

An Nguyen

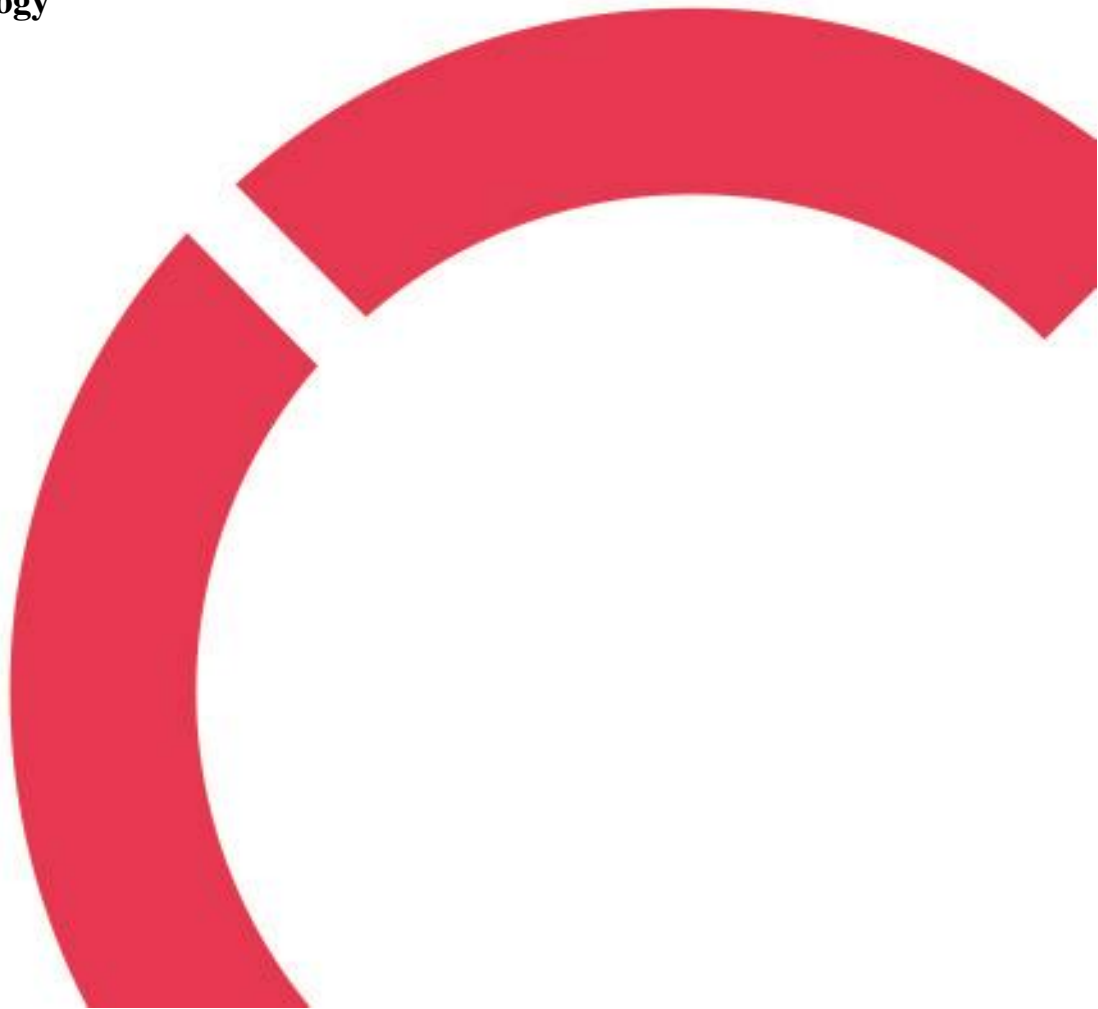
**BUILDING AN E-COMMERCE WEBSITE USING NEXT JS,
MANTINE, AND STRAPI**

Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Information Technology

May 2022



ABSTRACT

Centria University of Applied Sciences	Date May 2022	Author An Nguyen
Degree programme Information Technology		
Name of thesis BUILDING AN E-COMMERCE WEBSITE USING NEXT JS, MANTINE, AND STRAPI		
Instructor Jari Isohanni	Pages 45	
Instructor representing commissioning institution or company Jari Isohanni		
<p>The thesis focused on planning and building an e-commerce website using Jamstack architecture. Due to this architecture, the web page would benefit from performance, security, scalable capability, and maintainability. The framework used was Next Js. Performance of the web page could be archived thanks to the capability of server-side rendered and statically generated pages. The styling of the website was quickly built by using a React components library named Mantine. The images and product descriptions would be managed by using a headless content management system named Strapi.</p> <p>This thesis had three sections. The first one was a literature review of the technologies and framework used in the thesis. The next one was planning actual web page basis functionalities, which included viewing and searching products, adding to the cart, and confirming the order. Before building the application, the UX/UI, databases, models, and API routes are designed and planned. There are some example lines of code about the front-end and back-end setup and implementation in the next section. The final section is the conclusion, which overviews the Next Js framework and further suggestions to improve the project.</p> <p>Consequently, the thesis is a quick introduction to the Next Js and related technologies in building a complicated website such as an e-commerce platform.</p>		

<p>Keywords Jamstack, Next Js, React Js, Mantine, Headless CMS, Strapi, REST API, MongoDB, MySQL, Amazon S3</p>
--

CONCEPT DEFINITIONS

List of Abbreviations

API	Application Programming Interface
CDN	Content delivery network
CMS	Content management system
CNCF	Cloud Native Computing Foundation
CPU	Central processing unit
CSR	Client-side rendering
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated development environment
ISR	Incremental Static Regeneration
JSON	JavaScript Object Notation
NPM	Node Package Manager
ORM	Object relational mapping
RDBMS	Relational database management system
REST	Representational state transfer
SCSS	Syntactically Awesome Stylesheet
SEO	Search engine optimization
SSG	Static site generation
SSR	Server-side rendering
TTFB	Time to First Byte
UI	User interface
URL	Uniform resource locator
UX	User experience
WASM	Web Assembly

ABSTRACT
CONCEPT DEFINITIONS
CONTENTS

1 INTRODUCTION.....	1
2 PROJECT VISION.....	2
2.1 Functionalities	2
2.2 Non-functionalities	2
2.3 Use-case diagram	3
2.4 Activities diagram	4
3 TECHNOLOGY ARCHITECTURE.....	8
3.1 Jamstack	8
3.2 Next Js	10
3.2.1 Image optimization	10
3.2.2 Next Js compiler	11
3.2.3 Multiple pre-render strategies	12
3.2.4 File-system routing and API routing.....	13
3.3 Headless CMS.....	14
3.4 RESTful API.....	16
4 CONTENT.....	19
4.1 Site structure	19
4.2 Copyright and IPR.....	26
4.3 Maintenance	26
4.4 Project directory structure.....	28
5 FRONT-END IMPLEMENTATION.....	31
5.1 “_app” and “_document” component.....	31
5.2 Homepage	32
5.3 Search bar component.....	34
5.4 Use-form hook	36
6 API IMPLEMENTATION	39
6.1 Next Js Application	39
6.2 Strapi Application.....	40
7 CONCLUSION	45
REFERENCES.....	46

FIGURES

FIGURE 1. Use-case diagram.....	3
FIGURE 2. Login activity diagram	4
FIGURE 3. Register activity diagram.....	5
FIGURE 4. User edit information activity diagram.....	5
FIGURE 5: User search item activity diagram	6
FIGURE 6. User view item activity diagram.....	7
FIGURE 7. Site structure	19

TABLES

TABLE 1. API routes examples (Next Js Developers 2022)	14
TABLE 2. Comparison between Headless CMS and Traditional CMS (Wessling 2020)	16
TABLE 3. GraphQL and REST in comparison (AltexSoft 2019).....	17

PICTURES

PICTURE 1. Comparison between traditional and Jamstack websites (Ismail 2022).....	9
PICTURE 2. Jamstack applications heavily rely on APIs (ChecklyHQ Developers 2022).....	9
PICTURE 3. Comparison of download trend between Next Js, Gatsby Js, and Nuxt Js on NPM trends	10
PICTURE 4. Next Js configuration with SWC.....	11
PICTURE 5. “getServerSideProps” function gets “id” from the request, then get the product data and returns props for the rendered page	12
PICTURE 6. “getStaticProps” function gets data from the server and renders the page using the return props.....	13
PICTURE 7. Comparison of monolithic, decoupled, and headless CMS architecture (Strapi developers 2022)	15
PICTURE 8. Send “GET” request to Strapi API and return content.....	18
PICTURE 9. UI/UX prototype - Homepage (top cropped)	20
PICTURE 10. UI/UX prototype - Account Management page (top cropped)	21
PICTURE 11. UI/UX prototype - Shopping cart (top cropped)	22
PICTURE 12. UI/UX prototype - Confirm order page.....	23
PICTURE 13. UI/UX prototype - Product page	24
PICTURE 14. UI/UX prototype - Categories page (top cropped).....	25
PICTURE 15. CSV file contains 9510 lines of the product description.....	26
PICTURE 16. Packages of Strapi application from “package.json”	27
PICTURE 17. Packages of Next Js application from “package.json”	28
PICTURE 18. Project directories.....	29
PICTURE 19. Amazon S3 bucket for product images and Strapi images.....	29
PICTURE 20. Next Js directory.....	30
PICTURE 21. “_document.tsx” with a “link” tag to get Google fonts.....	31
PICTURE 22. “_app.tsx” with “MantineProvider”, “ModalProvider”, “NotificationProvider” and “AuthProvider”.....	32
PICTURE 23. Home page implementation	33
PICTURE 24. Home page styles file	34
PICTURE 26. Show suggestions while typing the “poke” word.....	35
PICTURE 27. Displayed results for the keyword “poke”	36
PICTURE 28. Initialize form object for Login form	37
PICTURE 29. Login form and handle submit function	38
PICTURE 30. API routes in the Next Js application	39
PICTURE 31. “/auth/user/check” API route handles only the “GET” request method	40
PICTURE 32. Example product query request from client-side	41
PICTURE 33. Strapi logs for searching a product by name	41
PICTURE 34. Content-type product - fields and their datatype.....	42
PICTURE 35. Products collections samples.....	43
PICTURE 36. AWS S3 plugin to upload images	43
PICTURE 37. New category route to return the total number of categories in the database	44

1 INTRODUCTION

Information technology is constantly transforming with new concepts and architectures. Online visitors are becoming more critical of their website experiences. Websites with poor user interface, performance, and security have low conversion rates. A third-party tool such as WordPress that creates landing pages is required to meet the business standards of loading speed and search engine optimization friendliness (SEO friendly). It is also a popular platform for small businesses to start their website without worrying about costs. However, this tool restricts designers and developers from implementing their ideas and functionalities. Jamstack appears like an excellent tool for developers to build web pages faster, easier to scale, and more secure. It is worth trying to update with new knowledge in modern web development.

Next Js are associated with the Jamstack methodology for developing modern web pages and applications. It was first introduced early in 2016 and only had the capability of server-side rendering. New features and enhancements have been added in every update, including automatic routing pages and multiple methods of page rendering and data fetching. It gained popularity because it used React Js and a single language to build a full-stack application. Netflix, Uber, Starbucks, and Twitch are a few of the world's most well-known organizations that employ Next Js. It is also one of the fastest-growing React frameworks, which has recently been a hot topic in the web development community (Niżyński 2022). Based on the report of wappalyzer.com, 167000 company websites are empowered by Next Js today.

The thesis focuses on rebuilding an e-commerce web page that can interact with users. The interactions are searching, viewing, adding to the cart, and making orders. The user interface and user experiences are designed and planned in Figma. The e-commerce application is implemented in React Js using the Next Js framework. The content is distributed from a headless content management system named Strapi and is saved in the MySQL database. Product images are kept in Amazon Simple Storage Service (Amazon S3). Data of users and orders are stored in MongoDB, a NoSQL database. All data is accessed through REST API or GraphQL requests.

2 PROJECT VISION

In order to avoid the expansion of the project, the thesis project will limit the functionalities and non-functionalities of the web page. Each smaller header will describe the functionalities and non-functionalities. There are use cases and activities diagrams that will be shown to clarify the logic behind the functions of the site. This project is focused on the point of view of customers shopping on a business website. There are no pages for businesses to sell their products on this platform. Due to the limited time, the project is built and optimized for large screen devices such as desktops and laptops. This project provides a quick guide to building a shopping website using cutting-edge technologies.

2.1 Functionalities

The core function requirements of the web page include the essential functions of a shopping web page, such as user authentication, saving data, viewing, and searching items. It allows a guest to register and authenticate without a password. After the register or authentication step, the system will send a link with the session token via email. Secondly, the system allows registered users to save their information such as name, phone, email, and delivery address in the permanent database. Moreover, registered users can save their favorite items, current shopping cart, and previous orders. Finally, all users can view product information and images, add items to the cart, and confirm the order.

2.2 Non-functionalities

With the purpose of expanding further aspects of the web page, there are several non-functional requirements. First, the web page needs to verify the session token to give back correctly the user information. The session token and login code need to be encrypted using JSON Web Token and secrets. The session token is saved in the cookie and will be expired in seven days, and the login code is used only once. Registered users can delete their accounts and data. Product information is available for anyone to find and view concerning the availability. The image database is stored separately from the product description, which prevents both from failing on the same request and is easier to maintain in future development.

2.3 Use-case diagram

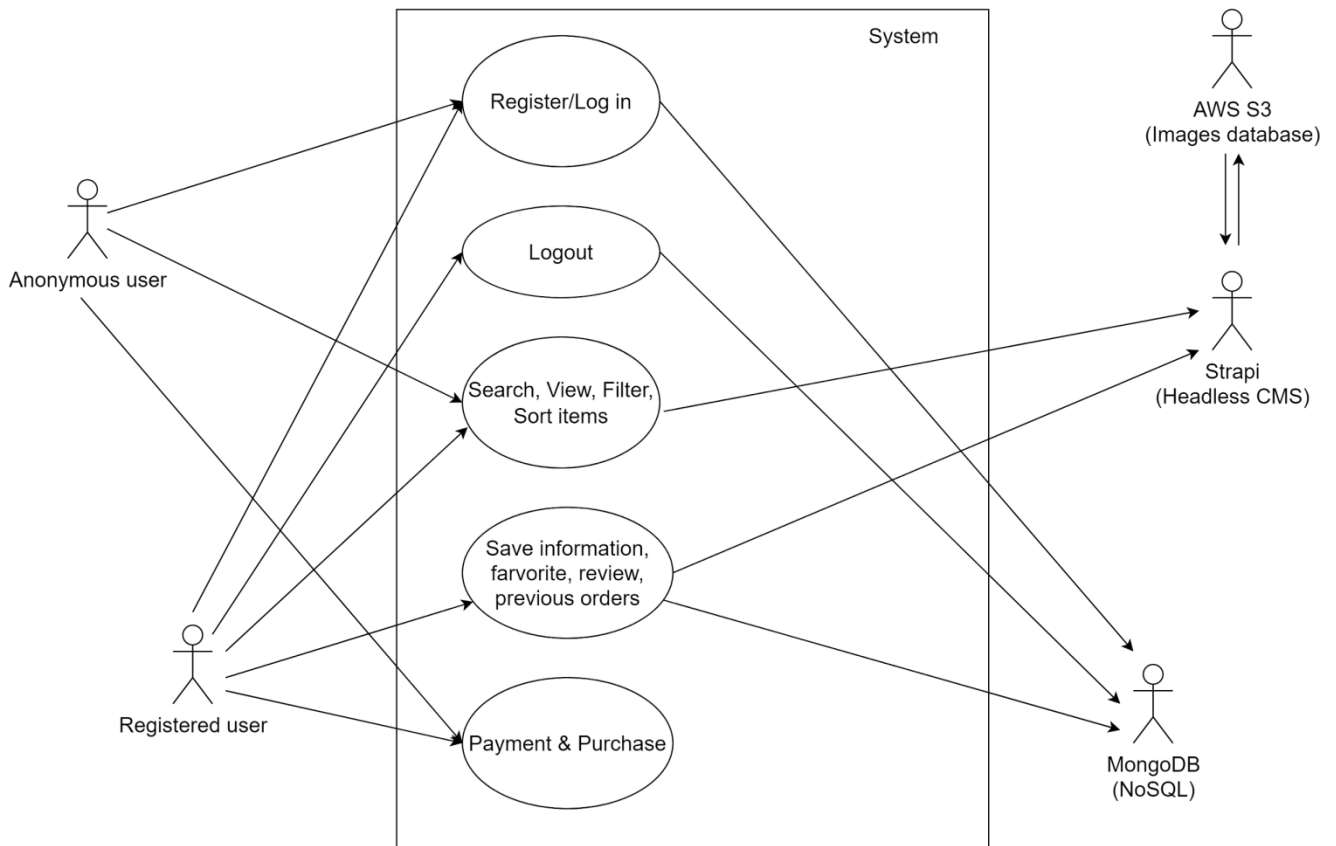


FIGURE 1. Use-case diagram

Following FIGURE 1, anonymous users can interact with the website with allowed actions. They can register a new account to save their basic information such as name, phone number, and delivery address. They can browse and search for items without suggestions in the search panel. The product data is returned from Strapi CMS while the product images are stored in the AWS S3. They can read the product information and reviews and add the item to the cart with a quantity. While browsing categories, they can sort or filter products by prices and review rating. They can go to the confirm page to purchase their order without an account.

Registered users have every permission of an anonymous user according to FIGURE 1. They can save their favorite products, write reviews of purchased products, and save delivery information. The registered users' data is saved in the MongoDB database. When they log in, the browser will save the authorization token in the cookie and use it for further requests. Then, the data from the cloud will be loaded and saved on the website. Users can access to modify their information on the account

management page. When they log out, the session token will be deleted in the header of the request and cloud database.

2.4 Activities diagram

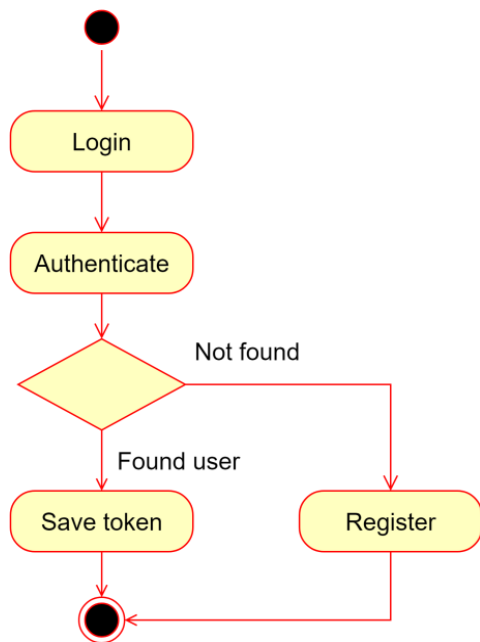


FIGURE 2. Login activity diagram

FIGURE 2 describes the login activity when a user login the system. An authentication request will be sent with the registered email. Then, the posted email will be searched in the MongoDB database. A login code will be generated for only one-time usage if the email is found. There is an email with a login code which will be mailed immediately. The user can use the provided link to log in and save the created session token in the browser. The link is only valid for ten minutes. After that, users need to try to log in again to create a new login code. On the contrary, there is a notification that no data is discovered, and the user can register a new account.

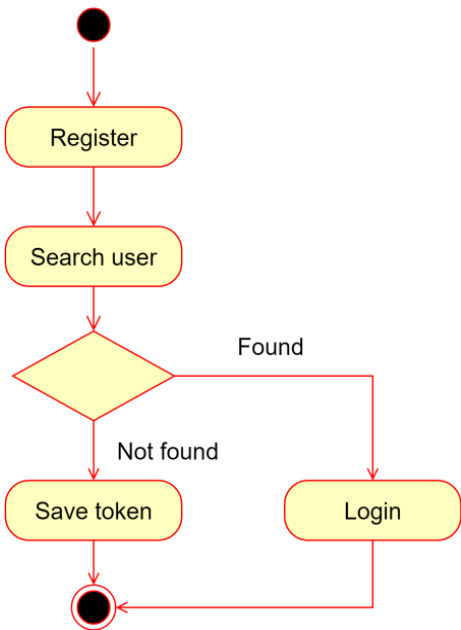


FIGURE 3. Register activity diagram

The system needs to check if a duplicate email exists when new users register, following FIGURE 3. The registering route is available to guests only. Registered users will be redirected to the homepage as long as they try to navigate to this route. Suppose there is an account associated with this email. A notification will be shown to guide the user to log in. If not, there will be a form for the user to input more information immediately or later. The information contains the user name, phone number, and address. Then, the new session token will be generated in the back-end. The session token will be saved in the cookie and the cloud database.

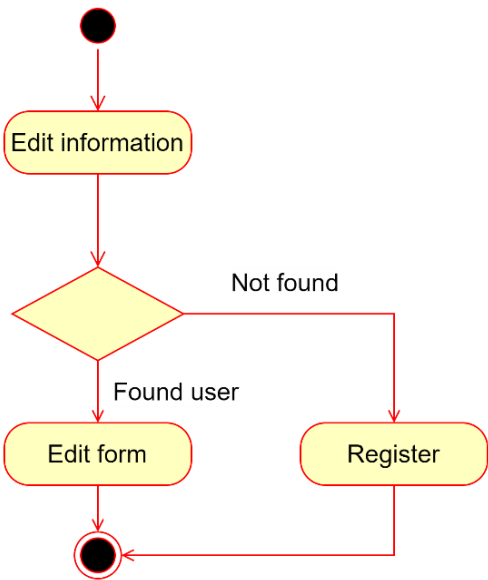


FIGURE 4. User edit information activity diagram

Registered accounts can edit their information on the account management page. After the user login, the user information will be saved browser. Hence, the system does not need to authenticate the user according to FIGURE 4. Users can open the profile page from the menu and footer links. If the user information is available in the context API, the user can edit and save it to the cloud database. Registered users can log out and delete their accounts on this page. Once the logout is sent, the session token is deleted from the cookie. It is the same case as the delete account request in which the account is removed from the MongoDB database. Otherwise, guests see the login and register buttons.

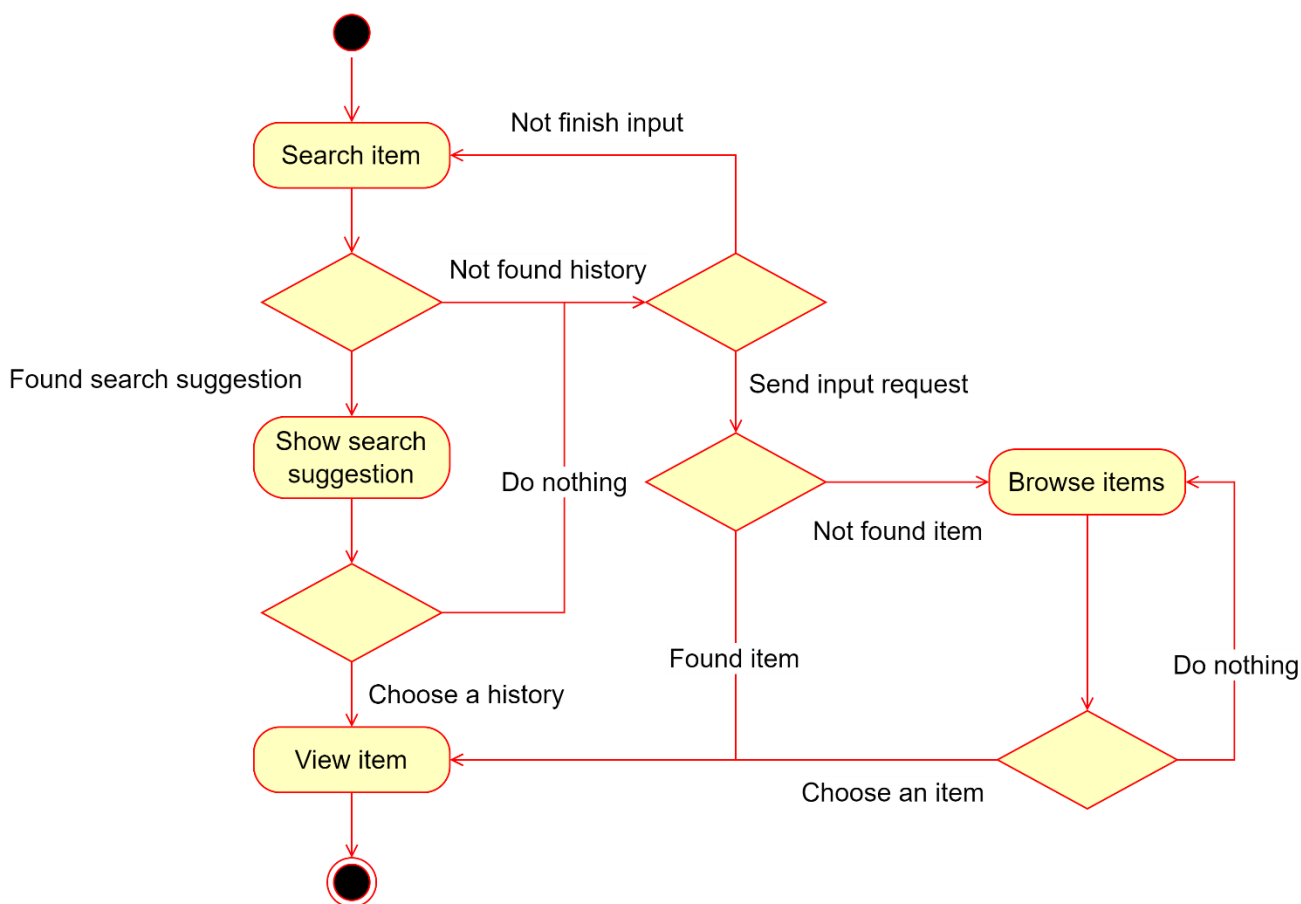


FIGURE 5: User search item activity diagram

FIGURE 5 describes the activity of the system while the user inputs on the search bar. While doing the search function, the search panel will display the product list as suggestions. This feature is available for all users. The product list will be shown after 800 milliseconds after the input keywords received. The request will be sent to Strapi CMS and return a list of products name that contains the keyword. If only one item is found, the user will be redirected to the product page. If many products are detected, the user

will be forwarded to the categories page with a filtered list. The user can view the product by clicking on a specific product. The filtered list is divided into many pages with the purpose of optimizing loading speed. The user can view more by clicking on the load more button at the end of the page.

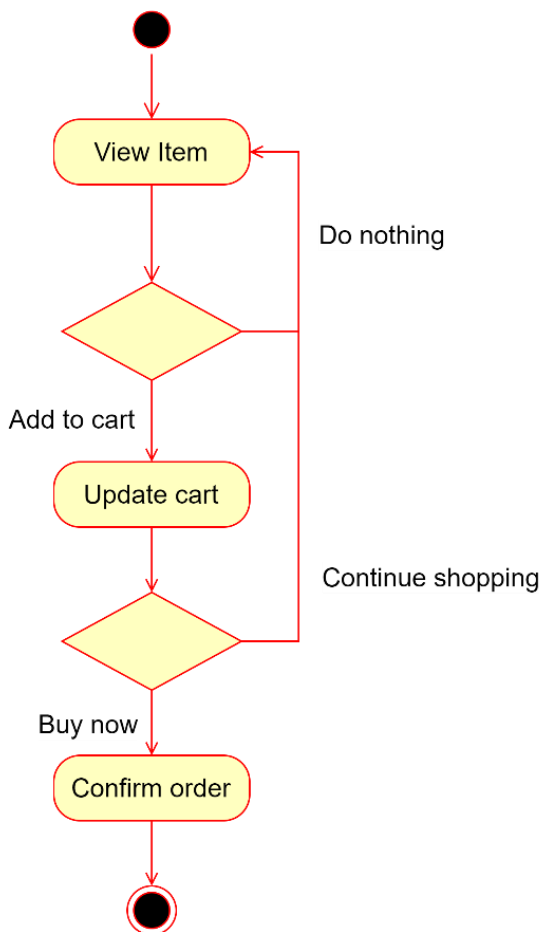


FIGURE 6. User view item activity diagram

Users can add this product to the cart with the desired quantity on the product page. The shopping cart will be updated to save the product and amount according to FIGURE 6. There is a notification at the bottom to inform the action is successful. Suppose that the user clicks on the “Buy now” button, the current shopping cart will be updated, and the user will be redirected to the confirm order page. On the contrary, the user will be on the current page. The current cart is saved in the React State and local storage for all users. Consequently, users still keep the current cart even if they exit the page.

3 TECHNOLOGY ARCHITECTURE

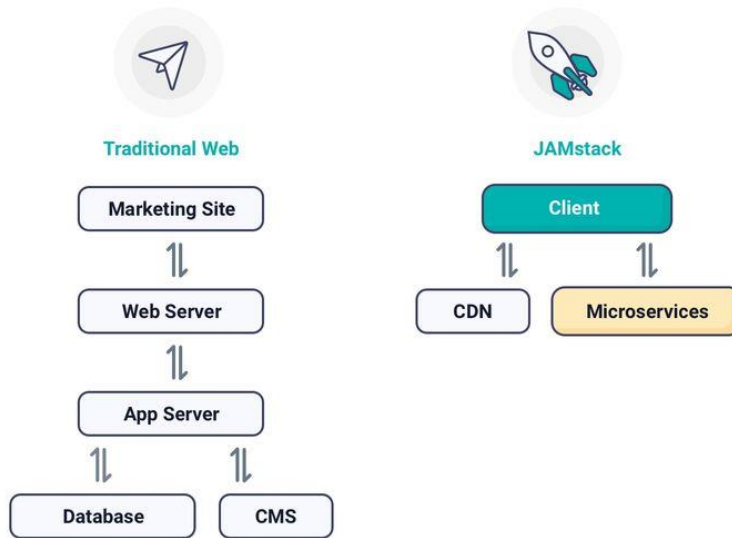
In this section, there are explanations about the technology architectures, frameworks, and libraries used. The description will be Jamstack architecture, Next Js framework, Headless content management system Strapi, and RESTful API. The content declares the advantages of the technologies. Jamstack architecture and Next Js are used to build a modern website with performance, security, and scalable capabilities. Headless CMS such as Strapi opens the freedom of choices and deployment ideas. Finally, REST API is a simple and flexible way to interact with many web services.

3.1 Jamstack

This architecture eliminates several moving elements and systems from the hosting infrastructure, reducing the number of servers and systems that need to be hardened against attack. The front-end is pre-built into optimized static pages and assets during the build process. Serving pre-generated pages and assets allow read-only file attribute, which helps to prevent attack vectors (Alfaro 2021). On the performance aspect, Jamstack sites minimize the need for the server to provide page views at the desired time by generating pages earlier in the build process. It makes the loading speed faster and has a more pleasant user experience and more conversions for the business (Anshu 2021).

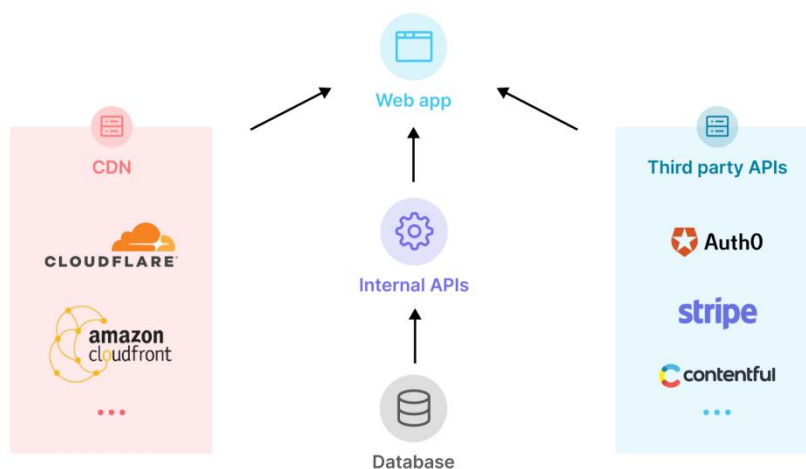
Since there is no sophisticated logic or workflow to determine which assets can be cached and when Jamstack sites can be cached in a content delivery network (CDN). It beats famous architecture when dealing with heavy traffic loading. In addition, the scalable capability of Jamstack sites can be shown when there is no limitation in the infrastructure of servers. It can be hosted on a range of different hosting services, even simple static hosting. (Anshu 2021.)

PICTURE 1 describes the differences between the architecture of traditional and modern websites. The traditional website is built with a monolithic architecture. The website has access to the resources in only one way, from top to bottom. The database and CMS have to go through the application server and web server before it comes to the clients. In reverse, JAMstack architecture opens a flexible accession to resources from the client-side. The database and CMS are distributed through the CDN and microservices. (Ismail 2022.)



PICTURE 1. Comparison between traditional and Jamstack websites (Ismail 2022)

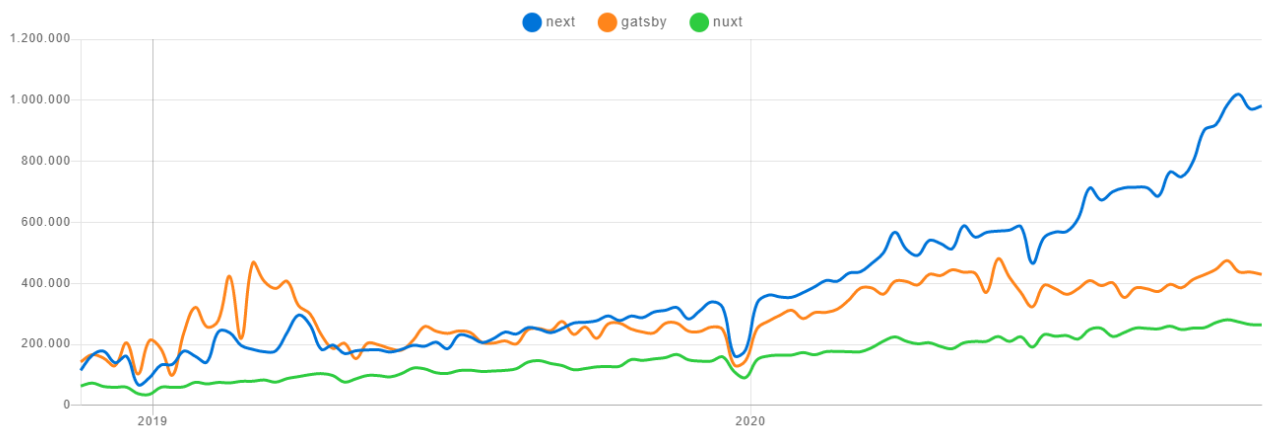
PICTURE 2 is another example of resource delivery in JAMstack architecture. Besides using microservices, the web application can use APIs from internal and third parties (ChecklyHQ Developers 2022). Some popular frameworks using Jamstack architecture can be named Next Js, Gatsby, Hugo, and Jekyll (Anshu 2021). Next Js will be used as a framework for development in this project. This framework uses its API routes for user authentication and order handling. The content of the site will be obtained via a Content Management System named Strapi. Strapi uses API routes to distribute the content to the web application with an authentication key (Strapi Developer 2022).



PICTURE 2. Jamstack applications heavily rely on APIs (ChecklyHQ Developers 2022)

3.2 Next Js

Next Js is a React framework to generate static sites. It is developed and supported by Vercel. Next js can be quickly learned by any developer who knows HTML, CSS, JavaScript, and React. The highlight tools are image optimization and multiple pre-render pages using server-side render (SSR), static-site generation (SSG) or incremental static regeneration (ISR), file-system routing, and API routes. There are other tools such as internationalization, zero-configuration, Typescript support, and built-in CSS and SCSS support (Next Js Developers 2022). PICTURE 3 demonstrates the trend of downloading between Next Js, Gatsby Js, and Nuxt Js from 2019. These frameworks support server-side render.



PICTURE 3. Comparison of download trend between Next Js, Gatsby Js, and Nuxt Js on NPM trends

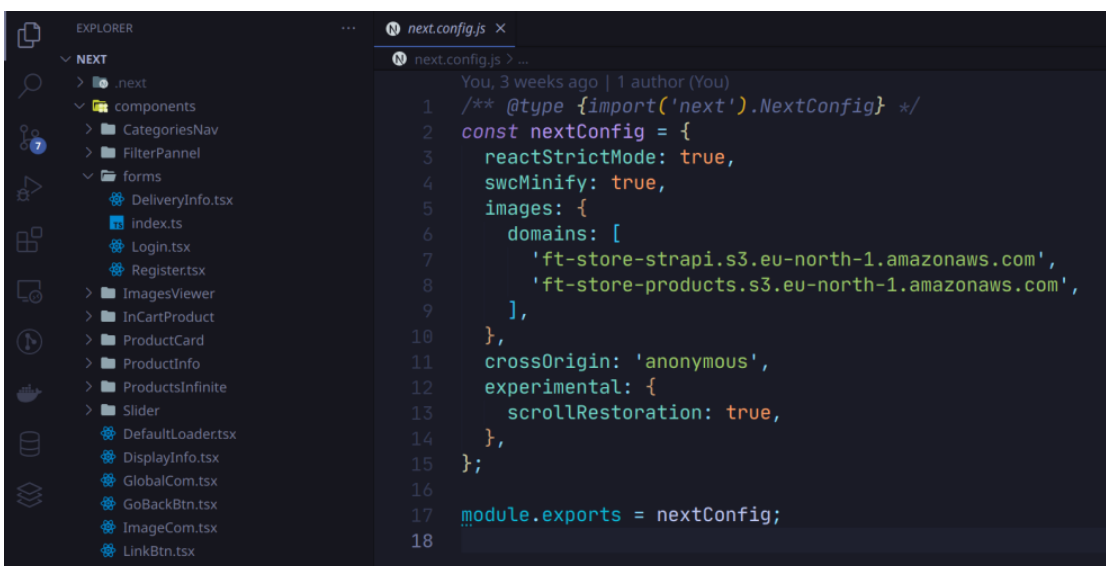
3.2.1 Image optimization

Next Js has a private image component named “next/image.” It comes with several built-in performance and optimization features essential for today’s modern web. It supports both self-hosting and remote sources. Developers can set the priority for the image component in order to optimize the loading performance. The image component serves correctly sized images for each device, using the modern image format. The images will be loaded whenever the user goes into the viewport, or they will be loaded first whenever it comes with a high priority tag. An optional choice is to show a blur placeholder while awaiting the full-size image to load. (Next Js Developers 2022.)

3.2.2 Next Js compiler

The Next Js Compiler, written in Rust with SWC, transforms and minifies JavaScript code for production. For individual files, this substitutes Babel and Terser for minifying output bundles. The Next Js Compiler compiles code 17 times quicker than Babel and has been enabled by default since Next Js version 12. The application will opt out of the Next Js Compiler and continue using Babel if the current configuration has an existing Babel or uses unsupported features. SWC is a Rust-based platform that may be extended to provide the next generation of rapid developer tools. SWC is designed to be extended and can be served for compilation, minification, bundling, and more. It is a function that can be used to execute code alterations (either built-in or custom). Higher-level technologies like Next Js benefit from carrying out the modifications. (Next Js Developers 2022.)

Extensibility, performance, web assembly, and community support are all advantages of SWC. Thanks to its extensibility, it may be used as a Crate inside Next Js, which eliminates the need to fork the library or workaround design limits. By converting to SWC, Next Js was able to accomplish nearly three times quicker in Fast Refresh and nearly five times faster builds in Next Js, with more space for optimization remaining to be done. PICTURE 4 shows the configuration example of “next.config.js” which enables SWC on line 4. The Rust community and its support for WASM are critical for assisting all platforms and spreading Next Js development. (Next Js Developers 2022.)



```

1  /** @type {import('next').NextConfig} */
2  const nextConfig = {
3    reactStrictMode: true,
4    swcMinify: true,
5    images: {
6      domains: [
7        'ft-store-strapi.s3.eu-north-1.amazonaws.com',
8        'ft-store-products.s3.eu-north-1.amazonaws.com',
9      ],
10   },
11   crossOrigin: 'anonymous',
12   experimental: {
13     scrollRestoration: true,
14   },
15 };
17 module.exports = nextConfig;
18

```

PICTURE 4. Next Js configuration with SWC

3.2.3 Multiple pre-render strategies

Next Js allows developers to build a powerful web application using Javascript and Typescript. The concern of developers is not about the back-end infrastructure but is the way to increase the performance (Alam 2022). This framework allows developers to view content on the site with three ways of data fetching. Server-side rendering, static-site generation, and incremental static regeneration are these strategies. Next Js enables developers to create hybrid apps that include both client-side and server-side rendering sites. Only the most recent version of Next Js supports the third option (Next Js Developers 2022).

With SSR, the page will be pre-rendered on each request using data returned from a function named “getServerSideProps.” This function returns a JSON object which is used to build a page. PICTURE 5 presents an example of SSR usage. Developers should use “getServerSideProps” if only it is needed to re-fetch the data on the page. For each page of a product, the server-side will take the “product id” from the path parameter of the request. Then the associated data will be fetched and returned. Time to First Byte (TTFB) will be higher than “getStaticProps” because the server must compute the result on every request. SSR is the best choice for fetching data on the client-side if there is no need to pre-render the data. (Next Js Developers 2022.)

```

89 export const getServerSideProps: GetServerSideProps = async (context) => {
90   const id = context.params?.id as string;
91   const [categoriesList, product] = await axios.all([
92     queries.categories,
93     queries.productById(id),
94   ]);
95
96   return {
97     props: {
98       product: product.data.data[0],
99       categoriesList: categoriesList.data,
100     },
101   };
102 };
103

```

PICTURE 5. “getServerSideProps” function gets “id” from the request, then get the product data and returns props for the rendered page

Using SSG, Next Js will pre-render the page at the build time using the return object from the exported function named “getStaticProps.” PICTURE 6 shows an example of SSG. The product data is gotten

randomly from a function. Then the data is returned as a “prop” before the webpage is built. When the essential data is available at build time, the data comes from a headless CMS, or the data can be publicly cached, the page must be pre-rendered for SEO and incredibly fast. (Next Js Developers 2022.)

```

64
65 export async function getStaticProps() {
66   const index: number = randomProducts() || 0;
67
68   const [products, categories, slider] = await axios.all([
69     queries.productsRandom(index),
70     queries.categories,
71     queries.slider,
72   ]);
73
74   return {
75     props: {
76       products: products.data.data,
77       slider: slider.data.data.attributes.images.data,
78       categoriesList: categories.data,
79     },
80   };
81 }
82

```

PICTURE 6. “getStaticProps” function gets data from the server and renders the page using the return props

Developers can use Next Js to construct or change static pages after they have been built. Incremental Static Regeneration enables static generation on a per-page basis without needing to rebuild the entire site. Developers can preserve the advantages of static while growing to millions of pages with ISR. In order to use ISR, it is needed to pass the “revalidate” parameter to “getStaticProps.” For example, if developers pass “revalidate: 10” in the “getStaticProps.” After the first request and before 10 seconds, all subsequent requests to the page are cached and immediate. The cached page will still be displayed after the 10-second limit has expired. In the background, Next Js allows the page to regenerate. Next Js will invalidate the cache and display the revised page after the page has been generated correctly. The old page will remain unchanged if the background regeneration fails. (Next Js Developers 2022.)

3.2.4 File-system routing and API routing

In Next Js, files placed in directory “pages” are available as routes. The router supports routing files as index route or “/” when a file is named “index.tsx”. For example, “pages/index.tsx” will be routed as “/” on the web browser address bar. Nested routes are supported if there is a nested file structure.

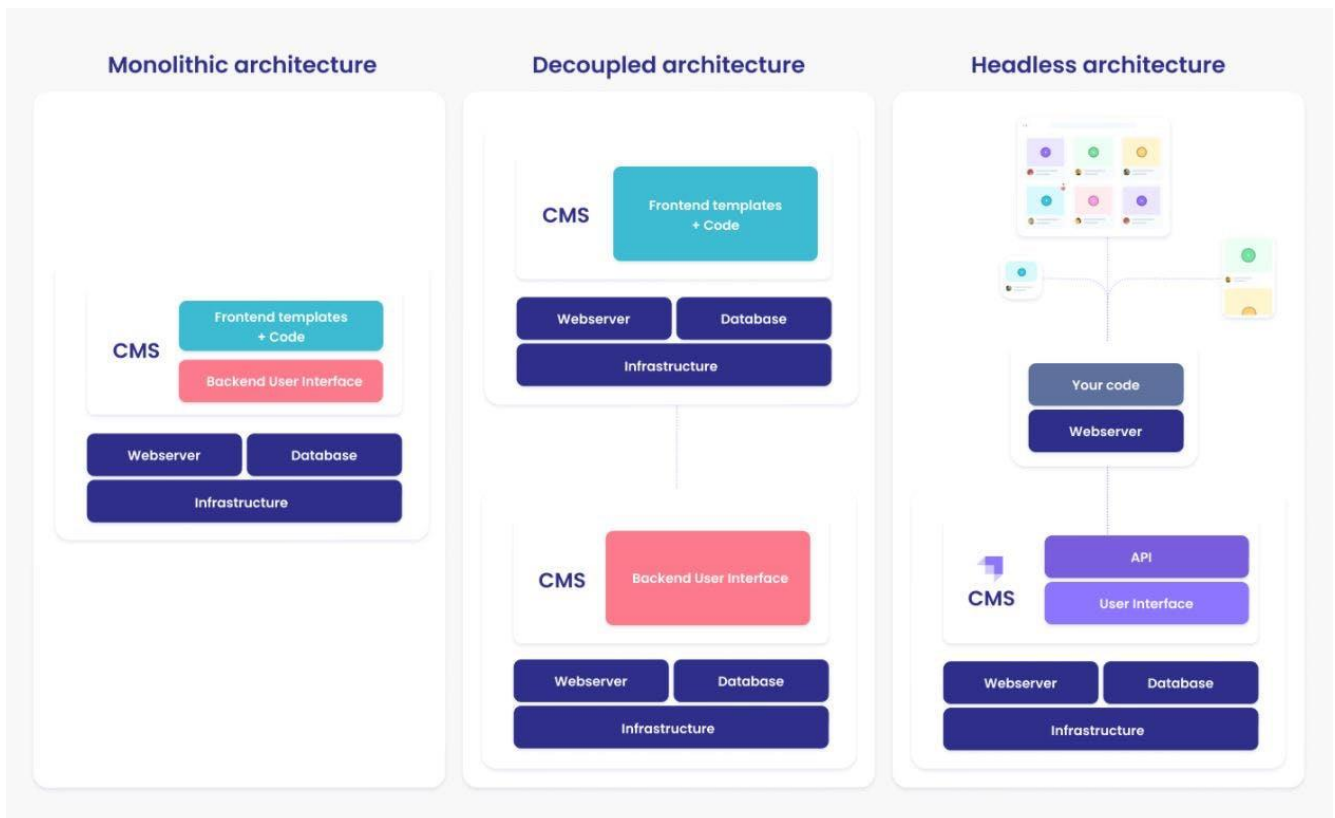
“pages/products/first-id.tsx” will be served as “/products/first-id.” The router also supports dynamic route segments by having bracket syntax in the file name. “pages/blog/[slug].tsx” will be turned into “/blog/:slug” (“/blog/hello-world”). By using bracket and spread parameters, developers can catch all requests to a nested page. For instance, the request to go to page “/products/a/b/c” can be caught in one file with the path being “pages/products/[...slug].tsx”. Similar to file-system routing, API routes are mapped to files inside the folder named “pages/api/.” TABLE 1 shows examples of file paths and API routes for a better understanding of the structure. (Next Js Developers 2022.)

TABLE 1. API routes examples (Next Js Developers 2022)

File path	API route
“pages/api/posts.ts”	/api/posts
“pages/api/posts/[post-id].ts”	/api/posts/post-id
“pages/api/posts/[...slug].ts	/api/posts/a /api/posts/a/b /api/posts/a/b/c

3.3 Headless CMS

A headless content management system (Headless CMS) is any form of back-end content management system in which the content repository’s “body” is separated from the display layer’s “head.” Content stored in a headless CMS is delivered through APIs requests. It helps content be served consistently on many devices (Wessling 2020). PICTURE 7 provides the differences between the three architectures of CMS. On the left-hand side, monolithic architecture distributed all the website’s front-end, back-end, and database. WordPress can be an example of this architecture. The decoupled architecture uses two CMS for front-end and back-end distribution in the middle. Drupal architecture that exposes content to other front-end platforms is referred to as decoupled Drupal (Drupal Developer 2022). Finally, headless architecture is the most flexible architecture that only manages the back-end. The front-end development is freedom of choice.



PICTURE 7. Comparison of monolithic, decoupled, and headless CMS architecture (Strapi developers 2022)

Transitional CMS such as WordPress saved content, photos, HTML, and CSS were all lumped together. This approach made the content disorganized with code and was impossible to recycle. As digital platforms have matured, more flexible solutions have become necessary. Moreover, old CMS organizes content into web page-oriented frameworks, and the same content cannot be adapted to other digital platforms. So, traditional content management systems have fallen behind. (Wessling 2020.)

Even though headless CMS allows developers to select an appropriate display layer for a digital platform, it does not address the fundamental issue of organizing content so that it may be consumed through channels and platforms. A multichannel method for effectively distributing dynamic content over a range of platforms and devices is headless architecture. The content stored in a headless architecture is raw and unformatted, with no front-end system dependence. TABLE 2 shows the general comparison between traditional and headless CMS. (Wessling 2020.)

TABLE 2. Comparison between Headless CMS and Traditional CMS (Wessling 2020)

	Traditional CMS (WordPress)	Headless CMS (Strapi)
Hosting & delivery	In house	In cloud
Development mindset	Project-focused	Product-focused
Content model	Build for single web page	Build blocks for many products
Supported devices	Limited	Limitless
Reach	One-to-one	One-to-many
Workflow	Waterfall	Agile
Updates	Scheduled	Continuous
Back-end system	Monolithic, all-in-one	Microservices, best-in-class
Investment	Large up-front cost	Quick proof of concept
Technical debt	Inherent to the system	Managed

3.4 RESTful API

A RESTful API is an application program interface (API) architecture that uses HTTP requests to access and utilize data. That information can be used with the “GET”, “PUT”, “POST”, and “DELETE” data types, which correspond to reading, modifying, creating, and removing resources. A website’s API is a piece of code that enables two software programs to connect. RESTful API, also known as a RESTful web service or REST API, is based on representational state transfer (REST), a communication architecture style and approach popular in web service development. Other related technologies are often preferred over REST technology. Because REST consumes less bandwidth, making it more appropriate for internet efficiency. Programming languages like JavaScript and Python can also be used to create RESTful APIs. (Gillis 2020.)

Browsers employ REST, which can be considered the internet’s language. Cloud customers use APIs to provide and organize access to online services as cloud usage grows. REST is an obvious choice for developing APIs that enable users to connect to, manage, and interact with cloud services in a dispersed context. For example, Amazon, Google, LinkedIn, and Twitter use RESTful APIs (Gillis 2020). TABLE

3 compares GraphQL and REST, which are two popular data fetching methods from API. There are some pros and cons for each use case.

TABLE 3. GraphQL and REST in comparison (AltexSoft 2019)

	GraphQL	REST
Architecture	Client-driven	Server-driven
Organized in terms of	Schema and type system	Endpoints
Operations	Query (reading data), Mutation (writing data), and Subscription (receiving data in real-time)	Create, Read, Update, and Delete
Data fetching	Specific data on a single query	Fixed data with multiple requests
Community	Growing	Large
Performance	Fast	Take more time
Development speed	Faster	Slower
Learning curve	Difficult	Moderate
Self-documenting	Yes	No
File uploading	No	Yes
Web caching	No (Via libraries built on top)	Yes
Stability	Less error-prone: automatically validation and type checking	A better choice for complex queries
Use cases	Multiple microservices and mobile apps	Single apps and resource-driven apps

REST is beneficial in cloud applications since the calls are stateless. If something goes wrong, stateless components can be easily redeployed, and they can scale to handle load variations. Any request can be addressed to any component instance, and nothing can be saved that the next transaction must remember. As a result, REST is better for online use. Because binding to a service through an API is a question of controlling how the URL is encoded, the RESTful paradigm is also helpful with cloud services. RESTful API architecture will almost certainly become the norm in the future, thanks to cloud computing and microservices. (Gillis 2020.)

PICTURE 8 presents the API routes in the Strapi application. The product route is “/api/products.” Strapi can restrict the accession permission using an authority key or open the accession for the public. As the picture shows, the data is returned using the REST method. All the information about the product is returned in one request. In order to prevent large bandwidth in the communication, Strapi limits the transferring of data which is only ten rows of MySQL database in one request (Strapi Developer 2022). Each product has its identification as the primary key in the MySQL database named “id.” The other description is contained in the “attributes” key on the returned result.

The screenshot shows the Postman interface with a GET request to `localhost:1337/api/products`. The response is a JSON array with two product objects. The first object is expanded to show its details.

```

1  {
2    "data": [
3      {
4        "id": 84134,
5        "attributes": {
6          "name": "DB Longboards CoreFlex Crossbow 41\ Bamboo Fiberglass Longboard Complete",
7          "category": "Sports & Outdoors",
8          "price": 237.68,
9          "about": "Make sure this fits by entering your model number. | RESPONSIVE FLEX: The Crossbow features a bamboo core encased in
          triaxial fiberglass and HD plastic for a responsive flex pattern that's second to none. Pumping & carving have never been so
          satisfying! Flex 2 is recommended for people 120 to 170 pounds. | COREFLEX TECH: CoreFlex construction is water resistant,
          impact resistant, scratch resistant and has a flex like you won't believe. These boards combine fiberglass, epoxy, HD plastic
          and bamboo to create a perfect blend of performance and strength. | INSPIRED BY THE NORTHWEST: Our founding ideal is chasing
          adventure & riding the best boards possible, inspired by the hills, waves, beaches & mountains all around our headquarters in
          the Northwest | BEST IN THE WORLD: DB was founded out of sheer love of longboarding with a mission to create the best custom
          longboards in the world, to do it sustainably, & to treat customers & employees like family | BEYOND COMPARE: Try our
          skateboards & accessories if you've tried similar products by Sector 9, Landyachtz, Arbor, Loaded, Globe, Orangatang, Hawgs,
          Powell-Peralta, Blood Orange, Caliber or Gullwing",
10         "specification": "Shipping Weight: 10.7 pounds (View shipping rates and policies)|ASIN: B07KMV3JK7| #474 in Longboards
          Skateboard",
11         "technicalDetails": "",
12         "productId": "4c69b61db1fc16e7013b43fc926e502d",
13         "createdAt": "2022-04-23T22:44:17.315Z",
14         "updatedAt": "2022-04-23T22:44:17.315Z",
15         "publishedAt": "2022-04-23T22:44:13.822Z",
16         "imageIds": "51wLHdwghfl.jpg|51j3fPQTQkl.jpg|31hKM3cSoSL.jpg|51FsyLRBzwl.jpg",
17         "weight": "10.7 pounds",
18         "avgRating": null,
19         "ratingAmount": null
20       }
21     ],
22     "id": 84135,
23     "attributes": {
24

```

PICTURE 8. Send “GET” request to Strapi API and return content

4 CONTENT

In this section, the site structure will be explained by images and descriptions for each page. As FIGURE 7 shows, the pages include the homepage, where users first land when they enter the website. The account management page is the place to edit user information in case they log in. The shopping cart contains all the products that are added during users on site. The payment terminal or confirmation order page is where to place the order. The product page shows a specific product description. The categories page divided the list of products using multiple filter conditions. Copyright and IPR of the tools and contents will be clarified. There is a brief description of what should be maintained in future usage in the maintenance.

4.1 Site structure

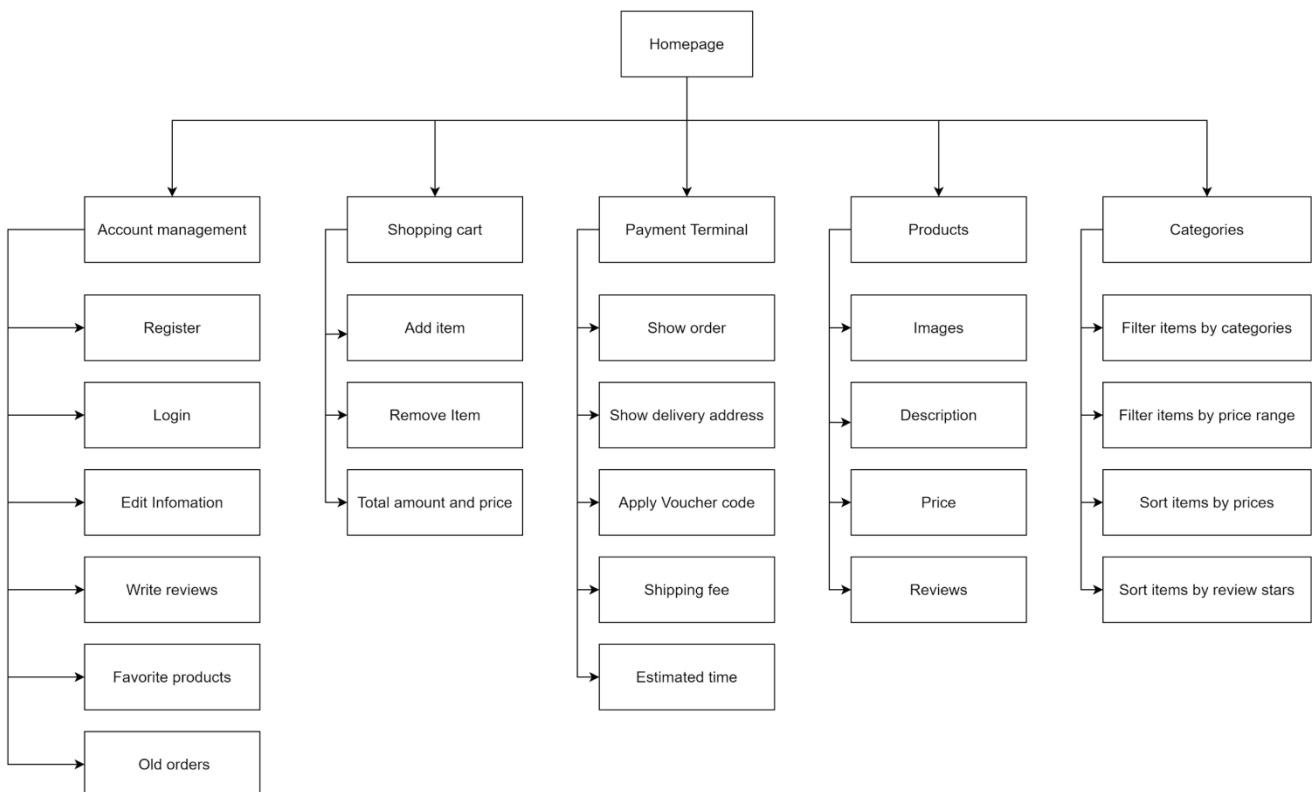
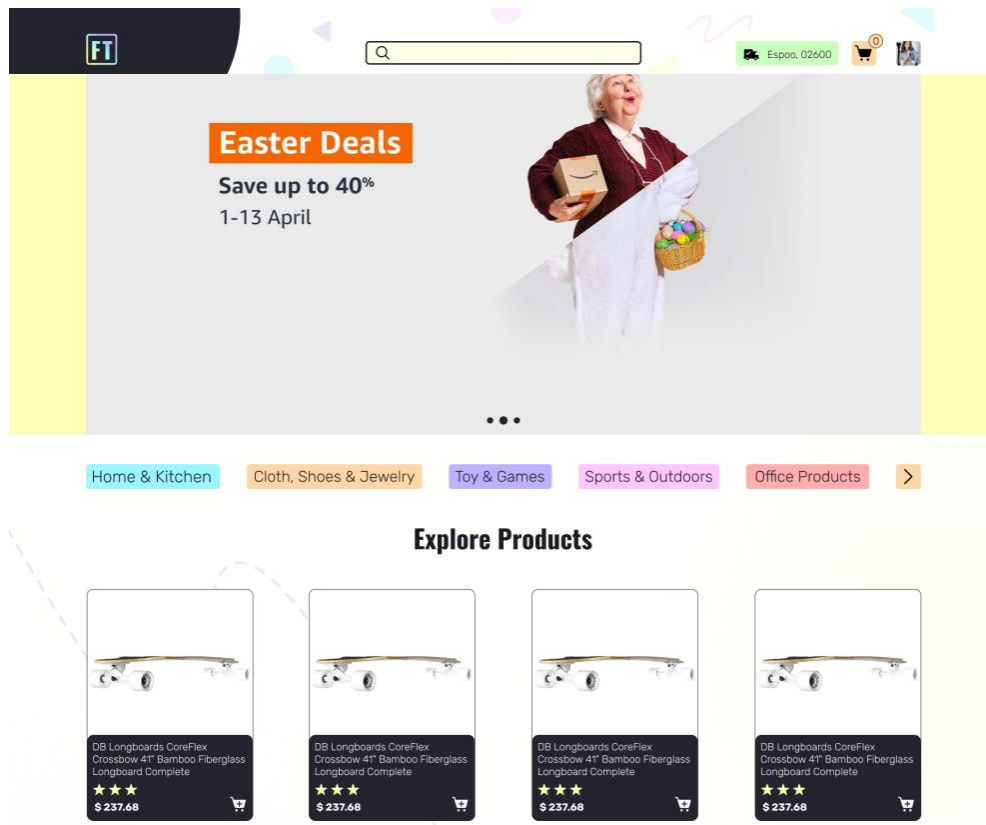


FIGURE 7. Site structure

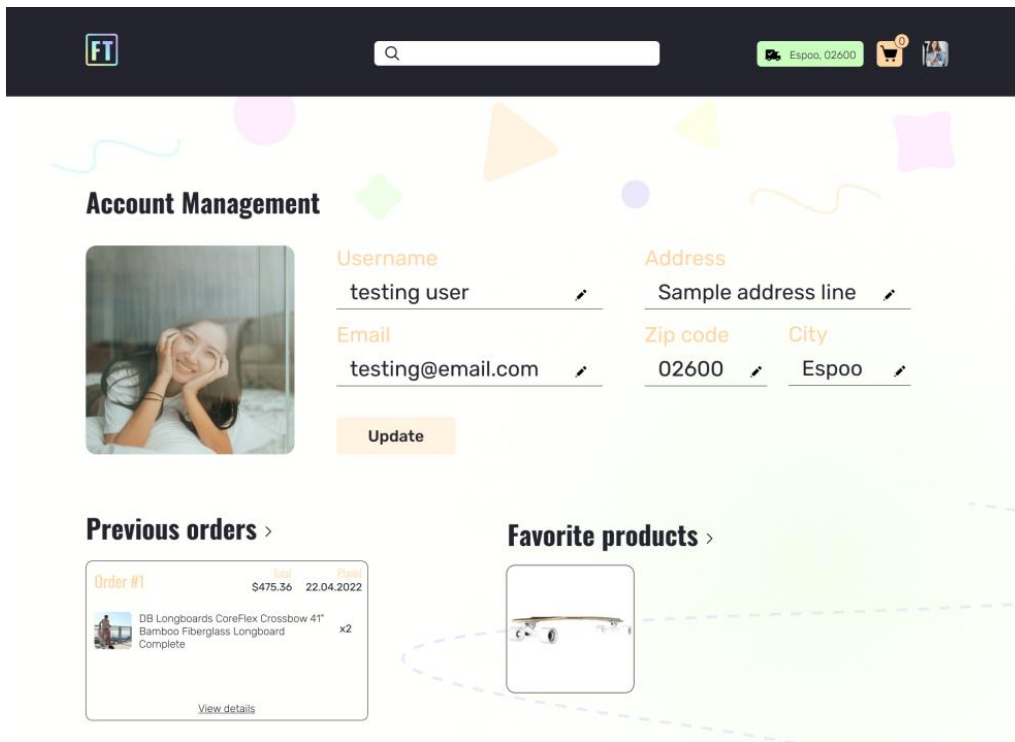
The homepage is the first page the users land on the website, and there is a prompt to introduce the website. PICTURE 9 shows the top of the home page. From the top to the bottom of the page, users can

see the navigation bar, hero images, quick categories links, random products section, previous orders, and footer links. Users can go to a specific category or product by clicking on the link or product card. Moreover, they can search for a specific product by using its name on the top search bar. They can also know how many products they have in the shopping cart. Logged users can change the delivery address in the top navigation. In addition, they can view their previous orders. The home page is used static site generation because the content on the site does not change for all users.



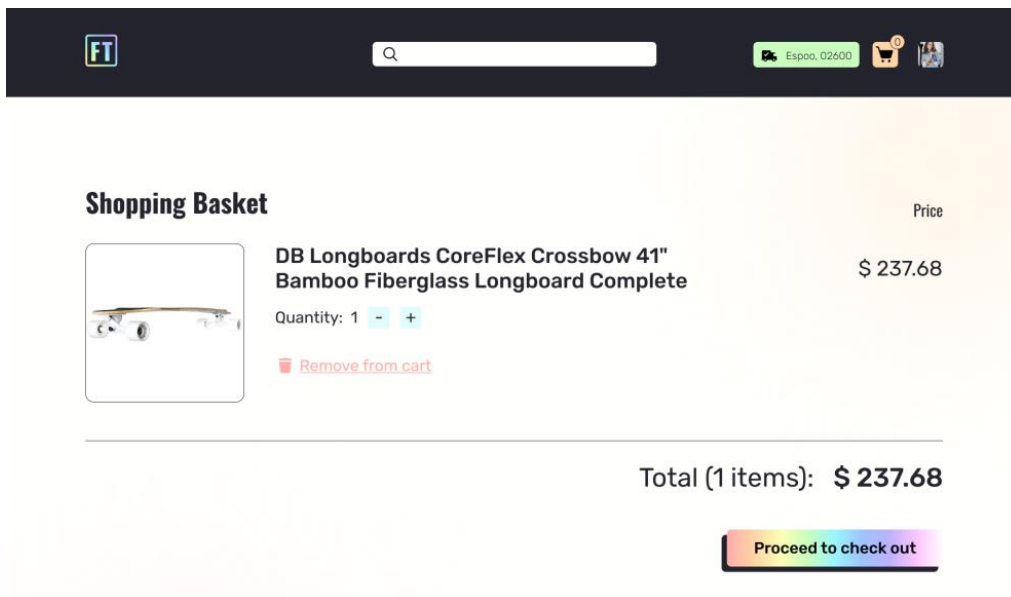
PICTURE 9. UI/UX prototype - Homepage (top cropped)

On the account management page, guests will be able to log in or register an account. PICTURE 10 shows the user interface prototype of this page. This page is different from the homepage in the middle content. The top navigation and the footer links are the same. In the middle content, registered accounts are able to view and edit their basic information, including username and email, address, written reviews, favorite products, and last orders. The updated information will be synced to MongoDB after the logged user confirms. Only the registered users who have bought an item can write a review and give rating stars. Further, they can log out and delete their account by clicking buttons. This page is rendered using static site generation because the content on the page is associated with a logged user identification string or guest. Moreover, it does not change during the user is on the website.



PICTURE 10. UI/UX prototype - Account Management page (top cropped)

Users can add items to their shopping cart and access the basket by clicking on the top navigation bar. The shopping cart will briefly describe the added product, prices, quantity, and total prices, as the PICTURE 11 shows. Users can continue shopping or make a payment by clicking on “Proceed to check out.” They can change the quantity of a product or remove items in the cart. The total price will be updated if there are changes happen. This page is available for all users. Logged user stored the current cart items in the cloud, and the guest stored them in the local storage of the browser. When the page is rendered, it takes the data from the context API, which saves the user data at the initial time.



PICTURE 11. UI/UX prototype - Shopping cart (top cropped)

Users can view all the items in their order, quantity, and final price on the confirm order page like the PICTURE 12. They can add the voucher to get a discount on the final prices. The delivery fee and estimated time will be calculated when there is a delivery address. Otherwise, users need to input a new delivery address via a form before placing the order. The delivery fee is free if the total order is more than 100 euros. There is a fake warehouse address for delivery distance calculation. The distance is calculated from the user and warehouse address via API named "DistanceMatrix.ai." Depending on the distance, the maximum estimated time starts from three days after the order date, and the delivery fee starts from five euros.

<

Review your order

Delivery information

Firstname Lastname

Delivery address

Postage code City

[Change](#)

Voucher & gift cards

[Apply](#)

Estimated delivery date: 06 April

DB Longboards CoreFlex Crossbow 41" Bamboo Fiberglass Longboard Complete

Quantity: 1

Order summary

Items	\$ 237.68
Delivery	\$ 10
Total	\$ 247.68
Voucher	-\$ 0

Order total

\$ 247.68

[Pay and Place order](#)



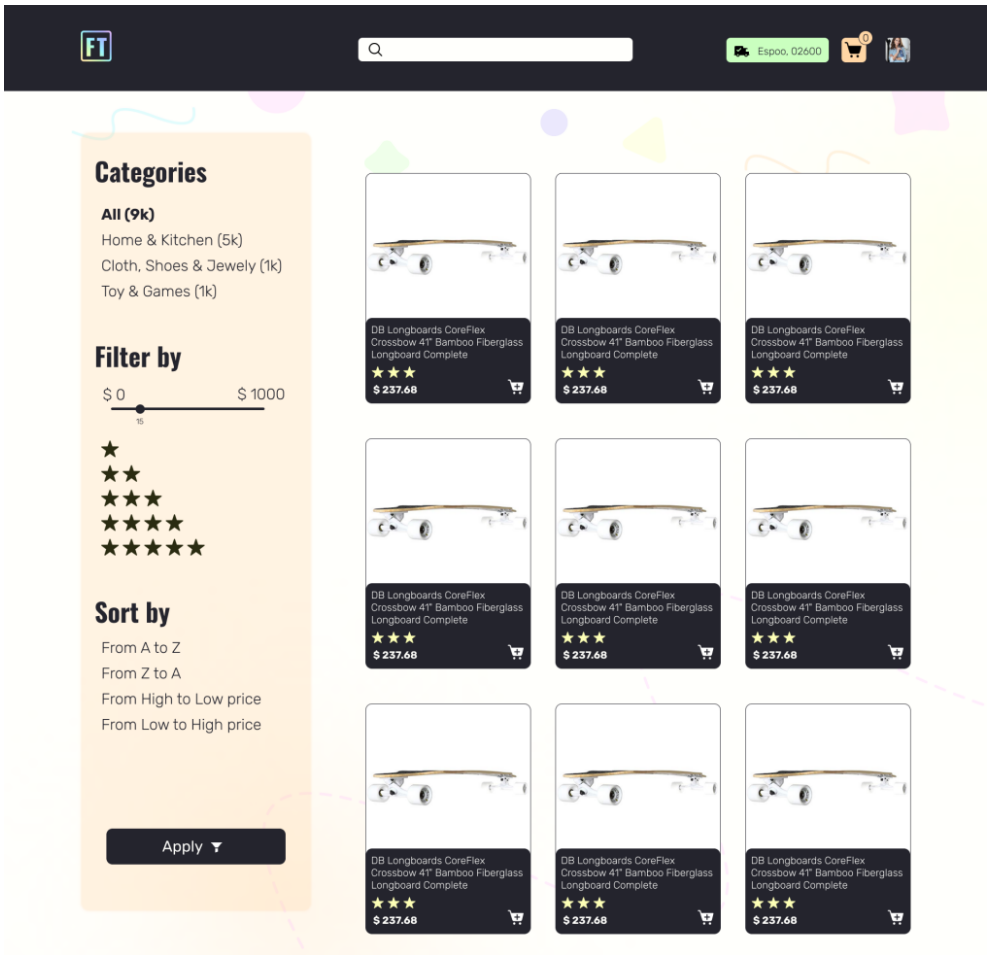
PICTURE 12. UI/UX prototype - Confirm order page

Users can view product images, descriptions, and prices on a specific product page like PICTURE 13. Users can add this item to their shopping cart with the amount they want. Users can add the product to their favorite products. Links to the product category will be displayed if the user wants to discover the related products. The reviews and rating stars will be shown. Customer who bought the product can add their review at the end of the page. The page is built using server-side render due to the changes in the content for each product. When the browser sends the product page request with a corrected identification string, the server will fetch the data from the Strapi CMS and build the page for the browser.



PICTURE 13. UI/UX prototype - Product page

Users can filter and sort items based on their needs on the categories page on the categories page. The items shown will be matched with the filter have been set. The amount of the items are filtered based on the condition. For example, there are more than nine thousand products if the filter is “All” for the category, and it is around five thousand for “Home and Kitchen.” According to the PCITURE14, sort functions are by name or prices of items. The list of items is divided into multiple pages. In case the users reach the end of the page, they can load more by clicking on the button. As long as the user does not reach the end of all pages, there is no message to inform them. The search result will be displayed on this page too. The filters and sort function are done in the Strapi CMS, and the client-side will generate the request parameters in accordance with selected conditions. Then, the created URL will be sent to Strapi to get the list of products.



PICTURE 14. UI/UX prototype - Categories page (top cropped)

4.2 Copyright and IPR

The development software is open-source software licensed to allow commercial use, although there is no commercial use. The Google fonts that are planned to use are licensed under the Open Font License, which allows free use for commercial purposes. Product descriptions and images are public and can be downloaded from the data.world (<https://data.world/promptcloud/amazon-product-dataset-2020>). It has a public domain license so that there is no violence to legal for the thesis usage. PICTURE 15 shows the shortage of products in the CSV file, including uniques identification string, name, category, selling price, general information, and image links.

1	A	B	C	D	E	F	G	H	I	J
1	Uniq Id	Product Name	Category	Selling Price	About Product	Product Specification	Technical Details	Shipping Weight	Image	Product Url
2	4c69611b11c16e7013b43692669202	DB Longboards CoreFlex Crossbow 41" Bamboo Fiberglass Longboard Complete	Sports & Outdoors Outdoor Recreation Skates, Skateboards & Scooters Skateboarding Standard Skateboards & Longboards Longboards	\$237.68	Make sure this fits by entering your model number. RESPONSIVE FLEX: The Crossbow features a bamboo core encased in triaxial fiberglass and HD plastic for a responsive flex pattern that's second to none. Pumping & carving have never been so satisfying! Plan 2 is recommended for people 120 to 170 pounds. COREFLEX TECH: CoreFlex construction is water resistant, impact resistant, scratch resistant and has a flex like you won't believe. These boards combine fiberglass, epoxy, HD plastic and bamboo to create a perfect blend of performance and strength. INSPIRED BY THE NORTHWEST: Our founding ideal is chasing adventure & riding the best boards possible, inspired by the hills, waves, beaches & mountains all around our headquarters in the Northwest BEST IN THE WORLD: DB was founded out of sheer love of longboarding with a mission to create the best custom longboards in the world, to do it sustainably, & to treat customers & employees like family BEYOND COMPARE: Try our skateboards & accessories if you've tried similar products by Sector 9, Landyachtz, Arbor, Loaded, Globe, Orangatang, Hivwo, Powell-Peralta, Blood Orange, Caliber or Gullwing	Shipping Weight: 10.7 pounds (View shipping rates and policies) ASIN: B07KMMVJK7 #414 Longboards Skateboard		10.7 pounds	https://images-na.ssl-images-amazon.com/images/I/51X3PQ10BL.jpg https://images-na.ssl-images-amazon.com/images/I/31KM3c5oSL.jpg https://images-na.ssl-images-amazon.com/images/I/51WHwag7L.jpg https://images-na.ssl-images-amazon.com/images/G011x-locale/commontransparent-pixel.jpg	https://www.amazon.com/DB-Lo-Longboards-Crossbow-Fiberglass-Longboards/dp/B07KMMVJK7
3	66499bbe90435a269e9f78e5957	Electronic Snap Circuits Mini Kits Classpack, FM Radio, Motion Detector, Music Box (Set of 5)	Toys & Games Learning & Education Science Kits & Toys	\$99.95	Make sure this fits by entering your model number. Snap circuits mini kits classpack provides basic electronic circuitry activities for students in grades 2-6 Includes 5 separate mini building kits - an FM radio, a motion detector, music box, space battle sound effects, and a flying saucer Each kit includes separate components and instructions to build Each component represents one function in a circuit; components snap together to create working models of everyday electronic devices Activity guide provides additional projects to teach students how circuitry works	Product Dimensions: 14.7 x 11.1 x 10.2 inches 4.06 pounds (Shipping Weight: 4 pounds (View shipping rates and policies) Domestic Shipping: Item can be shipped within U.S. International Shipping: This item can be shipped to select countries outside of the U.S. Learn More ASIN: B00A8KDASJ Item model number: 55234 #3032 in Science Kits & Toys	The snap circuits mini kits classpack provides basic electric circuitry information for students in grades 2-6. This classpack includes 5 snap-together building kits. Components snap together to create working models of everyday electronic devices. Kits included are an FM radio, a motion detector, a music box, space battle sound effects, and a flying saucer. Each mini kit comes with individual components, and an activity guide which includes instructions and additional project ideas. Each primary-colored component represents one function in a circuit. Activity kits are used by teachers and students in classroom and homeschool settings for educational and research applications in science, math, and for a variety of additional disciplines. Science education products and manipulatives incorporate applied math and science principles into classroom or homeschool projects. Teachers in pre-K, elementary, and secondary classrooms use science education kits, manipulatives, and products alongside science, technology, engineering, and math (STEM) curriculum to demonstrate STEM concepts and real-world applications through hands-on activities. Science education projects include a broad range of activities, such as practical experiments in engineering, aeronautics, robotics, chemistry, physics, biology, and geology.	4 pounds	https://images-na.ssl-images-amazon.com/images/I/51M0UJJKL.jpg https://images-na.ssl-images-amazon.com/images/I/5166SD80X.jpg https://images-na.ssl-images-amazon.com/images/I/61c51VnAL.jpg https://images-na.ssl-images-amazon.com/images/I/61N5AUJyCt.jpg https://images-na.ssl-images-amazon.com/images/I/51OMAD5tL.jpg https://images-na.ssl-images-amazon.com/images/G011x-locale/commontransparent-pixel.jpg	https://www.amazon.com/Electronic-Circuitry-Classpack-Motion-Detector/dp/B00A8KDASJ
4	2c55cae269aebf503839484b0d7d6911a	3Doodler Create Flexy 3D Printing Filament Refill Bundle (3X Pack, Over 1000' of Extruded Plastic) - Innovate	Toys & Games Arts & Crafts Craft Kits	\$34.99	Make sure this fits by entering your model number. Smooth 3D drawing experienced the best 3D drawing experience by only using 3Doodler Create Plastics with 3Doodler Create+ and create 3D Printing pen. [Safe to use] the 3Doodler Create Plastics, conforms to the health requirements of ASTM-D-4236 & require no additional labeling in accordance with the US Consumer Product Safety Commission's Regulations as mandated by Labeling of Hazardous Art Materials Act (LHAMA). [3Doodler very own type of plastic] the FLEXY plastic takes creativity to new levels! You can make flexible fun 3D creations! [Environmentally friendly] 3Doodler Create Plastic is made of corn and is 100% compostable! [125 strands of drawing fun] this bundle includes 5 refill filament packs, that's a total of 1043 ft. of 3D drawing and doodling fun! [The 3Doodler app] get an interactive experience! This is needed with doodling	ProductDimensions:10.3x3.4x0.8 inches Weight:12.8ounces ShippingWeight:12.8ounces Viewshipping ratesandpolicies ASIN:B0703p747F Manufacturerrecommended age:14yearsandup	show up to 2 reviews by default! No longer are you bound by the rigid constraints of hard plastic! Our FLEXY one you can now squeeze, stretch, and twist your creations providing a truly dynamic Doodling experience. Do you want to take your creativity to new levels? Explore the wide variety of FLEXY plastic refill colors for your 3Doodler Create 3D pen! Flexy Plastics are compatible with the 3Doodler V1, 2.0, and create 3D printing pens. Available in single & mixed color pack containing 25 strands each, and single color tubes containing 100 strands. 12.8	12.8 ounces	https://images-na.ssl-images-amazon.com/images/I/513eBC8PpL.jpg https://images-na.ssl-images-amazon.com/images/G011x-locale/commontransparent-pixel.jpg	https://www.amazon.com/3Doodler-Plastic-Innovate-Filament-Refill-Bundle/dp/B0703p747F

PICTURE 15. CSV file contains 9510 lines of the product description.

4.3 Maintenance

The application does not need any maintenance since using cloud services for hosting the back-end and the database means that these parts of the software are maintained automatically. The MongoDB database also provides a human-friendly dashboard for configuration and manual editing if that need ever arises (MongoDB Developer 2022). The most likely reason to do maintenance on the application is to patch security issues that arise in NPM packages from time to time. These issues can often be fixed

by updating the package versions to newer ones (Galle 2022). PICTURE 16 and 17 show the NPM packages used during the development of Strapi and Next Js applications.



```
package.json x
package.json > ...
You, 3 weeks ago | 1 author (You)
1  {
2  "name": "strapi",
3  "private": true,
4  "version": "0.1.0",
5  "description": "A Strapi application",
6  "scripts": {
7    "develop": "strapi develop",
8    "start": "strapi start",
9    "build": "strapi build",
10   "strapi": "strapi"
11  },
12  "dependencies": {
13    "@strapi/plugin-i18n": "4.1.8",
14    "@strapi/plugin-users-permissions": "4.1.8",
15    "@strapi/provider-upload-aws-s3": "^4.1.8",
16    "@strapi/strapi": "4.1.8",
17    "dotenv": "^16.0.0",
18    "strapi-plugin-placeholder": "^4.1.7"
19  },
20  "author": {
21    "name": "A Strapi developer"
22  },
23  "strapi": {
24    "uuid": "ec33ee41-41a5-4288-8775-61d441a4e377"
25  },
26  "engines": {
27    "node": "≥12.x.x ≤16.x.x",
28    "npm": "≥6.0.0"
29  },
30  "license": "MIT",
31  "devDependencies": {
32    "mysql": "^2.18.1"
33  }
34 }
35
```

PICTURE 16. Packages of Strapi application from “package.json”


```

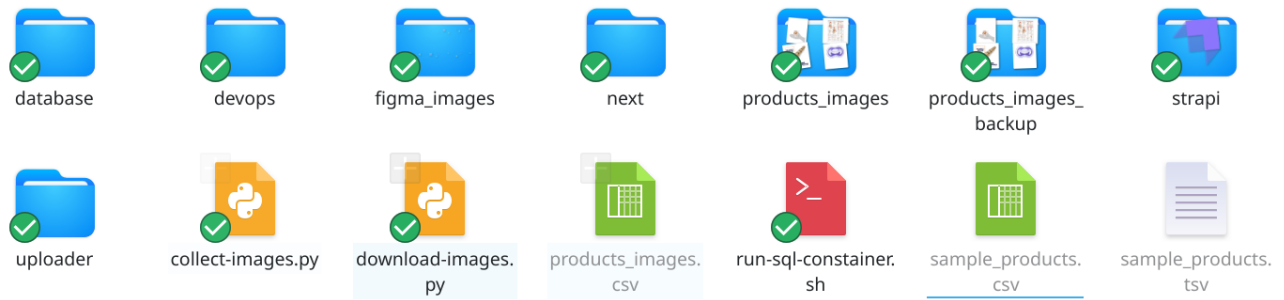
1  {
2    "name": "next",
3    "version": "0.1.0",
4    "private": true,
5    "scripts": {
6      "dev": "next dev",
7      "build": "next build",
8      "start": "next start",
9      "lint": "next lint"
10   },
11   "dependencies": {
12     "@mantine/core": "^4.1.5",
13     "@mantine/form": "^4.1.5",
14     "@mantine/hooks": "^4.1.5",
15     "@mantine/modals": "^4.1.5",
16     "@mantine/next": "^4.1.4",
17     "@mantine/notifications": "^4.1.5",
18     "@sendgrid/mail": "^7.6.2",
19     "@splidejs/react-splide": "^0.7.5",
20     "axios": "^0.26.1",
21     "crypto-random-string": "^4.0.0",
22     "js-base64": "^3.7.2",
23     "jsonwebtoken": "^8.5.1",
24     "lodash": "^4.17.21",
25     "mongoose": "^6.3.0",
26     "next": "^12.1.5",
27     "qs": "^6.10.3",
28     "react": "18.0.0",
29     "react-dom": "18.0.0",
30     "react-icons": "^4.3.1",
31     "swr": "^1.3.0",
32     "yup": "^0.32.11"
33   },
34   "devDependencies": {
35     "@types/jsonwebtoken": "^8.5.8",
36     "@types/lodash": "^4.14.182",
37     "@types/node": "17.0.25",
38     "@types/qs": "^6.9.7",
39     "@types/react": "18.0.5",
40     "eslint_d": "^12.0.0",
41     "eslint-config-next": "12.1.5",
42     "typescript": "4.6.3"
43   }
44 }
45

```

PICTURE 17. Packages of Next Js application from “package.json”

4.4 Project directory structure

The project directory contains eight folders, three file scripts, and three data files like PICTURE 18. They are databases of MySQL container, DevOps files for Kubernetes and deployment, Figma images for user interface design, Next Js application, product images, and Strapi application. Next is the uploader, which contains a small application to transfer the data from the file to the MySQL database. There are scripts written in Python and Bash language to support the progress. The rest files are the original data files.



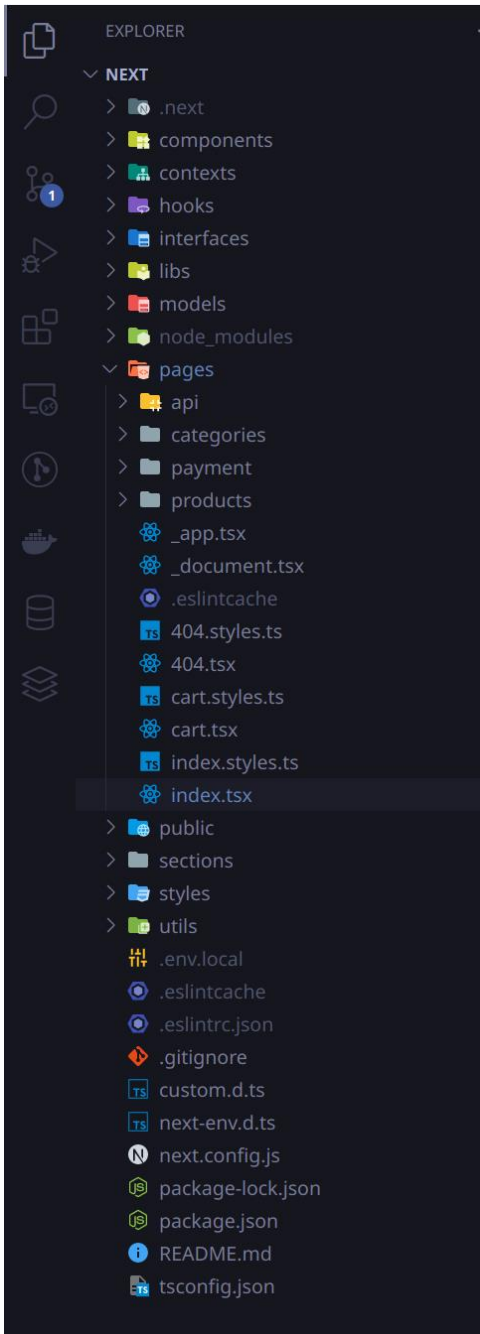
PICTURE 18. Project directories

The image URL would be extracted from the “sample_products.csv” file. Then, the script downloaded all images to the local directory. The images directory is synced directly to the Amazon S3 storage via the command line. PICTURE 19 shows two AWS S3 buckets used to contain product images and other images such as user avatars and public images. Other product information is read and saved in MySQL database via Strapi API and uploader application. In the development progress, MySQL runs in the container. It did not have a persistent memory for production on Kubernetes. The database was extracted to the “database” folder and will be back up to Kubernetes in the production stage.

Name	AWS Region	Access	Creation date
<input type="radio"/> ft-store-products	EU (Stockholm) eu-north-1	Public	April 16, 2022, 20:48:25 (UTC+03:00)
<input type="radio"/> ft-store-strapi	EU (Stockholm) eu-north-1	Objects can be public	April 16, 2022, 23:37:05 (UTC+03:00)

PICTURE 19. Amazon S3 bucket for product images and Strapi images

There are several folders in the “next” directory to keep the files tidy. From the top to the bottom of PICTURE 20, “.next” is the build production of the application. All the reusable components are kept in the “components” directory. Minor components are forms, sliders, link button, and filter panel. These components are imported to pages or sections. “Contexts” is saved React context used to share data globally between children components. “Hooks” contains the custom React hook. “Interfaces” is the description of Javascript objects in Typescript. “Libs” and “utils” are the small and reusable functions written in Typescript for both the front-end and back-end. “Pages” contains routes of pages and API. If users access “/”, the page “index.tsx” will be shown. If users go to unknown pages, page “404.tsx” will be displayed. The “public” contains such logos and “site.webmanifest”. “Sections” are the separated section components like the header navigation bar and footer. “Styles” are the global styling configurations.



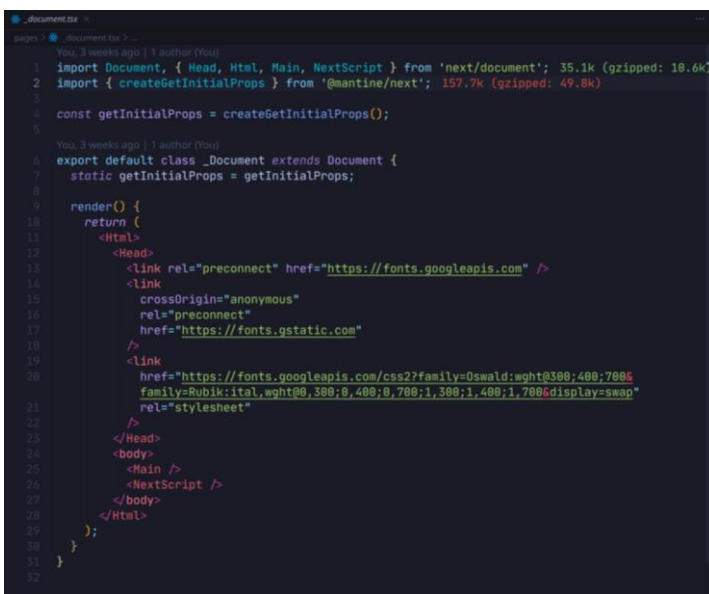
PICTURE 20. Next Js directory

5 FRONT-END IMPLEMENTATION

The previous sections have general views on the technologies and structure of the project. Building the web page will start with designing the user interface and the user experiences on Figma. The components and page implementation will be described. One page contains several smaller components in order to control the user interface functionalities. Due to the massive amount of components and pages, these sections will explain the “_app” and “_document” components, homepage, search bar component, and use-form hook.

5.1 “_app” and “_document” component

The “_app” and “_document” component is used to initialize the page in the Next Js project. They are in the “pages” folder. After project creation using “npx create-next-app@latest”, there is no file named “_document.tsx”. The developer can create this file if the project has custom fonts from Google Fonts like PICTURE 21. Additionally, Mantine supports server-side rendering, and it requires adding “MantineProvider” in the “_app” component and “createGetInitialProps” in the “_document” component. It has a global style component named “Global” to manage universal HTML tags such as “*”, “body”, “ul”, and “li” ... (Mantine Developer 2022). PICTURE 22 shows the page components are covered with “MantineProvider”, “ModalProvider”, and “NotificationProvider” from the Mantine library.



```

1 import Document, { Head, Html, Main, NextScript } from 'next/document'; 35.1k (gzipped: 10.6k)
2 import { createGetInitialProps } from '@mantine/next'; 157.7k (gzipped: 49.0k)
3
4 const getInitialProps = createGetInitialProps();
5
6 export default class _Document extends Document {
7   static getInitialProps = getInitialProps;
8
9   render() {
10    return (
11      <Html>
12        <Head>
13          <link rel="preconnect" href="https://fonts.googleapis.com" />
14          <link
15            crossOrigin="anonymous"
16            rel="preconnect"
17            href="https://fonts.gstatic.com"
18          />
19          <link
20            href="https://fonts.googleapis.com/css2?family=Oswald:wght@300;400;700&
21            family=Rubik:ital,wght@0,300;0,400;0,700;1,300;1,400;1,700&display=swap"
22            rel="stylesheet"
23          />
24        </Head>
25        <body>
26          <Main />
27          <NextScript />
28        </body>
29      </Html>
30    );
31  }
32 }

```

PICTURE 21. “_document.tsx” with a “link” tag to get Google fonts

```

5 import { ModalProvider } from '@mantine/modals'; 5.1k (gzipped: 1.8k)
6 import override from '../styles/globals';
7 import GlobalCom from '../components/GlobalCom';
8 import UserProvider from '../contexts/users';
9
10 function MyApp({ Component, pageProps }: AppProps) {
11   return (
12     <Head>
13       <title>FT Store</title>
14       <meta
15         You, 3 weeks ago • feat: header nav component ...
16         name="viewport"
17         content="minimum-scale=1, initial-scale=1, width=device-width"
18       />
19       <link rel="apple-touch-icon" sizes="192x192" href="/icon-192x192.png" />
20       <link
21         rel="icon"
22         type="image/png"
23         sizes="192x192"
24         href="/icon-192x192.png"
25       />
26       <link
27         rel="icon"
28         type="image/png"
29         sizes="256x256"
30         href="/icon-256x256.png"
31       />
32       <link
33         rel="icon"
34         type="image/png"
35         sizes="384x384"
36         href="/icon-384x384.png"
37       />
38       <link
39         rel="icon"
40         type="image/png"
41         sizes="512x512"
42         href="/icon-512x512.png"
43       />
44       <link rel="manifest" href="/site.webmanifest" />
45     </Head>
46     <MantineProvider withNormalizeCSS withGlobalStyles theme={override}>
47       <ModalProvider>
48         <NotificationsProvider>
49           <UserProvider>
50             <GlobalCom />
51             <Component {...pageProps} />
52           </UserProvider>
53         </NotificationsProvider>
54       </ModalProvider>
55     </MantineProvider>
56   );
57 }

```

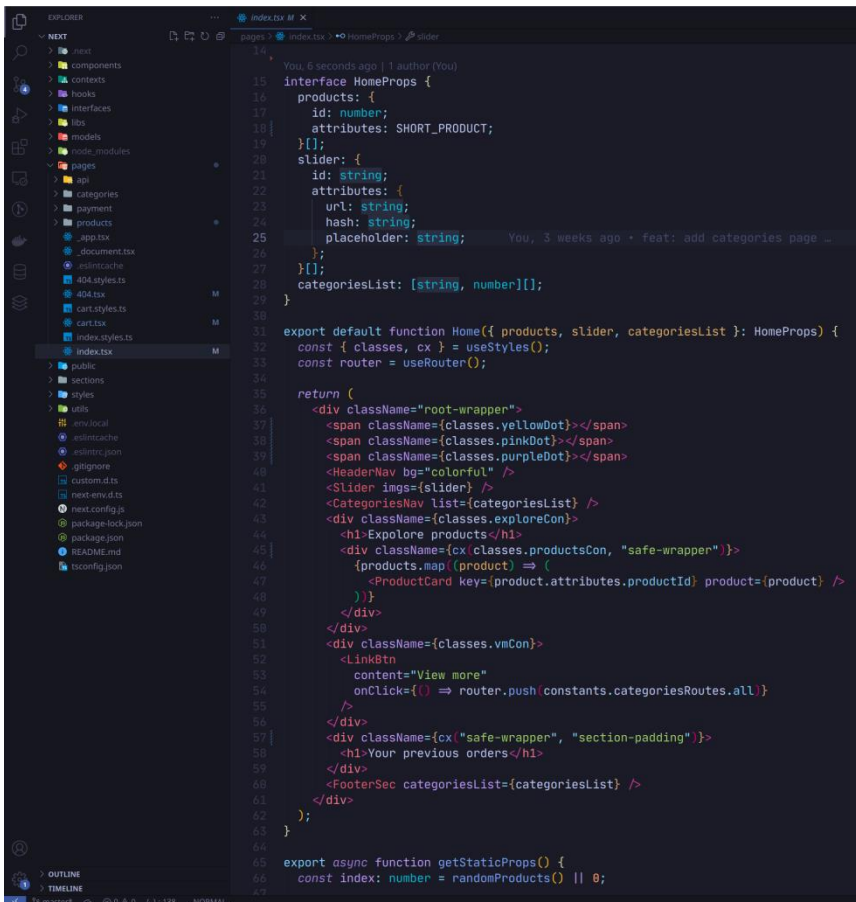
PICTURE 22. “_app.tsx” with “MantineProvider”, “ModalProvider”, “NotificationProvider” and “UserProvider”

5.2 Homepage

On every page, most lines of code are abstracted to multiple smaller components. The homepage received the data props from the server-side before the page was built. The props were random products, slider images, and a categories list. The random products are used in showing product cards. The idea is that the product cards and the categories suggestion appear arbitrary every time users come to the home page. The slider images and categories list are obtained from the Strapi using Axios.

Styling for each page is written in an apart file as “page name.styles.tx”. For the purpose of using the class name from the styling file, It is needed to import and extract the “classes” variable from the

“useStyles()” function. Besides, the developer can use a universal class name from global components. The children with a universal class name will have attributes like any HTML element having the same one. Furthermore, Mantine has its class name merging function, “cx”. There is no need for a new library to combine many class name attributes into one HTML tag. PICTURE 23 shows the home page implementation. There are minor components such as “HeaderNav”, “Slider”, “CategoriesNav”, “ProductCard”, ...



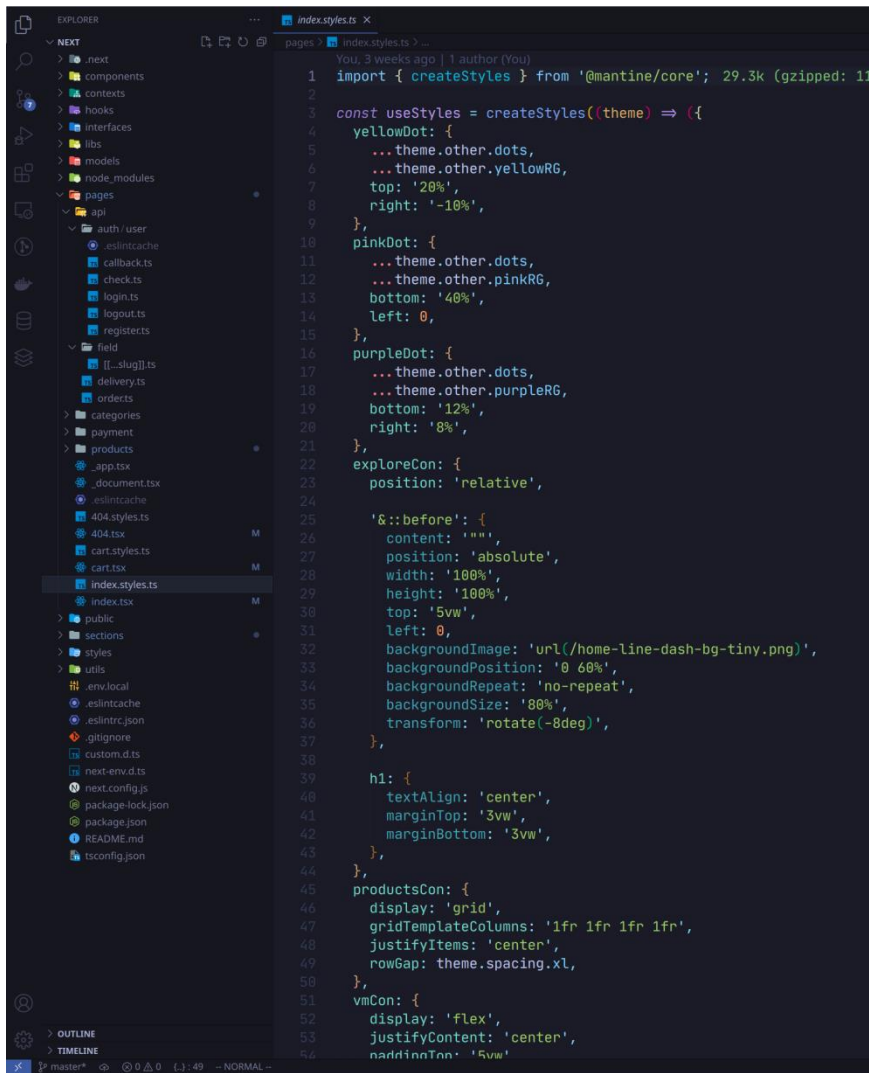
```

14  You, 6 seconds ago | 1 author (You)
15  interface HomeProps {
16    products: {
17      id: number;
18      attributes: SHORT_PRODUCT;
19    }[];
20    slider: {
21      id: string;
22      attributes: {
23        url: string;
24        hash: string;
25        placeholder: string;
26      };
27    };
28    categoriesList: [string, number][];
29  }
30
31  export default function Home({ products, slider, categoriesList }: HomeProps) {
32    const { classes, cx } = useStyles();
33    const router = useRouter();
34
35    return (
36      <div className="root-wrapper">
37        <span className={classes.yellowDot}></span>
38        <span className={classes.pinkDot}></span>
39        <span className={classes.purpleDot}></span>
40        <HeaderNav bg="colorful" />
41        <Slider imgs={slider} />
42        <CategoriesNav list={categoriesList} />
43        <div className={classes.exploreCon}>
44          <h1>Explore products</h1>
45          <div className={cx(classes.productsCon, "safe-wrapper")}>
46            {products.map((product) => (
47              <ProductCard key={product.attributes.productId} product={product} />
48            ))}
49          </div>
50        </div>
51        <div className={classes.vmCon}>
52          <LinkBtn
53            content="View more"
54            onClick={() => router.push(constants.categoriesRoutes.all)}
55          />
56        </div>
57        <div className={cx("safe-wrapper", "section-padding")}>
58          <h1>Your previous orders</h1>
59        </div>
60        <FooterSec categoriesList={categoriesList} />
61      </div>
62    );
63  }
64
65  export async function getStaticProps() {
66    const index: number = randomProducts() || 0;
67  }

```

PICTURE 23. Home page implementation

Developers can use the syntax from CSS or SCSS to style a class in Mantine. PICTURE 24 shows the home page styling file using SCSS. After importing the “createStyles” function from Mantine, the developer can pass “theme” as a parameter to use global styling objects. The syntax from the standard CSS file will need to change to “camelCase,” where is no hyphen. For example, “background-color” will change to “backgroundColor”. CSS selectors such as “:hover”, “::before”, and “::after” can be written by using quotation marks.



```

1 import { createStyles } from '@mantine/core'; 29.3k (gzipped: 11
2
3 const useStyles = createStyles((theme) => ({
4   yellowDot: {
5     ...theme.other.dots,
6     ...theme.other.yellowRG,
7     top: '20%',
8     right: '-10%',
9   },
10  pinkDot: {
11    ...theme.other.dots,
12    ...theme.other.pinkRG,
13    bottom: '40%',
14    left: 0,
15  },
16  purpleDot: {
17    ...theme.other.dots,
18    ...theme.other.purpleRG,
19    bottom: '12%',
20    right: '8%',
21  },
22  exploreCon: {
23    position: 'relative',
24
25    '&::before': {
26      content: '',
27      position: 'absolute',
28      width: '100%',
29      height: '100%',
30      top: '5vw',
31      left: 0,
32      backgroundImage: 'url(/home-line-dash-bg-tiny.png)',
33      backgroundPosition: '0 60%',
34      backgroundRepeat: 'no-repeat',
35      backgroundSize: '80%',
36      transform: 'rotate(-8deg)',
37    },
38
39    h1: {
40      textAlign: 'center',
41      marginTop: '3vw',
42      marginBottom: '3vw',
43    },
44  },
45  productsCon: {
46    display: 'grid',
47    gridTemplateColumns: '1fr 1fr 1fr 1fr',
48    justifyItems: 'center',
49    rowGap: theme.spacing.xl,
50  },
51  vmCon: {
52    display: 'flex',
53    justifyContent: 'center',
54    padding: '5vw'
55  }
56

```

PICTURE 24. Home page styles file

5.3 Search bar component

The search bar is a component in the Header navigation component. It uses the “Autocomplete” component from Mantine like PICTURE 25. “Autocomplete” component contains smaller components such as “root”, “wrapper”, “input”, “rightSection”, “dropdown”, “item”, and “hovered”. Mantine allows developers to make different classes and add to “classNames” of the parent component to styling for inner components (Mantine Developer 2022). The functions to handle components status, such as input changing and item submitting, are separated from the component parameters.

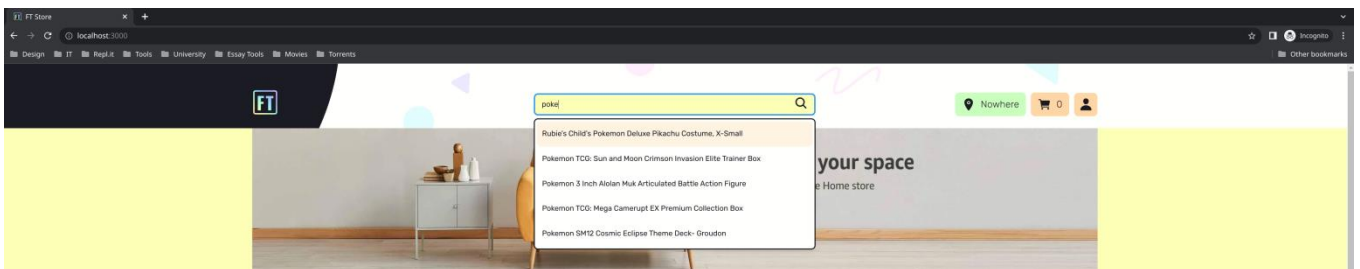
```

184     if (searchItem) {
185       router.push(`/products/${searchItem.id}`);
186     } else {
187       if (value) {
188         const query = clientQuery.queryByName(value);
189         router.push(
190           `${constants.categoriesRoutes.query}${Base64.encodeURI(query)}`
191         );
192       }
193     }
194     setValue('');
195     setSearchItem(null);
196   };
197
198   return (
199     <Autocomplete
200       placeholder="Do you want to find something?"
201       type="text"
202       ref={inputRef}
203       data={searches}
204       classNames={{
205         root: classes.root,
206         wrapper: classes.wrapper,
207         input: cx(
208           classes.input,
209           bg === 'colorful' ? classes.bgYellow : classes.bgWhite
210         ),
211         rightSection: classes.rgtSec,
212         dropdown: classes.dropdown,
213         item: classes.item,
214         hovered: classes.itemHovered,
215       }}
216     >
217       <UnstyledButton onClick={handleClick}>
218         <IoSearch color={theme.black} />
219       </UnstyledButton>
220     </Autocomplete>
221
222     <itemComponent={CustomItem}
223       onChange={handleChange}
224       onKeyDown={handleEnter}
225       onItemSubmit={(item: AutocompleteItem) => setSearchItem(item)}
226       dropdownPosition="bottom"
227       transition="slide-right"
228       transitionDuration={800}
229       transitionTimingFunction="ease"
230     />
231   );
232 }

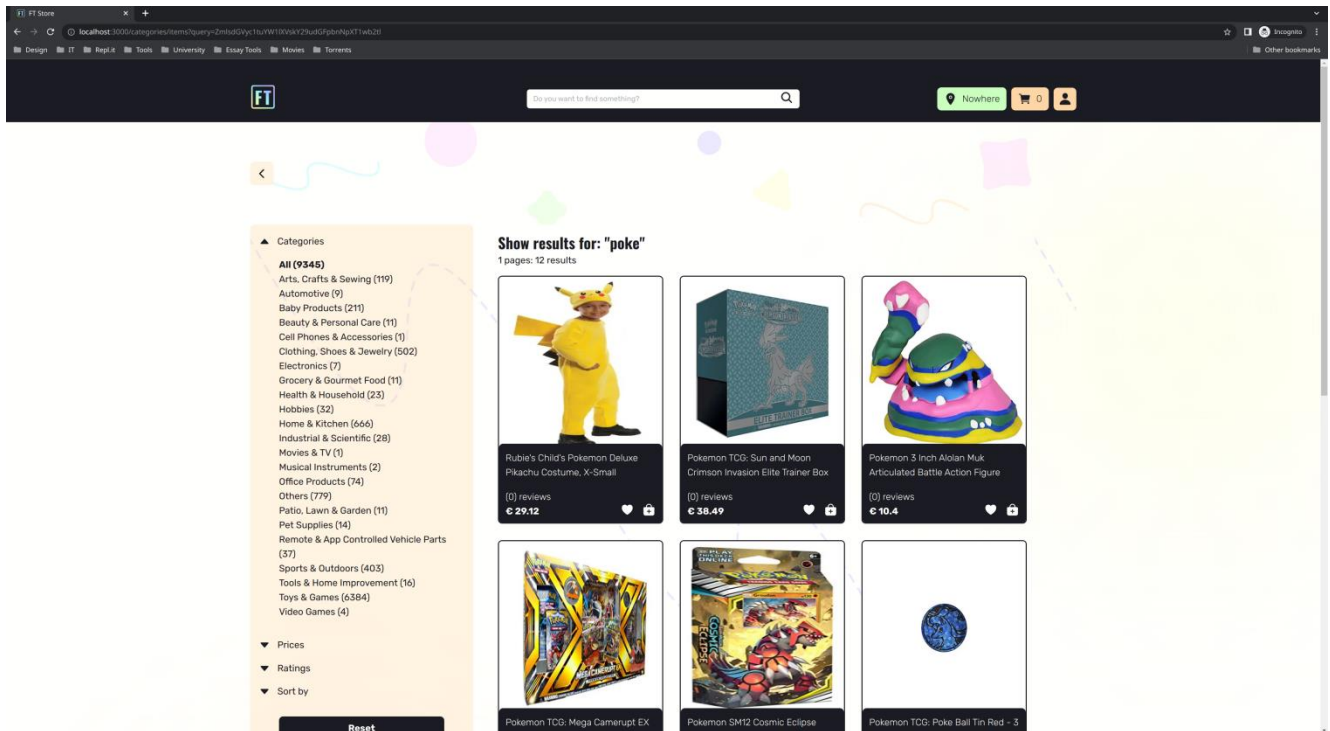
```

PICTURE 25. Search bar implementation using the “Autocomplete” component from Mantine

While users are typing in the search bar, there is a delay of 800 milliseconds before displaying the suggestions. The request will be created and sent to Strapi, and then it will return the list of related results. Five items limit the list, which helps boost the performance of returning the result. For example, on PICTURE 26, the user search for the keyword “poke”. The suggested products will be displayed at the bottom of the search bar. Then, on PICTURE 27, the user will be navigated to the categories page, which views related products to the search keyword. In case the user clicks on a suggested item, users will be redirected to the product page.



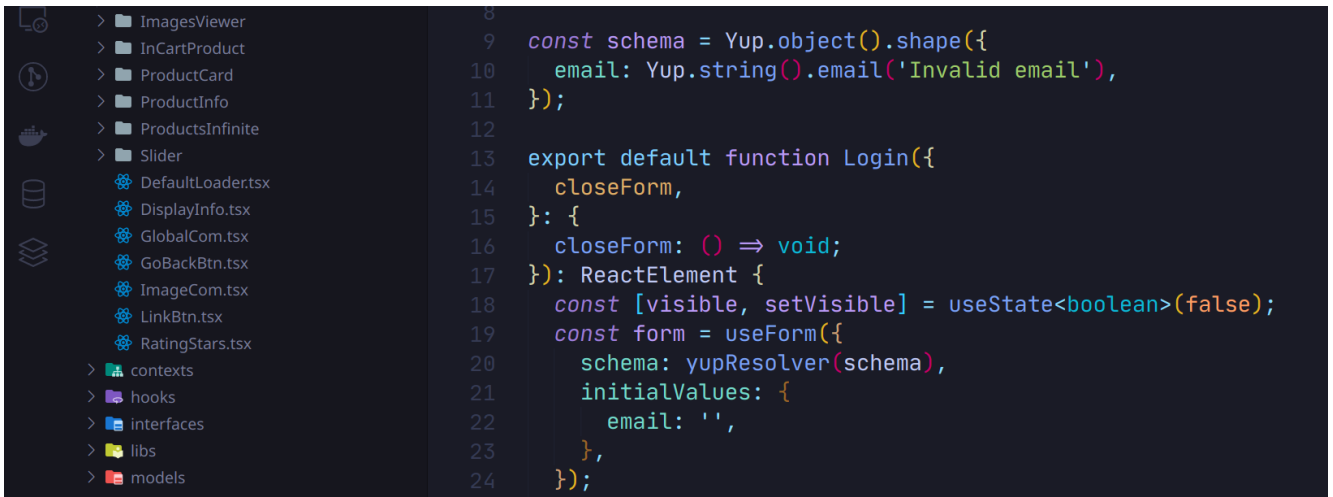
PICTURE 26. Show suggestions while typing the “poke” word



PICTURE 27. Displayed results for the keyword “poke”

5.4 Use-form hook

“Use-form” is a valuable custom hook from the Mantine library to control the form. Building a form includes setting up the state of inputs, validation, and arbitrary action to submit the form. Creating a form from scratch needs a lot of effort. “Use-form” hook creates a form object with initial values, schema validation, errors state and submits action. Form validation can use a popular library such as Joi, Yup, and Zod (Mantine Developers 2022). The syntax of building a form uses HTML tags. PICTURE 28 shows the beginning set up for the “useForm” hook. The schema uses the Yup validation object, and the initial value of the email is the blank string.



```

8
9  const schema = Yup.object().shape({
10   email: Yup.string().email('Invalid email'),
11 });
12
13 export default function Login({
14   closeForm,
15 }): {
16   closeForm: () => void;
17 }: ReactElement {
18   const [visible, setVisible] = useState<boolean>(false);
19   const form = useForm({
20     schema: yupResolver(schema),
21     initialValues: {
22       email: '',
23     },
24   });

```

PICTURE 28. Initialize form object for Login form

On the login form, there is a loading overlay component. It shows an overlay loading animation effect so that the user knows that the system is processing. In user experiences design, the design should always keep users informed about what is going on through appropriate feedback within a reasonable amount of time. The overlay is enabled when the user submits the form. This component is pre-built in the Mantine core package. The developer needs to pass a variable to control when the overlay appears. (Payne 2021.)

The login form requires an email. PICTURE 29 shows the handle submission form function. The inputted email will be verified using the Yup validation schema. When the email is correct, the user can click the button to submit it. The email will be sent to the back-end via the Axios function and “POST” method. Then, an overlay will be visible to show the submitting progress, and it will disappear when the process is done. If the authentication progress happens successfully, the magic link will be mailed to the registered account. Otherwise, the error messages will be shown in the form by using the hook.

```

25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
};

const handleSubmit = async (values: typeof form.values) => {
  setVisible(true);

  try {
    const res = await axios.post(
      `${constants.apiRoutes.authUser}/login`,
      values
    );

    // Reset and close form
    if (res.status === 200) {
      setVisible(false);
      form.reset();
      closeForm();
    }
  } catch (err) {
    const res = (err as AxiosError)?.response;

    setVisible(false);

    if (res?.status === 404) {
      form.setErrors({
        email: 'Registered email does not exist',
      });
    }

    if (res?.status === 500) {
      form.setErrors({ email: 'Something went wrong! Please try later' });
    }

    form.errors;
  }
};

return (
  <LoadingOverlay visible={visible} loader={<DefaultLoader />} />
  <form onSubmit={form.onSubmit(handleSubmit)}>
    <TextInput
      name="email"
      label="Email"
      placeholder="example@example.com"
      required
      {...form.getInputProps('email')}
    />
    <UnstyledButton type="submit">Send magic link </UnstyledButton>
  </form>
);

```

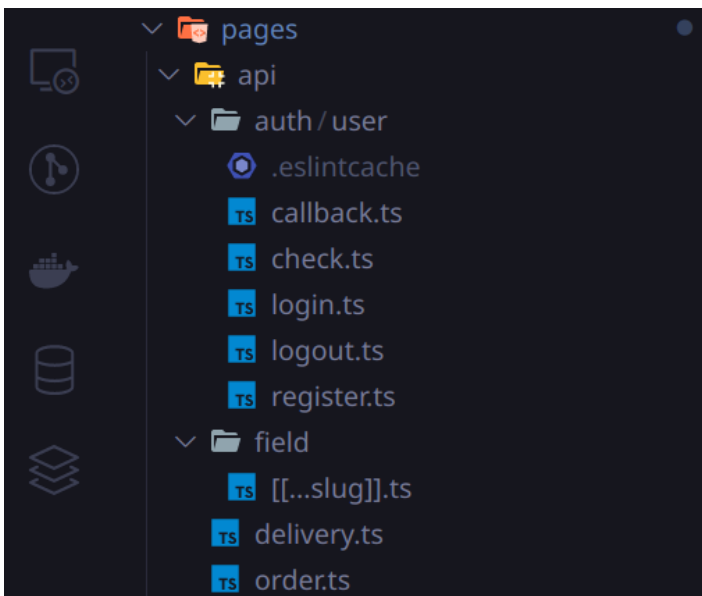
PICTURE 29. Login form and handle submit function

6 API IMPLEMENTATION

In this section, API implementation is the progress of building the API routes and database for the application. The first part of the section is about the Next Js application, and the rest is about Strapi. However, the given information is one example for each application due to the massive project API routes. The Next Js application describes checking user or guest function, and the Strapi application clarifies product content-type and custom category API route.

6.1 Next Js Application

PICTURE 30 shows the structure of the API route in directories. The “auth/user” folder is the route for user authentication and authorization in the API folder. The “field” folder is for updating the user data field. Besides the folder as routes, files are also served as API routes. “delivery” file handles the request that needs to calculate delivery distances and fees. Moreover, the “order” file controls adding new orders. Th route “/api/auth/callback” is handled by “/api/auth/callback.ts”. The files will bracket in the name such as “[[...slug]].ts” can catch all routes “/field/userId/name”, “/field/phonenummer”. The developer can create a new file to handle the new API route.

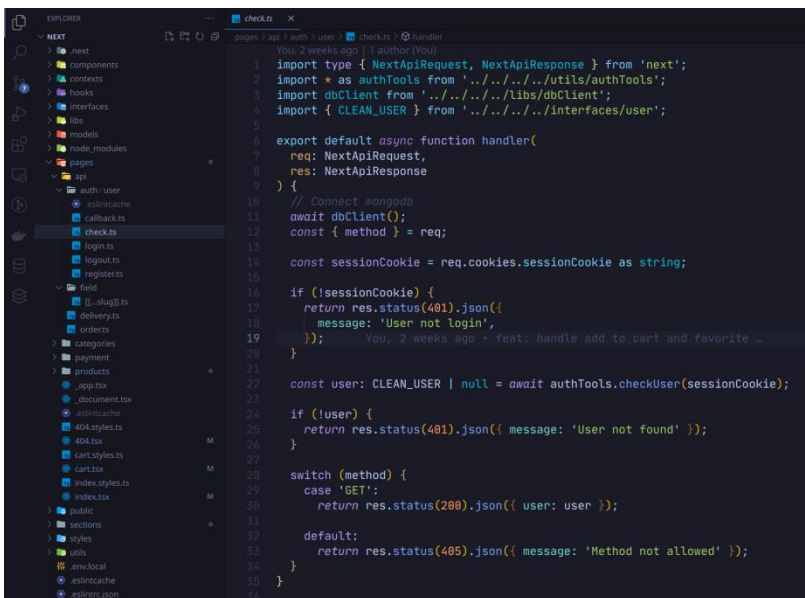


PICTURE 30. API routes in the Next Js application

Each route can handle all HTML request methods such as “GET”, “POST”, “PUT”, “DELETE”, ... The request handler receives “req” and “res” as a parameter to extract the information of the request and

respond the data to the front-end. The response from the back end needs to have status codes to indicate if an HTTP request is completed. Response status codes can be grouped into five classes. The first group “1xx” is the informational response. The next group “2xx” is the successful response. Third, group “3xx” is a redirect response, and group “4xx” is the client error response. Finally, group “5xx” is the server error response. (Mozilla Developers 2022.)

The “check” route runs whenever the user has access to the website. PICTURE 31 shows the implementation of the “check” route. If the user has a session token in the request cookie, it will need to verify using the “jsonwebtoken” library. Every token has an expiration time without auto-refreshing. After seven days, the user will need to log in again. A valid token contains the user id and session code which can be found on MongoDB. If both requirement strings are correct, user data will be loaded to the application. Or no data will be returned, and the application will mark the current user as “Guest” and save their browsing data in the local storage of the browser.



```

1 import type { NextApiRequest, NextApiResponse } from 'next';
2 import * as authTools from '../utils/authTools';
3 import dbClient from '../libs/dbClient';
4 import { CLEAN_USER } from '../interfaces/user';
5
6 export default async function handler(
7   req: NextApiRequest,
8   res: NextApiResponse
9 ) {
10   // Connect mongodb
11   await dbClient();
12   const { method } = req;
13
14   const sessionCookie = req.cookies.sessionCookie as string;
15
16   if (!sessionCookie) {
17     return res.status(401).json({
18       message: 'User not login',
19     });
20   }
21
22   const user: CLEAN_USER | null = await authTools.checkUser(sessionCookie);
23
24   if (!user) {
25     return res.status(401).json({ message: 'User not found' });
26   }
27
28   switch (method) {
29     case 'GET':
30       return res.status(200).json({ user: user });
31
32     default:
33       return res.status(405).json({ message: 'Method not allowed' });
34   }
35 }

```

PICTURE 31. “/auth/user/check” API route handles only the “GET” request method

6.2 Strapi Application

Strapi automatically creates API endpoints when developers create new content types. Filter, sort, and paginate results using API parameters, as well as pick fields and relations to populate. Specific parameters about optional features, such as publishing state and locality, can also be used. API routes have four HTTP methods, including “GET”, “POST”, “PUT”, and “DELETE” functions (Strapi

Developers 2022). PICTURE 32 is the implementation of requests from the client to the Strapi application. There are many parameters, so they need to be wrapped using the “qs” library. Furthermore, PICTURE 33 contains multiple lines of the request sent to Strapi. A “GET” request to product route “/api/products” is on the last line. The result needs to filter by name containing the keyword “poke”. The result is divided into various pages, and the size for each page is twelve products.

```

1 import qs from 'qs'; 31.9k (gzipped: 10.1k) You, 3 weeks ago
2 import axios from 'axios'; 35.3k (gzipped: 12.1k) You, 3 weeks ago
3
4 import * as constants from './constants';
5
6 export const productsRandom = (index: number) =>
7   axios.get(
8     `${constants.cmsAPI}/products?` +
9     qs.stringify(
10      {
11        fields: ['product_id', 'name', 'price', 'image_ids'],
12        populate: {
13          review: {
14            fields: ['rating'],
15          },
16        },
17        pagination: {
18          start: index,
19          limit: 12,
20        },
21      },
22      {
23        encodeValuesOnly: true,
24      }
25    )
26  );
27
28 export const productById = (id: string) =>
29   axios.get(
30     `${constants.cmsAPI}/products?` +
31     qs.stringify(
32      {
33        filters: {
34          product_id: {
35            Seq: id,
36          },
37        },
38      },
39      {
40        encodeValuesOnly: true,
41      }
42    )
43  );

```

PICTURE 32. Example product query request from client-side

```

Welcome back!
To manage your project, go to the administration panel at:
http://localhost:1337/admin

To access the server, go to:
http://localhost:1337

[2022-05-13 16:18:39.383] http: GET /api/slider?populate[images][fields][0]=hash&populate[images][fields][1]=url&populate[images][fields][2]=placeholder (33 ms) 200
[2022-05-13 16:18:39.416] http: GET /api/products?fields[0]=product_id&fields[1]=name&fields[2]=price&fields[3]=image_ids&populate[review][fields][0]=rating&pagination[start]=4033&pagination[limit]=12 (59 ms) 200
[2022-05-13 16:18:39.477] http: GET /api/category (142 ms) 200
[2022-05-13 16:18:51.120] http: GET /api/products?filters[name][$containsi]=te (44 ms) 200
[2022-05-13 16:18:56.697] http: GET /api/products?filters[name][$containsi]=pokem (42 ms) 200
[2022-05-13 16:11:26.584] http: GET /api/slider?populate[images][fields][0]=hash&populate[images][fields][1]=url&populate[images][fields][2]=placeholder (16 ms) 200
[2022-05-13 16:11:26.633] http: GET /api/category (67 ms) 200
[2022-05-13 16:11:26.635] http: GET /api/products?fields[0]=product_id&fields[1]=name&fields[2]=price&fields[3]=image_ids&populate[review][fields][0]=rating&pagination[start]=1877&pagination[limit]=12 (62 ms) 200
[2022-05-13 16:11:31.345] http: GET /api/products?filters[name][$containsi]=text (39 ms) 200
[2022-05-13 16:11:36.129] http: GET /api/products?filters[name][$containsi]=poke (33 ms) 200
[2022-05-13 16:13:40.712] http: GET /api/products?fields[0]=product_id&fields[1]=name&fields[2]=price&fields[3]=image_ids&populate[review][fields][0]=rating&pagination[start]=3134&pagination[limit]=12 (19 ms) 200
[2022-05-13 16:15:00.374] http: GET /api/slider?populate[images][fields][0]=hash&populate[images][fields][1]=url&populate[images][fields][2]=placeholder (9 ms) 200
[2022-05-13 16:15:00.441] http: GET /api/category (78 ms) 200
[2022-05-13 16:15:00.590] http: GET /api/products?filters[name][$containsi]=poke&pagination[page]=1&pagination[pageSize]=12 (17 ms) 200
[2022-05-13 16:18:30.976] http: GET /api/products?filters[name][$containsi]=poke&pagination[page]=1&pagination[pageSize]=12 (22 ms) 200
[2022-05-13 16:19:29.629] http: GET /api/products?filters[name][$containsi]=poke&pagination[page]=1&pagination[pageSize]=12 (30 ms) 200

```

PICTURE 33. Strapi logs for searching a product by name

The product description is built as “content-type” in Strapi so that it will have API endpoints. PICTURE 34 is the product content type which has eight fields, including “name”, “category”, “price, about”, “specification”, “technicalDetails”, “productId”, “reviews”, “imageIds”, “weight”, “avgRating”, and “ratingAmount”. In order to send the query request, the application will need an authorization token, or assets are available to the public. (Strapi Developers 2022.)

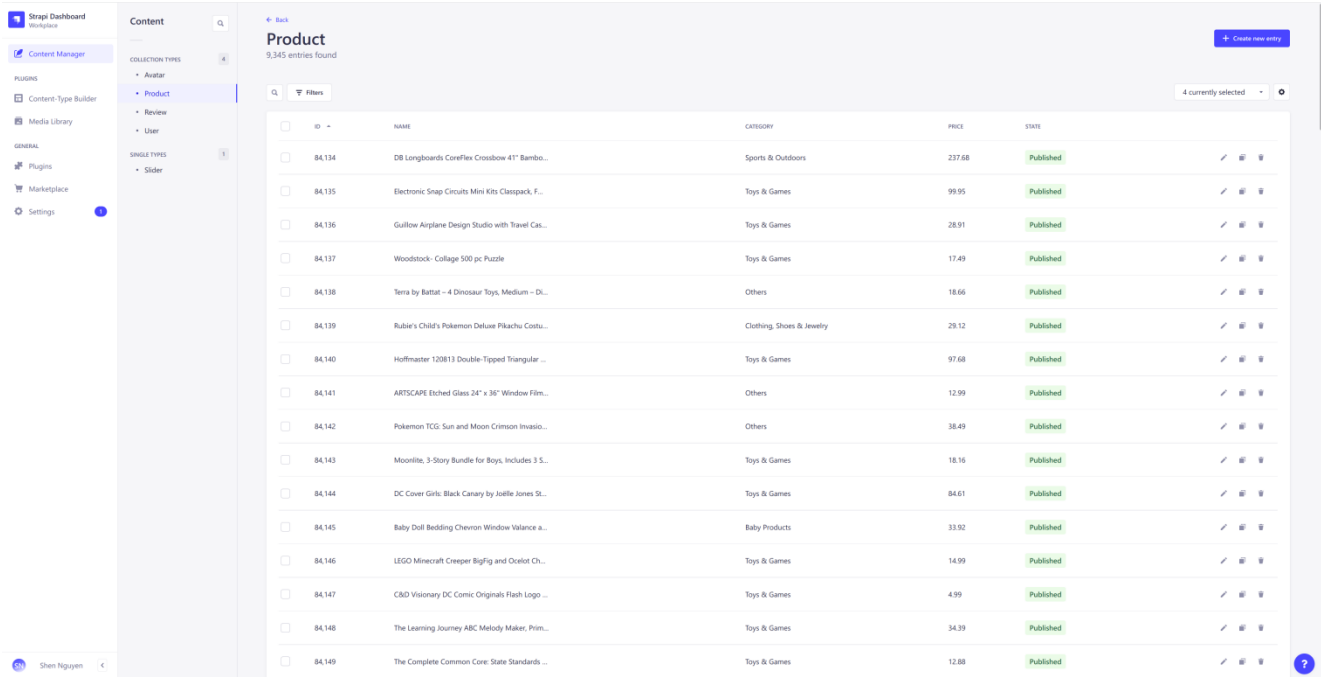
NAME	TYPE
Ab name	Text
Ab category	Text
123 price	Number
≡ about	Rich text
≡ specification	Rich text
≡ technicalDetails	Rich text
🔑 productId	UID
🔗 reviews	Relation with Review
≡ imageIds	Rich text
Ab weight	Text
123 avgRating	Number
123 ratingAmount	Number

+ Add another field to this collection type

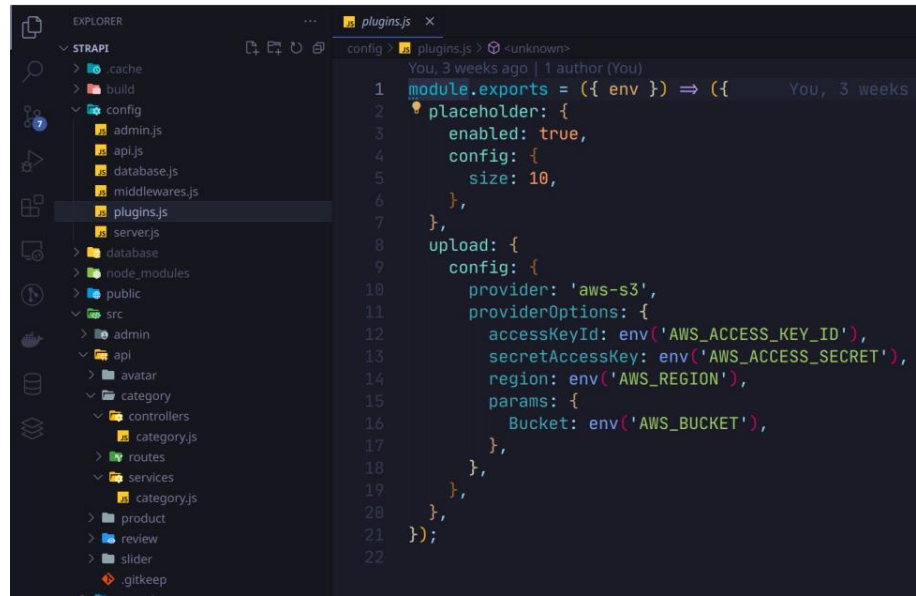
PICTURE 34. Content-type product - fields and their datatype

Strapi has a friendly user interface for regular users and developers to add or manage the content. Authority users can add new content by creating a new content type and its type. Then, a button named “create new entry” shows the form and input to save new content like in PICTURE 35. Developers can directly access RDBMS data programs such as MySQL and SQL Server. Then new data can be added using SQL queries. Developers can custom Strapi to their purposes and usages. For example, the RDBMS data program can be changed from SQL Server to MySQL by editing the config file

“database.js”. Besides saving the images in the current local storage, Strapi supports plugins to upload the images to cloud providers such as Cloudinary and Amazon S3. PICTURE 36 shows the implementation connecting Strapi to AWS S3. It requires the provider name, access key identification string, secret access key, region of the bucket, and the bucket name. There is a plugin to enable GraphQL queries. (Strapi Developers 2022.)

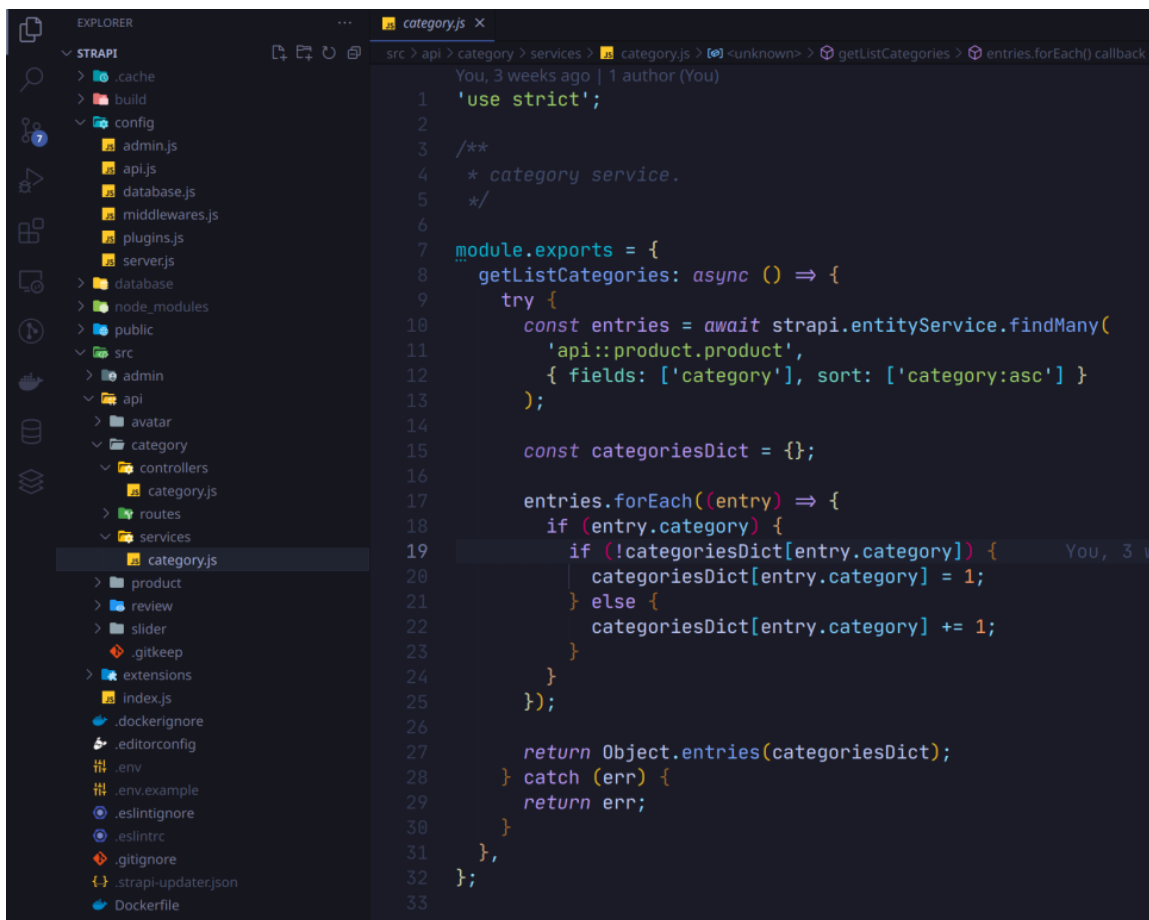


PICTURE 35. Products collections samples



PICTURE 36. AWS S3 plugin to upload images

Strapi also allows developers to establish new API routes by following the Koa back-end framework's documentation. The requests received the routes that handle the requests and trigger the execution of their controller handlers. The policies can block access to a route. The middlewares can control the request flow and the request before moving forward. The controllers execute code once a route has been reached. The services are used to build custom logic reusable by controllers. Finally, the models that represent the request can all be customized in the Strapi back-end (Strapi Developers 2022). PICTURE 37 is a new custom API route name "category". This route returns the category list and the items in each category. It is used in the footer and categories page of the client.



```

1  'use strict';
2
3  /**
4   * category service.
5   */
6
7  module.exports = {
8    getListCategories: async () => {
9      try {
10       entries = await strapi.entityService.findMany(
11         'api::product.product',
12         { fields: ['category'], sort: ['category:asc'] }
13       );
14
15       const categoriesDict = {};
16
17       entries.forEach((entry) => {
18         if (entry.category) {
19           if (!categoriesDict[entry.category]) {
20             categoriesDict[entry.category] = 1;
21           } else {
22             categoriesDict[entry.category] += 1;
23           }
24         }
25       });
26
27       return Object.entries(categoriesDict);
28     } catch (err) {
29       return err;
30     }
31   },
32 };
33

```

PICTURE 37. New category route to return the total number of categories in the database

7 CONCLUSION

This thesis aims to provide an overview of e-commerce web development using the benefits of Next Js and Jamstack. Through extensive study, the practical section is used as an introduction to serverless, containers, and microservices architecture for newcomers. Next Js developers need unique challenges to planning how and when content should be rendered for a specific page. Using a modern tool for the modern web development makes the user experiences, SEO, and security better, cleaner, and more user-centric. It does not matter if the web page serves several or millions of users. This technology can benefit online businesses in the most efficient ways.

There are several points for the further development of this project for better usage and reality factors. The first suggestion is in the authentication methods. The current authentication used the magic link to log in to save data. More methods can be implemented, such as usernames and passwords, and logging in with Google, Facebook, or Apple. The authentication information will need to be encrypted and kept in a safe place to prevent breaches. In addition, the website was designed and implemented for the desktop view only. It is necessary for mobile and tablet versions in the digital era where users are most active on handheld devices. Mantine library support using media query in the “createStyles” function and “MediaQuery” component. It is a flexible way to apply styles when the screen width meets the configured breakpoints.

The web page uses both SQL and no-SQL databases such as MySQL and MongoDB for database implementation. There are differences in the syntax when querying the data and the repetitive models between them. The work can be done more effectively by using object-relational mapping (ORM) such as Prisma. Developers can manipulate data in both databases with few errors and less boilerplate. Lastly, unit testing is an essential part that is missing from implementation. Unit testing is the testing from the developer to make sure each component units are usable. It is the first level of testing to find the errors early and save time and money in the production stage.

REFERENCES

- Alam, I. 2022. Best practices to increase the speed for next.js apps. Stack Overflow Blog. Available at: <https://stackoverflow.blog/2022/03/30/best-practices-to-increase-the-speed-for-next-js-apps/>. Accessed 26 May 2022.
- Alfaro, B.A. 2021. What is Jamstack and why is revolutionizing web development? Brayan Arrieta Alfaro's Blog. Available at: <https://brayanarrieta.hashnode.dev/what-is-jamstack-and-why-is-revolutionizing-web-development>. Accessed 27 March 2022.
- AltexSoft. 2019. GraphQL: Core features, architecture, pros, and cons. AltexSoft. Available at: <https://www.altexsoft.com/blog/engineering/graphql-core-features-architecture-pros-and-cons/>. Accessed 28 March 2022.
- Anshu, A. 2021. What is the meaning of Jamstack. codenanshu. Available at: <https://codenanshu.in/blog/what-is-the-meaning-of-jamstack>. Accessed 27 March 2022.
- ChecklyHQ Developers. 2022. API monitoring for the jamstack. Checkly. Available at: <https://www.checklyhq.com/guides/api-monitoring/>. Accessed 11 May 2022.
- Drupal Developer. 2022. What is decoupled. Drupal.org. Available at: <https://www.drupal.org/docs/develop/decoupled-drupal/what-is-decoupled> Accessed 27 May 2022.
- Galle, T. 2022. The basics of dependency maintenance in NPM/Yarn. DEV Community. Available at: <https://dev.to/th0rgall/the-basics-of-dependency-maintenance-in-npm-yarn-4c11>. Accessed 27 May 2022.
- Gillis, A.S. 2020. What is Rest Api (restful API)? SearchAppArchitecture. Available at: <https://www.techtarget.com/searchapparchitecture/definition/RESTful-API>. Accessed 5 May 2022.
- Haq, F. 2019. Why (and when) you should use Kubernetes. HackerNoon. Available at: <https://hackernoon.com/why-and-when-you-should-use-kubernetes-8b50915d97d8>. Accessed 31 March 2022.
- Ismail, K. 2022. Jamstack for marketers: What is a Jamstack CMS? Storyblok. Available at: <https://www.storyblok.com/mp/jamstack-for-marketers>. Accessed 11 March 2022.
- Kubernetes.io Editor. 2021. What is Kubernetes? Kubernetes. Available at: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>. Accessed 31 March 2022.
- Lin, J. 2020. Deploying a scalable web application with Docker and Kubernetes. Medium. Available at: <https://medium.com/better-practices/deploying-a-scalable-web-application-with-docker-and-kubernetes-a5000a06c4e9>. Accessed 31 March 2022.
- Mantine Developers. 2022. Mantine Documentation. Available at: <https://mantine.dev/getting-started>. Accessed 13 May 2022.

MongoDB Developer, 2022. Why use mongodb and when to use it? MongoDB. Available at: <https://www.mongodb.com/why-use-mongodb>. Accessed 27 May 2022.

Mozilla Developers. 2022. HTTP response status codes - HTTP: MDN. HTTP | MDN. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>. Accessed 14 May 2022.

Next Js Developers. 2022. Next Js Documentation. Available at: <https://nextjs.org/docs>. Accessed 27 March 2022.

Niżyński, D. 2022. What is Next JS and why should you use it in 2022? Pagepro. Available at: <https://pagepro.co/blog/what-is-nextjs/>. Accessed 25 May 2022.

Patel, A. 2021. Kubernetes-architecture overview. Medium. Available at: <https://medium.com/devops-mojo/kubernetes-architecture-overview-introduction-to-k8s-architecture-and-understanding-k8s-cluster-components-90e11eb34ccd>. Accessed 31 March 2022.

Payne, S. 2021. Your loading spinner is a UX killer! Here's an alternative. Boldist. Available at: <https://boldist.co/usability/loading-spinner-ux-killer/>. Accessed 16 May 2022.

Redhat.com. 2019. What is GraphQL? Red Hat - We make open source technologies for the enterprise. Available at: <https://www.redhat.com/en/topics/api/what-is-graphql>. Accessed 28 March 2022.

Redhat.com Editor. 2021. What is containerization? Red Hat - We make open source technologies for the enterprise. Available at: <https://www.redhat.com/en/topics/cloud-native-apps/what-is-containerization>. Accessed 31 March 2022.

Wessling, R. 2020. Headless CMS - explained in 1 Minute. Contentful. Available at: <https://www.contentful.com/r/knowledgebase/what-is-headless-cms/>. Accessed 27 March 2022.

Strapi Developers. 2022. What is a Headless CMS? Strapi.io. Available at: <https://strapi.io/what-is-headless-cms>. Accessed 11 May 2022.

Strapi Developers. 2022. Strapi Documentation. Strapi.io. Available at: <https://docs.strapi.io/developer-docs/latest/getting-started/introduction.html>. Accessed 11 May 2022.