Bachelor's thesis

Information and Communications Technology

2022

Rami Alsabki

FRONT-END WEB DEVELOPMENT FOR SHOPIFY

– Using React, Polaris, and Other Modern Technologies

TURKU AMK
TURKU UNIVERSITY OF
APPLIED SCIENCES

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2022 | 48 pages

Rami Alsabki

FRONT-END WEB DEVELOPMENT FOR SHOPIFY

- Using React, Polaris, and Other Modern Technologies

Front-end web development has undergone major development since the inception of the web, changes that altered the user experience and the web development process. New technologies are continuously introduced, and many tech companies, such as Shopify, have taken steps to improve user experience and enhance the development process by introducing their own tools.

This thesis is targeted at coders and the general public interested in front-end web development in general and Shopify in specific. The thesis covers, in addition to theoretical background and review, the development process of the front end of a Shopify app (named ‘Flashify’), followed by a discussion of the results. Technologies used during implementation included React, Polaris, and Shopify CLI, while a Kanban board was utilized to manage workflow.

The objectives were to build the front end of the app ‘Flashify’ and cover the project in a thesis form, allowing utilization by readers to start a Shopify app development project or to enhance knowledge related to the topic. The app development objective was successfully implemented. The app was tested with mock data and it is ready for the next development stage which was planned to take place after the thesis is published.

Keywords:

Front End, React, Polaris, Shopify, e-Commerce, Web Development

Contents

Figures

Tables

List of Abbreviations

CERN	the European Organization for Nuclear Research (Abbreviation derived from the French name of the organization ‘*Conseil Européen pour la Recherche Nucléaire*’)
CLI	Command-line interface
CRO	Conversion Rate Optimization
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HQ	Headquarter
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JS	Javascript
MERN	MongoDB, Express, React, Node (Web development Stack)
NPM	Node Package Manager
OS	Operating System
TS	TypeScript (a programming language)
UI	User Interface
USD or $	United States Dollar (a currency)
UX	User Experience
VSCode	Visual Studio Code (an IDE)
W3 or WWW	the World Wide Web

1 INTRODUCTION

1.1 The Significance of Front-End Web Development

Over the years since the beginning of the internet era, web development has seen rapid changes. The phenomenon is distinctly visible to the user simply by comparing how websites (such as Google, Yahoo..etc) used to look and function a few years back and how they do now. Technologies that contribute to the modernization of web development are on the rise in terms of the development of existing ones and the creation of novel ones. At the general level, developers aim to build optimal User Experiences (UX). In terms of the front end, this happens by, for example, providing a smooth, modern, and easy-to-use User Interface (UI).

While many backend concepts are also significant in web development, front-end development plays a crucial role in building remarkable websites with outstanding UX. The front end of a website is what the end user sees and interacts with. It is all the functionalities that a user would use from what they see on a web page. Therefore, the front end must be developed with user needs in mind, or in other words, while utilizing user-centric concepts. Programming languages for front-end development include Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS), although there are also more sophisticated technologies such as angular.js, react.js, backbone.js, and others.

Front-end web development is a crucial area for developers and experts in the e-commerce field. According to a Cloudflare article, front-end concepts like page design, page layout, text, and images on the page play a significant role in conversion rate optimization (CRO) [1]. In the business context, a higher conversion rate conduces to higher profits. Due to that, one way to be exceptionally prominent as a developer in the current business world is to demonstrate a decent understanding of business concepts such as CRO. By utilizing the understanding of CRO and similar business concepts, a developer could be seen as an asset to the business as the developer in that case would be capable of approaching technical tasks and projects with user-centricity and a business-oriented mindset, which subsequently results in positive outcomes in terms of finances. [2]

1.2 Motives and Objectives Behind the Study

The writer of this thesis has been in the e-commerce field for a few years with experience in establishing e-commerce stores on Shopify and WooCommerce. Coming from a business background and being currently in the web development field, gaps in existing e-commerce solutions, when detected, tend to be utilized to conceptualize app ideas which are then analyzed from both business and IT standpoints.

There were several reasons behind the selection of the topic. From the technical aspect, although significant steps were taken by Shopify to enhance developers' experiences in developing for Shopify, the amount of support (including community support) and documentation is succinct when compared to competitors such as WooCommerce. When existing resources are examined, many were found to be outdated. The Shopify website currently offers satisfactory general support through technical materials and manuals, however, it is not sufficient for more specific issues, although community support can be obtained on an individual basis through community support websites such as StackOverflow. On the other hand, some relevant YouTube videos are also available but in relatively small quantities and are often outdated. With the amount of use, merchants, and transactions happening through Shopify, there is a lot of space for more support and materials.

From a business standpoint, Shopify app development is a career opportunity. Theme development is another significant opportunity due to the small number of official themes on the Shopify theme store. However, the thesis does not cover theme development at all. According to an article posted by TechCrunch, Shopify developers collectively earned 233 million USD in 2020. They are currently charged no fee for the first 1 million US dollars made by the developer [3]. In addition, Shopify reported hosting millions of online stores in over 175 countries and the number has been growing [4]. By developing for the Shopify ecosystem, developers could theoretically serve millions of Shopify merchants. Business-wise, it is a great opportunity to monetize app ideas.

Taking the above points into consideration, the thesis aims to provide the following outcome (by completion of the thesis and its project):

- A ready functional thesis that can be utilized by the target group and possibly other readers.
- A Shopify app (called ‘Flashify’) that adds features related to sales to the admin panel built on the front end using modern web technologies and ready for the next development stage (backend development).

The thesis does not aim to be targeted solely at web developers with previous experience in the field but also at individuals of the general public with an interest in front-end web development as a broad concept or Shopify in particular. As a result, basic concepts that are likely to be already known by individuals in the technology field may be covered in some chapters.

To achieve an insightful and satisfactory result relevant to one specific area of web development, the thesis focuses on the front-end development of Shopify apps without addressing the backend.

1.3 The Structure of the Thesis

This thesis covers front-end development as a general concept while focusing on front-end app development for Shopify. Over the years, a variety of tools have been introduced by Shopify as steps toward ensuring a continuously-improving user experience (UX) for the users of their platform. Tools introduced by Shopify included libraries and dependencies used in the development projects of Shopify apps. For example, the Shopify CLI execute many of the repetitive tasks automatically to provide a ready app template. Shopify has also introduced Polaris which offers Shopify Developers access to all the necessary components to build feature-rich themes or apps.

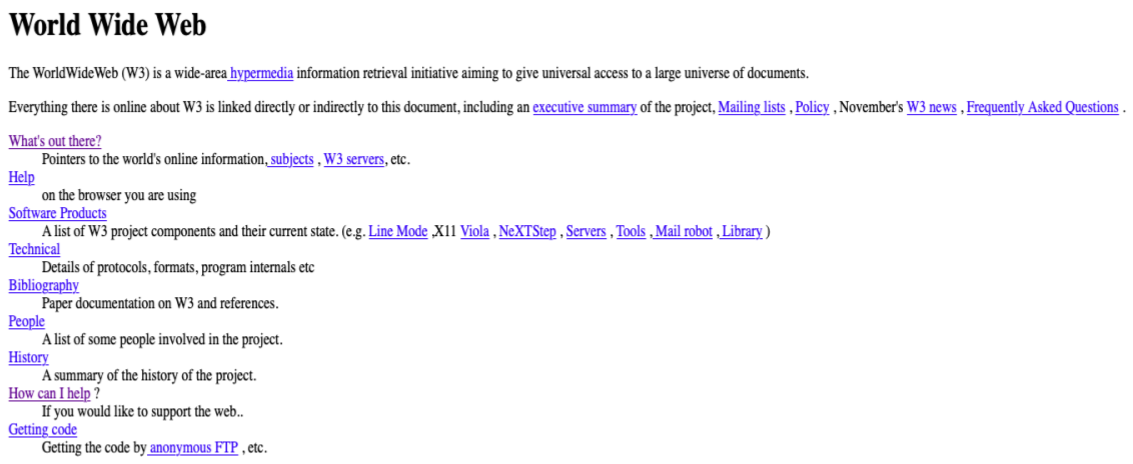
The thesis starts with introducing the main concepts (Front end web development and Shopify App development), followed by establishing goals, objectives, purposes behind the study, and other relevant knowledge such as methodology and background information about the main concepts. The background chapter includes a brief history of front-end development and Shopify. The main concepts are discussed further in the theoretical review sections.

Secondly, the thesis covers the implementation of an app called ‘Flashify’. The app, when ready for publication, would aim to help store owners easily create flash sales in their stores, combined with other useful features related to sales. Finally, the thesis discusses the outcome and results, evaluates them from a critical point-of-view, suggests improvements, and offers possible future continuations to the thesis. The critical review includes a discussion of concerns and issues that occurred during the writing of the thesis and the implementation of the app. It also discusses how they could have been avoided.

2 HISTORICAL BACKGROUND

2.1 The World Wide Web (W3): A Brief History

The World Wide Web (referred to as WWW or W3) is a global medium used to share information using computers connected to the internet. It was incepted in 1989 by the computer scientist Tim Berners-Lee (also known as TimBL) who also published the first web page in the history of the web. [5] The first page created by TimBL is represented in the next figure.


World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

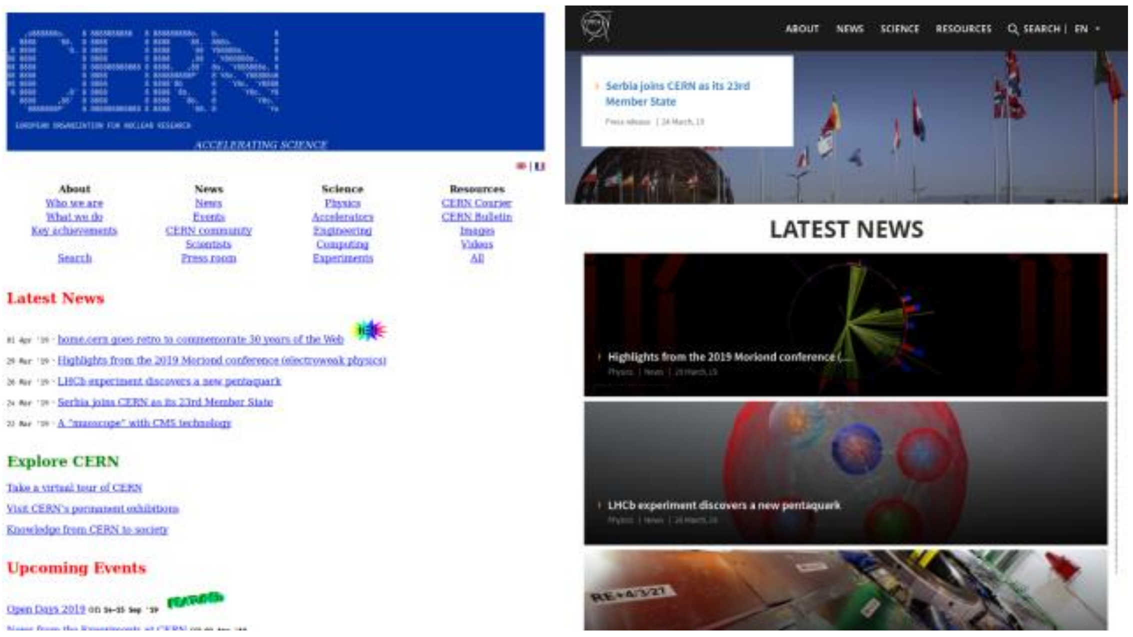
What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Figure 1. The first web page in history created in August 1991 by the inventor of the web TimBL. Fetched from [6]

By 1990, a functioning web system was implemented. [5] On the 06th of August 1991, the web became available worldwide and for public use [7]. Although the major transformation happened in 1993 when CERN (European Organization for Nuclear Research) announced that moving forward, the W3 is available for everyone to use and develop at no cost, a key announcement that changed the world [8]. The royalty-free use and development of the web, in addition to the introduction of the first browser that allowed embedding images in texts, called Mosaic (which later became Netscape Navigator), resulted in a sharp uptick in its use and development. Mosaic was particularly credited with the internet boom not only due to the introduction of texts and images on the same web page but also due to being particularly easy to use and install. [9]

2.2 The Evolution of Front-End Web Development

Front-end development has gotten relatively sophisticated in comparison to how it started, the image below represents the major changes that happened to the CERN website as web technology advanced over time.



CERN

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

ACCELERATING SCIENCE

About	News	Science	Resources
Who we are	News	Physics	CERN Courier
What we do	Events	Accelerators	CERN Bulletin
Key achievements	CERN community	Engineering	Images
	Scientists	Computing	Videos
Search	Press room	Experiments	All

Latest News

01 Apr '19 · home.cern goes retro to commemorate 30 years of the Web
29 Mar '19 · Highlights from the 2019 Moriond conference (electroweak physics)
26 Mar '19 · LHCb experiment discovers a new pentaquark
24 Mar '19 · Serbia joins CERN as its 23rd Member State
22 Mar '19 · A "marscope" with CMS technology

Explore CERN

Take a virtual tour of CERN
Visit CERN's permanent exhibitions
Knowledge from CERN to society

Upcoming Events

Open Days 2019 on 14–15 Sep '19

ABOUT NEWS SCIENCE RESOURCES SEARCH | EN

Serbia joins CERN as its 23rd Member State
Press release | 24 March, 19

LATEST NEWS

Highlights from the 2019 Moriond conference (...
Physics | News | 29 March, 19

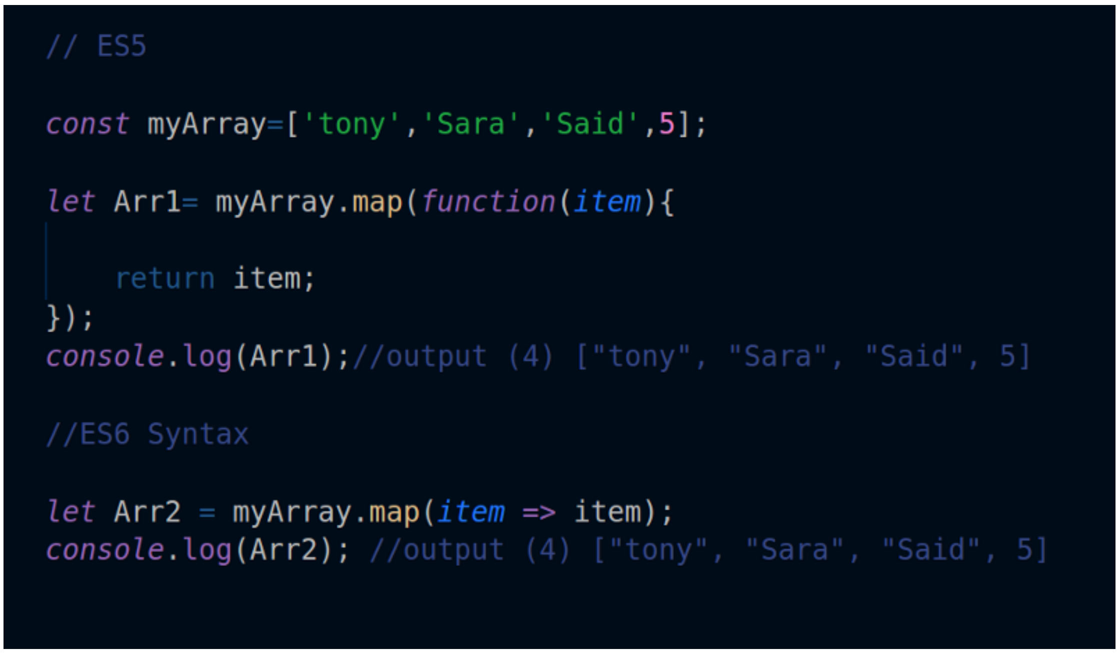
LHCb experiment discovers a new pentaquark
Physics | News | 26 March, 19

Figure 2. An early version of the CERN website (on the left) vs. A later modern version (on the right). [10]

The first page of the web created by TimBL was basic HTML in its first version. HTML was officially released in 1993, although other non-public versions existed before. Moving on from that point, HTML has gotten new releases up to the current version (HTML5) at the time of writing this document. The first version of HTML was referred to as HTML 1.0 Strict. The second version of HTML was introduced in 1995 with new features added. While that version was significantly improved, issues started to rise among browsers due to the introduction of extension tags made by browser developers. In 1997, HTML 3 was introduced and approved by many major browsers at the time. The third version of HTML included many additional features such as tables, text flow around images, subscripts and superscripts. Two years later in 1999, the fourth version of HTML was released. It was the most revolutionary at the time. HTML 4 supported more and more modern features such as styling, scripting, and printing facilities. The current version of HTML (known as HTML5) was released in 2014 and it has gone through a significant amount of development since its predecessors. At this point, HTML can handle videos, offline use of web applications, and graphics. [11], [12]

As HTML could do next to nothing in terms of design, CSS was first proposed in 1994 by Håkon Wium Lie. Per the proposal, HTML would continue as is, and CSS would define the way a website would look. The official release of CSS happened in 1996 and during the same year, W3C issued an official statement recommending the use of CSS. The following versions of CSS introduced brand-new features related to positioning, Z-index, and media elements, in addition to other improvements. Finally, CSS3 was launched in 1999 which is the latest official CSS version by W3C. This version is the most advanced version of CSS with better features related to font support, formatting, and module division. [13]

To add interactivity to websites, JS was introduced to web development. JS was created by Brendan Eich in 1995. Development of JS started with Netscape 2 (a major browser at the time) and later became an ECMA standard. New ECMA standards were periodically released and Javascript development has moved forward with more compatibility and support for various browsers. As of now, the current ECMA standard is ES6 with the syntax shown below in the next figure (compared to ES5 syntax). [14]



```
// ES5

const myArray=['tony','Sara','Said',5];

let Arr1= myArray.map(function(item){

    return item;
});
console.log(Arr1);//output (4) ["tony", "Sara", "Said", 5]

//ES6 Syntax

let Arr2 = myArray.map(item => item);
console.log(Arr2); //output (4) ["tony", "Sara", "Said", 5]
```

Figure 3. Basic JS code showing JS Syntax (ES5 and ES6 compared). [15]

Many modern technologies are based on JS, such as React, Node.js, Vue.js, and Angular [16]. Currently, JS is more than just the scripting language used to add interactivity to the web, since it can also run on the server. A JS developer can now develop apps for various platforms with JS as a result of the rapid development of JS and its frameworks.

2.3 Shopify Inc.: Where It Was and Where It Is Now

Shopify is a multinational e-commerce company founded in 2006 by Tobias Lütke, Daniel Weinand, and Scott Lake. The headquarter (HQ) of the company is based in Canada. Shopify offers the end-user a platform of services needed to establish an online store such as payment gateways, marketing systems, shipping and customer engagement tools. In other words, customers have an all-in-one solution to get started selling online through Shopify. [4], [17] Shopify started with 5 employees before growing over the years to over 10,000 current employees. The company operates worldwide with total transactions worth 543 Billion United States Dollars (USD). [4]

Shopify’s app store, which acts as the marketplace for additional plugins and features, currently has over 8,000 apps available for all merchants to use. It is estimated that there are currently over 1.75 million merchants using Shopify as an e-commerce solution. [4]

3 IMPLEMENTATION METHODOLOGY

3.1 Selection Criteria

Since the thesis is categorized as a functional thesis, it does not aim to answer questions concerning the best methodology to use or the best technologies to choose from. The reader should always keep in mind that every technology, approach, and model has its advantages and disadvantages. Solutions and selections were based on many prerequisites, such as individual preferences, skills, and experiences, technical needs, project requirements, and the nature of the project. The selections of methodologies and the technologies used during the implementation were mainly based on the following:-

- Official recommendations by Shopify
- Available technologies for Shopify app development
- Technical skills of the developer
- Career objectives of the developer
- Learning goals of the developer
- Thesis scope
- Scalability
- Personal preferences of the developer

3.2 Model and Approach

The project work was conducted mainly using the Kanban model, with a few principles adapted from the Rapid application development (RAD) approach. This methodology was selected due to the nature of the project and the timeframe available to the developer. The selection was done after careful consideration of the benefits and disadvantages of the approaches and determining the right methodology for the individual project in question.

3.2.1 Kanban Approach

The word ‘Kanban’ comes from the Japanese language which means ‘Visual sign’ or ‘Visual board’. The approach has been in use since the 1950s. It was first developed and adopted by the car manufacturer ‘Toyota’. The Kanban system is designed to define, manage, and improve services that deliver knowledge work using lean workflow management, while applying many agile principles. Through Kanban, work is visualized, efficiency is maximized, and continuous improvement is encouraged. Work is represented on Kanban boards, allowing work delivery optimization across multiple teams and the handling of even complex projects in a single environment. [18]

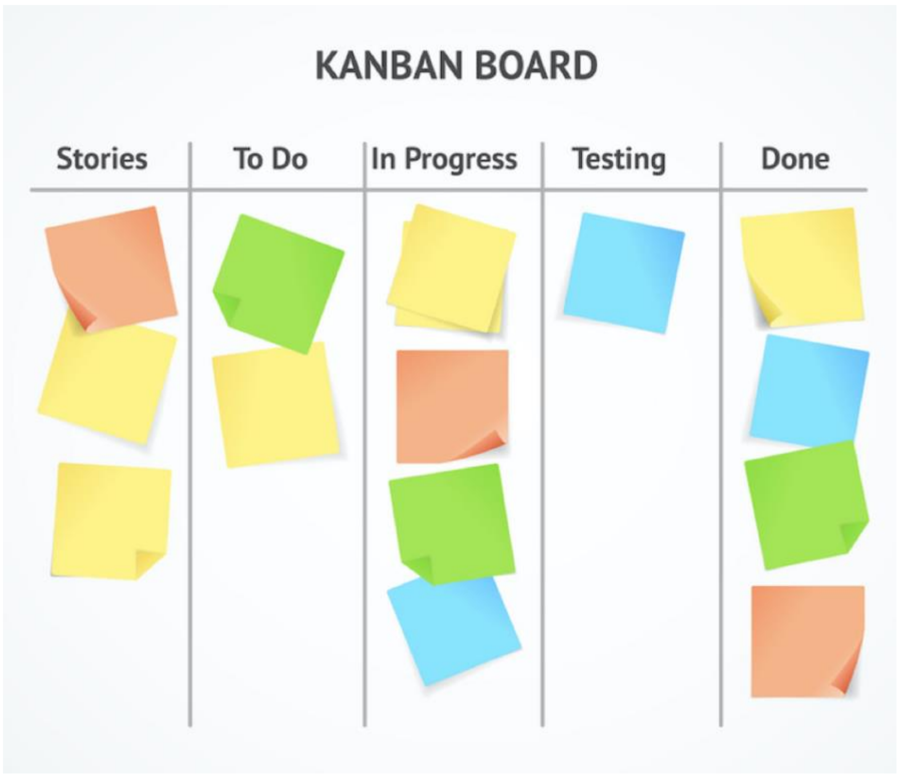


Figure 4. An example of a Kanban board. [19]

The Kanban method is most suitable for work where small tasks need to be completed quickly. This is especially true where requirements change frequently, or where changes need to be made as soon as possible. According to [20], for a successful Kanban implementation, six core practices must take place:

1. Visualize the workflow
2. Limit ‘work-in-progress’
3. Manage flow
4. Assure policies explicitness
5. Adopt feedback loops
6. Improve collaboration

According to an article by [21], a study done by ‘State of Kanban’ showed that enhancing the visibility of work and continuous improvement are the main reasons for adopting the Kanban method. The study included a survey aiming to understand the main benefits of Kanban. The results were as follows:

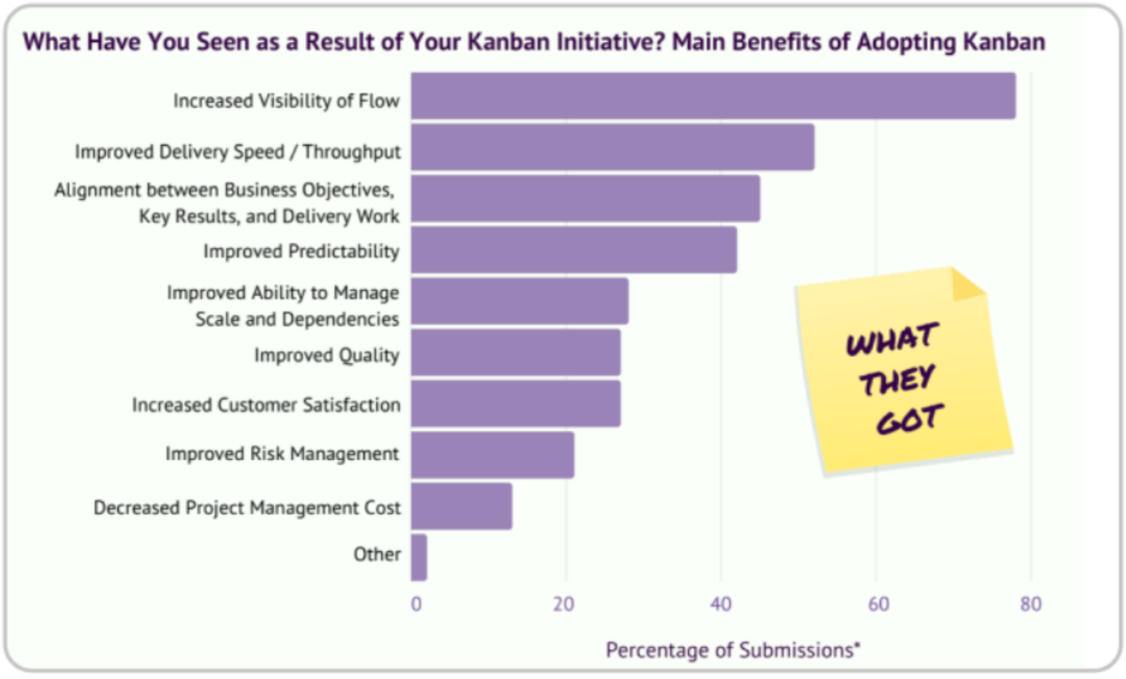


Figure 5. Result of a survey question of a study conducted by 'State of Kanban' aiming to understand the benefits of the Kanban model. Adapted from [21]

3.2.2 Rapid Application Development (RAD) Model

As opposed to long-drawn-out development and testing cycles, Rapid Application Development (RAD) emphasizes rapid prototyping and immediate feedback. With RAD, the software can be updated repeatedly without having to start from scratch. As a result, the final result is more quality-focused and aligned with the needs of the end users/customers. RAD was developed in the 1970s and 1980s as a response to the waterfall model. [22], [23] The figure below visualizes the general workflow in the RAD approach.

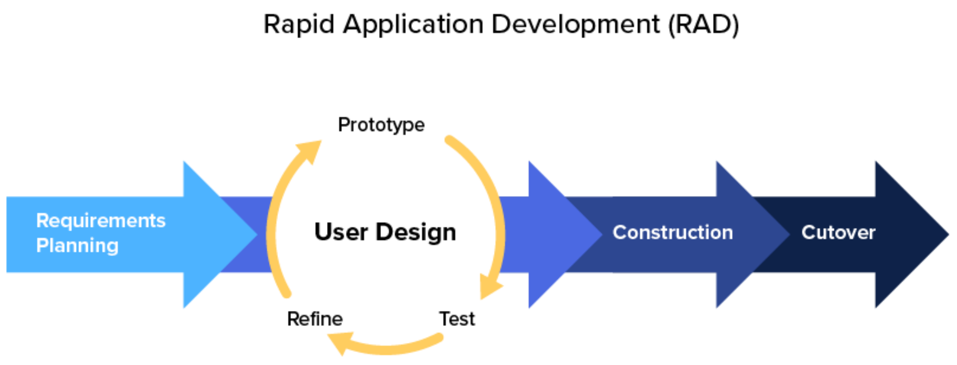


Figure 6. RAD workflow and steps. [22]

According to figure 6 and [22], the RAD model includes 4 main steps as follows:

1- Defining product requirements: Unlike many other models, the requirements do not have to be strict and can be broad. They can act more like guidelines.

2- Prototyping: Work starts, aiming to come up with an initial model and prototypes. Creating a working design for the client is the primary goal. To ensure that the client's needs are met, developers and designers work together until the final product is ready. The final product is produced only at the finalization stage, once the client and developer agree.

3- Construction and feedback: The rapid construction process involves coding, testing, and integrating prototypes and beta systems into a working version.
4- Final Stage: At this stage, developers address technical debt accumulated during earlier prototyping from previous stages, optimising implementation to improve stability and maintainability. In addition, components are moved to live production and full-scale testing takes place.

Advantages of the RAD model	**Disadvantages of the RAD model**
Requirements are modifiable at any point.	Requires strong team collaboration.
Customer feedback is encouraged and prioritised.	Not ideal for large teams.
Reviews are quick.	Requires highly skilled developers.
A significant reduction in development time is achieved.	Requires user requirements throughout the life cycle of the product.
Allows more productivity with fewer people.	Suitable for short-term projects only.
The time between prototypes and iterations is short.	More complex to manage (compared to other models).
Integration starts from the beginning of the project.	A rapid application development process can only be used to develop modularized systems.

Table 1. Advantages and disadvantages of RAD model. Adapted from [22]

4 FRONT-END WEB DEVELOPMENT: TECHNOLOGIES AND CONCEPTS

A front-end developer creates a UI, or basically what a user would see and interact with. The basic languages and technologies that comprise the front end are HTML, CSS, and JS. Thus, a programmer always starts learning those technologies as they make up the core and the base of front-end web development. [24]

Today’s modern web technologies are not limited to the languages above. Frameworks and libraries have become essential in front-end web development. Below are examples of modern libraries and frameworks:

- Bootstrap (CSS)
- React (JS)
- Angular (JS)
- jQuery (JS)
- Semantic-UI (CSS)

There are many additional libraries and tools currently available and each has its advantages and disadvantages. As a developer, the choice of which technology to use also depends on what technologies have been utilized during the developer’s educational and career life. An example of a popular JS library currently used for web development is React.

React is an open-source Javascript library built by Facebook and used in front-end development to build interactive UIs. The core idea of React is to be able to develop and reuse components to minimize the amount of code in the software, while only updating/re-rendering the relevant components when necessary. [25]

Another key concept in modern-day web development is ‘mobile-first’ design. This is because visiting websites from mobile devices is the new norm. Mobile phones are significantly different from desktops in terms of screen size, memory, and processing power, making the web development process not as straightforward. This mobile-first approach considers mobile phones with their screen size and capabilities as the target of the development process, before considering desktops and other larger devices. In other words, developers who are using a mobile-first approach start designing and prototyping firstly for smaller devices, such as phones, followed by bigger devices and screens, such as desktops. [26]

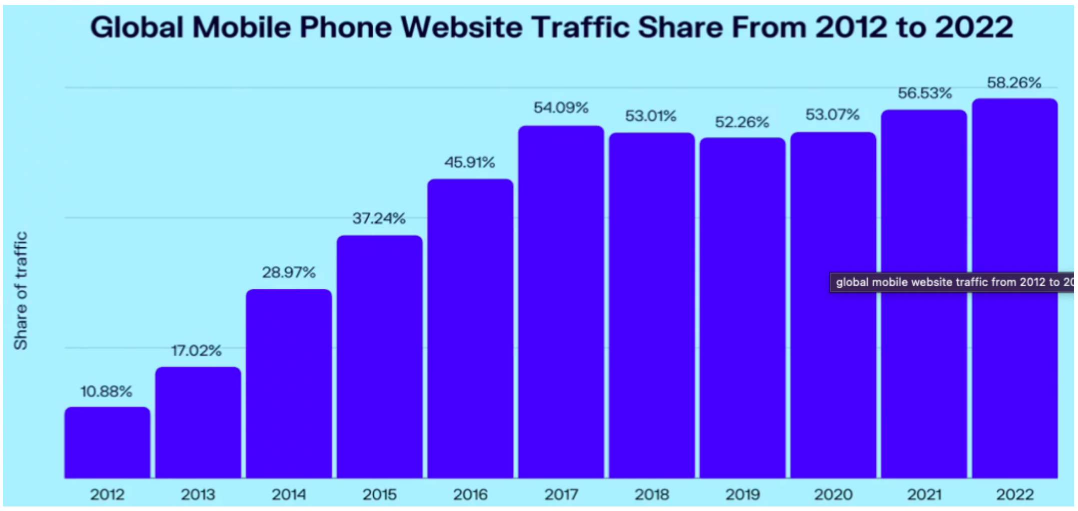


Figure 7. Global mobile phone website traffic share between 2012 and 2022. Adapted from [27]

Following the increasing use of mobile phones to access the web, the mobile-first approach emerged distinctly. At some point before that, the assumption was that visitors to the web would access websites from desktops. According to a previous statistical study, mobile devices have taken over desktops in terms of market share almost every year between 2016 and 2022 [28]. Other studies support that the usage of mobile phones is rapidly increasing worldwide. The previous statements show the significant importance of the mobile-first approach from a technical point of view and a business point of view alike which requires the developer to keep in mind and apply various practices and techniques while developing. [27]

5 SHOPIFY DEVELOPMENT: TOOLS AND DEPENDENCIES ENGINEERED FOR SHOPIFY

The main purpose of Shopify app development is to extend the existing features and enhance the merchant’s/customer’s experience. As a result, Shopify offers a selection of tools available for Shopify developers on the official developer’s site to expedite and enhance the development process. The thesis covers the tools based on relevance to the practical part. As a result, not all tools developed by Shopify were included.

In terms of the design system, Shopify uses Polaris both internally at Shopify and externally for developers to build Shopify apps. Polaris is comprised of design guidelines, code libraries, developer opinions, and API documentation. [29] As the Shopify app needs to communicate with the rest of the Shopify Admin interface while also needing to get rendered through the Admin panel, Shopify App Bridge is used to achieve that. The App Bridge library offers components used to enable Shopify apps to be embedded in the Shopify admin panel. [30]

In terms of starting the coding process, Shopify offers a practical command-line interface called Shopify CLI. It is used to do common development tasks such as generating Shopify app templates and app extensions. While the developer can use other technology stacks to build the app (e.g. PHP or Ruby), Shopify CLI relies on Node.js for installation and dependency management. CLI currently allows developers to generate templates written in Node, Ruby, and PHP, contributing to an expedited start to the coding process. [31], [32]

Shopify libraries include many APIs that allow using the platform’s built-in features and extensions such as reading and writing merchant data. Shopify APIs require authentication only in some cases, depending on the API used. As for app development, the Admin API is the main way apps interact with Shopify. The Admin API provides comprehensive access to data about Shopify stores such as products, inventory, discounts, and more. It also allows adding features to the app. The Admin API supports both REST and GraphQL. [33], [34]

6 IMPLEMENTATION WORK

This section of the thesis discusses the planning process of a Shopify application, the tools and technologies used during the front-end development, the implementation stage, and the application itself in terms of features and use cases. The chapter does not aim to represent the only way to create a Shopify app but rather discusses some of the available technologies that can be employed (and were chosen for this project) to build a Shopify app.

6.1 Planning

The project aims to develop the front end of a Shopify app called ‘Flashify’. The app, when in the production-ready stage (the process will extend beyond the thesis timeframe), is supposed to have two main uses:

- Merchants can create discounts for their flash sale campaigns.
- Merchants can create QR codes for their flash sales.

By defining use cases, main features can be defined in order to help divide tasks into smaller segments:

- Adding discount with a defined start-time, end-time, discount value and unit and other necessary specifications required.
- Generating QR codes for created discounts that will lead a customer to the cart page with discounts and products already applied and added to the cart, making the checkout process as seamless as possible. The merchant will define some required specifications for the QR code generated during the creation process.
- Viewing current discounts.
- Viewing generated QR codes.

The features defined above were used to define task segments. The tasks included creating main components and child components within the parent components.

- The main page of the app.
- A navigation bar with buttons to create a new discount/QR code.
- Smaller components within the main page (e.g. Layouts or Cards) to view currently existing QR codes and discounts.
- A page with a form to create a QR code.
- Smaller components (e.g. sections, cards, and text fields) for the QR code creation form.
- A page with a form to create a discount.
- Smaller components (eg. sections, cards, and text fields) for the discount creation form.

To achieve an organized workflow, A paper version Kanban board was used with 5 main sections.

- Use cases (tagged with the necessary features and components)
- Front-end components to build
- Components in development progress
- Components in the testing phase
- Completed components

Tasks with higher priorities were placed higher on the board within the relevant section. This was done to maintain as high efficiency as possible by breaking tasks into smaller pieces and moving them to the next stage as efficiently as possible. The method was chosen as it was determined to be the most efficient for this project according to the advantages of the Kanban method discussed previously in this document. The choice was shown to provide satisfactory support to the development process along the implementation timeline, although the implementation was done by one developer.

6.2 Hardware Setup

The hardware setup was defined as per the availability of existing hardware. The device used for the development process was the developer’s MacBook Air (2017 edition, 13 inches) holding model number A1466. The laptop has a

1.8GHz Dual-Core Intel i5 processor with 8 GB 1600 MHz DDR3 memory. The hardware used was confidently believed to be sufficient for this project.

The operating system (OS) used at the time of development is macOS Monterey (Version 12.X) which did not have any major issues that would negatively impact the development process or the outcome.

The hardware setup is normally a personal choice decided by each developer. This choice should not have a direct impact on the project itself in most cases, with exceptions of course but the exceptions are irrelevant to this specific project.

6.3 Integrated Development Environment (IDE)

Visual Studio Code (commonly referred to as VS Code) was chosen to be the IDE/Code editor for this project. According to freecodecamp.com, *'an IDE is a handy piece of software that acts as a text editor, debugger, and compiler all in one. IDEs are designed to make coding easier for developers'* [35]. Keeping this definition in mind, VS Code precisely fits the definition taking into account the previous experiences and preferences of the project's developer and thesis author.

The IDE was also chosen while taking into account the hardware setup and its capabilities. VS Code is a powerful yet lightweight code editor available on Windows, macOS and Linux. It has many extensions available to install, in addition to many useful features that a developer might need or find useful within the development process, such as keyboard shortcuts including custom ones. VSCode comes with built-in support for JS, TypeScript (TS), and Node.js yet still allows extensions for other languages such as C++, Python, and PHP. [36]

This is another situation where a choice is more relevant to the developer than to the project. For web development projects, there are many alternatives to VS Code but the final choice is generally decided by the developer.

6.4 Technologies in-use

As a requirement to build a Shopify app, a developer must choose a Node.js package manager from the three options listed by Shopify which are: npm, Yarn1.x, or pnpm. Npm was the Node package manager selected for this project as it was the familiar option for the developer with previous experience using it and the pre-existence of the package manager already installed on the development device. In addition to a package manager, the version of Node.js required is set to be 14.13.1 or higher at the time of this project. Shopify also requires developers to have the latest version of Chrome or Firefox installed in addition to Git (without any version recommended officially by Shopify). [37]

Shopify CLI was the most efficient way to get started with building the app, as it creates an app template with a significantly lower amount of commands. With a simple command (npm init @shopify/app@latest), an app template with the necessary files and dependencies to get started is generated, after inserting the app name and the relevant template to be used.

Shopify CLI automatically installs the necessary dependencies and libraries such as Polaris, while also importing some components relevant to the app template such as useNavigate, TitleBar, and Loading. Some other libraries installed include App Bridge, Polaris Icons, and React (with some components and hooks pre-imported such as useState and useCallback).

Additional libraries were installed manually through the terminal during the project as they became necessary. @shopify/react-form package was installed to create forms through React hooks and Day.js library was installed to deal with date and time parsing, validating, manipulating, and displaying on modern browsers. Full confidence with the use of every package and library used in this project did not seem to be necessary but a general idea and some familiarity would be required which can be achieved through developer documents on the Shopify developer’s website. Packages used were as follows: React, React-router-dom, App Bridge (and its utilities), Polaris, React-query, Polaris Icons, React-form, Day.js, and React-hooks.

For version control, GitHub was used as a version control hosting system since it is free and open source. GitHub has a desktop app that provides a neat GUI allowing several tasks such as fetching, pushing, forking…etc. Some git commands were used in the terminal as well.

6.5 Implementation and Development

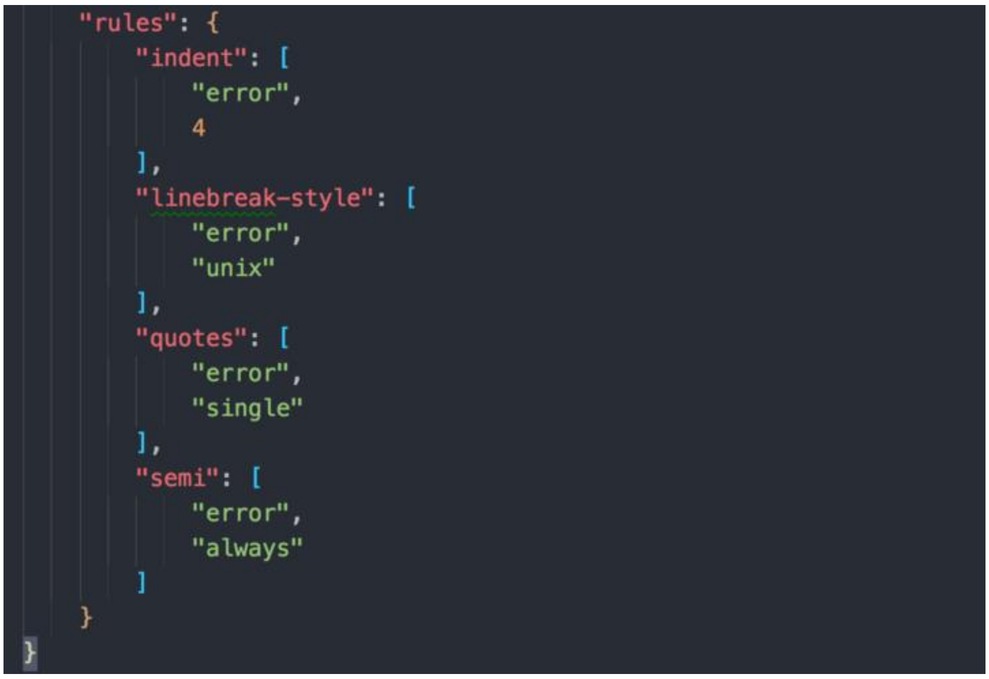
The section discusses the main crucial technology related to the head topic of the thesis, which may not elaborate on each technology used. A web development project typically includes so many libraries, tools, and technologies used simultaneously and following the developer's experience, confidence in the main technologies (referring to the stack as an example) is normally sufficient to proceed.

6.5.1 ESLint

ESLint is an open-source linting solution that helps developers find issues in their JS codes, automatically fix some coding errors, and allow code configuration and syntax rulings. By using ESLint, developers can avoid many errors and bugs and report them, as well as allow setting a unified rule sheet defining syntax patterns. [38] This is especially useful when working within a team where code patterns can be different from one developer to another. An open-source code acts as a perfect candidate that must use a linting tool such as ESLint. Open-source projects get a tremendous amount of contributions from all over the world. In such a case, linting would be necessary to ensure reusability and readability.

The use of ESLint was considered at the beginning of the project but it did not occur. The reason for dismissing the use of ESLint was the fact that the development stage in this project was carried out by one person. Linting offers several benefits in the development process but it is especially useful when more than one developer is involved in the development process in order to unify the style of coding. As the project in this thesis did not include several developers, linting did not seem necessary to the developer.

The decision to not use linting did not appear to be a good decision from a critical point of view considering that, firstly, it is generally a good development practice to involve linting in every project and secondly, the aim to continue working on this project after the thesis is published (more developers could get involved). If ESLint was used, a rule sheet which would be saved in the root folder of the project could look like below



```
    "rules": {
        "indent": [
            "error",
            4
        ],
        "linebreak-style": [
            "error",
            "unix"
        ],
        "quotes": [
            "error",
            "single"
        ],
        "semi": [
            "error",
            "always"
        ]
    }
}
```

Figure 8. An example of an ESLint rule sheet. [39]

6.5.2 Node.js

As mentioned previously, Shopify has set some prerequisites to develop a Shopify app and one of them was a Node.js runtime environment version 14.13.1 or higher. Node.js can be installed from the official site, or if installed already, the version can be confirmed from the terminal as is the case with package managers. Node.js was already installed with NPM as a package manager due to previous use of the runtime environment and the package manager. The version installed was compatible with Shopify’s requirements.

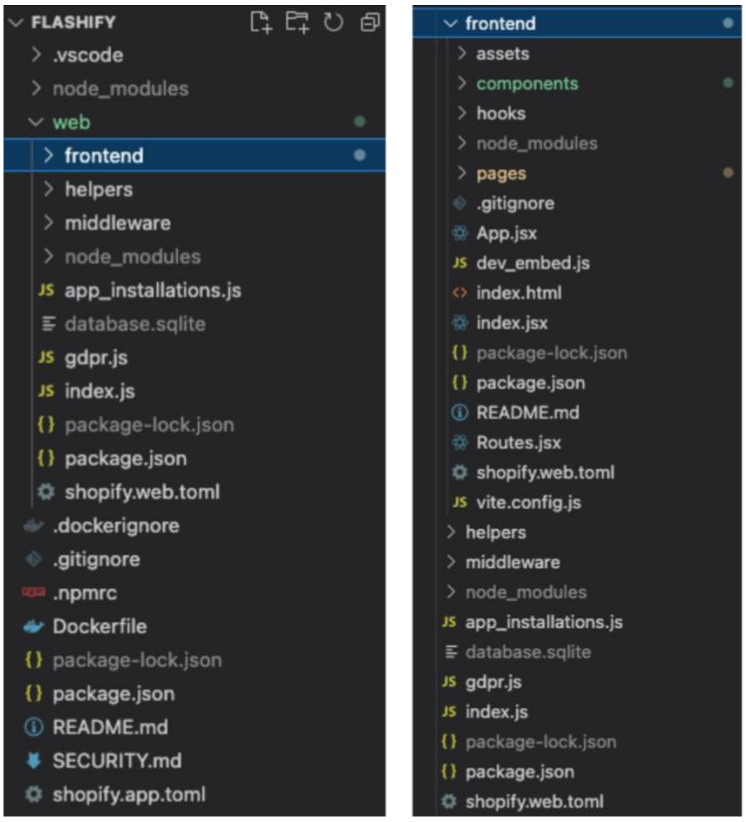
Node.js is a free, open-source, cross-platform JS runtime environment. It uses an asynchronous programming approach, meaning it can accept a new request even when the previous one is not complete yet (unlike PHP, where the request must be complete and handled before accepting the next one). Node.js runs on several operating systems such as Windows, Unix, macOS, and Linux. While JS was initially created as a scripting language to improve the interactivity of the web, Node.js allows JS to run on the server and it comes with built-in modules by default, although more can be added as needed and new modules can be created from scratch. Node.js packages/modules are managed by for example NPM. NPM acts as an online repository for node.js projects as well as a command-line utility for installing, managing, and interacting with modules and packages, NPM is also used as a dependency manager and version manager. [40], [41]

The interaction with Node.js files was minimal in this project since the project would as planned, remain in the front-end development phase for the whole time of the thesis. However, NPM as a command-line utility was used to install libraries such as the ones mentioned in section 6.4.

6.5.3 Shopify CLI

Shopify CLI is an open-source command-line tool that helps developers generate app templates as well as automate many Shopify app development tasks, resulting in an accelerated development process. It also creates a tunnel to allow viewing the current in-development version of the app in the development store and updates the view as files update in the development repository. The tunnel created is a Ngrok tunnel and the link is generated each time the command (npm run dev) is submitted on the terminal. Shopify CLI consists of a set of node packages that are: *@shopify/CLI* and *@shopify/app.* [42]

The Shopify CLI was used to create a node app template using the relevant command and the name ‘Flashify’ was selected during the creation process. Shopify CLI also requires developers to log in to their developer account in a browser tab that is automatically opened. After template creation, visiting the tunnel was necessary to complete the installation of the app in the development store, followed by a view of the app in the Shopify Admin Panel. App view is lively updated automatically after each file save. Additional manual installation for tools like nodemon was not necessary. At this point, the coding process could start any time only after a few-minute process and here the benefit of Shopify CLI was distinctly highlighted. The picture below represents the directories and files generated automatically by Shopify CLI



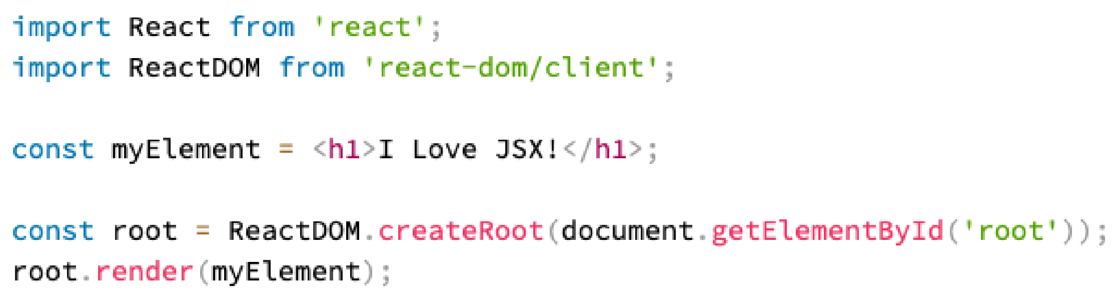
```
FLASHIFY
  > .vscode
  > node_modules
  v web
    > frontend
    > helpers
    > middleware
    > node_modules
    app_installations.js
    database.sqlite
    gdpr.js
    index.js
    package-lock.json
    package.json
    shopify.web.toml
  .dockerignore
  .gitignore
  .npmrc
  Dockerfile
  package-lock.json
  package.json
  README.md
  SECURITY.md
  shopify.app.toml
```

```
v frontend
    > assets
    > components
    > hooks
    > node_modules
    > pages
    .gitignore
    App.jsx
    dev_embed.js
    index.html
    index.jsx
    package-lock.json
    package.json
    README.md
    Routes.jsx
    shopify.web.toml
    vite.config.js
  > helpers
  > middleware
  > node_modules
  app_installations.js
  database.sqlite
  gdpr.js
  index.js
  package-lock.json
  package.json
  shopify.web.toml
```

Figure 9. Files and directories autogenerated by Shopify CLI (on the left) and the autogenerated front-end directory with its files (on the right).

6.5.4 React

React, a JS library, developed by Meta (Previously known as Facebook Inc.), is used to create user interfaces. One of the main advantages of React is the creation of reusable components. React uses JSX syntax which is officially recommended by Meta as the creators of React, although it is not required. JSX provides developers with the advantage of combining markup and logic instead of separating them into different files. [43], [44] The picture below represents a basic React app that uses JSX



```
import React from 'react';
import ReactDOM from 'react-dom/client';

const myElement = <h1>I Love JSX!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

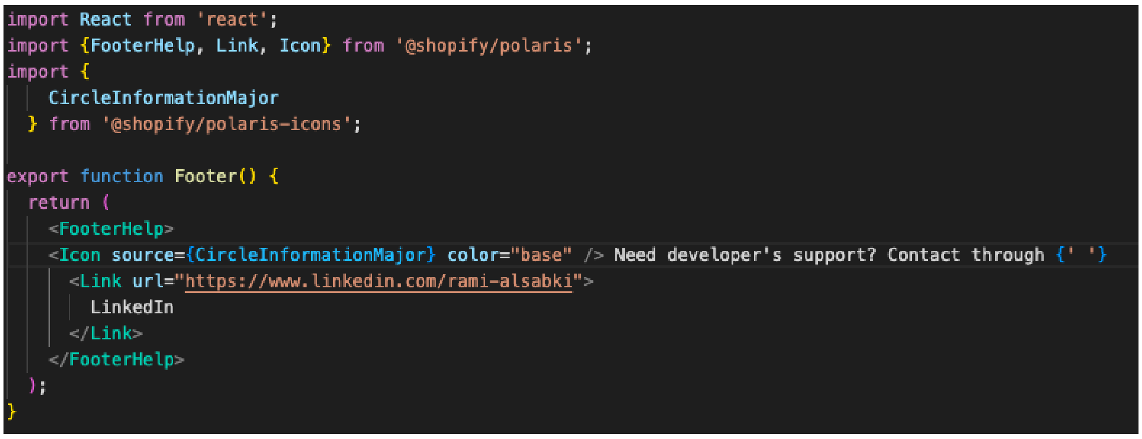
Figure 10. An example React code with a JSX. Fetched from [45]

React takes advantage of the memory by creating a virtual DOM. This allows it to do all the necessary changes there before changing the browser’s DOM instead of making changes to the browser’s DOM directly. In addition, React does not re-render the whole page when a change is made. Instead, it changes/re-renders only what needs to be changed/re-rendered. [44] Here is where concepts like state and props are essential and mandatory to be familiar with. As represented by [46], in React, rendering generally occurs in these three scenarios:-

- An example app has a state with a variable, and that variable is changed through setState. The component in that case gets changed and re-rendered to reflect changes.
- React utilizes props. Like the previous case, when props change, it leads to a change in state. As a result, a component is re-rendered to reflect changes.

- Assuming the example app also includes a parent component with several child components. A change to the parent component will cause both the parent component and the child components to re-render.

React was the library used to build this project with other tools added such as the Polaris design system which will be discussed in the next section. JSX was the syntax used and the development process generally did not differ from any React app development process. As a rule, a React component holds a name starting with an uppercase, this is necessary to distinguish React components from other built-in elements such as <div>. The picture below represents a react component created during the implementation work. The component is a footer that was used within the app.



```
import React from 'react';
import {FooterHelp, Link, Icon} from '@shopify/polaris';
import {
    CircleInformationMajor
  } from '@shopify/polaris-icons';

export function Footer() {
  return (
    <FooterHelp>
    <Icon source={CircleInformationMajor} color="base" /> Need developer's support? Contact through {' '}
      <Link url="https://www.linkedin.com/rami-alsabki">
        LinkedIn
      </Link>
    </FooterHelp>
  );
}
```

Figure 11. The footer component created for Flashify app.

6.5.5 Polaris

As previously mentioned in a previous section. Polaris is the official design system created and maintained by Shopify Inc. The foundation blocks that create Polaris, according to an official page from the Shopify website [47], are as follows:

- Patterns: They address the most common situations that merchants face and provide guidance on the best practices for dealing with these situations from a design aspect.

- Primitives: The smallest units in the Polaris design system. They represent codified design decisions and reusable visual elements.
- Components: Packaged in @Shopify/Polaris, which is a React library. They are highly composable and reusable. Usually assembled to create a designed user experience.
- Tokens: They are reusable design decisions such as colour, spacing, and typography. Tokens are used across the platform to unify the merchant's UX.

Using Polaris for this project contributed positively to the development process by accelerating it due to having a well-founded design system. In addition, it allowed designs to blend in with the Shopify user experience while also leaving some room for developers to make some design decisions.

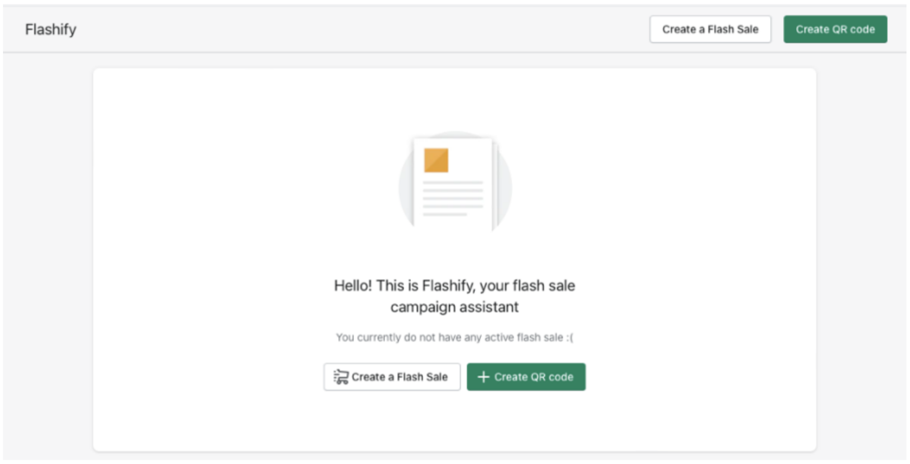

Flashify
Create a Flash Sale
Create QR code
Hello! This is Flashify, your flash sale campaign assistant
You currently do not have any active flash sale :(
Create a Flash Sale
Create QR code

Figure 12. The main page of the Flashify app (non-final version within the development process).

The picture above shows the main page of the app 'Flashify', assuming the user has not generated any discount or QR code yet (in that case, no mock data was entered). The page consists of many components from the Polaris system such as buttons, texts, and an empty state. The page also includes icons fetched from the Polaris icon library which is imported separately to the pages where it is needed. Props were added following the design decisions made. For example, the two buttons shown in the middle of the picture above have different props and as a result, different designs, which is an example of how Polaris helps make the front-end coding process more time-efficient.

7 TESTING AND RESULTS

As the implementation part was completed, ‘Flashify’ was officially ready on the front-end side and could move on to the next development stage after final testing. The plan set was to complete the following stages such as backend development outside the thesis timeframe, as the backend was not included in the thesis scope.

The project utilized manual functional testing methods to test the app. Functional testing methods are used to verify functionality against requirements and specifications predefined [48]. The testing was executed by the developer both during and after the development stage. The app was tested mainly with mock data as the app could not receive any data from the server (backend) yet. As a result, code access was required to simulate scenarios such as page loading (the component would receive props defining that loading data is in progress) and data added. The testing combined the following techniques:

1. Unit Testing: it tests software’s components individually. It is time-efficient and cost-efficient, but it does not catch every error in the whole app, especially integration errors [49].
2. Smoke Testing (also referred to as “build verification testing” or “Confidence Testing”): it is an easy-to-perform technique that verifies the functionality/stability of main features. It also minimizes risks and is time-efficient [50].

In order to conclude the test, the following steps were required (according to [51]):

- Understanding the functional requirements.
- Defining test cases.
- Defining test data based on requirements.
- Defining the expected outcomes using the test data.
- Executing test cases.
- Comparing test results and expected results.

After following the steps, tests were defined. the main tests are summarized in the next table:

SCENARIO	**STEPS**	**EXPECTED RESULT**	**ACTUAL RESULT**	**STAT-US**
No data exists.	Open the main page when there is no mock data.	The page suggests creating a discount or a QR code.	The page did not have any data and showed two buttons to create a new QR code/discount.	Passed.
Data exists.	Open the main page when there is mock data (QR as an example).	The page shows tables with mock data represented clearly.	The page showed the QR data that were used as mock data.	Passed.
“Create a new QR” button navigates to the right form.	Click on “Create a new QR” button on the navigation bar.	The app navigates to the form used to create a new QR code.	The app navigated to the form page requested.	Passed.
404 page shows when necessary.	Navigate to a random resource (example: QR number 793).	The app shows the 404 error page created during the development process (not the generic Shopify 404 error page).	The 404 page shown matched the page developed during the implementation stage.	Passed.

Table 2. Summary of tests.

8 DISCUSSION AND FURTHER DEVELOPMENT

The thesis was self-evaluated by the author (also the developer) of this thesis. And the app was tested in order to determine the success in reaching the technical objectives of the thesis. This section discusses the evaluation from technical and theoretical standpoints.

Despite the several difficulties and challenges that occurred during the implementation process, the final front-end version of the Shopify app ‘Flashify’ has been successfully implemented. The backend of the app is outside the scope of the thesis and would be the next stage which extends beyond the thesis timeframe. The features discussed before implementation were all successfully developed. As a result, the use cases discussed in this thesis are achievable.

Challenges and difficulties occurred along the way. As with any other project, problems can occur, whether foreseeable or unforeseeable. This might slow progress or temporarily halt it. In this project, halts occurred during issues related to the code or getting support with coding issues. One of the aims of the thesis is to act as a supporting document for Shopify app development due to the relatively low amount of community support existing. As a result, less support is available when it came to implementing the thesis. This issue was solved by attempting to acquire personalized community support through StackOverflow.

According to the author, the thesis managed to focus on Shopify's front-end development while also discussing front-end development as a general concept. Although some backend concepts were necessary, the thesis accurately followed the main topic. The target group for this thesis was the general public as well as developers. Taking that into account, the author tried to balance the level of the text to remain not too simple for experts in the IT field yet not too complicated for the general public. The success factor cannot be examined by the author and can only be determined by third parties such as the general public once the thesis is published.

The resource selection was influenced by the author’s previous personal experiences in earlier projects and studies. This was necessary in order to achieve the best possible results in a time-efficient manner utilizing the author’s existing knowledge and skills. The information, when possible, was checked through other resources to confirm the validity of the pieces of information used during this study.

The disadvantage of documentation in the IT field analogous to this one is that as technologies evolve, some of the information mentioned in the study gets outdated. This is especially applicable to JS frameworks where evolution happens at a fast pace.

Despite the fact that the author is satisfied with the outcome of the thesis, there were several points where it seemed that implementation could have been either done differently or better conducted.

- As mentioned in section 6.5.1, linting could have been utilized from the beginning of the implementation stage. The author plans to fix this by configuring ESlint and a rule sheet at the beginning of the next development phase.
- Testing occurred during and after the development stage. However, not all tests were summarized in table 2 due to the similarity between some tests (e.g. testing the form to create a QR code was similar to testing the form to create a discount).
- No more testing methods and techniques were involved as the developer did not believe it was necessary at that point (however, it will be necessary as the app moves to the next development stages).
- A more illustrative referencing style could have been used for writing this thesis.

The results can be further developed by developing the backend of the application and connecting it to the front end. This is currently planned to take place after the thesis submission. The same idea can also be further developed through the use of another technology, such as PHP or Ruby, as a programming language. This is especially possible since Shopify CLI already offers app templates for those programming languages.

As a side note, Shopify tools are evolving quickly, which makes some of the existing materials 'out-of-date'. In the meantime, while this study was underway, Shopify had already introduced a major version of its CLI (currently version 3.x, the successor to version 2.x) [52]. And the developer's page, if tracked through the web archive, shows the significant changes that occurred in a relatively short time.

9 CONCLUSION

The web has gone through so many changes since it started. The first web page in history shown in this thesis consisted mostly of text and was minimal. Throughout the years, the evolution continued until the current status where we see smart interactive web apps like Shopify handling online financial transactions.

The thesis project aimed at utilizing modern web technologies to create the front end of a Shopify app offering specific features related to sales. The development tools used may have had alternatives as the goal was not to determine the only right way to develop for Shopify. The main aim was to use modern technologies supported by Shopify to build the front end of a Shopify app. Technologies used for this specific project included React, Polaris, Shopify CLI, Node.js, and others. The study updates several previous resources available online. As technologies tend to evolve at a fast pace, some previous studies may be considered at least partially outdated.

This study, like many others in the IT field, may eventually become outdated. However, for some time, it will remain beneficial and function as a useful resource for individuals interested in web development in general, and Shopify app development in particular, in addition to coders interested in building apps for Shopify (especially developers with MERN stack experience). This thesis can help developers get started with Shopify app development and achieve the same results, so they can continue developing their ideas. It can also be utilized to enhance knowledge related to the topic.

The next development stage will involve developing the backend and publishing the app to the Shopify app store for monetization.

References

[1] Cloudflare, “How website performance affects conversion rates,” [Online]. Available: https://www.cloudflare.com/en-gb/learning/performance/more/website-performance-conversion-rates/. [Accessed 02 August 2022].

[2] K. Prinsloo, “Why Every Frontend Developer Should Learn CRO,” 04 July 2021. [Online]. Available: https://swd.hashnode.dev/why-every-frontend-developer-should-learn-cro. [Accessed 02 August 2022].

[3] S. Perez, “Shopify drops its App Store commissions to 0% on developers’ first million in revenue,” 29 June 2021. [Online]. Available: https://techcrunch.com/2021/06/29/shopify-drops-its-app-store-commissions-to-0-on-developers-first-million-in-revenue/. [Accessed 05 August 2022].

[4] Shopify Inc., “Company Info,” [Online]. Available: https://news.shopify.com/company-info. [Accessed 09 August 2022].

[5] CERN, “Web@30: The 30-year anniversary of an invention that changed the world,” 04 March 2019. [Online]. Available: https://home.cern/news/news/computing/web30-30-year-anniversary-invention-changed-world. [Accessed 15 August 2022].

[6] CERN, “Home of the first website,” [Online]. Available: http://info.cern.ch. [Accessed 15 August 2022].

[7] N. Couldry, Media, Society, World: Social Theory and Digital Media Practice, London: Polity Press, 2012.

[8] M. Bryant, “20 years ago today, the World Wide Web opened to the public,” The Next Web (TNW), 06 August 2011. [Online]. Available:

https://thenextweb.com/news/20-years-ago-today-the-world-wide-web-opened-to-the-public. [Accessed 15 August 2022].

[9] S. S. McPherson, Tim Berners-Lee: Inventor of the World Wide Web, Minneapolis: Twenty-First Century Books, 2009.

[10] A. Rao, “The new look of home.cern from 1 April 2019,” CERN, 01 April 2019. [Online]. Available: https://cds.cern.ch/images/CERN-HOMEWEB-PHO-2019-026-2. [Accessed 15 August 2022].

[11] Tutorialstonight, “A Brief History Of HTML,” [Online]. Available: https://www.tutorialstonight.com/html/history-of-html. [Accessed 15 August 2022].

[12] J. Harwood, “The Evolution of HTML,” Medium, 06 October 2018. [Online]. Available: https://medium.com/@jasmineharwood/the-evolution-of-html-837f85e6c1ee. [Accessed 16 August 2022].

[13] W3C, “20 Years of CSS,” 17 December 2016. [Online]. Available: https://www.w3.org/Style/CSS20/. [Accessed 15 August 2022].

[14] W3Schools, “JavaScript History,” [Online]. Available: https://www.w3schools.com/js/js_history.asp. [Accessed 16 August 2022].

[15] S. Hayani, “JavaScript ES6 — write less, do more,” FreeCodeCamp, 20 April 2018. [Online]. Available: https://www.freecodecamp.org/news/write-less-do-more-with-javascript-es6-5fd4a8e50ee2/. [Accessed 16 August 2022].

[16] S. Roddewig, “The 5 Best JavaScript Frameworks of 2022 [+ How to Pick the Right One],” Hubspot, 27 December 2021. [Online]. Available: https://blog.hubspot.com/website/javascript-frameworks. [Accessed 09 October 2022].

[17] J. McLeod, “Shopify holds a healthy chunk of pot sales' upside, says COO.,” *The Financial Post,* no. 30th October 2018, p. 2, 2018.

[18] Kanbanize, “What Is Kanban? Explained for Beginners.,” [Online]. Available: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban. [Accessed 18 August 2022].

[19] Cathy, “The Kanban method in IT development projects,” 29 July 2020. [Online]. Available: https://www.bocasay.com/kanban-method-it-development-projects/). [Accessed 21 August 2022].

[20] Wrike Inc., “The Core Kanban Principles and Practices,” [Online]. Available: https://www.wrike.com/kanban-guide/kanban-principles-practices/. [Accessed 26 October 2022].

[21] N. Tsonev, “State of Kanba Report,” 2021. [Online]. Available: https://kanbanize.com/blog/state-of-kanban-report/. [Accessed 21 August 2022].

[22] Kissflow, “Rapid Application Development (RAD) Model: An Ultimate Guide For App Developers in 2022,” 22 August 2022. [Online]. Available: https://kissflow.com/application-development/rad/rapid-application-development/. [Accessed 21 August 2022].

[23] J. Fredrick P. Brooks, “No Silver Bullet Essence and Accidents of Software Engineering,” 1986. [Online]. Available: http://www.sci.brooklyn.cuny.edu/~sklar/teaching/s10/cis20.2/papers/brooks-no-silver-bullet.pdf. [Accessed 22 August 2022].

[24] W3Schools, “How TO - Become a Front-End Developer,” [Online]. Available: https://www.w3schools.com/howto/howto_blog_become_frontenddev.asp. [Accessed 26 August 2022].

[25] Meta Platforms, Inc., “React - A JavaScript library for building user interfaces,” [Online]. Available: https://reactjs.org. [Accessed 26 August 2022].

[26] Mozilla, “Mobile first,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Responsive/Mobile_first. [Accessed 27 August 2022].

[27] J. Unadkat, “Mobile First Design: What It Is and How to Implement It,” BrowserStack, 01 June 2022. [Online]. Available: https://www.browserstack.com/guide/how-to-implement-mobile-first-design. [Accessed 27 August 2022].

[28] Statcounter, “Desktop vs Mobile vs Tablet Market Share Worldwide,” [Online]. Available: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-200901-202209. [Accessed 27 August 2022].

[29] Shopify Inc., “About Polaris,” [Online]. Available: https://polaris.shopify.com/getting-started/polaris-101. [Accessed 22 October 2022].

[30] Shopify Inc., “Shopify App Bridge,” [Online]. Available: https://shopify.dev/apps/tools/app-bridge. [Accessed 01 September 2022].

[31] Shopify Inc., “App developer tools,” [Online]. Available: https://shopify.dev/apps/tools. [Accessed 01 September 2022].

[32] Shopify Inc., “Shopify CLI for apps,” [Online]. Available: https://shopify.dev/apps/tools/cli. [Accessed 01 September 2022].

[33] Shopify Inc., “Apps overview,” [Online]. Available: https://shopify.dev/apps/getting-started. [Accessed 01 September 2022].

[34] Shopify Inc., “Shopify Admin API,” [Online]. Available: https://shopify.dev/api/admin. [Accessed 01 September 2022].

[35] H. Nyakundi, “What is an IDE in Programming? An IDE Definition for Developers,” freecodecamp.com, 28 June 2021. [Online]. Available:

https://www.freecodecamp.org/news/what-is-an-ide-in-programming-an-ide-definition-for-developers/. [Accessed 04 September 2022].

[36] Microsoft, “Getting Started,” [Online]. Available: https://code.visualstudio.com/docs. [Accessed 04 September 2022].

[37] Shopify Inc., “Create an app,” [Online]. Available: https://shopify.dev/apps/getting-started/create. [Accessed 05 September 2022].

[38] OpenJS Foundation, “ESlint,” [Online]. Available: https://eslint.org. [Accessed 06 September 2022].

[39] A. J. A. Alvarez, “Eslint Basic Configuration,” 12 May 2018. [Online]. Available: https://medium.com/alturasoluciones/eslint-basic-configuration-18b2109d98ec. [Accessed 08 September 2022].

[40] W3Schools, “Node.js Introduction,” [Online]. Available: https://www.w3schools.com/nodejs/nodejs_intro.asp. [Accessed 07 September 2022].

[41] OpenJS Foundation, “About Node.js®,” [Online]. Available: https://nodejs.org/en/about/. [Accessed 07 September 2022].

[42] Shopify Inc., “Shopify CLI for apps,” [Online]. Available: https://shopify.dev/apps/tools/cli. [Accessed 10 September 2022].

[43] Meta Inc., “Introducing JSX,” [Online]. Available: https://reactjs.org/docs/introducing-jsx.html. [Accessed 11 September 2022].

[44] W3Schools, “React Introduction,” [Online]. Available: https://www.w3schools.com/REACT/react_intro.asp. [Accessed 11 September 2022].

[45] W3Schools, “React JSX,” [Online]. Available: https://www.w3schools.com/react/react_jsx.asp. [Accessed 11 September 2022].

[46] Geeksforgeeks, “Re-rendering Components in ReactJS,” 23 September 2021. [Online]. Available: https://www.geeksforgeeks.org/re-rendering-components-in-reactjs/. [Accessed 12 September 2022].

[47] Shopify Inc., “About Polaris,” [Online]. Available: https://polaris.shopify.com/foundations/about-polaris. [Accessed 30 August 2022].

[48] Javatpoint, “Functional Testing,” [Online]. Available: https://www.javatpoint.com/functional-testing. [Accessed 26 October 2022].

[49] T. Hamilton, “Unit Testing Tutorial – What is, Types & Test Example,” 27 August 2022. [Online]. Available: https://www.guru99.com/unit-testing-guide.html. [Accessed 26 October 2022].

[50] T. Hamilton, “What is Smoke Testing?,” 31 August 2022. [Online]. Available: https://www.guru99.com/smoke-testing.html. [Accessed 26 October 2022].

[51] T. Hamilton, “What is Functional Testing? Types & Examples,” 14 September 2022. [Online]. Available: https://www.guru99.com/functional-testing.html. [Accessed 26 October 2022].

[52] Shopify Inc., “Introducing Shopify CLI 3.0,” 22 June 2022. [Online]. Available: https://shopify.dev/changelog/introducing-shopify-cli-3-0. [Accessed 15 September 2022].