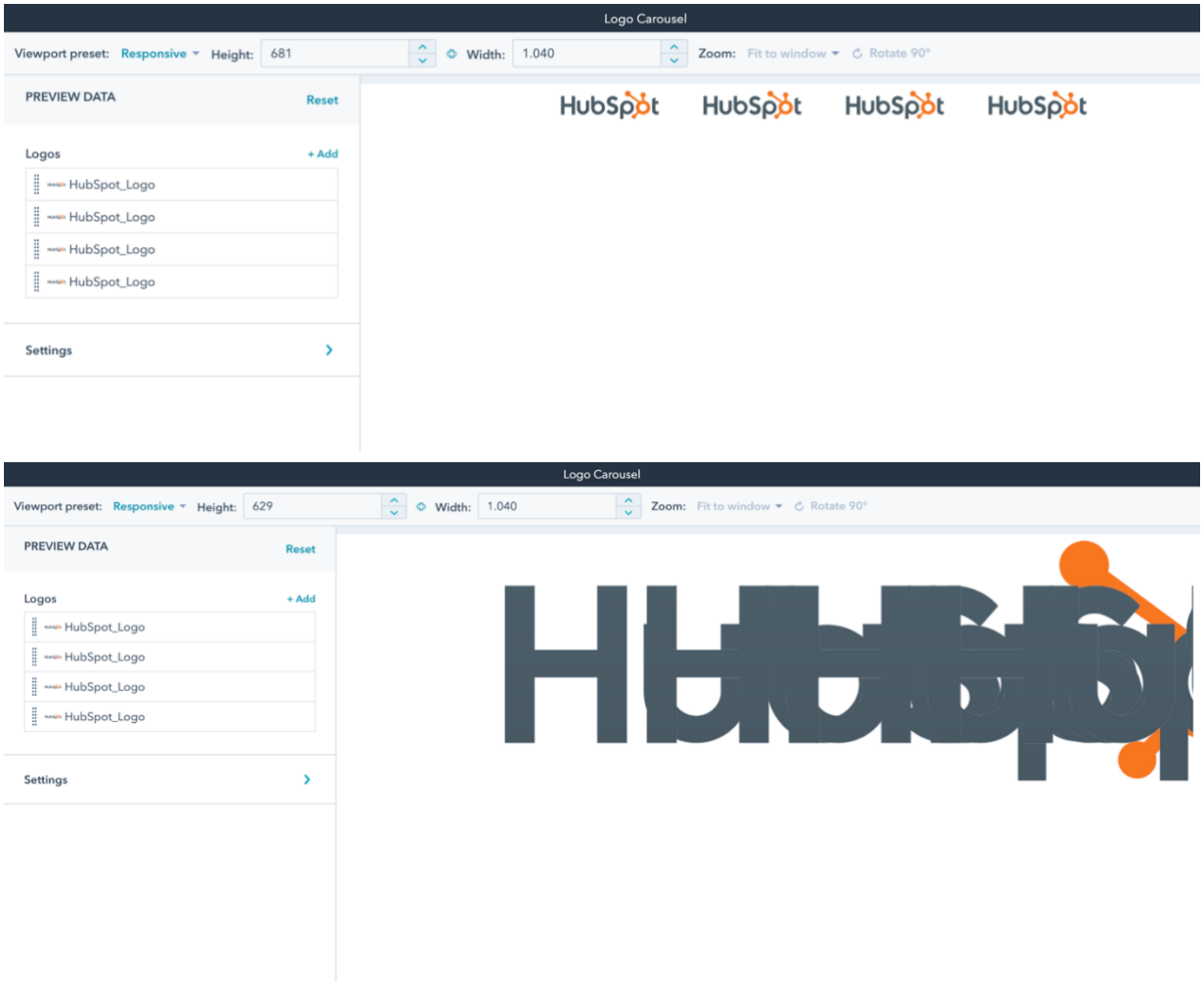
Instead of creating a new module, I could use the HubSpot carousal and edit the styles as necessary. This logo carousal could be added to the developer tools using the HubSpot marketplace. I found the module and added it to the developer tool, but then I ran into an issue changing anything in the added module.

I chose to clone the module into the hailer-2022 theme and change everything as necessary from there. After cloning the module, I made the mistake of removing all the CSS styles. I assumed that the Slick slider module did most of the styling; therefore, all the styles currently found were to style the logo carousal in the way HubSpot wanted.

The unedited module and the module with all styles removed can be seen below in Figure 2, with the unedited version on the top and the bottom being the styling removed image. This was incorrect. I undid my editing and slowly read through all the CSS classes to find which styles were necessary and which were not.

Part of checking, which was necessary, was removing the class and seeing what would happen. After I had gone through and removed all the unnecessary styling, I added the module to the hero block template partial. Now that the logo carousal was complete, I moved on to the hero image block.

Figure 2. Logo carousal from HubSpot marketplace



The hero image block is a simple design of two columns. I decided to use a grid layout again because it would set the columns to equal size. This is the first fully editable block I have created so far. This means that all the content needs to be changeable by content managers. This includes the image, title, text and CTA button. I added all these fields in the design manager and then added them in their relevant spaces in the HTML.

I decided to make most of the styling customisable because if the content changes, there is a high chance that the spacing, alignment, colours and fonts will also need to be changed. Therefore, I added multiple style fields to this module. The major changes will revolve around the spacing of the

different elements, but I added the option to change the colours of the background, button background and text, and hover colours.

For button hover colours, I added a Boolean choice field for content managers to decide whether they wanted to add specific hover colours. I did this because we have set hover colours, but for any reason, these could change, or the content manager may want the hover colour to darken instead of lightening.

I also added two different colour options for desktop and mobile. The reason for this is due to the design in Figma. While the design is consistent in Figma, there are slight differences. While creating the desktop colours, I realised that, in some instances, the content manager might want slightly different colours on mobile. This was not necessary, but I felt that adding this option increased the usability for the content managers.

Once these fields had been added and the layout set up, I started translating the title over the image as is done in the design. I ran into a significant issue here, though, as the translation of the title needed to change whenever the screen size changed. I created multiple media queries to try and solve this issue. Whenever I opened the preview and changed the screen size, I had to add more media queries. I decided to discuss this with the Marketing coordinator, who agreed that spending so much time on these media queries made no sense when the design could be changed.

I then spoke to the designer, who agreed to change the design. I, therefore, left that part of the hero block alone and went on to style the CTA button. The CTA button did not need much styling, but it changed in size and position depending on the screen size, so I had to change the preview's screen size to find the breakpoint at which the button needed to change. Included in this was changing the grid layout from two columns and one row to one column and two rows.

This is another reason I like using a grid layout. Whenever you need to change the layout due to screen sizes, you change the grid template and, if necessary, change the grid positions of the elements. I chose to keep it as a grid layout because I wanted the image section to be a specific size ratio compared to the text, and flexbox tended to make the image too big.

At this point, I had mostly completed my daily goal – although the block was not complete, I was waiting for the designer to redo the designs and could not continue.

Tomorrow, I plan to work on some header block fixes. When I tested this hero block on the home page, I noticed that the buttons (signup and login) in the header are spaced strangely and that it is impossible to have the colours different as they are meant to be on hovering. So, I will also need to fix these issues.

3.4.3 Wednesday 25 May 2022

The goal for the day was to complete the header block as well as fix the buttons in the header block. Most of today was spent discussing how the design will need to change with the designer. She was busy with other designs yesterday, so she did not have time to work on the designs and did not fully understand why the current layout would not work. We discussed the issue early in the day, but she asked that I use an SVG instead of text.

I tried that fix; however, it still ended up with many different media queries. I messaged her again, and we had a Google Meet about the website with the marketing coordinator in the afternoon, where I showed them the issue. As you change screen sizes, the design jumps due to the media queries. The other problem was that over 15 media queries were necessary just for this SVG.

Another issue is that the SVG needs to be changed to a different SVG on the mobile screens, as the design differs. This is not an issue, but it does mean we have to load two separate image SVGs and switch between them as necessary. Usually, this would not be too much of a problem, but on top of this – using an SVG means that the content managers can never change this text. This is a significant issue because the hero block title might vary depending on campaigns, times of the year and according to the different languages. Before we had this call, I also created the Finnish version of the home page. To do this in HubSpot, create a cloned multi-language page wherein you change everything to Finnish. When I changed the SVG to the Finnish version, the media queries no longer worked.

This meant that there should be two completely different modules for the various language sites, which is unnecessary extra work. Each module would need a lot of media queries to get the title to work, and in general, it would make the content manager's job harder. The marketing director also said she wanted the title to be text so that she or the content managers can change it whenever they want instead of creating multiple SVGs and changing them around and hoping that the media queries work for them.

Therefore, the designer had to redraw the designs of the hero block completely. I will likely only get the reworked designs next week. Unfortunately, I spent the entire day working on these media queries and the discussions with the designer and marketing director, so I did not have time to work on the button bugs.

3.4.4 Thursday 26 May 2022

Non-working day (public holiday).

3.4.5 Friday 27 May 2022

The goal for today is to fix any current bugs, specifically the button bugs I noticed the past few times when I loaded the header into a ‘live’ site. When I added the buttons into the header block, I assumed the blocks would work as usual modules and allow me to edit the button individually.

Unfortunately, this was not the case. I then had to create two buttons for the login and signup buttons. I made two cloned buttons from the already created customisable button module. I removed some of the style fields I had created for the customisable button, as they were unnecessary. I then added these two buttons to the header block. To check that the buttons worked, I used the live preview site. When I opened the site, I noticed that the buttons were spaced strangely, not in line with the other elements.

I then spent many hours researching why this was happening and possible fixes. I downloaded various HubSpot marketplace modules to try and see what I was doing wrong. I deleted all the styling and still could not figure out why the buttons were popping ‘over’ the edge of the header.

Eventually, I contacted one of the intermediate developers in the team. He got onto a call with me and immediately noticed the issue. I had not used the correct display property. I used display flex instead of display block. I also needed to add overflow: none to prevent the button from overflowing. These fixes immediately fixed my bug. I felt pretty defeated and frustrated because I had spent an entire working on this, and the fix ended up being extremely simple.

3.4.6 Weekly analysis

This week was filled with minor and annoying fixes for me. While I had done much work during the week, due to Friday being so frustrating, I was disappointed and annoyed with myself for the week's progress. Most of the week was spent discussing with various developers and the UI/UX designer. I had to contact the designer because of the necessary design changes constantly.

Unfortunately, this is often the case when developing or redesigning features. When working with new features, especially in new languages, it is not necessarily possible to know how it would work out and if the designs will even work. It is especially true with CMS, as there is code built into the CMS that cannot be overridden, which means that the designs are sometimes impossible to complete due to this code.

Designing new websites is always challenging, especially when working in a CMS – however, the UI/UX designer's designs are highly detailed and well-documented. Therefore, when creating the designs in HubSpot, I am fully aware of the sizes, spacing, colours and layouts the designer has created.

Initially, the UI/UX designer designed the desktop site but gave me no designs for the mobile or tablet. I then requested that she create designs for the smaller screen sizes. The main reason I asked that she create designs for these sizes is that the desktop version needed to be more responsive to be used on mobile and tablet sizes.

According to Hailer HubSpot statistics, 27% of our users interact with our website through their mobile phones and 19% through their tablets. Due to the differences in display size (and resolution), it is practical and almost essential to create modern, scalable websites which can be used on any device and screen size to create a compelling experience on all devices (Decker, 2020; Braccialini, 2022).

Responsive web design focuses on having an overall design which changes shapes, spacings, sizes and layouts according to the screen size (W3Schools, no date; Decker, 2020). I expected the UI/UX designer to create a responsive version of the site, a scaled-down version with slight changes. However, she opted to go for an adaptive design. Adaptive website design focuses on having set layouts for different screen sizes that are only shown/hidden at a specific screen size (Soegaard, 2020; vickykumar7061 and annieahujaweb2020, 2022).

Neither adaptive nor responsive are necessarily better than the other (Soegaard, 2020; Lien, 2021; vickykumar7061 and annieahujaweb2020, 2022) and should be used according to the preferences of the designer and developer. There are pros and cons to both designs' methodologies. Mainly, responsive design is considered easier to code, has less coding required (simply needing to create media queries instead of new layouts for every screen size) and is widely used (Lien, 2021; vickykumar7061 and annieahujaweb2020, 2022). Adaptive design, on the other hand, is arguably a better user experience, as the layouts are adjusted automatically to the device type and screen size (Soegaard, 2020; Lien, 2021; vickykumar7061 and annieahujaweb2020, 2022).

Generally, I have gone for responsive designs simply because I did an advanced course in CSS, and the lecturer did all the lectures using responsive mobile-first design. Mobile-first design is when the website is created first to fit the mobile devices and then scaled up to accommodate larger screen sizes (Ferguson, 2021). Whereas desktop-first design is first created to fit a desktop screen size (Ferguson, 2021) (1080px and up) and then is scaled down to work on smaller screens.

The choice between mobile-first and desktop-first is, again, irrelevant. Either works but requires slightly different mindsets and media queries. It can be argued that the design choice should be based on what your users use the most (desktop or mobile), but overall, it is a personal preference.

The UI/UX designer created three website versions for me to implement. Generally, there is a minimum of six designs in adaptive web design (Lien, 2021), so this design is not quite adaptive.

However, it is also not responsive design because there are significant differences in the layout, especially in the header, global navigation bar, and footer sections (Appendix 1, Appendix 2).

The design of the header is quite different and will require at least two versions. The first version will be the desktop header (Appendix 1), a simple row of information including the Hailer logo, navigation menu, CTA buttons and language switcher. The other version would be that of the mobile (and tablet) header (Appendix 3). The mobile version requires an entirely different HTML layout, with overlays, hidden elements, and various animations. The first iteration of these header and navigation sections has already been created this week, and I will likely need to do many more revisions once I test it on the live site.

3.5 Observation week 5

3.5.1 Monday 30 May 2022

Non-working day

3.5.2 Tuesday 31 May 2022

While working on the buttons on Friday, I noticed that the theme settings I had set up were not pulling through to the website. I invested this in the morning and made it my goal to make the theme settings work correctly on the website.

The issue was that I had to use the theme settings from the fields.json file in the theme.overrides file. I had to pull the data from the JSON file, set them to variables, and then use those variables to set the various CSS properties.

This is a lengthy process as I need to create these variables for each element I want to have theme settings. I have created the JSON file fields already, so I need to pull that data into the theme.overrides.

I started with the typography overrides and got confused about setting the 'simple' changes for the H1. The basic variable declaration can be seen in Figure 3, and the use of the code can be seen in Figure 4.

Figure 3. Example code of the theme overrides file

```
{% set primary_font = theme.global_fonts.primary %}
{% set secondary_font = theme.global_fonts.secondary %}

{# headline 1 #}
{% set h1_font = theme.text.h1.font %}
{% set h1_text_transform = theme.text.h1.transform %}
{% set h1_text_lineheight = theme.text.h1.line_height %}
```

Figure 4. Example code of using the variables set in theme overrides

```
h1,
.h1,
h1 > span {
  {{ h1_font.style }};
  color: {{ h1_font.color }};
  font-size: {{ h1_font.size ~ h1_desktop_font.size_unit }};
  text-transform: {{ h1_text_transform }};
}

h1,
.h1 {
  line-height: {{ h1_text_lineheight  ~ 'rem' }};
}
```

This simple setup took most of the morning to figure out. This took me so long because there is very little documentation on the theme overrides file in HubSpot.

While there is documentation on the use of various elements and what you can edit in the fields.json file – declaring the variables and knowing what to use where and when was very confusing. I originally copied the default theme overrides, so the {{font.style}} was understandable. However, as the designer has a set line height and transformations for the various typographies, I also had to set those.

At the same time, I decided to change the units of the font styles from pixels to rem. I find it is better to use rems for fonts because then when a user has set their browser font size differently from what we expect (due to poor eyesight etc.) – the font size will always work with the default font size of the browser. Using rems also means that when you zoom in or out of a page, the font sizes adjust accordingly.

Therefore, I spent the entire day today firstly changing the font style units to rem and ensuring all the font fields in the fields.json file had all the associated font weights, transformations, line heights, sizes and default colours. I had done most of them, but I double-checked and found a few fields in which I had not added properties.

This was tedious and painstaking work and took much longer than I realised it would. The end of the day came much faster than I realised, and I had only completed the font import and declaration. I still needed to check that the font changes were working, however. So, I needed to find out whether my changes were pulling through.

3.5.3 Wednesday 1 June 2022

Today I planned to complete the theme settings updates to the units as well as importing and declaring the various CSS properties and classes. However, I was asked by the finance director to write up some documentation on the employment procedures in Hailer. The reason she asked for my help is due to two reasons. The first being that she had yet to work with our internal site, which I had set up for documentation, and the second being that I tended to be the one to ask her odd questions regarding employment and other procedures.

In this specific case, it was regarding the leave policy, as I was planning my summer holiday leave and needed clarification on how the process works. This is also the first time I am working 'full-time' at a Finnish company and receiving paid time off – so I had many questions. Eventually, we brought the other new developer into our meeting so he could tell us what other questions or information he would like on the documentation site. Like me, he is a foreigner and not used to the way everything is done in Finnish labour law. This meeting took most of the morning, so I spent the rest of the day continuing yesterday's work. Firstly, I checked that the changes I was making were, in fact, being shown on my live preview site.

Luckily, the changes were pulling through – which meant I was doing it all correctly and could move on to completing the declaring of the variables. The rest of the variables ended up being far more complex, as multiple CSS properties, such as border, background colour, colour etc., needed to be set.

However, I finally managed to complete the declarations today. I am annoyed and frustrated with this project because I feel I am not getting much done or seeing the end anywhere near. I am not a fan of working with a CMS, and while I am happy to be learning a new system – I wish I could be creating this website myself without the need for this 'low code' solution,

I feel that this low-code solution creates more work, meaning I need to write even more code than I typically would for a website. The constant back and forth between modules and the generally clunky design tools manager frustrates me.

3.5.4 Thursday 2 June 2022

As I had made some significant changes to the styles of the website, I needed to go through the already created modules and fix various CSS properties. This was mainly because the font line heights were changed and made much larger than the standard line heights. I had to go through each already completed module and first ensure that the correct type of typography was being used and then also make sure the preview worked and looked as it should.

To ensure every element already created was using the correct typography, I had to go through the various modules – header, hero, footer, menu, and buttons. I had to also follow the design standards from Figma. The best way I found to do this was to start with one module and go through each element in the module in the preview and ensure the font size, line height etc., matched the design information in Figma.

While doing so, I also added new classes to the theme overrides, specifically for elements like the footer list items, which are smaller than usual list items and have their own specific class.

I ended the day having completed all these checks but felt quite disappointed and frustrated with working in HubSpot because I need to go through so many different modules, settings and other files and folders to be able to make simple changes and then ensure the changes occur.

I am used to creating websites from scratch, with no previous code and nothing preventing or blocking me in the way the hidden CMS code tends to do. While the day's goal was achieved, I feel I am not getting anywhere and am struggling to focus and keep the energy going for this project.

3.5.5 Friday 3 June 2022

Today I took a break from working on the HubSpot site as I was feeling very frustrated and disappointed and could not get the energy to work on the project today. Today was also retro day – which meant half the day was spent discussing what was achieved during the retro. Unfortunately, this made me feel even more despondent and frustrated because it felt that everyone else was making progress while I was stuck on the website and not really having anything to show for my months of work. However, part of retro was discussing the possibility of me working on other features in Hailer as I was now full-time over the summer. This was appealing, and I hope I get the chance to do so.

My self-appointed task for the day was to ensure all the document templates were updated and converted to the backend versioning. This meant going through the over 40 templates currently in production. We also did not have a git repository for these converted templates, which I chose to create today. After creating the branch, I asked my senior developer to add all the templates he had converted to the branch. Most of the ones he converted are templates that have not needed changes since we started the conversion.

The easiest way to ensure all the converted templates are the latest is to remove the converted template and redo the conversion based on the production code. Therefore, I needed to go into the app.hailer frontend and copy each of the files of the document templates and convert them using the instructions set out by my senior developer.

As I am the only one working on templates, I know what templates changed and which did not. Therefore, I only focussed on the templates that had been changed in the last month. There were around eight templates that I did this for; even if I thought the conversion had already occurred, I wanted to do the updates anyway, just in case. Having spent my day doing this semi-mindless task, I felt better about working on the website next week.

3.5.6 Weekly analysis

This week was frustrating for me. Firstly, the theme settings were not pulling through, which meant I spent the entire week figuring out how to use the theme settings and then having to ensure that they worked. Secondly, the button fixes, which another developer so easily fixed, was a massive blow to my confidence.

While I did get quite a bit done this week, I do not feel I am on track in any way whatsoever. I also am struggling very much with imposter syndrome. Imposter syndrome is when you feel you are an imposter and do not belong in the job/career you are currently in (Raypole, 2021; Cuncic, 2022). Many developers struggle with this, but more so junior developers when they are first starting. While I know I can code, and I know I can do the work – I truly feel that I am not doing enough or being productive enough in my job at current.

Imposter syndrome affects up to 70% of people (Psychology Today Contributors, no date; Owens, 2021; Tulshyan and Burey, 2021; Cuncic, 2022; Wilding, 2022) and is thought to be more apparent in high-achieving women (Clance and Imes, 1978; Raypole, 2021). The original study into imposter syndrome was initially conducted on women only (Clance and Imes, 1978), and therefore it was considered an issue that primarily women face (Raypole, 2021; Tulshyan and Burey, 2021; Cuncic, 2022).

The primary issue with imposter syndrome is that it can lead to depression, guilt and anxiety (Psychology Today Contributors, no date; Cuncic, 2022). By consistently doubting yourself and your abilities, you find reasons you are not as good as everything thinks, and the vicious cycle continues. Therefore, it is essential to know the characteristics or symptoms of imposter syndrome so you can combat these feelings.

The main characteristics include the following:

- Overachieving (Owens, 2021; Raypole, 2021; Cuncic, 2022)
- Setting unrealistic goals and being disappointed when you are unable to reach them (Owens, 2021; Cuncic, 2022)
- Agonising over the smallest detail (Owens, 2021; Cuncic, 2022; Wilding, 2022)
- Perfectionism (Owens, 2021; Raypole, 2021; Cuncic, 2022)

- Saying other or external factors are the reasons you are successful (Owens, 2021; Cuncic, 2022)
- Inability to honestly assess your abilities (Cuncic, 2022)
- Pervasive self-doubt over past, current and future experiences (Owens, 2021)
- Despite your success, you are constantly worried about being outed as a fraud (Owens, 2021; Raypole, 2021; Cuncic, 2022)

These characteristics are by no means the only ones people face when they experience imposter syndrome, but it is the generalised list of what most people will experience. Imposter syndrome is also tied into various anxiety disorders, such as social anxiety or general anxiety syndrome. (Psychology Today Contributors, no date; Owens, 2021; Raypole, 2021; Cuncic, 2022)

According to various studies, there are five main types of imposter syndrome (Cuncic, 2022; Wilding, 2022). Each type has slightly different emotional reactions, but the underlying feeling is the same: you feel that you are not worthy and experience self-doubt (Raypole, 2021; Cuncic, 2022). Each type also has more specific ways of dealing with the feeling of being an imposter ranging from accepting your work is never going to be perfect or meet your unsuitably high standards to interacting more with your co-workers (either through asking for help, becoming a mentor or mentee) (Wilding, 2022).

Overall, the best way to combat this feeling of self-doubt and fraud is to question yourself, reframe your thinking and ask for help (Owens, 2021; Raypole, 2021; Cuncic, 2022; Wilding, 2022). Embracing your success and listing your achievements is one way of reframing your thinking. In my case, I am still trying to find the best way for me to overcome these feelings. While I discuss it with my friends and some of my colleagues, they disagree with my self-doubt and thoughts of not being enough – it is tough for me to rationalise this and accept it. I think I will end up needing to work on this for a very long time, but it is helpful to know that I am not alone in this feeling.

3.6 Observation week 6

3.6.1 Monday 6 June 2022

This week started a new sprint, and my focus for the sprint is still the website and document templates. I am frustrated because everyone else seems to be doing much more exciting work, and I also do not have anyone who can help me with the HubSpot syntax and various issues that point up while I am coding.

Working in a team where you can rely on and ask for help from other developers is a huge bonus, and working alone in HubSpot has driven that home for me. I always knew that it is better to work in a team, but I never experienced development in any other form than in a team (apart from boot camps, but that is a different situation).

Today I also met with the CMO and a marketing agency specialising in HubSpot development. This agency contacted our CMO to employ their services in content creation, website development and marketing automation. The CMO immediately thought of me and first met with the agency to get information on what services they supply regarding website design and development. She meant that she wanted to find some support for me and my questions; however, the agency seemed slightly misunderstood and assumed she wanted them to take over the website development from me. She told them that I am new to HubSpot development, so I think they assumed that she was looking to replace me with someone more experienced.

Luckily, the CMO is quite happy with my progress and work and wanted to find ways to support and help me while I figure out how to work in HubSpot. The agency, unfortunately, was not hired, although we had an hour-long meeting. There were other issues, but the main reason we eventually declined is that they do not give me the support and help that the CMO was looking for. The plan from the agency was to take over the development entirely. The CMO does explicitly not want that scenario again, as a contractor created the current website, and she is unable to do a lot of the things which I am now enabling for her in the content editor, such as customising the colours, fonts, layouts and adding more/less features.

Above that, having a contractor complete the work I have already completed would be pretty frustrating and disappointing for me. I have not only been learning a lot during the HubSpot development, but the website is slowly taking shape and looking nice, and I want to see it through. After the meeting, I had to complete some document template changes to an existing template. The template was throwing strange errors wherein there was data, but it came out as [Object Object].

This is likely due to a change in the input field on the Hailer side, and I was correct. The field was changed from being an array to being an object array. Therefore, I had to destructure the array in the template into a string. The string could then be used in the template. I also had to check that the same field was not used in other templates and would require the same fix as in this template. After testing that the templates worked in production, the template with the issue was the only one needing the fix. I then pushed the template changes to production (through my senior developer, as I do not have production privileges) and converted the changed template into the backend version.

My plan for the day had been to work on the template fix and then work on the website – but unfortunately, sprint planning, the meeting and the following discussion with the CMO took more time than I realised, and I was not able to complete the planned tasks for the day.

3.6.2 Tuesday 7 June 2022

Non-working day

3.6.3 Wednesday 8 June 2022

Today I plan to focus on the website again, particularly block 2 – features (Appendix 1). My plan for the features block involves using a grid and repeating cards, which the content manager can add/remove as necessary.

The grid layout will have to be dynamic because I am unsure how many cards there will be. In the Design Tools field tab, it is possible to add a repeat option to a grouped set of fields or a single field. I plan to create a grouped card including a heading, text and an image. This card group will be made repeatable with a minimum of four cards.

Along with the cards, there will be a CTA block with a heading, CTA and optional text box. I based the grid layout on the design of block two, which can be seen in Appendix 1. Block 2 has a basic grid of 3 x 2 (columns x rows). However, two columns are larger than the others, so I opted to create a grid pattern of 7x2. I initially tried the design with a grid of 3x2, with the larger columns having larger fractal ratios (fr) than the others. However, this layout made the other cards look thin and less neat than the design. I then experimented with 5x2 and found the other cards were larger and less squashed but still relatively thin.

Therefore, I did the 7x2 layout, and it worked perfectly. I set the grid positions of each card to ensure the cards went where I wanted them to by adding grid-areas to the card classes. I then created an if statement in the module, allowing the content creator to move the cards to different grid positions if they wanted to. This optional step had to be toggled on using a Boolean selector. If

the content creator wanted a specific card larger than the others, the Boolean selector had to be checked, and the card positions changed accordingly.

I then created media queries which changed the grid pattern to 2x5 (2 columns x 5 rows) to mirror the design on tablets (800px and smaller screens). The CTA block takes up the two columns in the first row, and the cards take up one column each, except for the last one, which takes up the entire row. Again, I set the grid-areas for each card to ensure that this layout was set. The final media query mirrored the mobile design, with 1 column and multiple rows. I did not specify the number of rows, as this was unnecessary. If the row amount is not set, the grid implies multiple rows and will allow as many rows as necessary to display the contents.

Most of the styling was not complete, and only the layout of the grids was completed today – but the if statement took a long time to get right. I finished most of the day's goal, although my goal was to complete the entire features block – the main layout is now complete, and I can work on the specific styles and fix any bugs on Friday.

3.6.4 Thursday 9 June 2022

Non-working day.

3.6.5 Friday 10 June 2022

Today I continued with the features block. My goal for the day is to complete it and have it added to the home page template by the end of the day. I finished the grid layout on Wednesday, so I only need to add the styling today. The styling of these cards includes a background image, border, various spacing(s) and grid gaps. I first focused on ensuring all the information necessary for the features block was inserted into the card repeater, as I had used a basic setup on Wednesday to create the grid layout.

The CTA block was mainly complete; however, I added more options to the CTA button, including more colour customisations (colour change on hover) and link options that I thought might be required in this block. The card repeater was also mainly done; however, I edited the default wording in the heading and text to include more content, so I can ensure the padding and margins in the cards are correct and will work for however much content the content manager wants to add. I added multiple spacing style fields in the style tab and some default values, which added spacing to the entire block wrapper, the CTA block, the CTA button, and the card wrappers. This block needs to be more customisable in the spacing and styles because it will be reused throughout the website on different pages for different content.

I also added the option to add background colours to the entire block wrapper and CTA block by adding a Boolean selector, which, if toggled on, gave the content manager a colour choice. The cards are designed to have a background image, so I did not add this option.

The background in the cards was the first time I used the background image style field, and it required a lot of documentation reading. In HubSpot, you can add .css to various style fields, which HubSpot will then parse into usable CSS code when compiled (HubSpot, 2022b). However, not all style fields do this – and luckily, the background image field does. Therefore, I needed to add the .css to the snippet, and the background image worked.

We chose to use SVGs as background images to speed up the site load; however, when exporting the background images from Figma – I ran into an interesting issue when exporting them as SVGs. The problem was that when I exported it as a PNG, the image was blurred and filtered correctly. However, when I exported it as an SVG – the blur and filter disappeared, which meant the SVG was now made up of 3 different coloured balls instead of the blurred gradient that the UI/UX designer had created.

I tried multiple ways of solving this issue, including adding a filter blur using CSS in the module (which unfortunately blurred the entire card) but eventually messaged the UI/UX designer who exported the file as an SVG. She did the export precisely the way I did – but she managed to get it to export correctly. We joined a Google Meet, and I showed her what I did, and she showed me, but we got two completely different SVGs. We are unsure why this happened, but it does not matter, as she downloaded the SVG for me, which solved the issue.

Finally, the features block was complete. I tested it on the different device sizes and found no issues, so I added it to the home page template. I then previewed the ‘live’ site to check that the block worked – and there were no issues I could see. Therefore, I considered my work complete for the day and was happy with my progress.

3.6.6 Weekly analysis

This week involved a lot of HubSpot development and a lot of back and forth with the designer because of changes necessary due to the HubSpot CMS. Overall, I completed a decent portion of the website and am finally making decent progress. However, I am struggling in general with HubSpot development firstly because I have never developed or coded anything in a content management system (CMS). Secondly, because the HubL documentation is not as easy to understand or use, in my opinion, I must go searching for how to use various syntaxes as well as how to get information from the built-in functions and datasets that HubSpot has available (such as

page information). Lastly, and probably my biggest frustration in HubSpot, is the development environment.

I am used to Visual Studio Code (VS Code), Visual Studio, and Atom integrated development environments (IDEs), and most notably, I have a thing about correctly aligned and neat code. Unfortunately, Design Tools does not have a code formatter – so any formatting must be done by hand. It also does not have code completion. In VS Code, there is a built-in extension called Emmet which both suggests the completion of tags and uses abbreviations to create tags with classes, ids and other properties (Visual Studio Code Documentation, 2022). HubSpot does not have such extensions and cannot add extensions to Design Tools.

It could be argued that I rely too much on code completion extensions, which is likely true. It could also be argued that the code format and layout do not matter as much – which I disagree with wholeheartedly. Correctly formatted code means you can better read the code, find errors and make further adjustments (Martin, 2016). It also means that when someone else takes over from you – they will be able to read and understand your code with little struggle (Martin, 2016).

HubSpot development can occur locally in your IDE, and before starting this thesis, I did attempt to set this up. The setup is relatively simple – you must download (pull) the theme from HubSpot to your local machine using the command line. Once on your machine, you can make changes and then upload (push) the changes to the theme.

However, I quickly ran into issues with this approach. Firstly, creating new modules, templates or partials is impossible locally. You must create them in Design Tools and then pull the changed theme to your local machine. I thought I could get around this by implementing a GitHub pipeline where changes would automatically be pulled/pushed as necessary. However, this pipeline needed much more work than I was willing to do.

Another issue I had with the local development was the syntax. VS Code read the HubL syntax as incorrect and kept telling me there were errors in my code. Luckily, HubSpot has a VS Code extension that supports HubL syntaxes and snippets (HubSpot, 2021).

For me, the major flaw in the local development, and the reason I elected instead to use the Design Tools, is that the file structure in local development is ridiculously complex. The general CSS, HTML and JavaScript structure is similar to Angular. Still, the fields connected to the modules are in JSON, and there is not much explanation on the documentation pages regarding local development using the fields in JSON. After struggling to figure out what to use, when and how – I gave in and used the Design Tools.

In Design Tools, adding fields and style fields in the inputs tab is easier. These fields and styles are then accessible to content managers, who can change content and styling. The fields and styles are specific to a module, which is then used in a template or template partial. The templates/partials are then used as web page templates (HubSpot, no date). These web page templates are the end products of HubSpot development but are completely customisable in the content editor. Content managers can remove, add or edit modules on any web page. Content managers cannot create new modules in the content editor; that is only possible in the Design tools (HubSpot, no date).

Creating a new website in HubSpot generally means creating a new website theme comprised of templates, partials and modules. Theme fields are set in theme settings, a global file that overrides any other styles used within the theme. These settings are set in the Design Tools theme overrides file and can be edited further by content managers if changes are required.

Modules include module fields (inputs) and CSS and JavaScript functions relative to the module. Each module should be coded as a reusable but customisable component (HubSpot, no date). There are two separate module fields – style fields and input fields. Style fields are found under the styles tab in the module, while input fields are termed ‘fields’, and I am simply calling them input fields for clarity. Input fields include text inputs, images and icons. Style fields include colours, background images, gradients, spacing and font styles. It is also possible to create loops within modules, such as making one specific field grouping that can be repeatedly added to a module (HubSpot, no date). Choice and Boolean selectors are other handy features; these selectors allow you to hide/show fields in the content editor and add specific elements in the module depending on the choice/Boolean (HubSpot, 2022d).

Overall, HubSpot development is an interesting sphere. While I enjoy learning how to use and implement the HubL syntax – I am not a fan of it and will likely not pursue being a HubSpot developer, which the CMO suggested.

3.7 Observation week 7

3.7.1 Monday 13 June 2022

During daily planning today, our development operations (DevOps) developer asked for help from anyone in the development team in refactoring the current logging system. We currently have a very basic error logging system, and the DevOps developer has been updating it with more versions and information. He now needs help going through the entire backend, removing the old logger, and replacing it with the new one. As I needed a break from coding in HubSpot due to last week being so frustrating, I offered to help him. He still has some setup and will write up a manual on the various logging levels and error message information that I should add. So, I will likely work on the refactoring later in the week.

My goal for the day is to work on block 3 of the website – the statistics block (Appendix 1). Firstly, I created the basic layout and used a 4x3 grid layout. However, this ended up not working with the bottom row as it is designed to have only three cards and be a carousal on smaller screens. Therefore, I redid the layout in a 1x3 structure for the wrapper. The wrapper also has a background image, which I set as customisable for the content manager, and the outer spacing.

I then created a repeatable card option for the statistics themselves but made a Boolean selector for the content manager to decide if they want to add more statistics. We only plan to have four blocks of statistics, and it is unlikely we will need to add more which is why I added this Boolean. The cards are made of 2 text area fields (heading and content), and I want the heading to always be in line with each other, no matter the size/about of text in the heading. I also want the content text to always start in line with each other. To achieve this, I created two grid layouts. The first one is for all four cards, 4x1, to ensure the cards are always in line. The second grid is for the card itself, with a 1x2 layout. I set the top row height (the heading row) using min/max rem values. The minimum was set as though only one line would be added to the heading as in the design (1rem), and the max is double that. The content row was done using auto because the actual height of this row does not matter – only the start points. Adding min/max to the grid layout ensured that the headings would always start at the same place and content.

The tablet and mobile versions of these statistics cards were simple changes, with the grid columns/rows of the wrapper changing to 2x2 (Appendix 2). The inner grid layout did not need to change, and using rems instead of pixels ensured that the heights scaled with the screen sizes. I added spacing and a borderline to the styles field for the content manager to change as they see fit. The second row in the statistics block is a simple text area for the block's heading. It takes up

the entire space and only requires spacing for the content manager to change (apart from the text itself).

The final row was a bit trickier. It is a simple block or grid layout on the desktop and then transitions into a carousal on the tablet and below. I used Slick slider again, as I had already figured out how to use it for the logo carousel. I created a wrapper for the block (termed advantages from hereon), and within the wrapper added repeatable cards consisting of a number, heading and content. At first, I created a text input field for the number; however, if the content manager opts to move the cards around – it would be necessary to change the numbers constantly.

Therefore, I chose to dynamically set the numbers in the for loop. According to the design, numbers below 10 have a 0 in front of them. Initially, I added a '0' before the number and expected it to work. In JavaScript, if you add a string (i.e., a variable in quotes) to a number, the output is a string. However, this is not the case in HubL, and the outcome was simply an error.

I knew what needed to be done – the number variable needed to be converted from a number type to a string type. The number variable itself had to remain a number for me to be able to increment it depending on how many cards are added – so I could not just change the variable itself to a string.

I needed to find how to convert variables to strings in HubL. In JavaScript, there are multiple ways to convert variables (MDN Contributors, 2022e). It is possible to set a variable to a different type (var number = 23; number = '23'; the number is now a string) or using toString() (e.g., var number = 23, var numberAsString = number.toString()) (MDN Contributors, 2022e). I tried using toString(), and it did not work. Therefore, I read the HubSpot Developers CMS reference documentation and found the correct conversion type – '|string' (HubSpot, 2022d). In HubL '|' denotes a function or filter (HubSpot, 2022d, 2022b). Using this type conversion, I created a new variable (advantage_number) and used this in the output design.

Once I had completed the content for the slider, I needed to set the breakpoints and ensure the slider was working correctly. Unfortunately, if the outer wrapper is in a grid format – the slider does not work correctly in smaller screen sizes. The slider requires the wrapper to be in block format, so I changed the display property from the grid to block on smaller screens.

Apart from that minor issue, this block was relatively easy to do. I attribute that to it being a simple block in general and the fact that I am becoming more familiar with the Design Tools and generally developing in HubSpot. The goal for the day was to complete this block, and for once, I could complete the block fully in one day.

3.7.2 Tuesday 14 June 2022

Today the goal was to focus on refactoring the logging in Hailer. The DevOps developer had written a ‘How to’ for me to use, and the refactoring itself is relatively simple. Unfortunately, I cannot use the refactoring option in VS code simply because the new logger version allows us to add more information, so I need to add this information manually. I also cannot find and replace all the old logger codes.

Therefore, I searched the entirety of the Hailer backend for the version 2 (v2) logger and went through each file. Each v2 logger was replaced with the version 3 (v3) logger, and more information, including messages on the file, function, feature version and, if possible, the session information, was included in the v3 message.

This is a repetitive but fun task. I enjoy refactoring, as it allows me to go through the entire backend and discover how the backend runs and how the data is obtained and used. I am not very experienced in backend development, so this was an excellent start to learning how to work with the Hailer backend. I worked overtime today because I was so engrossed in my refactoring that I did not realise the day was over. However, I completed the refactoring and sent the branch to be reviewed by the DevOps developer. I hope to hear from him tomorrow or next week.

3.7.3 Wednesday 15 June 2022

Tomorrow I am going on a short holiday, and therefore do not want to do anything drastic to the website, so I planned to do bug fixes and other minor improvements. However, I noticed a CTA field while exploring the different field types in the Design Tools. I added the CTA to a block for testing and realised that the CTA tracked clicks, allowed the content managers and marketing to determine where a specific contact interacted with the website and provided various other information and statistics on the button itself.

Therefore, I decided to change all the blocks already created with basic buttons and links using this CTA field. Creating CTAs in HubSpot is done in the marketing section, and while making the CTA, you choose the CTA name, link(s), and styles, and you can create smart versions that change depending on various meta-data obtained by HubSpot from a user (HubSpot, 2022a). Specifically interesting for Hailer is the possibility of changing the CTA text according to preferred languages. I created one primary CTA but realised quickly that the set styles for the CTAs in HubSpot cannot be edited, and the only way to edit the styles is to add styles under the advanced section. This means that every time any salesperson or anyone in the marketing team wants to create a new CTA, either I need to create it or they need to figure out how to copy the code I use.

According to the HubSpot community forum, it is suggested to duplicate one CTA button and change the text/links (Köhler, 2021). While this is a solution, it requires all the salespeople and the marketing team to remember to duplicate and not create their own buttons. After a discussion with the CMO, we elected to go a different route simply because she expects each salesperson to create and operate their own CTAs. If they duplicate existing buttons or code, there is too much room for errors.

Therefore, I decided to use the no-style version of the CTA and style it in the website theme. Unfortunately, adding a class to this CTA button is still necessary. Still, I created a Confluence documentation write-up for the sales team to ensure they knew about the CTA changes. I made a global CTA class in the main CSS file, which will be added to all the CTA buttons. This default class ensured the background colours, text colours, padding, text alignment, text transformations, and other styles would inherit the settings from the container div where I placed the CTA button.

Once this default class was created, I added a container div and the CTA field in each module, which currently had a button and link acting as a CTA (hero block and features block). I then added style options for the content manager to edit, such as colours, hover colours, text transformations, spacing, and alignment. These styles were applied to the container div, and because the CTA button has the class that sets it to inherit everything – the CTA button was then styled to match the theme.

I also created a login and sign-up CTA buttons but used specific classes to ensure they remained distinct and different from the general CTA buttons. The login and sign-up buttons would gather information when a user interacts with them but would link the user to the Hailer application or the sign-up form.

For each of these new CTAs, I created two smart versions in Finnish and Swedish. We only have content in Finnish, but we plan to make the website more international and accessible in multiple languages. I wanted to use these two smart versions as testers to see how they worked.

To see the smart content in action, I needed to create multi-language versions of the ‘live’ site in Finnish and Swedish. This is simple in HubSpot, as I select the page and select – ‘Create multi-language variation’. HubSpot will then duplicate the page but will not alter the content. The content needs to be changed manually to the other languages. The smart CTA buttons change automatically when you change the page language, which was the goal.

I had planned not to do much today but ended up working on optimising the analytics of the Hailer website. The CMO was quite happy to be able to use the CTA buttons more effectively. Currently, the CTA buttons do not match the website's theme and are rarely used because of this. Using the

CTA buttons means the marketing team can effectively track and monitor users and user interactions, as well as potentially obtain new leads and create more targeted marketing campaigns based on the statistics obtained.

The DevOps developer was still going through my changes, so I did not do anything regarding the logger today and will likely only do fixes or changes next week.

3.7.4 Thursday 16 June 2022

Nonworking day

3.7.5 Friday 17 June 2022

Nonworking day

3.7.6 Weekly analysis

This week was very short due to my holiday, but I feel I achieved quite a bit. The statistics block is completed, and I redid the CTA buttons throughout the website. These CTA buttons will make lead tracking and contact management easier for the marketing and sales teams, so this is a significant improvement to the website. It is frustrating that it is impossible to style the buttons dynamically without needing to do the 'hacky' fix I made. This fix works, but it is frustrating because the buttons must be set in the various modules and cannot be set in the CTA button menu itself. Ideally, the buttons would just take on the theme settings, but it, unfortunately, does not.

I did learn a lot about logging, though, which was interesting. As I said previously, I do not have much experience with logging and the backend, so this project was very interesting. Logging is vital in ensuring regulatory compliance, increasing security and, in general, knowing what is going on in your software (Chuvakin, Schmidt and Phillips, 2013). According to Chuvakin, Schmidt and Phillips (2013), logs are often unappreciated but are a vital source of information on the software's resources, information and security. The term logging applies to collecting information from the system while functions are being completed or run in the system (Chuvakin, Schmidt and Phillips, 2013). This collection of information is logged or recorded and known as the log data. Logs are a collection of log messages which contain the log data (Chuvakin, Schmidt and Phillips, 2013). There are multiple forms of log messages, from informational to debugging (Chuvakin, Schmidt and Phillips, 2013).

In Hailer v2 logging, we had simple log messages logged by the logging library – Winston. Winston is a JavaScript logging library that, according to the documentation: 'aims to decouple parts of the logging process to make it more flexible and extensible' (Robbins, 2022). I am unsure exactly why we use Winston because I am not involved in that side of the development team. By the time I

became involved in this logging task, the decisions had already been made, and I was brought on to refactor the codebase to match the new logger version. However, Winston is one of the most popular logging libraries (Chege, 2020; Manandhar, 2021; Ulili, 2022) and is known for its flexibility and malleability, so this is likely why it was chosen. Winston allows us to have multiple storage devices (transports) for the logs. These transports include the AWS console, private databases and files and the console (Robbins, 2022; Ulili, 2022).

Winston also allows for multiple logging levels (Robbins, 2022). Logging levels are the priority or type of log; the higher the number, the higher the priority (i.e. 1 is a critical error). These priority levels are set numbers according to the Syslog Protocol (Gerhards, 2009), which states the protocols relating to various system logging standards. There are seven standard logging levels described in this protocol (Gerhards, 2009), and Winston has implemented these levels to a degree (Robbins, 2022).

According to the Winston documentation (Robbins, 2022), there are seven logging levels:

0. Error (critical error/system is unusable)
1. Warn (warning of errors)
2. Info (information such as ‘user logged on’)
3. Http
4. Verbose
5. Debug (debugging information, only available when in debug mode)
6. Silly

Our v2 logging system had two versions of log messages; error and debug. This meant that anything that was not a debug message was recorded as an error message. Therefore, we had copious amounts of error logs that the DevOps engineer had to search through to find true errors because often, these errors were information logs instead of error logs. In v3 logging, the DevOps engineer has created four different logging levels; error, warn, info and debug. These levels were what he deemed critical or valuable information.

While working on the refactoring, I chose to do one file and then asked the DevOps engineer to check that I was doing the refactoring correctly. Thankfully, I did this because I had misunderstood the levels he had created and used error (level 0 / critical) as the main level throughout the file. I misunderstood that level 0 was critical errors only and that level 2 (warn) was the majority of other errors. So I had to redo this file, and then I asked him to recheck it. While waiting for him to check the file changes, I did a quick reading on level logging and what it meant, which made it easier for me to determine which old error messages would be on which level. The development team also asked me not to remove any of the log messages currently being used, as most were funny

anecdotes or had some nostalgic meaning to various team members. I made sure to keep these messages and just added more information to them.

Before starting this task, I should have read through the Winston documentation to understand exactly what I was doing before just diving in, but this is a recurrent issue of mine. I tend to dive in and do it instead of reading the documentation or doing tutorials. The documentation for Winston is very well written, with multiple examples, and every part is explained. I would have easily understood what was happening if I had read through it before starting.

3.8 Observation week 8

3.8.1 Monday 20 June 2022

Non-working day

3.8.2 Tuesday 21 June 2022

Non-working day

3.8.3 Wednesday 22 June 2022

Today was my first day after my holiday, and I had turned off any notifications from Hailer while I was away. Therefore, I was flooded with messages from the last few days. My goal for the day was to catch up on what happened while I was away and take care of any urgent tasks.

After reading through the multitude of messages and activity discussions for the morning, I started working on template changes for a customer. This customer has a very complex template for invoicing, with multiple linked activities, data that is transformed and changed in the template itself and, in general, a very confusing layout of tables with the data being input in bizarre ways.

This template was created at the beginning of the template generator implementation; therefore, it is one of the earliest templates in Hailer. How pdfmake utilised the data and created documents were only partially understood by those making the templates at the time, so there is also a lot of hardcoded data in these templates. I often work with these specific clients' templates, as they request changes around once a month.

This change is significant in adding more columns to the complex structure. Currently, the template has 12 columns with three sub-columns in 5 of the 12. This means the table is entirely using the space of the A4 document, although there are standard page margins on either side of the table. The change requested by the client is to add another sub-column to the five columns.

This is a significant change because the table is already so big; it means a complete change in the page margins, paddings and font sizes of the columns, as well as changing the dimensions of the columns themselves. Currently, the table's column sizes are hard coded to ensure that everything fits in the document with enough space to read all the information.

To make the requested changes, I first needed to add the data links from the clients' workflow and then change the column widths and potentially all the sizing and font sizes to accommodate the extra sub-columns. I set up the basic plan for this before the workday ended and will likely spend the entirety of tomorrow working on this change.

3.8.4 Thursday 23 June 2022

Today is retro day, and the next sprint will be exceptionally long as most of the other developers will be on holiday for most of the following month. This afternoon we will discuss what we have completed, and we are currently developing multiple new features for the DigiOne project. I will be one of the only developers available for the summer without long breaks. Therefore, I need to pay attention to this retro to ensure I know what is happening so I can work on these projects.

Before the retro, however, my goal is to get further along in the template changes I began yesterday. While I got the data pulled into the template – the template cannot be printed currently because the table overflows the page margins and throws errors. My first go-to fix was to remove the page margins altogether and see if this at least allowed me to see the table (even though I was fully aware the widths would not work at all).

This fix worked, and I could see the mess I had managed to create in the template. Removing the margins meant that the header, footer and other tables (which are not as large) and any other content or information were now pushed right to the edge of the document. The large table was also squashed and illegible in various columns, and the totals at the bottom of the table (which is an entirely different element from the table itself) were all over the page instead of in line with the relevant column.

However, I could at least see how much space I could use for the table, of which there was not a lot. This meant the table would need to have reduced font sizes to allow more of the columns to be readable. I decided to start with fixing the layout of the other contents before working on the table simply because they are faster fixes.

For pdfmake, you can set margins in different places. I had removed the page margins in the page styles area, but I could add the same margin to the content to make the document look the same as it did previously (although the table still looks terrible). This was a long process due to the number of different content components in this template. While I was doing these margin changes, I also decided to refactor the template a bit to make it more readable and easier to change in future.

The main changes I made were removing repeated styles and creating a global style class. pdfmake allows you to either set styles per line or per component, and these styles can be set the same as a style tag in HTML (in-line styles) or as a CSS class. Essentially, I created a CSS class and removed all the in-line styles. This took a long time because the in-line styles were not written in the same order (e.g., font size, colour, spacing/padding, colour, font size), which meant I could not simply ‘find all and replace’. I searched for a specific colour or property and then had to select

the inline style and replace it with the CSS class. I only got halfway done with these changes before retro started, so I will need to continue these changes next week.

3.8.5 Friday 24 June 2022

Non-working day

3.8.6 Weekly analysis

This week was filled with more refactoring but of a different kind. Last week I refactored the logger, and now I have refactored a document template. This document template has been bothering me for a while because the overall layout and how it has been coded are extraordinary and makes no sense. Whenever I have been working on it previously, I have slowly but surely been making the layout and general coding more readable.

The readability of code is an essential part of being a developer. If your code is eligible to your peers or those who come after you, it is impossible to use and implement your code. I have always been a stickler for these guidelines and standards, which is why this template has frustrated me for quite a while. I have read various blogs and books about clean code and how ensuring your code is clean, DRY, and easy to read is imperative for well-functioning code. In particular, *Clean Code* by Robert Martin (2016) is one of the main contributors to how I code.

The general takeaway rules that I always keep in mind from *Clean Code* (Martin, 2016) include the following:

- Keep it simple stupid (KISS), meaning reduce complexity (Tilburg Science Hub, no date; webpro, no date).
- Be consistent in naming and layouts.
- Choose descriptive and unambiguous names.
- Use code comments to add more information, don’t leave others in the dark
- Keep things that are linked close to each other. Declare your variables close to where you will use them. Similar and dependent functions should be close to each other. Sequential functions should be sequential.

Code standards are the rules that govern how a company codes (Bost, no date; Hanson, 2019; Sayan Kumar Pal, 2022), and they are vital in ensuring that everyone’s code is the same. Consistency throughout a company ensures that there is less bloat and that when a new person is brought on to the team, it is easy for them to read the code as well as add their code in a sensible and understandable fashion (Hanson, 2019; Sayan Kumar Pal, 2022). Code standards also mean that when a piece of code is not read for a while, it is still understandable for whoever wrote it and is now reading it (Hanson, 2019; Sayan Kumar Pal, 2022).

In Hailer, we do not have a very detailed guidelines of code standards; therefore, we have some differences in how everyone codes. However, for the most part, it does not affect anyone too much as we use ESLint, which formats important files for us. ESLint is a linter, essentially a tool that enforces various rules on your code (ESLint Contributors, no date). Linters help reduce the time taken to format code and ensure that all your code looks the same (Fayock, 2019). Linters also help you to find syntax errors, provide suggestions for fixes or best practice principles and can help you keep track of code comments (Fayock, 2019). Most linters rely on a set of rules, which the company can customise, or a list of rules that the linter has already pre-programmed. Not all linters fully analyse and format your code, depending on which one you choose.

ESLint analyses the code in a file after you tell the IDE to format the file (ESLint Contributors, no date). ESLint then either auto-formats the code or informs you that there are errors using the standard red underscore. Our ESLint rules mainly focus on the TypeScript files and SCSS files in the frontend and the JavaScript files in the backend. Our HTML files do not have many set rules, and this is a feature I intend to add later on.

There are multiple linting libraries that can be used, but the main ones include ESLint, JSLint and Prettier. All three are found as extensions in VS Code; while ESLint is automatically built into VS Code (Microsoft Contributors, 2022), JSLint and Prettier must be added. Prettier and JSLint rules can be edited, but it is not necessary to create rules as they have set rules that are employed until they are changed. ESLint, however, requires more setup. To use ESLint, it is necessary to initiate the library and choose specific rules and guidelines (ESLint Contributors, no date; Microsoft Contributors, 2022). After which, an ESLint file is created in your project directory. This ESLint file can then be used to add more rules or edit current ones (ESLint Contributors, no date). To enable linting in VS Code, navigate to the linting settings under the Tool Options menu (Microsoft Contributors, 2022). In this menu, you can choose which files are to be linted. After enabling ESLint and selecting the files – the initiation of the library begins with a series of questions and then downloads the necessary dependencies (Microsoft Contributors, 2022).

In my opinion, linting is not taught very often in programming courses, and when you head out into the programming world, you are expected to understand what linting is and why it is important. Linting is not only helpful in making your code look nice but also for reducing the number of bugs and discrepancies that creep in when coding large features. I think learning what linting is, how to create new rules, and what the linter is telling you is vital in ensuring your code is practical, efficient and readable.

3.9 Observation week 9

3.9.1 Monday 27 June 2022

Today was sprint planning, and this sprint will be longer than usual due to the summer break. Most of the developers will be away during the summer, so only a few of us will be around for the next month and a half. Therefore, this sprint is five weeks instead of the usual 3. During the sprint planning, we discussed who would be taking over various tasks from the developers going away over the summer. One of the features we plan on releasing during or after summer must be styled to suit the designs the UI/UX designer created. The developer in charge of this feature is going away for 1.5 months; therefore, to meet the planned release date, someone needs to take over the styling of this feature.

I offered to do this feature styling because I have a lot of experience in styling from various courses, and I really enjoy UI/UX work. I am also one of the only Hailer developers to remain during the summer. The remaining people working this summer are the contractors, who have their tasks.

This sprint planning took most of the morning because we needed to discuss what the contractors needed to do and how things would work over the summer. After sprint planning, I met with my senior developer to discuss the newly released backend template generator. We discussed how to test the templates, and he told me I had to double-check check all the templates were still working in production. This meant I had to either inform the sales team and the project team that they needed to check their various clients' documents to ensure they were working or go through each client's workspace and check the templates myself. As I could not access all the clients' workspaces, I opted to check the ones I did have access to and message the sales/project team member in charge of the other client projects to check their documents.

Luckily, there were no issues, so I could continue working on the template changes from last week. These specific template changes are pretty complex, so I did not expect to finish them today. However, I added the additional columns and information with little issue and needed to further adjust margins and text sizes for the table to fit on the page. This surprised me because I planned to work on these changes for the next day or two. However, I was pleased that I completed it so quickly.

This template change was also the first time I had used the backend document template generator, so I did the updates and double-checked all the changes before making them live on production. There were no bugs or errors, so I happily removed this from my to-do list.

3.9.2 Tuesday 28 June 2022

Today my goal was to complete another block on the Hailer website, and the chosen block for the day was the ‘Get Hailer’ block. This block is the final CTA block on the main page and is found just above the footer (Appendix 1). This block is the last thing a customer would see on the website and needs to be eye-catching and make them want to click on the download buttons. The design is very nice, in my opinion, but it remains to be seen how complex it will be to implement.

I started this block by doing my usual layout using divs and slowly adding the necessary content. The image used in Appendix 1 is that of Hailer on a phone and is actually a full PNG of a phone and not just half the phone, as depicted in Appendix 1. To move the image downwards, as in the design, I needed to use absolute positioning and translate/move the image down. Five values can be used for the position property; static, relative, fixed, absolute and sticky (MDN Contributors, 2022g). A statically positioned element is the default value and means the element is positioned according to the document's structure (Coyier, 2008; MDN Contributors, 2022g). A relatively positioned element can either be translated relative to its static position or used as the relative base for another element (Coyier, 2008; MDN Contributors, 2022g). Absolute positioning an element means the element is set precisely where you want it to go – relative either to its parent element (if the parent element is relatively positioned) or relative to the page (Coyier, 2008; MDN Contributors, 2022g). A fixed element is essentially absolutely positioned according to the viewport (i.e., the screen) (Coyier, 2008; MDN Contributors, 2022g), while a sticky element behaves like a static element until you scroll past it (Coyier, 2008; MDN Contributors, 2022g). The sticky element will remain in the viewport until you completely scroll past the parent (relative) element (Coyier, 2008; MDN Contributors, 2022g).

At first, I used flexbox to set the block layout and the image position as absolute relative to the parent wrapper block. This worked fine for the desktop screen size. However, when I moved on to do the media queries, I quickly realised I would need to add copious amounts of media queries to get the block to remain in the correct position(s). I then switched to using a grid layout of 2 columns, one for the image and one for the content. Using the grid layout allowed me to change the size of the image according to the screen size and the horizontal or vertical positioning of the container div without needing to change the actual positioning of the image.

I also added the usual layout options for the content managers so they can design where the image will be. I also made it possible for the background to be changed as necessary. This block will likely be edited for various campaigns, with different content, CTA buttons and images, so it needs to work with varying images and amounts of content. Having completed this block, I felt my

work for the day was successfully completed and felt very accomplished at having completed this block so quickly.

3.9.3 Wednesday 29 June 2022

When I started work today, I was asked by one of the other (intermediate) developers if I could please take over the styling of his feature (the feature I mentioned I might be taking over in the sprint planning). Initially, another developer was going to work on this styling, but he became unavailable, so the leading developer asked me to work on it. So I could switch focus and take a break from the website to work on this styling.

The first thing I did with this styling was get to terms with what was happening in the feature. I started my frontend using the leading developers' branch and realised I could not run any of the functions. Of course, this was a new feature, so there was a new backend for it as well. So I switched to the correct backend and restarted my frontend. The feature then worked, and I immediately saw issues with the styling. Specifically, there was weird spacing between the fields and inputs, which drew my eye immediately. I started writing a list of things I noticed, and then when I was about to ask the UI/UX designer about the designs, I realised she had already created a list of tasks that needed to be fixed. So, I had just done some unnecessary work, but it meant I had at least seen how the feature worked.

From the Figma designs shared by the UI/UX designer, I could see that the feature had a lot of styling work that needed to be done. I started by simply reading the HTML file of the feature and figuring out which component of the feature was coded where and how. This is a new feature, so any changes I make to these components will not affect the rest of Hailer, so I did not need to worry about that. I did need to ensure that any class names I used were not the same as other global classes, so I immediately went to the main SCSS file and found an absolute mess.

Unfortunately, it seems most of the developers in Hailer have not worked with SCSS nor understand the power or abilities of SCSS. SCSS is a CSS superset (SASS-LANG, no date), which makes coding CSS easier (in my opinion). It allows you to nest classes and elements and create functions which set the styles dynamically (SASS-LANG, no date). The goal of using SCSS is to have a set of global files that include all the styles for the product, with variables that are set in a variables file or at the start of the specific SCSS file. These variables make it easier to make whole-scale changes to an application. For example, you can set a border in one file and use the variable name in any file that needs that specific border. Then if the design ever changes (colour, size etc.), you change it in the one file, and it updates automatically throughout the entire product.

When looking at the SCSS files in Hailer, I found that no variables were used, even when the same numbers or styles were used throughout the application, styles were duplicated across multiple files and components, and there was almost no global styling. This both shocked and frustrated me, as I really love SCSS and how easy it makes styling, and I assumed that other developers had enough experience that they would know about SCSS and how it works, and how to use it effectively and reduce the time it takes to style components.

I spent the entire day reading through the various components and files and spent a long-time creating a list of changes that needed to be made. I will need to redo a lot of the already completed work of the feature because the current HTML will not work with the designs.

3.9.4 Thursday 30 June 2022

Today I continued going through the code of this new feature and making the list of changes which need to be done. I started by making notes in the code itself to see what belonged where and did what. I also made notes to myself about moving repeating HTML code into components or Angular templates. I want to be able to start changing the styling on at least one of these components today, but I am still not quite sure how Angular works, so I will also need to spend most of the day figuring out Angular and how to code in Angular and TypeScript.

After a while, I decided to dive in, as I tend to do, and just start with changing the styling. I was annoyed with trying to figure out what was happening and felt it would be easier if I just changed things and saw what happened. I picked a simple number input field and followed the component to its base component. This base component was a simple number input field, and I did not need to make any changes. Therefore, I moved ‘up’ the hierarchy until I found an incorrectly styled component according to the Figma designs. I found the component on my local frontend and made changes using the developer tools. Once I had figured out what needed to be changed using the developer tools, I made the style changes in the components’ SCSS file.

However, I again noticed the strangeness of the SCSS file and how there were duplicated styles, so I decided that tomorrow, I would create a global SCSS file only for this feature and remove all the current styles and start from the beginning.

3.9.5 Friday 1 July 2022

As I decided yesterday, I started refactoring the styling in the entire component. I rearranged the folder structure of the SCSS to better match the 7-1 folder architecture with which I am familiar. The 7-1 folder structure consists of seven folders and one main.scss file (Epicodus, no date). The folders contain multiple files, while the main.scss file consists of import functions from the other folders’ files (Epicodus, no date). Some of this folder structure was already set up but was not

being utilised. I added a few more folders and added files under them. The current main.scss file has many styles coded in there when it is only supposed to contain import functions. Therefore, I moved most of these styles into the newly created folders and files. As I was refactoring, I checked that my changes were not impacting the local Hailer frontend I was running. Occasionally I had to move a styling set back into the main.scss file because it was not pulling through from the folder file. This took the entirety of my day, but I utterly enjoyed it. I really love styling, and seeing how my changes reduce the number of lines of code is just pure joy for me.

Today was also the last day for the leading developer on this feature before his summer break, so he checked if I needed any help. I said no, although I did need some help understanding what was going on in Angular; I was too shy or scared or embarrassed to say I had no clue what was going on with Angular and TypeScript. I feel my imposter syndrome kicked in again, and it definitely made me feel like I would be a fraud or everyone would become aware that I do not actually know anything. So instead of asking for help, I foolishly said I was okay. The developer went on holiday, and I was then left to my own devices for the rest of the summer.

3.9.6 Weekly analysis

This week was primarily spent working on the Hailer frontend. It was quite a switch for me to move from HubL and the website to such complex features and components. This was my first time genuinely working in Angular and TypeScript, so I had some issues adapting and understanding what was happening. While I have made some progress and will continue making progress next week, this week was slow going. I first had to understand what the leading developer had done in the HTML and TypeScript files, which meant I firstly needed to understand what was going on in Angular.

My primary learning curve in Hailer has been using Angular, the development platform used in the Hailer frontend. Before Hailer, I mainly worked in node.js and React and did a short 3-month internship in ASP.NET. At first, I assumed Angular would be no issue because it would be the same as React. I quickly learnt that this assumption was wrong. Firstly, in Angular, it is far more complex to create components. From my experience, creating React components involves simply creating a file component and coding the functions for the component. I expected Angular to be similar.

Angular components are (like React) building blocks of an application (Angular Contributors, 2022) and are comprised of three files:

- Styling (CSS or SCSS in Hailer’s case)
- TypeScript class
- HTML template

The TypeScript class file has an Angular-specific Component decorator comprised of a selector (how the component is called), and an HTML template instructs Angular how to render the template (Angular Contributors, 2022). Apart from the Component decorator, the TypeScript file also has a class which determines the component's behaviour (the functions it performs). The HTML template file is like the HTML of any web application with standard HTML tags and properties. However, there are additional syntax elements in the HTML template file, such as ngIf, ngClass and ngFor. These additional elements were a significant learning curve for me, as I had never encountered them before.

Property bindings allow you to set values for HTML element properties and attributes (Angular Contributors, 2022) and are set using square brackets. These can be dynamically created in the HTML template file or the TypeScript class file or set in the HTML template. Event listeners are declared in parenthesis – which surprised me as I have never had to do this. The first few times I created event listeners in Angular, I did not add the parenthesis and was incredibly confused about why the event was not triggering. Angular directives such as ngIf and ngFor dynamically modify the HTML structure (Angular Contributors, 2022). I am used to using if statements and for loops within the component's logic instead of the render structure, but I have used some functions such as map and filter in React logic, so this was not a significant change for me.

Another difference is that the functions and dynamic values within the template are rendered and compiled in the TypeScript class file (Angular Contributors, 2022). This differs from React, wherein the functions are run inside the component file.

While all these differences are hard to master, the Angular documentation is clear and concise. I have hugely benefitted from reading the documentation and undergoing their free tutorials. However, I should have done the tutorials before working extensively on the Angular frontend, as the learning curve was steep without the documentation and tutorials. This was a mistake, but it taught me always to check the tutorials and documentation of new languages or frameworks before diving in and coding in them. Before working in Angular, I could ‘dive in’ to React, node.js and JavaScript because I followed web developer boot camps, essentially the same as reading the documentation and watching the tutorial videos.

When I began working in HubSpot, I immediately watched all the tutorial videos, read most of the documentation, and created a copy of the default website theme from HubSpot. I think doing this before starting to work in HubSpot was a significant help and allowed me to get to work a lot faster than if I had followed my previous method of simply ‘diving in’.

3.10 Observation week 10

3.10.1 Monday 4 July 2022

Today I continued refactoring the new feature. I am still reworking all the styles, and while this is likely not what the leading developer wanted me to do when he handed his project over, I feel that reworking these styles will help ensure that the styling is consistent. I have already noticed that the typography is working correctly throughout the entire local site. Previously the typography was not set globally; it was set in each component's SCSS file, which meant that if the developer had not set the typography – it would simply not be there.

I added a base SCSS file which normalised various styling properties such as the line-height, global typography (simply setting the font-family) and moved the box-sizing property into this file. Box-sizing specifies how the total width and height of an element are calculated. If you do not set the value for box-sizing, an element's computed width and height do not include any border, padding or margins (MDN Contributors, 2022a). Setting the box-sizing value to the content-box is the same as the default (or no value). Setting it to border-box means that any padding, margin or border additions to the element are added to its calculated width and height. This is a typical CSS styling trick to make it easier to set the widths and heights of an element. (MDN Contributors, 2022a)

After removing many styles from the main.scss file, I got a bit tired of doing this refactoring and felt guilty I was doing this instead of working on the feature styling as I was supposed to be doing. I went back to the feature and started by removing all the current styles used in the various components' SCSS files. Once I had achieved a 'blank slate', I added a separate component file in the SCSS folder architecture and started fixing the layout of the feature.

By the end of the day, I had only managed to do the general structure fixes – particularly the flex layouts or grid layouts of the major components. I still needed to do a lot of work on the smaller structure changes and move various HTML elements around to achieve the looks of the designs.

3.10.2 Tuesday 5 July 2022

The day's goals centred around continuing the refactoring, but I decided to go into the deep end and move the HTML elements that needed to be adjusted. This will take a lot of work and will likely result in many bugs and errors because I need to move entire components around and remove and add various elements to said components.

My chosen victim for my refactoring today is the field elements. A field element comprises an input element, label, tooltip (if present), unit (if present), and a function field button. When the field is

being edited, it must also include an ‘x’ that will remove all the information from the field. The field element had the correct elements within it but had the tooltip in the wrong place or the function field in the wrong place. I then moved these elements to their correct positions. One of the tasks that the UI/UX designer had created was to align all of these elements correctly. They are all haphazardly aligned at the moment, based on the various elements within the field element itself. This meant I needed to create a set grid layout which would force these elements to be in alignment, even if the other elements were not present.

This grid layout took me a long time to figure out because only some of the field elements had the same number of columns necessary, and not all elements would always be showing, but if they were not showing, the grid layout got messed around a bit. Eventually, I settled on a series of different grid layouts used only when the element fulfilled certain conditions. To achieve this, I used a mix of SCSS mixins and Angular ngClass and ngIf. I spent the entire day simply getting the field element to align perfectly, no matter how many elements it had within it.

3.10.3 Wednesday 6 July 2022

After my success yesterday, where I changed one of the field elements layouts, I intended to change the rest. In Hailer, we have various types of input fields, from text, text-area, number, and number with units to dropdowns with set values. These inputs must be refactored and styled to work with this new feature. The issue with some of these inputs is that other components use them, so I must be careful whenever I make changes to the root component and extensively test it.

Yesterday I changed the text field element, so today, I decided to work on the text-area element. This was the same as the text field, except it can extend vertically to allow for more text. I used the same grid layout as the text field, so this was a relatively simple refactor.

Then I moved on to the number field. We have two different number formats in Hailer, with and without units. I tried the text field grid layout; surprisingly, it worked very well with the number field. However, I had yet to check the Figma design file, and after going there, I quickly realised that I had been doing this entirely wrong. The designs depicted the number field(s) as includes/between/excludes type filters. The number fields have three radio button options which filter the outputs. I had created a text filter wherein the number was always an includes filter. So I had to go back to the HTML and completely rewrite this entire field. I also had to create a custom radio button because the option from the Mat-UI library used in Hailer did not match what the designer wanted.

I spent the entire day reworking this field, and near the end of the day managed to get at least the layout correct. However, the functionality needs to be fixed. It will need backend fixes, and I am not

confident enough to work on the backend at present. Besides, I would prefer that the leading developer do it because he understands exactly what is going on there, and I worry that I will further break the feature if I mess around with the backend.

I created a new development task for the leading developer when he returns and will use it to list all the elements I break when I rework them, so he will know which elements' functionality needs to be fixed.

3.10.4 Thursday 7 July 2022

Today I plan to continue rewriting the number fields. Yesterday the styling was mostly done, but there are a few tweaks I still need to make. I specifically need to change the colours used in this feature. Therefore, I need to add these colours to the Hailer palette and use the variable name in the SCSS file.

While doing these colour changes, I realised that the colour palette files were also messy. We have three themes in Hailer – light (default), dark and Dracula. It seems that most developers have only been setting new colour variables in the light theme and not adding them to the dark or Dracula themes. I decided to go through each variable, add the missing ones to the Dracula and dark themes, find the equivalent colour code for the theme, and fix it.

It was strange once I had done these fixes because I suddenly saw how much better the dark and Dracula themes were looking, and I wondered how I had not noticed the slightly off colours beforehand. For example, the hover colour on the discussions was not added in the dark and Dracula modes, so it used the light blue variable from the light mode, which was very strange when you have a dark grey discussion and when you hover over it, it suddenly goes light blue.

I then continued adding the new or additional colours to all the themes and ensured that all the design colours were now in the Hailer palettes. This was a small and simple change, but it took me the entire day to sort it out due to the missing colour variables in the themes. I also moved the variables around so that the same variable was set in the same place in each theme palette. This was to make it easier when updating or changing colours in future.

3.10.5 Friday 8 July 2022

Today I decided to take on a bit of a tedious task that was bothering me – the icons. The icons in the Hailer design system are all custom, but for some reason, the icons in the Hailer application range from custom icons to Mat-UI icons. This is (in my opinion) a jarring change because the custom and Mat-UI icons are not in the same style.

To change the icons, I had to download them from Figma and add them to the Hailer application. I was unsure how to do this, so I chose an existing custom icon and searched for all mentions of this custom icon. This quickly pulled up the icon model file and was easy enough for me to understand, so I just added the icons that were supposed to be used. I then went through the feature, systematically removed the Mat-UI icons, and replaced them with custom icons. After doing this, I checked my local frontend and noticed that the colours of the icons were all black. They were not supposed to be black but to follow the colours of the text they were next to. I dug around in the developer tools in Chrome to try and figure out where the colour was being set. Eventually, I discovered that the icons (which are in SVG format) had a default colour set. So, I went into each icon and removed the default colour. Then checked the frontend and saw that none of the icons were showing. After more developer tool digging, I realised I needed to set the colour property to currentColor, making the SVG inherit the colour set for the div or the sibling text. I again went and made the changes in all the icons, and it worked.

The icons in this feature were now all correct and looking much better and in line with the design system, look and feel of the Hailer application. This issue was fun but frustrating to fix because it took me a long time to figure out what needed to be fixed, and then I needed to change each icon individually.

Today is the last day of my thesis observation weeks, and I am slightly sad that I did not get to do more of the Angular and TypeScript work I will continue doing as the summer goes on. I enjoy fixing, refactoring, making the new feature polished, and following the designs. I enjoy seeing that my change are slowly but surely aligning the feature with the designs and the overall Hailer design system.

3.10.6 Weekly analysis

this week involved a lot of styling work in Angular and TypeScript, and I am genuinely enjoying myself for the first time in a long time. The website is an interesting project, but it is not fun and tends to be more frustrating than anything else. This task is pure fun for me, however. Seeing the fixes and changes as I make them, refactoring and genuinely reducing the load time of the styles and making it pretty really sparked my joy for coding once again.

While working on these changes, I kept various common programming principles in mind. I have previously mentioned KISS, but multiple other principles are key to ensuring you are efficient and produce quality code. I find that not many young developers are aware of these, and I have been lucky enough to know many other developers who told me about these principles, which is why I know about them and can implement them in my code.

One of my favourite principles is ‘Write programs for people, not computers’ (Tilburg Science Hub, no date). Otherwise known as code for the maintainer (webpro, no date). While code is created to be used by computers, you must not forget that the next person to read your code is a person – not a computer (Tilburg Science Hub, no date). Computers will understand what you are telling them to do if the syntax is correct. People will not know what you are trying to do unless you explain it. To ensure that you do not have to repeat yourself constantly and that whoever works on your code after you needs to write your code so that other people can read and understand it. This includes adding comments to explain what a function does, using easily understood and unambiguous naming, and keeping in mind the principle of least astonishment (POLA) (Tilburg Science Hub, no date; webpro, no date). POLA means that the component should not astonish users (Raymond, 2003), including the readers, testers or maintainers of your code.

The rule of 3 is an easy way to know when you should refactor and create a function or component. This rule states that if you write a function or component 3 or more times, you should create a reusable function or component instead of repeating yourself. (Tilburg Science Hub, no date)

Continuing with reducing repeating code is the DRY principle. DRY stands for Don’t Repeat Yourself (Tilburg Science Hub, no date; webpro, no date). DRY is probably one of the principles I hear about the most, but in general, people forget that it is just one of the many principles that make you a better programmer. DRY follows the rule of 3 but also states that business rules, metadata, lengthy statements etc., should only be coded and found in one place and should instead be called or used as a function in the application (webpro, no date; Grant, 2021). DRY is essential in reducing maintenance issues due to duplication and ensures that when changes occur – the changes only need to happen in one place.

This principle is why I was so frustrated and surprised by the state of the SCSS files. There were countless duplications in the files, with some SCSS component files being completely the same apart from class names. This is against all the coding conventions I know and will likely be my pet project to fix in the future.

Another important principle that I tend to ignore, in some instances, is that of YAGNI. YAGNI stands for You Aren’t Gonna Need It; essentially, this principle states that you only implement things when needed. Meaning only implementing something that the feature needs and ignoring what you think the feature will need in the future (Tilburg Science Hub, no date; webpro, no date). I tend to ignore YAGNI when styling because I will often add class names before I need them. However, while doing the refactoring work, I realised I needed to unlearn this. While doing various styling changes, I searched for class names and ended up not finding various ones, which meant I

spent time searching for these names that were not even used. If I apply POLA and writing for people, not computers – I see that ignoring YAGNI goes against these two principles. Therefore, I will be working on using YAGNI to prevent those who come after me from being frustrated at my excessive class name usage. YAGNI is also more applicable when designing a feature for a customer because you should never guess what the customer wants. That is how issues with requirements and goals occur, which usually ends badly. In that sense, I apply YAGNI to the document templates, where I do not add more features than they ask for.

These principles are important and should be kept in mind when coding, but they are not rules. They are guidelines that should shape the way you code and become ingrained in the way you code. I do my best to implement these, but I sometimes forget. This is part of the coding life, and as long as you are willing to admit your mistakes and fix the issues – you will do fine. Also, remember that your code is not yours, and you should not become emotionally attached to it. You are creating a product or feature and simply a cog in the machine. Instead, be happy with the output than the mechanisms.

One saying that I will never forget, which is probably my coding motto is by Martin Golding: ‘Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live’. I try my best to live by this saying and ensure that any code I create, refactor or work with is easy to read, simple to understand and includes the necessary information to ensure the violent psychopath will not be coming after me.

4 Discussion

At the start of this thesis, I was a novice HubSpot developer, barely able to create simple modules such as buttons or links. As this thesis progressed, I became more accustomed to developing in HubSpot, learned the syntax, and have slowly but surely become a decent HubSpot developer. While I will likely never focus on HubSpot development as a career, simply because I do not enjoy working in the syntax and IDE, developing the website with Hailer has been incredibly good for my self-esteem and knowledge growth.

Firstly, I learnt how to develop in HubSpot, but more than that – I was given a project all to myself, and I was the only one in charge. The CMO checked in and ensured we were on track – but overall, the website was purely my project. As a young developer, this massive self-esteem boost has made me more confident in my coding abilities.

Coupled with website development, I have developed a strong relationship with the UI/UX designer. We work well together and happily spend hours discussing (or obsessing, as the other developers call it) various styling changes and fixes we plan to implement in the Hailer app. At the writing of this thesis (November 2022), I oversee refactoring the entire Hailer application UI/UX and have been put in charge of the UI/UX design team – which did not exist until I joined Hailer. Upon releasing the new feature that I worked on, the entire team was happy with the design, functionality, and the small number of bugs the feature had.

According to the product owner, this release was the ‘best’ in Hailer’s history in terms of a smooth and bugless release. Therefore, I am proud and happy to have found my niche in the development world. I genuinely enjoy the styling and UI/UX aspects of frontend development, and I will continue to focus on this aspect of development. I will also focus on the backend development because I want to understand it and how it works, but I have realised where my talents and expertise lie.

After this thesis, I will continue developing in Hailer and will likely be the de-facto UI/UX engineer. However, I will also work with the backend team and improve my skills in all things backend related. The backend has always been my weakness, and I have never felt comfortable working with databases and data. Still, I am confident that the Hailer team will support my learning journey, as they have during this thesis and year.

This thesis observation lasted from 25 April 2022 to 9 July 2022. However, the submission was delayed due to various factors. The main contributing factor to the lateness is based on my work and university schedules. Unfortunately, working and studying took a higher toll than I realised, and I slowly but surely became burnt out.

Burning out is a common stress/anxiety reaction. It is not talked about often, but in my experience, I often hear the symptoms when speaking to other developers, particularly young developers trying to earn their place in the development world. Due to this burnout, the toll on my physical and emotional state led me to focus purely on working. I had to focus on one thing or the other, and unfortunately, working was more important.

Following my period of burnout (July 2022-September 2022), I got awful influenza which meant I was sick and bedridden for two weeks. I then worked on this thesis and caught up on my other studies. In October 2022, I contracted coronavirus 2019 (Covid19), which further delayed my submission and re-writing of this thesis.

However, while this has been a remarkably stressful period for me, writing this thesis has been enlightening. It enabled me to see how far I have come in only a few months, starting from not knowing anything about HubSpot or Angular – to having a fully functioning HubSpot website and being employed as the UI/UX engineer. I also learnt a lot about myself, and I need to know to say no. I have always had issues with this and ended up overloaded and unable to complete all the tasks I said I would. I have already begun saying no and instead delegating my tasks at work. I have found that this has massively decreased my stress and enabled me to focus on finishing my university work and this thesis.

Comparing my knowledge and experience to March 2022, I am far more capable and confident in HubSpot and Angular. I never had an issue with SCSS or CSS, but Angular was surprisingly hard to understand and learn. Likely because I did not do the tutorials, which is another lesson I have learnt. I need to do the tutorials or read through the ‘Getting Started’ tutorials.

While I will not be a HubSpot developer any time soon, I am happy to work as the Hailer HubSpot developer for the foreseeable future. The website was launched in November 2022, and I am proud of the flow, layout, and overall look.

There were some trying times during this thesis, but I am better off now, six months later. I am a better developer, and this thesis allowed me to truly see how I have changed and developed as a developer and person. I am confident in my abilities and can lead the UI/UX team in the future in updating the current UI/UX of the application itself.

References

Angular Contributors (2022) *What is Angular?*, *Angular Docs*. Available at: https://angular.io/guide/what-is-angular#what-is-angular (Accessed: 14 November 2022).

ASQ (no date) *Quality Assurance & Quality Control*, *ASQ Blog*. Available at: https://asq.org/quality-resources/quality-assurance-vs-control (Accessed: 8 December 2022).

Association for Project Management (no date) *What are agile methods/what are agile methodologies?*, *Association for Project Management*. Available at: https://www.apm.org.uk/resources/find-a-resource/agile-project-management/agile-methods/ (Accessed: 13 November 2022).

Association for Project management (no date) *What is agile project management?*, *Association for Project Management*. Available at: https://www.apm.org.uk/resources/find-a-resource/agile-project-management/ (Accessed: 13 November 2022).

Aston, B. (2017) *9 of the Most Popular Project Management Methodologies Made Simple*, *The Digital Project Manager*. Available at: https://thedigitalprojectmanager.com/projects/pm-methodology/project-management-methodologies-made-simple/ (Accessed: 13 November 2022).

Aston, B. (no date) *A Guide To Agile Project Management Methodology & Tools*, *The Digital Project Manager*. Available at: https://thedigitalprojectmanager.com/projects/pm-methodology/agile-project-management/#what-is-agile-approach (Accessed: 13 November 2022).

Beck, K. *et al.* (2001) *The Agile Manifesto*, *Agile Manifesto*. Available at: https://agilemanifesto.org/ (Accessed: 13 November 2022).

Bloomenthal, A. (2022) *What is the C Suite?: Meaning and Positions Defined*, *Investopedia*. Available at: https://www.investopedia.com/terms/c/c-suite.asp (Accessed: 8 December 2022).

Bost, K. (no date) *Demystifying Coding Guidelines: Intro to Coding Conventions*, *IntelliTect*. Available at: https://intellitect.com/blog/demystifying-coding-guidelines/ (Accessed: 11 December 2022).

Braccialini, C. (2022) *Why You Need a Responsive Web Design and How to Do It [+ Examples]*, *HubSpot Blog*. Available at: https://blog.hubspot.com/marketing/responsive-design-list (Accessed: 15 November 2022).

Chege, J. (2020) *Logging with Winston and Node.js*, *Section.io*. Available at: https://www.section.io/engineering-education/logging-with-winston/ (Accessed: 11 December 2022).

Chen, J. (2022) *Business-to-Business (B2B): What It Is and How It's Used*, *Investopedia*. Available at: https://www.investopedia.com/terms/b/btob.asp (Accessed: 8 December 2022).

Churchville, F. (2021) *User interface (UI)*, *TechTarget*. Available at: https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI#:~:text=The%20user%20interface%20(UI)%20is,an%20application%20or%20a%20website. (Accessed: 8 December 2022).

Chuvakin, A., Schmidt, K. and Phillips, C. (2013) *Logging and Log Management*. Edited by P. Moulder. Syngress.

Clance, P.R. and Imes, S.A. (1978) 'The imposter phenomenon in high achieving women: Dynamics and therapeutic intervention.', *Psychotherapy: Theory, Research & Practice*, 15, pp. 241–247. Available at: https://doi.org/10.1037/h0086006.

Coyier, C. (2008) *Absolute, Relative, Fixed Positioning: How Do They Differ?*, *CSS-Tricks*. Available at: https://css-tricks.com/absolute-relative-fixed-positioining-how-do-they-differ/ (Accessed: 11 December 2022).

Cuncic, A. (2022) *What Is Imposter Syndrome?*, *VeryWell Mind*. Available at: https://www.verywellmind.com/imposter-syndrome-and-social-anxiety-disorder-4156469 (Accessed: 11 December 2022).

Decker, A. (2020) *How Responsive Web Design Works*, *HubSpot Blogs*. Available at: https://blog.hubspot.com/marketing/responsive-web-design (Accessed: 15 November 2022).

DigiOne Project (2022) *About DigiOne*. Available at: https://www.digione.fi/digione-eng/ (Accessed: 14 November 2022).

Epicodus (no date) *7-1 Sass Architecture*, *Learn how to program*. Available at: https://www.learnhowtoprogram.com/user-interfaces/building-layouts-preprocessors/7-1-sass-architecture (Accessed: 11 December 2022).

ESLint Contributors (no date) *Getting Started with ESLint*, *ESLint*. Available at: https://eslint.org/docs/latest/user-guide/getting-started (Accessed: 11 December 2022).

Fayock, C. (2019) *What is linting and how can it save you time?*, *FreeCodeCamp*. Available at: https://www.freecodecamp.org/news/what-is-linting-and-how-can-it-save-you-time/ (Accessed: 11 December 2022).

Ferguson, B. (2021) *Design approaches of Mobile-first, Desktop-first, and Responsive Web Design*, *Dev.to*. Available at: https://dev.to/bryanalphasquad/design-approaches-of-mobile-first-desktop-first-and-responsive-web-design-516i (Accessed: 15 November 2022).

Fitzgerald, A. (2022) *What Is a CMS and Why Should You Care?*, *HubSpot Blog*. Available at: https://blog.hubspot.com/blog/tabid/6307/bid/7969/what-is-a-cms-and-why-should-you-care.aspx (Accessed: 8 December 2022).

Garner, R. (2022) *What Is a Pull Request & Why Is It Important for Code Review?*, *LinearB*. Available at: https://linearb.io/blog/what-is-a-pull-request/ (Accessed: 8 December 2022).

Gerhards, R. (2009) 'The Syslog Protocol', *Network Working Group* [Preprint]. RFC Editor. Available at: https://www.rfc-editor.org/rfc/rfc5424#section-6 (Accessed: 11 December 2022).

Grant, A. (2021) *10 Basic Programming Principles Every Programmer Must Know*, *Make use of*. Available at: https://www.makeuseof.com/tag/basic-programming-principles/ (Accessed: 11 December 2022).

Hailer Oy (2022) *Hailer Features*. Available at: https://www.hailer.com/features (Accessed: 14 November 2022).

Hanson, G. (2019) *Why Are Coding Standards Important?*, *Dev.to*.

Hayes, A. (2022) *Chief Executive Officer (CEO): What They Do vs. Other Chief Roles*, *Investopedia*. Available at: https://www.investopedia.com/terms/c/ceo.asp#:~:text=The%20chief%20executive%20officer%20(CEO,public%20companies%2C%20improving%20share%20prices. (Accessed: 8 December 2022).

HubSpot (2021) *HubSpot VS Code Extension*, *VS Code Documentation*. Available at: https://marketplace.visualstudio.com/items?itemName=hubspot.hubl (Accessed: 15 November 2022).

HubSpot (2022a) *Create button calls-to-action (CTA)*, *HubSpot Knowledge Base*. Available at: https://knowledge.hubspot.com/ctas/create-calls-to-action-ctas (Accessed: 16 November 2022).

HubSpot (2022b) *HubL Syntax*, *HubSpot Documentation*. Available at: https://developers.hubspot.com/docs/cms/hubl (Accessed: 29 September 2022).

HubSpot (2022c) *HubSpot Design Manager*, *HubSpot Documentation*. Available at: https://developers.hubspot.com/docs/cms/developer-reference/design-manager (Accessed: 29 September 2022).

HubSpot (2022d) *HubSpot Developers: CMS Reference Docs*, *HubSpot Documentation*. Available at: https://developers.hubspot.com/docs/cms (Accessed: 29 September 2022).

HubSpot (no date) *HubSpot CMS Overview*, *HubSpot Developer Documentation*. Available at: https://developers.hubspot.com/docs/cms/key-concepts (Accessed: 15 November 2022).

Interaction Design Foundation (no date) *User Interface (UI) Design*, *Interaction Design Foundation*. Available at: https://www.interaction-design.org/literature/topics/ui-design (Accessed: 8 December 2022).

jQuery API Documentation (no date a) *offset*, *jQuery API Documentation*. Available at: https://api.jquery.com/offset/ (Accessed: 14 November 2022).

jQuery API Documentation (no date b) *toggleClass*, *jQuery Documentation*.

JSON.org (no date) *Introducing JSON*. Available at: https://www.json.org/json-en.html (Accessed: 8 December 2022).

Juviler, J. (2021) *CRM vs. CMS: How to Use Both for Your Online Business*, *HubSpot Blog*. Available at: https://blog.hubspot.com/website/cms-crm-integration (Accessed: 10 December 2022).

Kanbanize (no date) *What Is Kanban? Explained for Beginners.*, *Kanbanize Blog*. Available at: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban (Accessed: 13 November 2022).

Kenton, W. (2020) *What a Call to Action (CTA) Is and How It Works*, *Investopedia*. Available at: https://www.investopedia.com/terms/c/call-action-cta.asp (Accessed: 8 December 2022).

Kenton, W. (2022) *B2C: How Business-to-Consumer Sales Works, 5 Types and Examples*, *Investopedia*. Available at: https://www.investopedia.com/terms/b/btoc.asp (Accessed: 8 December 2022).

Köhler, K. (2021) *CTA - editing the default Styles*, *HubSpot Forums*.

Ladas, C. (2008) *Scrumban*, *The Agile Alliance*. Available at: https://www.agilealliance.org/scrumban/ (Accessed: 13 November 2022).

Leaning, B. (2022) *50 Call-to-Action Examples You Can't Help But Click*, *HubSpot Blog*. Available at: https://blog.hubspot.com/marketing/call-to-action-examples (Accessed: 14 November 2022).

Lien, T. (2021) *Responsive vs. Adaptive Design: What's the Difference?*, *Spring Board*. Available at: https://www.springboard.com/blog/design/responsive-vs-adaptive-design/ (Accessed: 15 November 2022).

Manandhar, G. (2021) *Comparing 5 Node.js Logging Libraries for You To Make the Optimal Choice*, *Better Programming*. Available at: https://betterprogramming.pub/node-js-logging-libraries-5789c379640b (Accessed: 11 December 2022).

Martin, R.C. (2016) *Clean Code*.

MDN Contributors (2022a) *Box-sizing*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing (Accessed: 11 December 2022).

MDN Contributors (2022b) *CRUD*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Glossary/CRUD (Accessed: 8 December 2022).

MDN Contributors (2022c) *CSS: Cascading Style Sheets*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/CSS (Accessed: 8 December 2022).

MDN Contributors (2022d) *HTML: HyperText Markup Language*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML (Accessed: 8 December 2022).

MDN Contributors (2022e) *JavaScript data types and data structures*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#type_coercion (Accessed: 16 November 2022).

MDN Contributors (2022f) *JSON*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON (Accessed: 8 December 2022).

MDN Contributors (2022g) *Position*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/CSS/position (Accessed: 11 December 2022).

MDN Contributors (2022h) *SVG: Scalable Vector Graphics*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/SVG (Accessed: 8 December 2022).

MDN Contributors (2022i) *What is a URL?*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL (Accessed: 8 December 2022).

MDN Contributors (2022j) *What is JavaScript?*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (Accessed: 14 November 2022).

MDN Contributors (2022k) *Window.pageYOffset*, *MDN web docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Window/pageYOffset (Accessed: 14 November 2022).

Microsoft Contributors (2022) *Linting JavaScript in Visual Studio*, *Microsoft Docs*. Available at: https://learn.microsoft.com/en-us/visualstudio/javascript/linting-javascript?view=vs-2022 (Accessed: 11 December 2022).

Owens, A. (2021) *What is Imposter Syndrome?*, *PsyCom*. Available at: https://www.psycom.net/imposter-syndrome (Accessed: 11 December 2022).

Pacewicz, J. (2021) *HubSpot's CMS and CRM: Why use both?*, *ClickRay*. Available at: https://marketing.clickray.eu/blog/cms-crm#:~:text=HubSpot%20offers%20the%20following%3A,help%20you%20close%20more%20deals. (Accessed: 10 December 2022).

Pang, A. (2021) *TypeScript vs. JavaScript*, *Geek Culture*. Available at: https://medium.com/geekculture/typescript-vs-javascript-e5af7ab5a331 (Accessed: 14 November 2022).

pdfmake (no date) *pdfmake*, *pdfmake Documentation*.

Pratt, M.K. and Torode, C. (2020) *Agile Manifesto*, *TechTarget*.

Product Plan (no date) *Scrumban*, *Product Plan Glossary*. Available at: https://www.productplan.com/glossary/scrumban/#:~:text=Scrumban%20is%20a%20project%20management,agile%2C%20efficient%2C%20and%20productive. (Accessed: 13 November 2022).

Psychology Today Contributors (no date) *Imposter Syndrome*, *Psychology Today*. Available at: https://www.psychologytoday.com/us/basics/imposter-syndrome (Accessed: 6 December 2022).

Raymond, E.S. (2003) *The Art of Unix Programming*. Available at: http://www.catb.org/~esr/writings/taoup/html/ch01s06.html (Accessed: 11 December 2022).

Raypole, C. (2021) *You're Not a Fraud. Here's How to Recognize and Overcome Imposter Syndrome*, *Healthline*. Available at: https://www.healthline.com/health/mental-health/imposter-syndrome (Accessed: 6 December 2022).

Robbins, C. (2022) *Winston*, *GitHub*. Available at: https://github.com/winstonjs/winston#logging (Accessed: 11 December 2022).

SASS-LANG (no date) *SASS Basics*. Available at: https://sass-lang.com/guide (Accessed: 8 December 2022).

Sayan Kumar Pal (2022) *Coding Standards and Guidelines*, *GeeksForGeeks*. Available at: https://www.geeksforgeeks.org/coding-standards-and-guidelines/ (Accessed: 11 December 2022).

Shubel, M. (2022) *What is TypeScript*, *The New Stack*. Available at: https://thenewstack.io/what-is-typescript/ (Accessed: 14 November 2022).

Sirk, C. (no date) *CRM vs CMS: The Definitive Guide*, *CRM.org*. Available at: https://crm.org/crmland/crm-vs-cms (Accessed: 10 December 2022).

Soegaard, M. (2020) *Adaptive vs. Responsive Design*, *Interactive Design*. Available at: https://www.interaction-design.org/literature/article/adaptive-vs-responsive-design (Accessed: 15 November 2022).

Sutherland, J. and Schwaber, K. (2020) *The 2020 Scrum GuideTM*, *Scrum Guides*. Available at: https://scrumguides.org/scrum-guide.html (Accessed: 13 November 2022).

Techopedia authors (no date) *Create, Retrieve, Update and Delete (CRUD) TABLE OF CONTENTS*, *Techopedia*. Available at: https://www.techopedia.com/definition/25949/create-retrieve-update-and-delete-crud (Accessed: 8 December 2022).

The Agile Alliance (no date a) *Kanban*, *The Agile Alliance*. Available at: https://www.agilealliance.org/glossary/kanban/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'kanban))~searchTerm~'~sort~false~sortDirection~'asc~page~1) (Accessed: 13 November 2022).

The Agile Alliance (no date b) *Scrum*, *The Agile Alliance*. Available at: https://www.agilealliance.org/glossary/scrum/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'scrum))~searchTerm~'~sort~false~sortDirection~'asc~page~1) (Accessed: 13 November 2022).

Tilburg Science Hub (no date) *Principles for Good Coding*, *Tilburg Science Hub Blog*. Available at: https://tilburgsciencehub.com/building-blocks/develop-your-research-skills/tips/principles-good-coding/ (Accessed: 11 December 2022).

Tulshyan, R. and Burey, J.-A. (2021) *Stop Telling Women They Have Imposter Syndrome*, *Harvard Business Review*. Available at: https://hbr.org/2021/02/stop-telling-women-they-have-imposter-syndrome (Accessed: 11 December 2022).

TypeScript Contributors (2022) *The TypeScript Handbook*, *TypeScript Docs*. Available at: https://www.typescriptlang.org/docs/handbook/intro.html (Accessed: 14 November 2022).

Ulili, S. (2022) *5 Best Node.js Logging Libraries*, *Highlight*. Available at: https://www.highlight.io/blog/nodejs-logging-libraries (Accessed: 11 December 2022).

User Testing authors (no date) *UI vs. UX*, *User Testing Blog*. Available at: https://www.usertesting.com/resources/topics/ui-vs-ux#:~:text=%E2%80%9CUser%20experience%20(UX)%20is,a%20company's%20products%20and%20services. (Accessed: 8 December 2022).

vickykumar7061 and annieahujaweb2020 (2022) *Difference between responsive design and adaptive design*, *GeeksForGeeks*. Available at: https://www.geeksforgeeks.org/difference-between-responsive-design-and-adaptive-design/ (Accessed: 15 November 2022).

Visual Studio Code Documentation (2022) *Emmet in Visual Studio Code*, *Visual Studio Code Documentation*. Available at: https://code.visualstudio.com/docs/editor/emmet (Accessed: 15 November 2022).

W3Schools (no date) *HTML Responsive Web Design*, *W3Schools*. Available at: https://www.w3schools.com/html/html_responsive.asp (Accessed: 15 November 2022).

webpro (no date) *Programming Principles*, *GitHub*. Available at: https://github.com/webpro/programming-principles (Accessed: 11 December 2022).

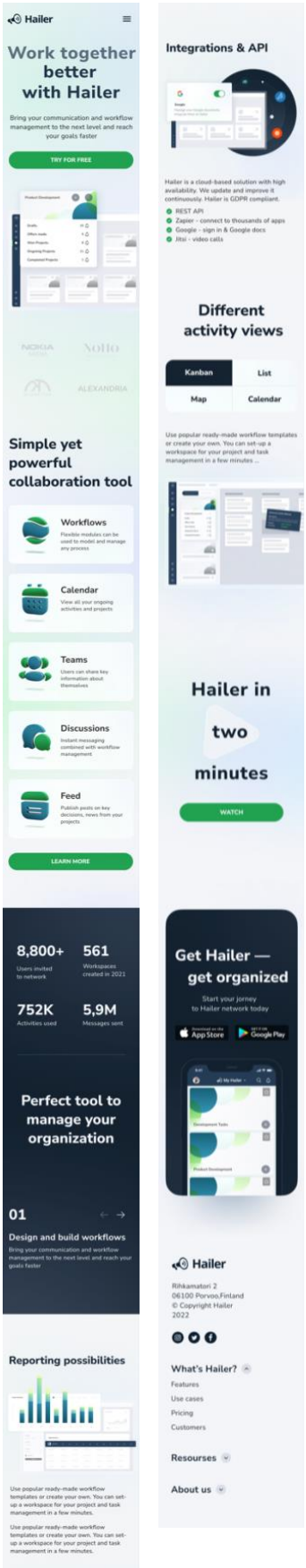
Wheeler, K. (no date) *Slick*, *GitHub*. Available at: https://kenwheeler.github.io/slick/ (Accessed: 14 November 2022).

Wilding, M.J. (2022) *5 Different Types of Imposter Syndrome (and 5 Ways to Battle Each One)*, *The Muse*. Available at: https://www.themuse.com/advice/5-different-types-of-imposter-syndrome-and-5-ways-to-battle-each-one (Accessed: 11 December 2022).

Appendices

Appendix 1 Hailer website design

Appendix 2 Hailer redesigned website in mobile version



Appendix 3 Hailer redesigned website in mobile - open menu version.