ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siége Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : 1-2 rue André Ampère - 2083 - Pôle Technologique - El Ghazala - Tél +216 70 250 000 - Fax +216 70 685454

esprit

Se former autrement

HONORIS UNITED UNIVERSITIES

2021 / 2022

Speciality :Computer Science

Design and Implementation of a Headless Design System

Presented by: Mohamed Neji GHAZOUANI

University Supervisor: Wafa GAIGI

Enterprise Supervisor: Hazem MDIMEGH

Academic supervisor	Technical supervisor
Mrs Wafa GAIGI	Mr Hazem MDIMEGH [illegible]

DEDICATION

I dedicate this project to :
My dear parents Bechir and Samia for their continuous love and support. Thank you so much for all the sacrifices you have made for me.
My dear sisters Sarah,Sondes and Souma who supported me all along the way and significantly contributed to my success.
To those who have supported me, encouraged me, appreciated my efforts and created the favorable environment and the jovial atmosphere for me to embark upon this journey.To all these lovely people, I would like to dedicate this modest work embroidered with my heartfelt gratitude and my sincerest recognition.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to my Hatch supervisors,**Mr. Hazem Mdimegh**,**Mr. Leonard Goldstein** and **Mr. Julian Willner,**. Since the first day I joined Hatch, they were always available to guide me, teaching me the best practices and helping me whenever I was blocked. I also want to thank them for their exceptional management skills and welcomeness.

I would like to thank Miss. Wafa Gaigi, my supervisor at the Private Superior School of Engineering and Technology (ESPRIT), for her precious guidance, intellectual and moral support as well as her valuable advice which never ceases to give me throughout the internship period.

I would like to express my gratitude to all the Hatch team, especially whom I worked closely with for the last 6 months.

Furthermore I would also like to acknowledge with much appreciation all of those who supported me somehow along the way. Family, friends, werewolf people, director... Thank you all, you are awesome.

TABLE OF CONTENTS

LIST OF FIGURES

ACRONYMS

CD Continuous Delivery/Continuous Deployment. xiii

CI Continuous Integration. xiii

CSS Cascading Style Sheets. xiii

GIT Global Information Tracker. xiii

HDS Headless Design System. xiii

HTML HyperText Markup Language. xiii

JS Java Script. xiii

NPM Node Package Manager. xiii

PR Pull Request. xiii

UI User Interface. xiii

UML Unified Modeling Language. xiii

UX User experience. xiii

GENERAL PRESENTATION

In 1968, Douglas Mcllroy the American mathematician presented in the software engineering conference of Nato a component-based development to solve the dilemma.Because it provided a solution to speed up the programming's potential by creating reusable code which will provide efficiency and ease to scale. This idea lowered the effort and boosted the speed of the development allowing software to better use the power of modern computers.Now, After more than 50 years, a similar challenge is faced but this time in design. A huge struggle to scale design with the applications it supports because the design is still bespoke tailoring solutions for individual problems.

A design system is the center of shared parts and procedures, such as components, patterns, and guidelines to build reliable and robust products. It is in the ecosystem where the design procedure happens and the fruit of the design thinking reaches its highest level. The term design system itself envelops all of the design, code, and content assets, they are tailored to organizational necessities. Also, design systems show the culture, team values, and visual language of a company.

Design systems empower teams to manufacture better products quicker by making design reusable. Reusability makes scale conceivable. This is the heart and essential value of design systems. A design system is an assortment of reusable components, guided by clear guidelines, that can be amassed together to build any number of applications. The goal of a design system is to build a solution that will help the teal to scale a product successfully. With a well defined and clear system, both teams of designers and developers will focus their efforts on giving what the user needs, instead of creating elements and inventing solutions that may be

done before[1].
This report is organized into four chapters, the first chapter is a context chapter that presents a description of the project. In Second chapter we elaborated the State of art of this design system. Furthermore, In chapter three, we analyzed the requirements and explained the architecture . Finally, the last chapter summarizes the project's implementation and the working environment.

CHAPTER 1

PROJECT SCOPE

Introduction

The purpose of this chapter is to provide an overview of the broad scope of our project. This contains a presentation of Hatch, the hosting company, and the departments that make up the startup as a whole. In addition, we will present our project, the issue that it attempts to address, and the environment in which it exists. After that, we'll talk about the methodology that we've used throughout this project.

1.1 Project Environment

In the final year of engineering at the Private Higher School Of Engineering And Technology (ESPRIT) students are required to do a final project internship in order to obtain their diplomas. The internship is done on duration of 6 months from April the 1st 2022 to October the 1st 2022 at Hatch Studio company in Berlin,Germany.

The project is realised by Mohamed Neji Ghazouani, supervised by Mrs Wafa GAIGI professor at ESPRIT and Mr Hazem MDIMEGH Full stack developer at Hatch and Mr Leonard GOLDESTIN Front-End Developer at Hatch.

1.2 Host company

1.2.1 Hatch studio

Founded in 2019, Hatch is a startup headquartered in Berlin, developing a wide variety of digital solutions and products. Through its expertise and a well balanced team, it provides assistance in the analysis and development of digital solutions for different needs.

Figure 1.1: Hatch studio

Hatch studio constitutes of three different departments:

- **Commercial Team** : Responsible for marketing and reaching to potential clients.
- **Designers Team**: Responsible for providing the best design possible for the best User Experience (UX).
- **Engineering Team** : Responsible for the development, the optimization and the delivery of the products.

1.2.2 Area of expertise

The main mission consists of giving companies and brands a competitive edge in the digitally connected world. Therefore the focus is on developing outstanding digital solutions that shape the world of tomorrow. So in a nutshell Hatch offers:

- **Consulting & strategy**
- **Branding & design**
- **Digital solutions development**

During the 2+ years in the market, Hatch has built great relations with many customers like Technical University of Munich (TUM),Valora, Bridl, MilFit, DeepGreen, Solavieve ...

1.3 Project Presentation

In this section, we will showcase the project by discussing its background and problematic, studying and criticising the existing, followed by a discussion of Hatch's solution and its limits.

1.3.1 Problematic

At Hatch, clients range from start-ups and small businesses to large corporations, and we work with diverse industries. Each client has a unique set of requirements, as well as unique user interfaces. Modern user interfaces are more complicated than ever. Clients expect compelling, personalized experiences delivered across devices. That means frontend developers and designers have to embed more logic into the UI. But UIs become unwieldy as applications grow. Large UIs are brittle, painful to debug, and time consuming to ship.

1.3.2 study of the existing

The study of the existing situation helps to identify the weaknesses and strengths of the current situation to identify the needs of the clients.

At Hatch, both of the design team and development team spend a huge amount of time on the creation of a new component from scratch where they attend large number of meetings for every project that the company is working on.

1.3.3 Criticism of the Existent

in this subsection we will criticize the existing situation on three aspects:

- **Time Loss:** with attending a lot of meetings , spending a huge time on the creation on the new components in every project and having a short time on comprehending the complexities of the project.
- **Quality Loss:**with the spending of more time on the creation of the components over and over again, the development team doesn't have time to manually test everything well enough and loses focus on developing complex features.
- **Money Loss:** with the time loss and quality loss the organisation will spend more time preparing the product with the possibility of dissatisfaction from the client that would result in a lot of money loss for the organisation.

1.3.4 Proposed Solution

Hatch HDS, A series of documented elements, components, and pages that include both design and front-end guidelines. The documentation contains live code examples, allowing cross-functional teams to easily reuse styles and components in several instances across an application. we present it by delivering a storybook for visualisation and documentation and an npm package as a component library and building an automated process to encourage the developers and designer to contribute to it.

1.4 Methodology

An organization places a high value on project management and project structure. Good project management methodology enables businesses to fulfill external deadlines and deal with unexpected developments. Failure to do so may have highly severe consequences, allowing projects to spiral out of control. Different industries have developed several models for customized solutions, some more efficient than others. However, it appears that the Agile Methodology, in particular, is one of the most powerful techniques to have ever emerged from the software engineering industry.

1.4.1 Agile Methodology

The Agile Methodology entails a number of components that are difficult to extract. There are several definitions and implementations available. Cross-collaboration and adaptive planning with short sprints and functional iterations are, nevertheless, common concepts.

The goal is to create a procedure that is both realistic and responsive. It is necessary to strike a balance between giving just enough structure to keep things on track and everyone successful while avoiding unnecessary restrictions or process bloat. This procedure is very useful to startups.

1.4.2 Scrum

Is an agile process that allows us to focus on delivering the most important value in the shortest time. It rapidly and repeatedly inspects actual working software.

1.4.2.1 Stages

The different stages of Scrum:

- **Daily scrum meeting (stand-up meeting)** : This is a daily meeting of about 15 to 20 minutes, allowing each member of the team to say what the issues he has encountered, what he has done, what he actually working on.
- **Weekly Retrospective meeting** : this is a weekly meeting of about 30 min to 1 hour that takes place at the end of the week. Its purpose is to discuss what went well,how can we improve and what could've gone better on the week.
- **The sprint meeting** : This is a bi-weekly meeting, the team leader assigns tasks to all members of the team and closes the old user's story and discusses the problems if there are any, and then the meeting will deal with the next iteration.

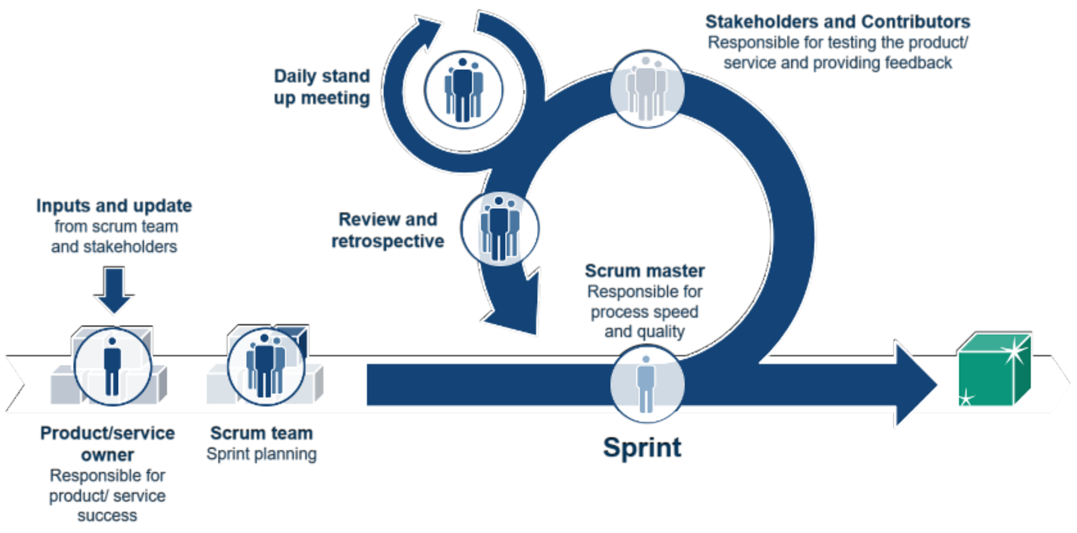

Stakeholders and Contributors
Responsible for testing the product/ service and providing feedback
Daily stand up meeting
Inputs and update from scrum team and stakeholders
Review and retrospective
Scrum master
Responsible for process speed and quality
Product/service owner
Responsible for product/ service success
Scrum team
Sprint planning
Sprint

Figure 1.2: Method SCRUM

1.4.2.2 Team

Scrum team members:

- **The "Product Owner"** : Responsible for the return on investment, he is the client and the owner of the final product.
- **The "SCRUM Master"** : the "SCRUM Master" enforces the values and practices of the method applied by the team.
- **The "SCRUM" team** : Composed of more multiple members, full time on the project, the team organises itself and its composition must not change during a Sprint.

1.4.3 Kanban

1.4.3.1 What is Kanban

Kanban is a methodology that aims to improve and manage the company's knowledge-based workflow. This technique helps organizations deal with their unfinished tasks rather than begin new ones, to focus on driving progressive change, and to limit the work in progress by visualizing their workflow frequently.

Kanban has different interesting principles:

- Focuses on understanding customers' needs.
- It gives the team the flexibility to organize their priorities.

- Daily meeting to observe the progress of the work.
- Define the **Work In Progress (WIP)** and limit the number of changes per commit.
- Improve the flow by starting with tasks that have the strongest priority.

By identifying the best possible team workflow, Kanban promotes continuous cooperation and encourages active and continuous learning and improvement.

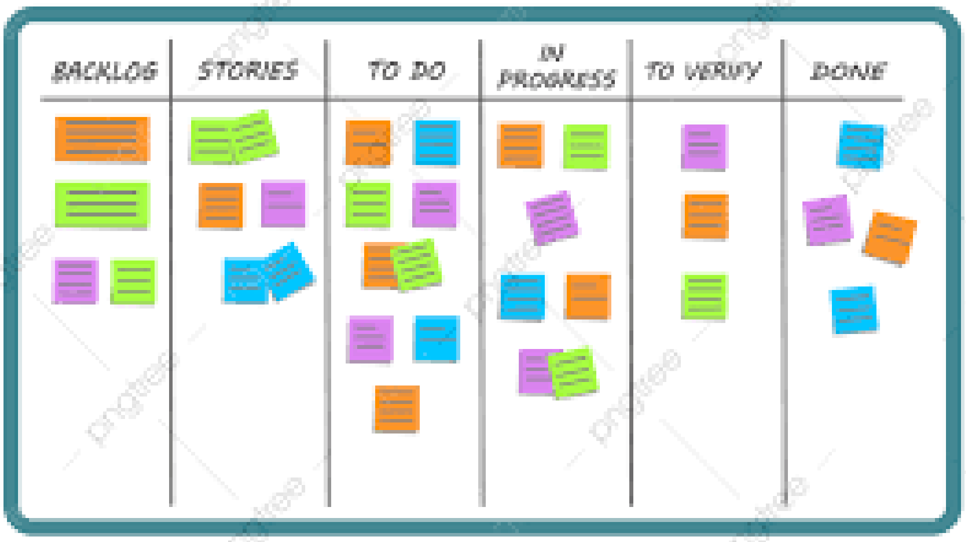


Figure 1.3: Kanban Board

1.4.3.2 Benefits of Kanban

Like all the agile methodologies, Kanban focuses on finishing the tasks in progress and not starting new ones. The approach presents different benefits such as:

- Shorter cycle times can deliver features faster.
- Responsiveness to change, which makes Kanban ideal since priorities change very frequently.
- Focusing on relevant tasks and avoiding wasting time and effort.
- Faster feedback enhances motivation, creativity, and efficiency of the team.

1.4.3.3 Why we chose Kanban

The choice of the development method is often determined by the organization, culture, and team dynamics.

Being a new startup, it is important for Hatch to create income and revenue from its products in order to survive. Building and deploying new features in frequent and continuous way is of equal importance as it would allow us to grow faster and be more marketable.

Given this, it seems that Kanban is the best choice since it is based on the Continuous Delivery approach contrary to other methodologies that are based on time-boxed frames. Kanban also allows priorities to change at any time which will help organize tasks continuously in order to deliver faster.

Defining a strict hierarchy of roles in the engineering team is not possible since it is limited to its team members. This allows everyone to participate in any task, making Kanban a straightforward choice.

Conclusion

In this chapter, we have presented the company and its main departments. We have also presented the context and the problematic behind our project.Additionally, we have studied and criticized the existing; and provided a proposed solutions. We have also described the methodology used to drive the project. For the next chapter, we will discuss the state of art of the design system.

CHAPTER 2

STATE OF THE ART

Introduction

In this chapter, we will begin by introducing UX methodology and the atomic design methodology then we will demonstrate the various design system Principles and we will finish by presenting the UI patterns and what they represent in the design system .

2.1 UX methodology

The idea of user experience predates the acronym UX and is a cycle of user-centered design that seeks to improve upon user research through methods such as surveys and interviews. Customer experience with products and services.

Sometimes looking for a methodology to create something innovative means looking elsewhere. Looking to other fields and industries can sometimes provide inspiration for big discoveries. Given this incredibly messed-up world we've created, it seems quite normal that different disciplines are grappling with the same problem and that we can benefit and be appropriate. It seems Surprisingly, a cluster of different disciplines such as industrial design and architecture have created an ingenious framework for assembling highly complex objects such as airplanes, boats and skyscrapers. Our technology goes back to smaller details of this complex world we have. A natural world that evokes memories of high school life It's just a chemistry class[2].

Chemical reactions are governed by chemical equations, as in Figure 2.1[2] which frequently shows how the atomic components combine to form the molecule. in the model underneath we perceive how hydrogen and oxygen consolidate together to frame water molecules.

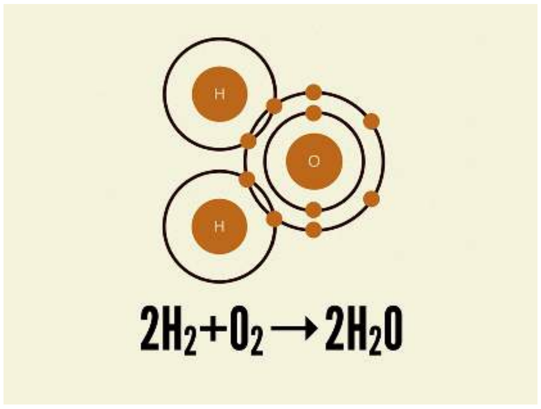


Figure 2.1: An example chemical formula showing that a hydrogen atom and oxygen atoms combine to form water molecules.

In a natural way, the atomic elements combine to form molecules. These molecules can be further consolidated into complex organisms in general. To clarify a little more :

- **Atoms:** The essential building blocks of all matter. Each chemical component has special properties and cannot be decomposed without losing its meaning. They are composed of many small fragments such as protons, electrons and neutrons, but atoms are the smallest functional units.

- **Molecules:** : A collection of at least atoms of at least two atoms joined by chemical bonds. Mixtures of these atoms have unique properties that make them more substantial and functional than atoms.

- **Organisms:** : are gatherings of molecules operating all collectively. These generally elaborate systems can pass from single-celled organisms all route as much as unbelievably present-day dwelling just like the human being.

In summary, atoms are part of collectively to form molecules, which in addition integrate to border organisms. This atomic speculation implies that each one troubles withinside the found out universe may be broken down right into a restrained set of atomic factors provided withinside the parent 2.2[2].

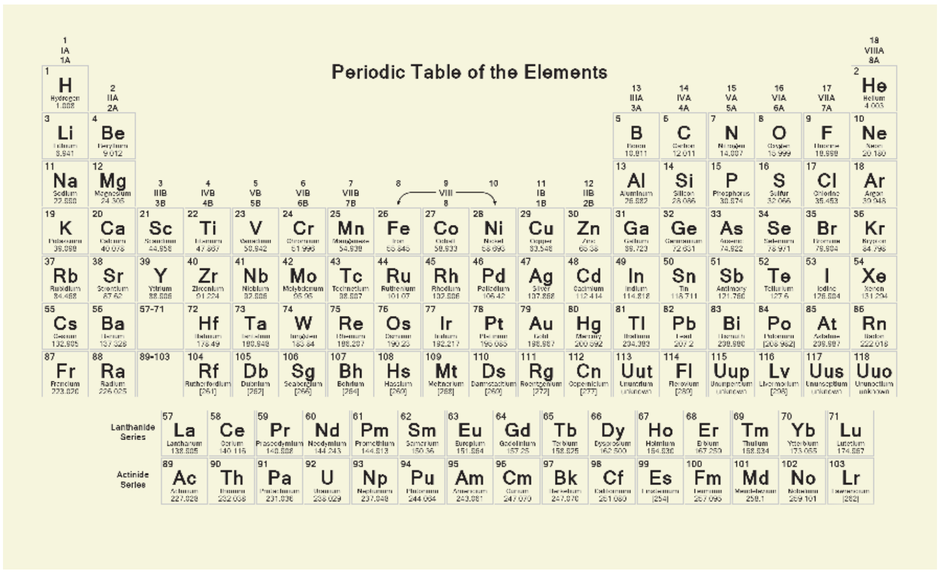

Periodic Table of the Elements
Lanthanide Series
Actinide Series

Figure 2.2: The periodic table of chemical elements.

2.2 The atomic design methodology

After speaking approximately chemistry and the atomic principle withinside the final section, it's time to explain greater what's the factor of bringing up all that. We mentioned how all troubles acknowledged to mankind may be separated right into a finite set of atomic factors. As it occurs, our interfaces may be damaged right into a comparative set of factors. Josh Duck an Australian engineering supervisor has flawlessly verbalized how the whole lot of our web sites and programs are made from comparable HTML factors in his Periodic Table of HTML Elements provided in discern 2.3[2].

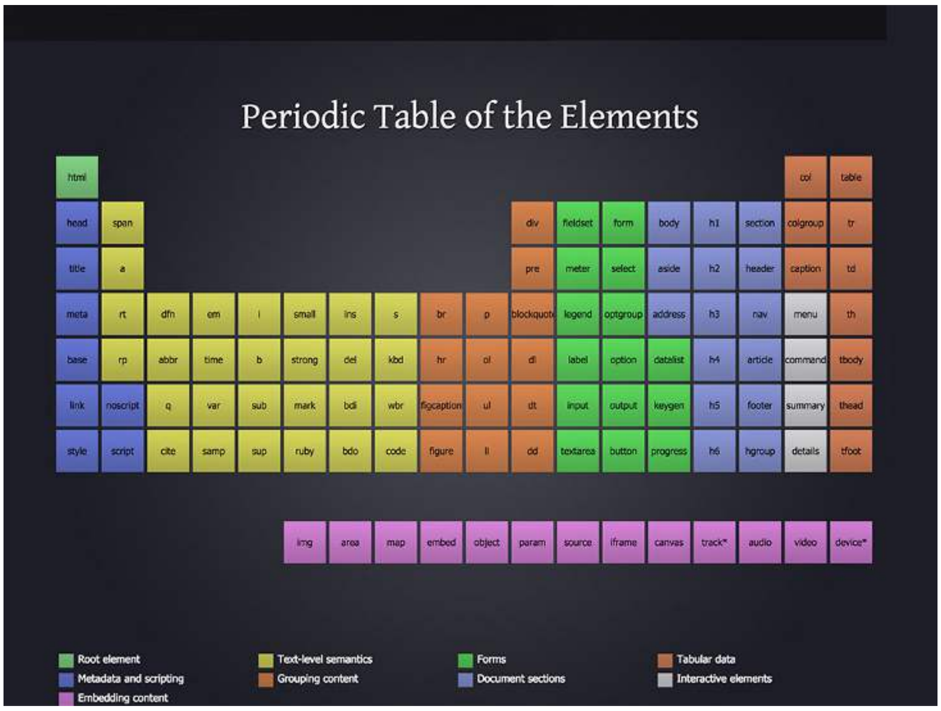


Figure 2.3: The periodic table of HTML elements

Since we commenced with a comparable technique with the aid of using a finite set of constructing blocks, we can via the equal procedure that has to do with the herbal international to shape and create user interfaces. And that is while the Atomic Design enters. The atomic layout is a technique made from 5 awesome degrees cooperating to make interface layout structures in a extra aware and numerous levelled way. The 5 levels are :

- **Atoms**
- **Molecules**
- **Organisms**
- **Templates**
- **Pages**

Atomic Design isn't a linear cycle, but instead an intellectual version to help us with thinking approximately our UIs as each a sturdy complete and a collection of components simultaneously. Every one of the 5 levels assumes a key component withinside the chain of command of our interface shape layout. We have to leap into every segment in greater fine-grained detail.

2.2.1 Atoms

f atoms are the basic building block of the problem, then at this point in the the atom of the interface fills in the basic building block that contains all of the user interface. These atoms contain important HTML elements such as form captions, inputs, buttons, etc. that cannot be further separated for their usefulness.

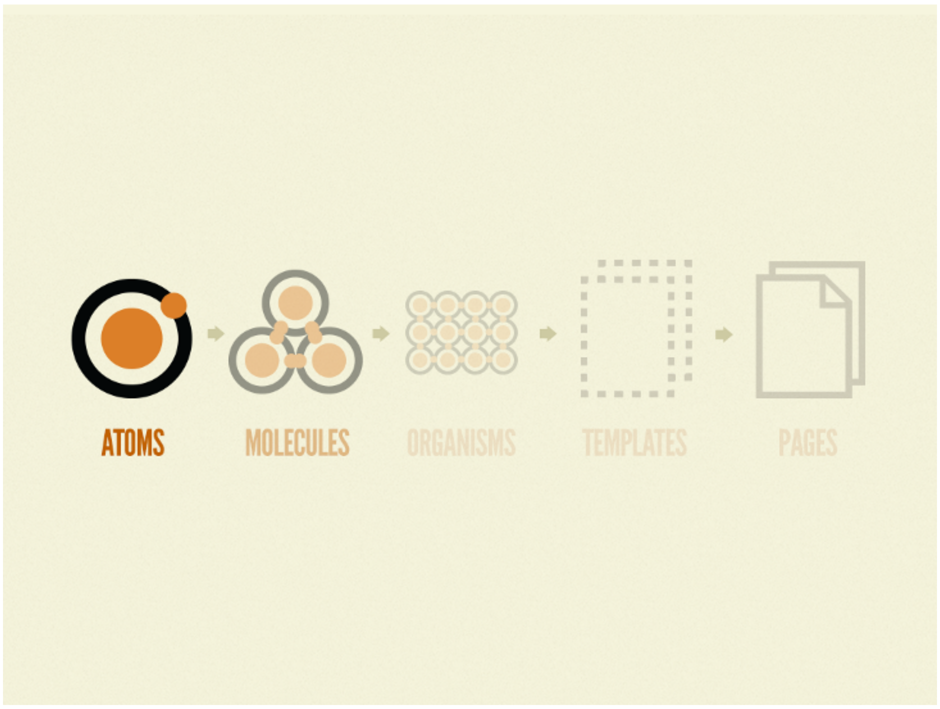


Figure 2.4: Atomic Design elements: Atoms

All atoms in nature have unique properties. A hydrogen atom has only one electron, and a helium atom contains two electrons. Each interface atom also has its own unique properties such as dimensions, font size, background color... These properties allow you to control how the element is applied to the user interface system.

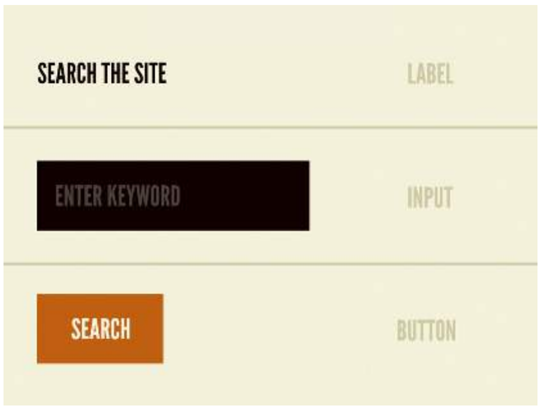


Figure 2.5: Atoms include basic HTML tags (inputs, labels, and buttons)

When it comes to the pattern library, Atoms initially includes all the basic styles, which are useful as a reference when creating and maintaining a design system. In any case, like atoms in nature, interfacial atoms do not exist in a vacuum, but are truly brought to life by the application.

2.2.2 Molecules

In science, a molecule is a collection of atoms that are strengthened together and have different new properties. For example, water and hydrogen peroxide molecules have unique properties and continue unexpectedly despite being composed of similar atomic elements (hydrogen and oxygen).

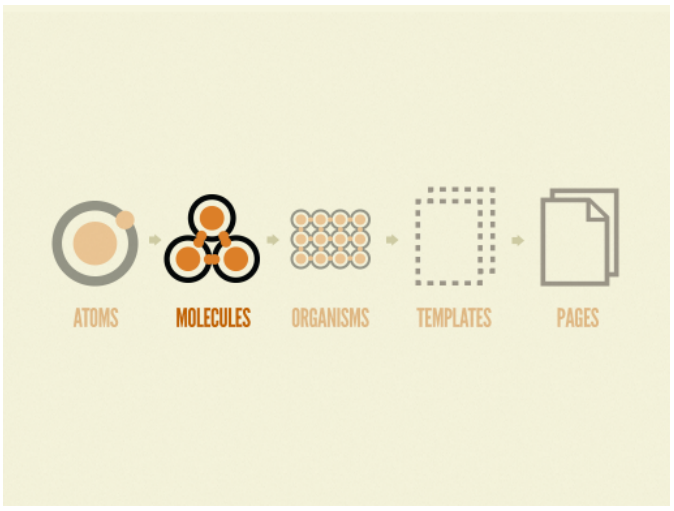


Figure 2.6: Atomic Design elements: Molecules

In interfaces, a molecule is generally a simple collection of user interface elements that work together. For example, form labels, search prompts, and buttons can be combined to form a search form molecule. Once they are connected, these abstract atoms have a purpose out of nothing.

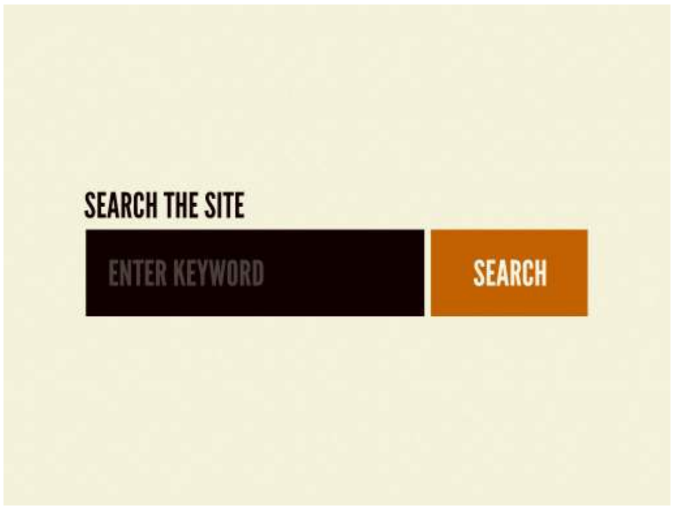


Figure 2.7: A search form molecule is composed inputs, labels, and buttons

Marker atoms now characterize the input. Tapping the button submits form . The result is a simple, versatile, reusable segment that you can drop anywhere you need searchable functionality.

2.2.3 Organisms

Organisms are moderately confusing UI components that consist of collections of molecules and atoms and other organisms. These creatures form different areas of the interface.

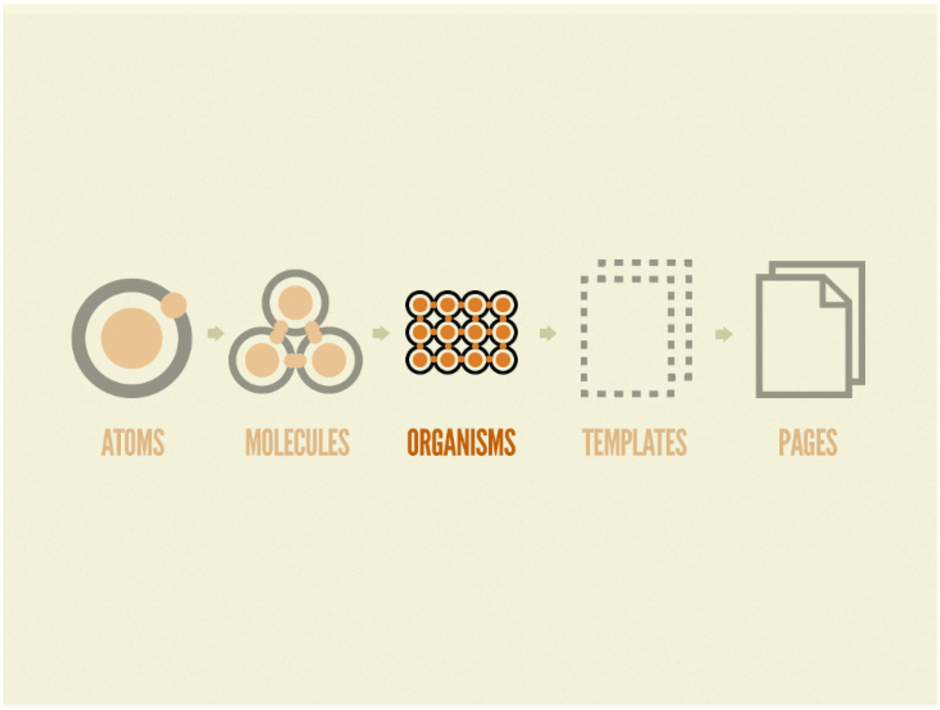


Figure 2.8: Atomic Design elements: Organisms

If you go back to the Molecule search form, you can usually find it in the header of many web applications. So when we put these molecules into the context of the headers in our app, we are talking about header organisms, as shown in the image below.

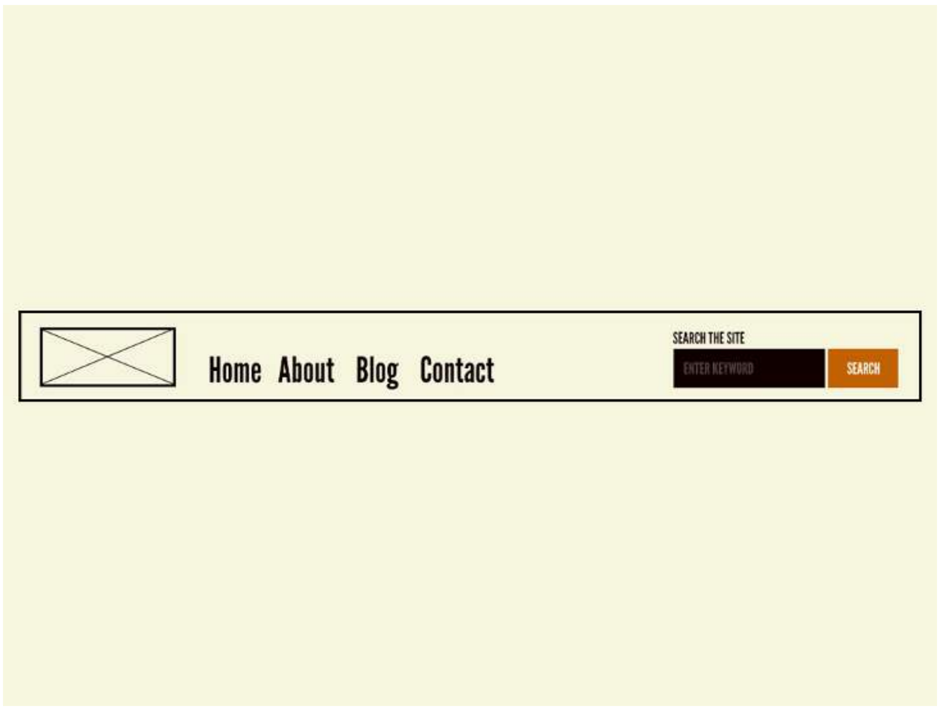


Figure 2.9: Header organism is composed of a search form molecule, logo atom, and primary navigation molecule.

The header forms a separate area of the interface, but contains smaller interface blocks with new properties and functionality. Living organisms can be made up of similar or different types of atoms. A header organization can consist of different elements such as a logo image, primary navigation list, search form, etc. . These types of creatures are present on almost every website available.

2.2.4 Template

Now is the perfect time to say goodbye to chemical analogies. The Atom language provides a useful hierarchy for intentionally building parts of the building system.

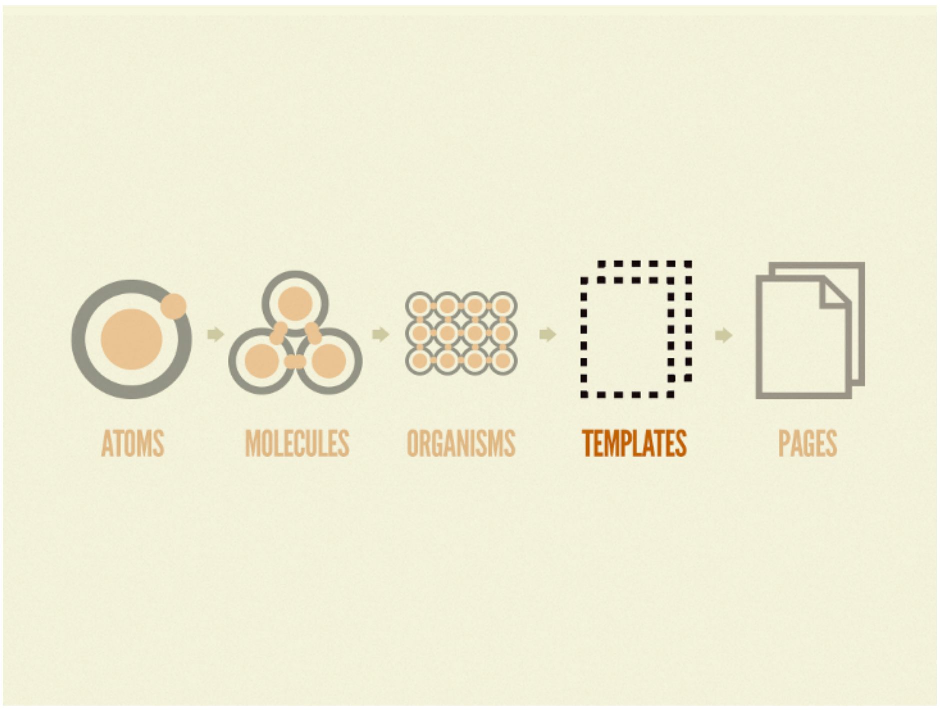


Figure 2.10: Atomic Design elements: Templates.

Formats are web page-degree gadgets that area additives right into a layout and well-spoken the layout's hidden content material shape. To amplify on beyond models, we are able to take the header organism and use it on a touchdown web page layout.

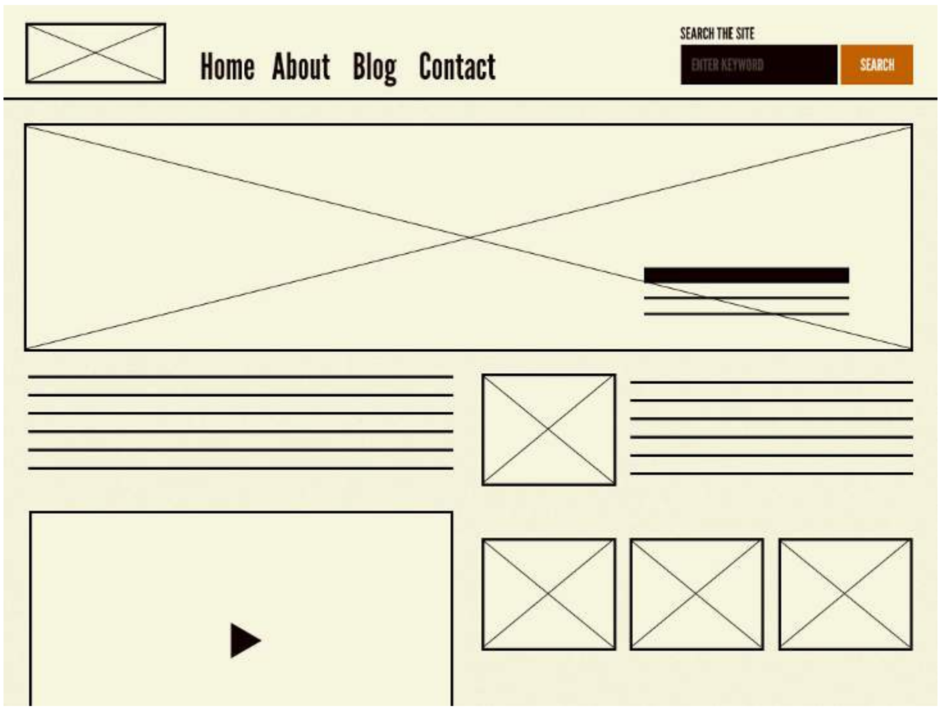


Figure 2.11: Atomic Design elements: Templates Page

The homepage template withinside the determination above suggests all of the critical web page additives running collectively, which offers a placement to those fairly summary molecules and organisms.

While creating an effective layout system, it's primarily to showcase what additives seem like and feature collectively almost about a layout to illustrate the components that characterize a well-running whole.

Another substantial characteristic of templates is they are middle across the web page's essential content material shape instead of the web page's very last content material. Design structures should report for the dynamic concept of the content material, so it's extraordinarily beneficial to verbalize substantial residences of additives like photo sizes and individual lengths for headings and textual content sections.

2.2.5 Pages

A page is an explicit instance of a template that shows what the user interface looks like with the actual content set up. As an extension of the previous model, the landing page template allows text, images, and media to flow into the template to actually display the actual content of the content in real life.

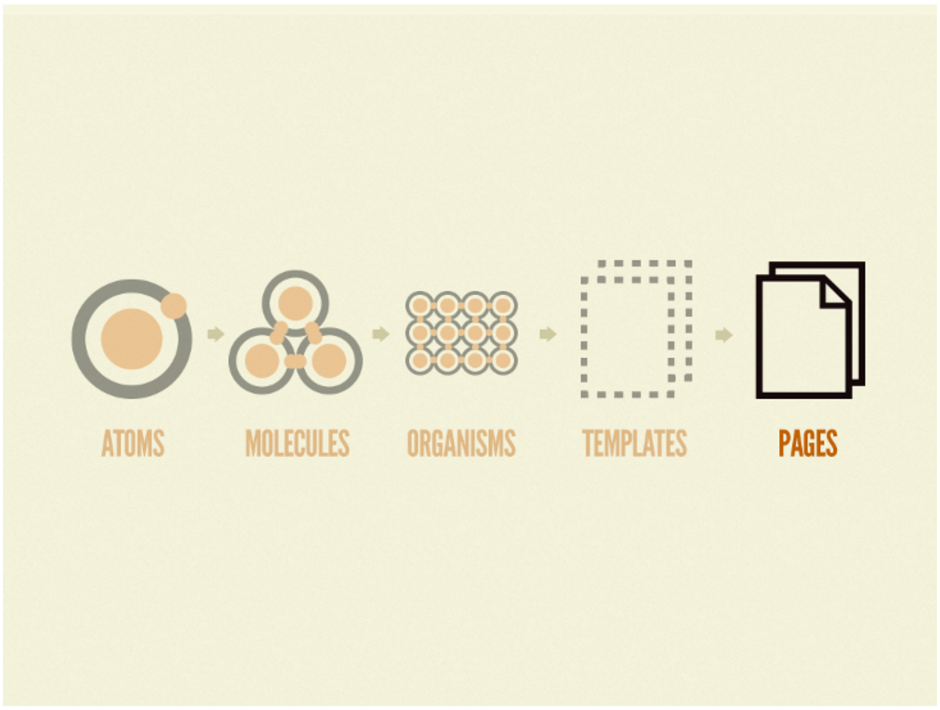


Figure 2.12: Atomic Design elements: Pages

The page phase is the most conservative phase of atomic design and is important for obvious reasons. Overall, this is what customers see and interact with when they visit our products. This is what our partner closes his on. Here you can also see how each of these components fit together to form a fun and useful user interface.

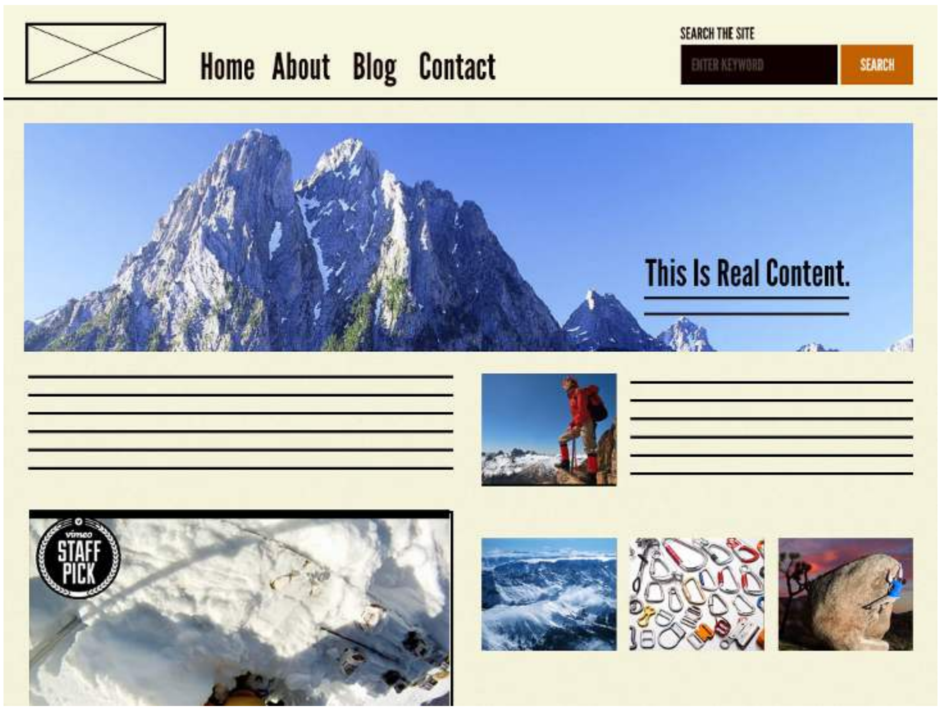


Figure 2.13: The page stage replaces placeholder content with real representative content to bring the design system to life

Besides demonstrating the final interface as our customers will see it, pages are simple for testing the adequacy of the powerful layout system. It is on the web page degree that we're geared up to analyze how every one of these examples holds up while actual content material is carried out to the layout framework. To ensure that the whole thing works because it should. If the solution is no, we will circle again and alter our molecules, organisms, and templates.

Finally, discern 2.14 is a super instance of the way Atomic Design become carried out to local cellular app Instagram.

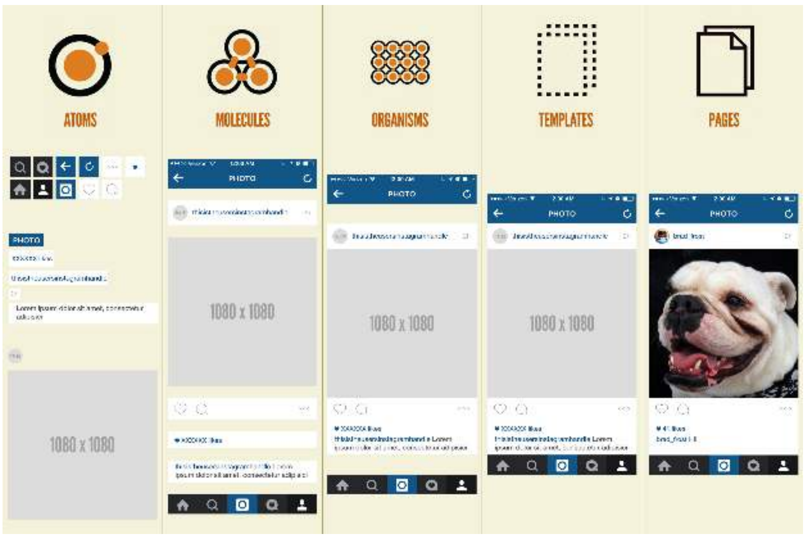


Figure 2.14: Atomic design applied to the native mobile app Instagram

2.3 Design System Principles

Current design systems are the outcome of several lengthy durations of advancement within-side the way we compose front-end code. A few years in the past maximum websites had been laboured with single-use, wasteful, delicate, and conflicting front-quit codebases. Through hard-received revel in and severa lengthy durations of joint attempt and emphasis, front-quit builders have constructed up extra current practices to compose and type out the code. Presently, there's a blast of front-end systems and gear to guide us to write better, extra possible HTML, CSS, and JavaScript.

Despite the improvements and gear in the back of them, a powerful layout device should follow sure centre values. Consistency, how additives are constructed and overseen follows a predictable pattern. Self-contained, it ought to be dealt with as an impartial dependency. Reusability, constructing factor that may be used later in specific contexts. Accessibility is important, utility constructed with the layout device could be utilized by as many humans as possible. And the layout device ought to be robust, and equipped to be carried out in any product or platform with good overall performance and less bugs.[3]

2.3.1 Consistency

The first and most important task is to characterize and record the standards of the system and ensure compliance by all. Once you have established well-documented code guidelines and best practices, your designers and engineers will definitely be able to leverage and contribute to your design system.

Code Style Guides

Code grammar is the arrangement of rules for structuring and styling code (e.g. curly braces are always on new lines). Code semantics provide standards for correct code (e.g. ordering of CSS property assertions). In any case, don't get into fights over tabs and spaces. The most important thing is not to achieve fictitious perfection, but to get a reliably constructed code.

Automating Code Style

To enforce our code standards and achieve consistency in our system, contributors must write code that follows the rules through linting and tooling. Linting is a process that analyzes code and throws errors and warnings if the code doesn't conform to syntax rules, is corrupted, corrupted, or corrupted. There are several different tools on the market such as: CSSLint and StyleLint for CSS and JSHint, ESLint for JavaScript. These tools can be run manually during the development process, or triggered automatically as automated pre-commit hooks before the code is checked into source control.

2.3.2 Self-Contained

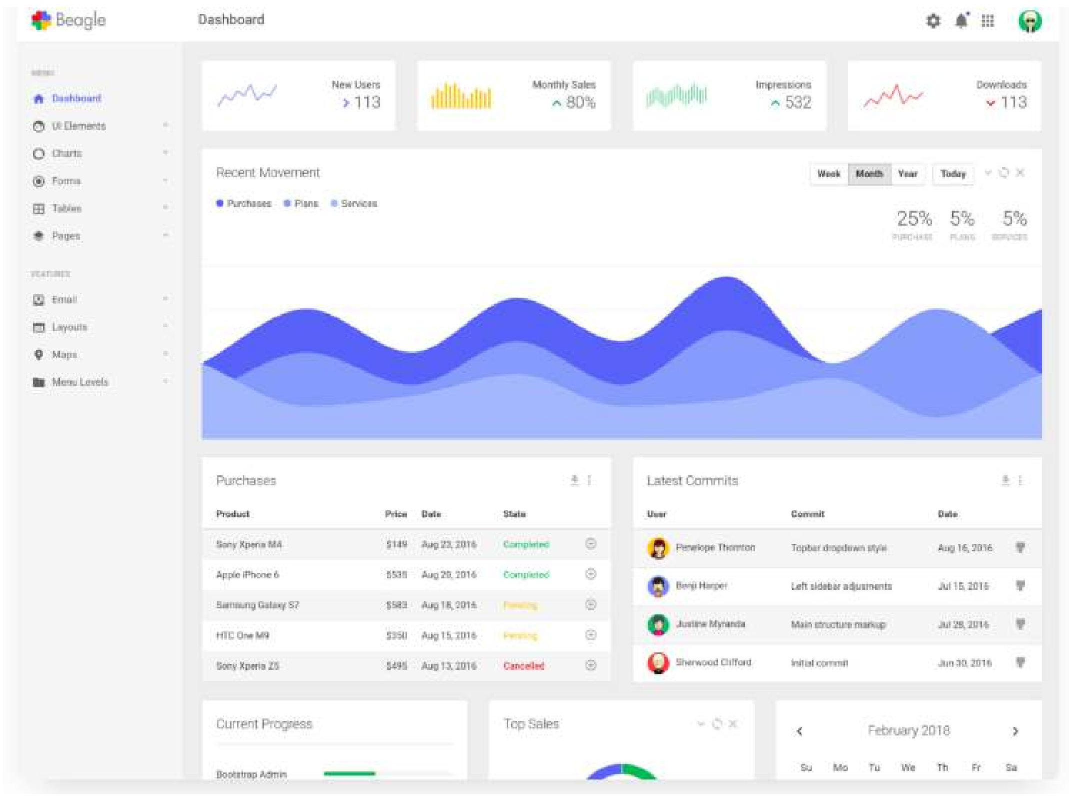
Our design system should reside in a separate source control repository from the main code base. It takes a lot more work to get it working, but the separate repository has many long-term benefits.

- Having versioned releases of our code can be needed later on
- Ability to share code across multiple teams and products
- Ease to build components isolated which will allow their usability
- Providing a good infrastructure for a powerful front-end testing architecture
- Forming a foundation for a living style guide website

An independent design system archive serves as a single source of truth. At this point, there are only locations that feature components shared as standalone dependencies in other codebases. All usages refer to a licensed implementation, so changes in one place propagate throughout the system. In a perfect world, all the code for each piece would be co-authored within the system: CSS, JavaScript, HTML formatting, and documentation would all live in the same location (probably in the same directory). The closer the parts are to each other, the easier it is to track and monitor the dependencies between the pieces of code, and the easier it is to keep them updated and up-to-date.

2.3.3 Reusability

Successful design systems are tremendously reusable. Bootstrap, the most-utilized front-end library ever, powers a massive range of webweb sites because it changed into architected in mild of usability. Building additives to be reused in diverse settings is imperatively significant, but difficult to development nicely-made additives excessively engaged for a unmarried use case or excessively inflexible, and customers will land up making their styles as we will see in determine 2.15[4] and the instance of Admin Template primarily based totally on Bootstrap.



Purchases

Product	Price	Date	State
Sony Xperia M4	$149	Aug 23, 2016	Completed
Apple iPhone 6	$535	Aug 20, 2016	Completed
Samsung Galaxy S7	$583	Aug 18, 2016	Pending
HTC One M9	$350	Aug 15, 2016	Pending
Sony Xperia Z5	$495	Aug 13, 2016	Cancelled

Latest Commits

User	Commit	Date
Penelope Thornton	Topbar dropdown style	Aug 16, 2016
Benji Harper	Left sidebar adjusments	Jul 15, 2016
Justine Myranda	Main structure markup	Jul 28, 2016
Sherwood Clifford	Initial commit	Jun 30, 2016

Figure 2.15: Beagle – Responsive Admin Template based on Bootstrap

To be reusable and adaptable, styles must be particular, composable, nonexclusive, and adaptable. Composable additives may be putten collectively to create new styles, they must deal with distinctive use cases. Flexibility is essential too for additives in order that be prolonged to paintings in distinctive contexts. A essential pleasant exercise withinside the software program engineering international is Don't Repeat our-self called DRY. When distinctive blocks of code are doing an equal job, the opportunity of insects is accelerated with the aid of using and the desired boom as nicely as the amount spent preserving functionality. The predominant cause of a very good layout gadget is to DRY up our improvement and decrease duplicating blocks of code with the aid of using making them reusable.

2.3.4 Accessibility

For a long time, accessibility or a11y has been misunderstood as a building product for a small group of supporting innovation customers, too often dismissed as overly complex, boring or "not for our customers". is the planet of , not just for a small group, but for 15% of people overall. In the United States alone, 56.7 million people suffer from a variety of permanent or temporary visual, hearing, and mental disabilities.

The Web is fundamentally designed to work for all people, whatever their hardware, software, language, culture, location, or physical or mental ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability.

WEB ACCESSIBILITY INITIATIVE (WAI) W3C

Fortunately, attitudes are changing and our industry recognizes the wider meaning of accessibility. Opening our website to users with disabilities enhances the experience of each individual who visits our website. If that's not enough, you can improve your SEO by improving your site's accessibility.

2.3.5 Robust

A robust design system has a solid set of tests behind it. Testing gives confidence in your code and drives adoption. Users will find that they can redesign or change the user interface without breaking it in unexpected ways. Additionally, our design system is very well suited to provide powerful testing capabilities for front-end code.

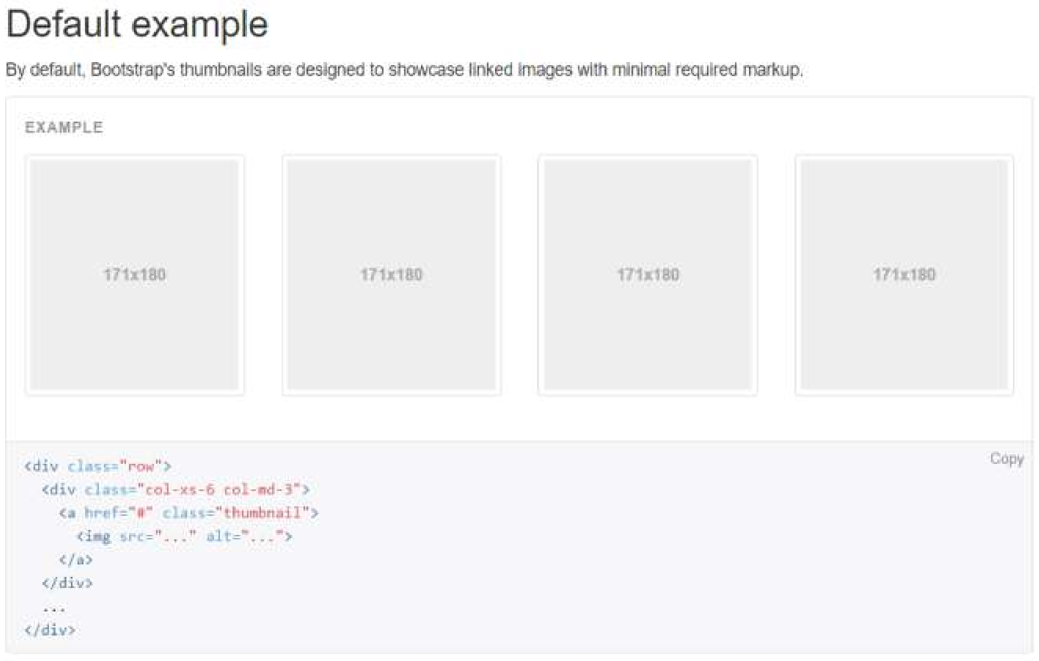
2.3.6 Documentation

A design system calls for many pointers to be documented. These pointers, alongside with live code examples might be the number one documentation for the theming machine.

Provides clean and well-described policies for a way every element need to be used, required properties, and fashion options. The consumer need to be capable of getting solutions to special questions, what, when, in which and why with every documented element. Including all implementation, policies are essential to guide users on a way to use the element correctly. This way that magnificence names, properties, and code snippets had been delivered with legitimate examples.

2.4 UI patterns

Digging deeper into UI and user interface patterns, we find them in the digital circle of websites, applications, and native mobile applications. They give a language to talk about interactive design. They provide function, interaction, and purpose. UI patterns document reusable parts of an interface with a purpose. To further understand his UI pattern for , see the image below and the bootstrap example on the thumbnail. component[5].



Default example

By default, Bootstrap's thumbnails are designed to showcase linked images with minimal required markup.

```
<div class="row">
  <div class="col-xs-6 col-md-3">
    <a href="#" class="thumbnail">
      <img src="..." alt="...">
    </a>
  </div>
  ...
</div>
```

Figure 2.16: Bootstrap's thumbnails component (default example)

As you can see in Figure 2.16, the thumbnail pattern or component (developers' preferred term) provides small image previews of the in the collection. Each image has a link to a larger size (high resolution image) resource. If the component is an item or product preview, the link takes the user to the details page. It can be a small preview of the video and will redirect you to the appropriate player to watch it (e.g. wetube homepage). Also, each pattern should have an important function. In the example, thumbnails have the following features:

- Small images
- Linked resources
- Collection to represent

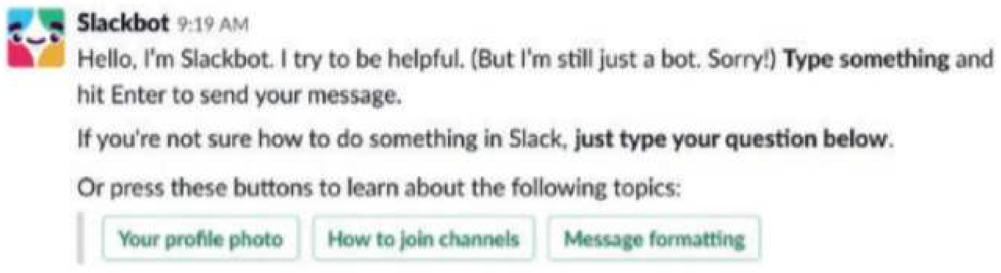
UI patterns are more abstract than visual styles. Patterns are usually recognizable by their visual similarity, which shows that these elements are generally not very natural: patterns represent behaviors that can be effectively separated from recognizable visual representations. For example, you can give your thumbnail collection a solid and sensational visual style, or give an understated and subdued touch.

2.4.1 UI Patterns benefits

Knowing the patterns and understanding the decisions made on them allows us to tap into the growing intelligence of every generation and company that has invested time in these patterns without rehashing problems that have already been solved. Done. The small, reusable UI solutions found in these patterns can form together to create a more powerful and visceral experience.[6]

2.4.1.1 Design efficiently

Understanding styles can help us to design effectively through swiftly perceiving the most perfect device for the task, expertise the estimation of some solutions, and fixing the biggest variety of problems at once. We can take the instance of autocomplete seek input, it helps customers navigate the content material of a internet site and realize what they're attempting to find with out understanding the precise time period or with the incorrect spelling. This permits them to pick a end result after typing some characters with out losing time on locating the precise name, getting to know this autocomplete thing will permit us to determine whilst we want to apply it and it'll be the identical with all UI patterns.



Slackbot 9:19 AM
Hello, I'm Slackbot. I try to be helpful. (But I'm still just a bot. Sorry!) **Type something** and hit Enter to send your message.
If you're not sure how to do something in Slack, **just type your question below**.
Or press these buttons to learn about the following topics:

Figure 2.17: Screenshot of Slackbot from Slack

2.4.1.2 Consistency

Using familiar patterns helps promote consistency. Directly control elements by dragging and dropping them using the familiar drag-and-droplet concept. A common use of drag and drop is to transfer an image by dragging it from the PC's file system to a drop target area in the UI. Most to-do apps let you drag and drop to do items to reorder or move them. As drag-and-drop interfaces become more inevitable across her web, users are more certain to understand the most effective ways to collaborate with them.

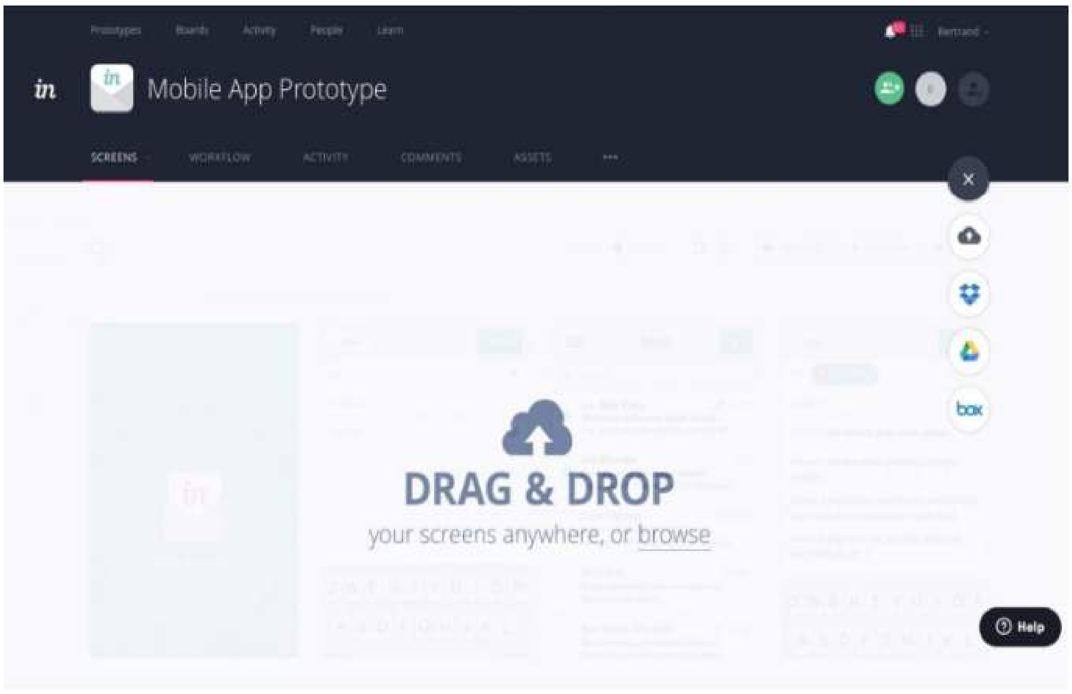


Figure 2.18: Screenshot of drag and drop option from Invision

2.4.1.3 Reusability

Patterns are shared so design solutions can be reused both visually and in code. Visual repetition helps build consistency into the interface, making a learning experience for users. Code reusability also saves time, allowing existing functionality to be refactored and enhanced rather than creating a new from scratch. A pattern combines many design decisions to address a problem, but programming encodes those decisions, ensures that the pattern does not repeat itself (DRY), and makes each design decision Do it only once for him.

2.4.2 UI Patterns properties

The UI pattern recipe is composed of three main ingredients :

- **Solution**: what the component do?
- **Problem**: why we need this pattern? What the problem that the user is facing?
- **Context**: when to use the pattern?

Based at the thumbnails thing we are able to recognize extra of those components one with the aid of using one. The answer is describing the thing proposes a group of small photo previews linked to large-sized resources. The hassle is that the consumer is having a hassle navigating a huge series of content material and simply choosing what he wants. The context is the consumer wishes a preview of the photo earlier than choosing or downloading a larger record or looking an extended video. To recognize extra, we are able to see in determine 2.19 an instance from the Pinterest domestic web page that makes use of thumbnails to show the gathering of photographs they may be offering.

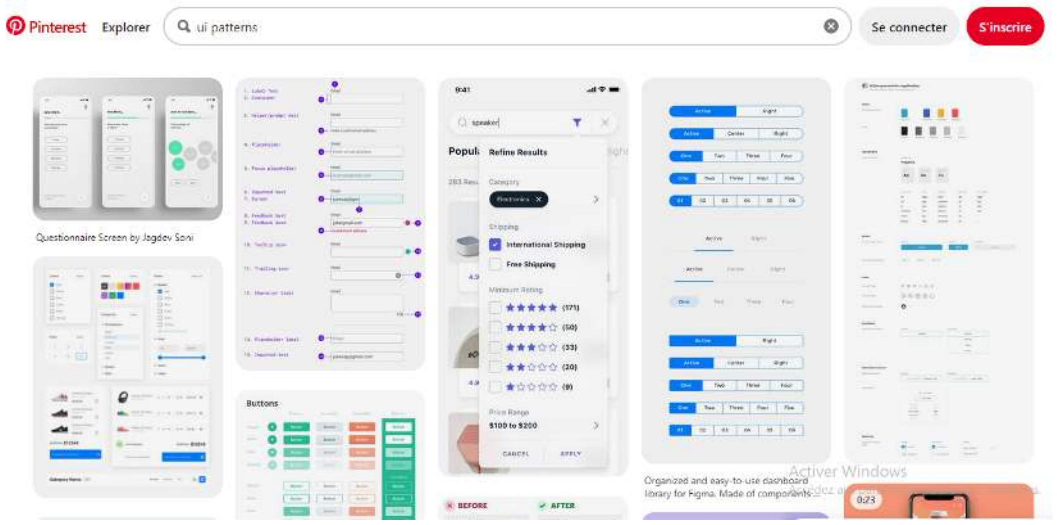

Pinterest
Explorer
ui patterns
Se connecter
S'inscrire
Questionnaire Screen by Jagdev Soni
Refine Results
Buttons
Organized and easy-to-use dashboard library for Figma. Made of components
Activer Windows

Figure 2.19: Screenshot from Pinterest

On the off threat that Pinterest constantly stacked high-decision photographs at their full length as opposed to thumbnails, that might gradually down the consumer experience. Pinterest wishes to introduce thumbnails to make surfing smoother and assist their customers to find out extra ideas. In a great series like that, figuring out what photo will display subsequently or if it's something the consumer wishes to locate in detail. By the usage of thumbnails, we are able to unexpectedly peruse a larger association of alternatives earlier than specializing in a selected item.

2.5 UI patterns in design systems

To understand what this section is doing, you first need to understand what a pattern library is. A pattern library is a collection of patterns used to teach and improve design decisions. It contains a reusable solution to issue that focuses on interaction and UX components. As digital experts, we may be familiar with other pattern-related design and code assets in the world: style guides, style manuals, brand manuals, identity guidelines, frontends. Style guides, laws, etc. For the most part, these are complementary ideas that work together beautifully. Otherwise, they are collectively called "pattern libraries" or "design systems". To familiarize yourself with the concept and answer the question of when and how to use it? You need to understand the difference between them.

2.5.1 Front-end style guides

Front-end style guides are extremely just like styles in actual existence internal an organization, transported with code snippets, design resources, and something else vital to complete everyday design and improvement errands influencing the frontend of a product.

2.5.2 Living style guides

Living style guides allude to publications which are in a country of concord with the manufacturing environment. For instance, if we alternate a element in a dwelling fashion manual and it's going to alternate in manufacturing over our entire site. They are made to present sufficient area for design thinking sharing approximately a few factors inclusive of typography desicions and retaining the tips on the equal stage with its real execution.

2.5.3 Code style guides

Code style guides or code requirements often centre across the code arranging and naming conventions of a product engineering team, for example, no matter whether or not they use tabs or areas to indent code and the way they call strategies. Code fashion publications are often very separated from design matters. Accordingly, they're often placed away independently from design-orientated fashion publications and design machine resources, housed in a codebase README web page or a code storehouse's wiki.

2.5.4 Documentation

Practically speaking, rather than making sample libraries and documenting styles from scratch, agencies make design systems that encode styles with a specific visual design language in factor libraries with documentation. Some of the time they join the equipment and belongings that cope with the wider sample but as soon as in a while, they do describe an summary sample, they describe a specific execution of a sample of their specific domain. With regard to growing and documenting new styles in a design system, we must focus the efforts on that one maximum vital one withinside the business, or the styles that reason the maximum dispute withinside the organization. Introducing the factor, bringing up the great practices, list the properties, and the way the person can customise primarily based totally on his needs. For example, while and the way to use a hyperlink, the props for a hyperlink factor can be the URL of the hyperlink and the way it is going to be showed (identical web page or new tab) and the person can customise the fashion associated with the factor (Changing the shadeation of the hyperlink and the movement while the mouse is over it).

2.5.5 Assembly

When we've an unfinished model of all styles, we are able to chip away at supplying them. A viable design device is knowledgeable through various contributors. Imagining a designer's inclination approximately builders the usage of a design device whilst it's miles horrible and untouched through the designers that could have the abilties to make certain its usability. Likewise, builders feeling approximately being pressured to depend upon a UI Kit this is continuously out of sync with manufacturing patterns wherein a few factors are previous and a few aren't healthy to be became out yet. As such, all events contributing and the usage of the design device need to have the liberty to apply it and upload to it. Some alternatives that are to be had to make certain the styles storage :

- **On cloud**: Having a folder on the cloud with the patterns collection is easly accessed but it might lack versioning.
- **Code repository**: : The most preferred option by developers, this option will ensure versioning and full control with code security for updates.
- **Internal website**: a normal website with commenting on the features.

2.5.6 Versioning

The designs, code, and primary UI styles will alternate after a few time. It's now no longer continually viable to show every pertinent alternative on the equal time, consisting of shifting a website beginning with one front-end framework to an extra effective or more recent one. As such, the design gadget may also have several versions: one the use of the antique framework for maximum styles and one the use of a more recent one for upgraded interactions. Like any software program bundle, we should use semantic versioning to denote patches, minor modifications, and full-size breaking modifications in the design system.

Conclusion

In this chapter, we presented the UX methodology and the atomic design methodology and we discussed the different principles of the design system then we talked about the UI patterns and what they represent in the design system.In the next chapter we will go in depth with the architecture of our solution.

CHAPTER 3

REQUIREMENTS ANALYSIS AND SPECIFICATION

Introduction

In this third chapter we will start with the functional and non-functional requirements, and we will introduce the main actors of our system then we will detail some use cases and scenarios using diagrams . Finally, we will present the architecture used during the development process of our project.

3.1 Requirements Analysis

Performing a needs analysis is a method of identifying valuable features that the system should have. The solution that is proposed must address both the functional and non-functional requirements outlined below.

3.1.1 Actors

In every system we can find one or more main actors that will interact with it during different phases of its life cycle, we identify three main actors:

- **Design team :** the design team will use the system to contribute with new designs and visualise the existing designs with their documentation from the storybook.
- **Development team :** the development team will use the system to contribute new components and to use the existing components from the component library using the storybook as the library documentation.
- **Client :** the client will use the storybook to visualise the future vision and direction of their product.

3.1.2 Functional specification

In the scoping phase of a digital project, the functional specifications present what the developed product will do:

- The system should provide a component library that contains existing modifiable components and pages.
- The system should provide a storybook that contains the documentation of the designs and components and a live demo of the several UX scenarios of the components.

- The system should provide an automated process to encourage the developers to contribute new components and designers to contribute new tokens.

3.1.3 Non functional specifications

In addition to meeting the functional specification that have been identified, our solution must also meet a number of other quality standards in order to be of the highest possible quality:

- **Maintainability :** Maintaining the code is one of the crucial parts when running a company since there are always developers working on it and in a clean code environment it would be easier to determine where the bug is or where the piece of code that is causing the slow downs and a good way to do that would be naming variables with names that are indicative of their behavior or their content as well as commenting the code.
- **Performance :** In terms of performance, the solution should have a minimal latency.
- **Availability:** The solution must be readily available for use in the production environment.
- **Scalability :** The solution must be capable of accommodating the growth and change of its constituents.
- **Security :** The solution must provide maximum security through the identification of users who have the right to access to the component library and its documentation.

3.2 Diagrams

Diagrams are a simplified way of representing information or how something works visually, they have been used historically by ancient egyptians and even older civilizations.

3.2.1 Global use case diagram

In this subsection, we will represent the main functionalities and features that are defined on our project through a global use case diagram that will give us a global overview about how our system is working and what it's providing for us.

The following figure illustrates the main axes of our system. Yet, it enables us to have a clearer vision about what the system is going to implement.

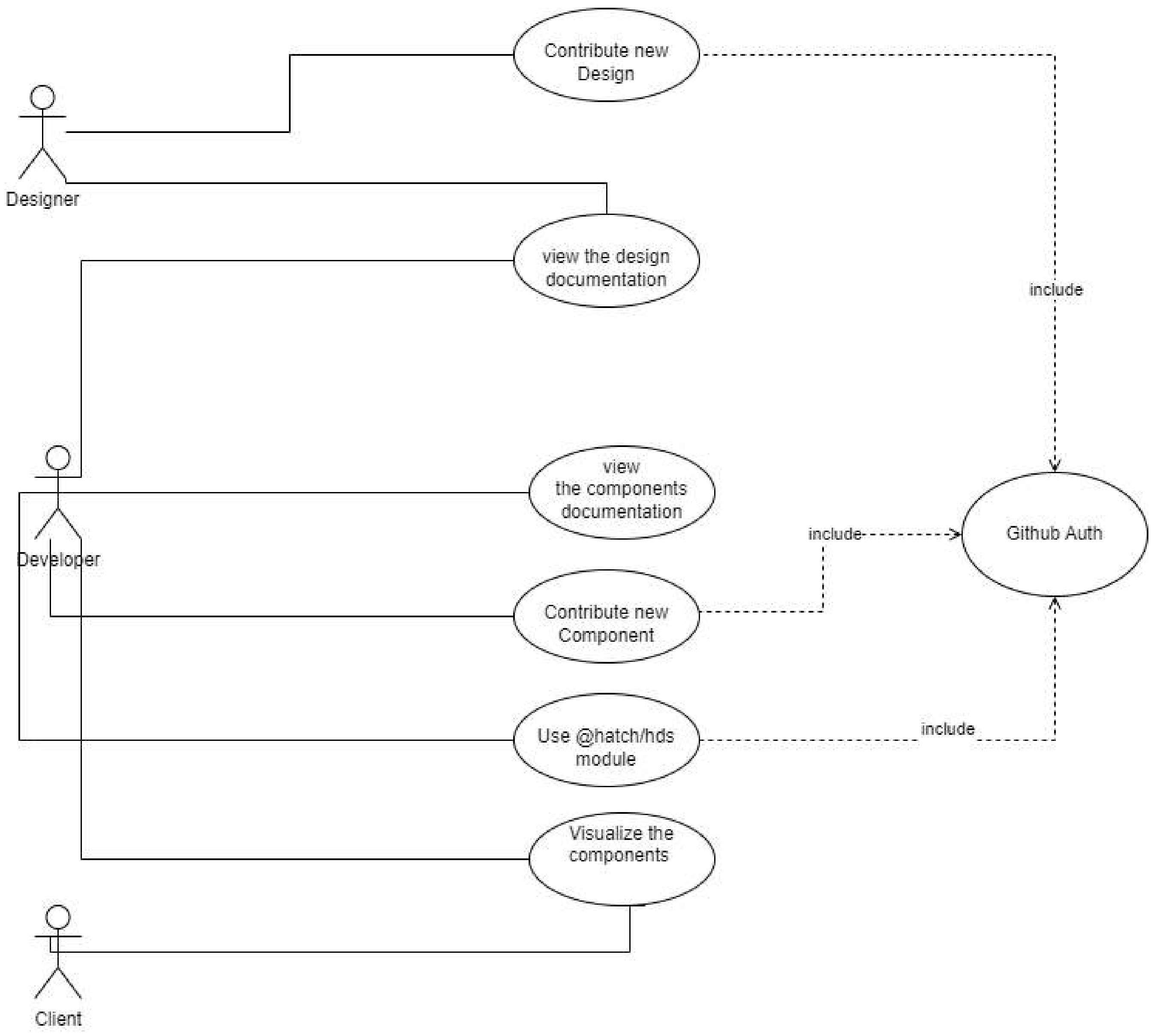


Figure 3.1: Global use-case diagram

3.2.2 Deployment diagram

The deployment diagram is the diagram that shows what software is being used on different nodes in the form of components and how these nodes communicate with each other, in our case we have a N tier architecture and N=4.

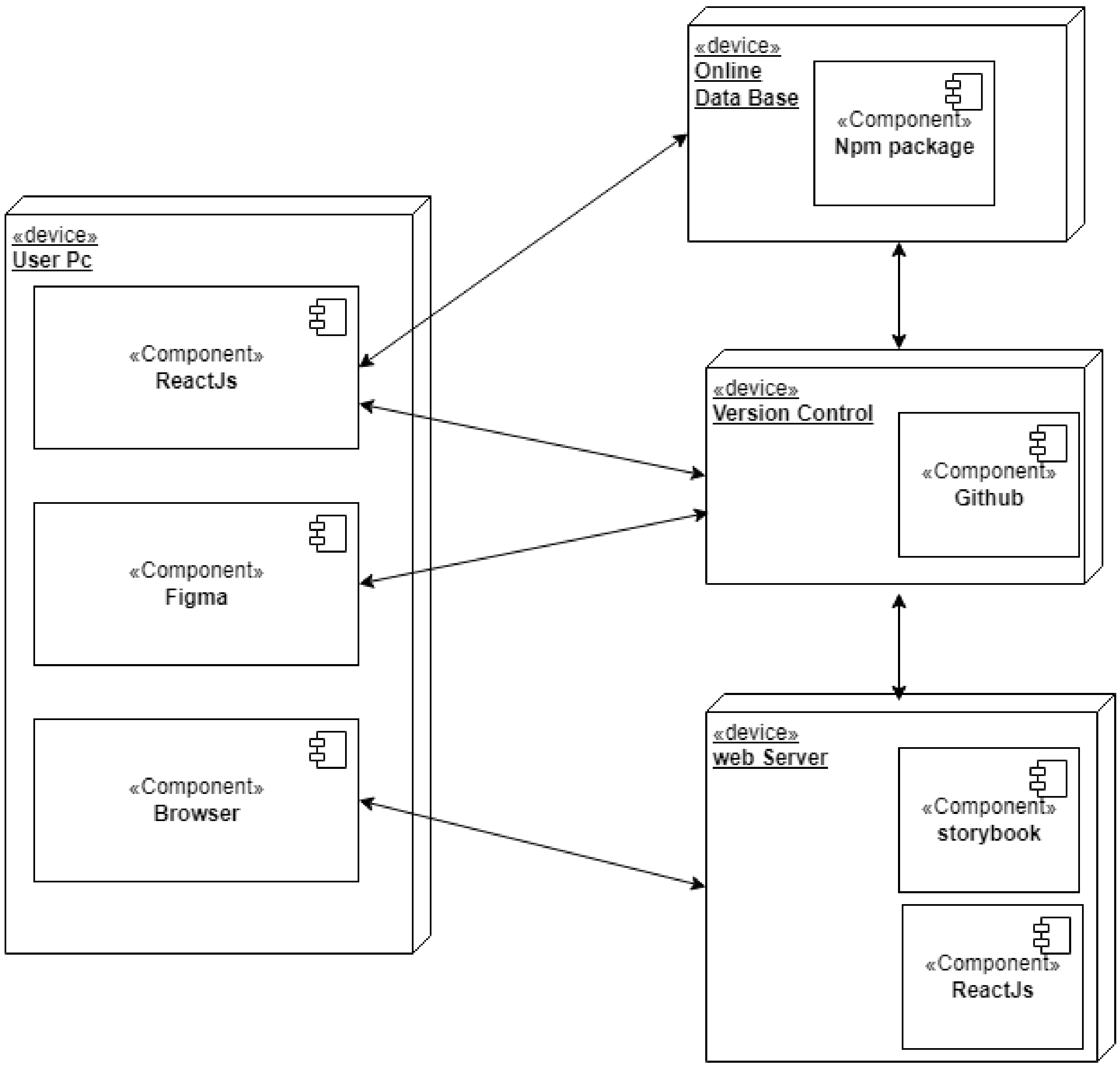


Figure 3.2: Deployment diagram

3.3 Architecture

The architecture of software outlines how the many components of an application will be organized and interacted with one another. As a result, during the design phase, this is one of the initial tasks in software development. It's usually done by a software architect or a solution architect, and it's an important part of the development process.

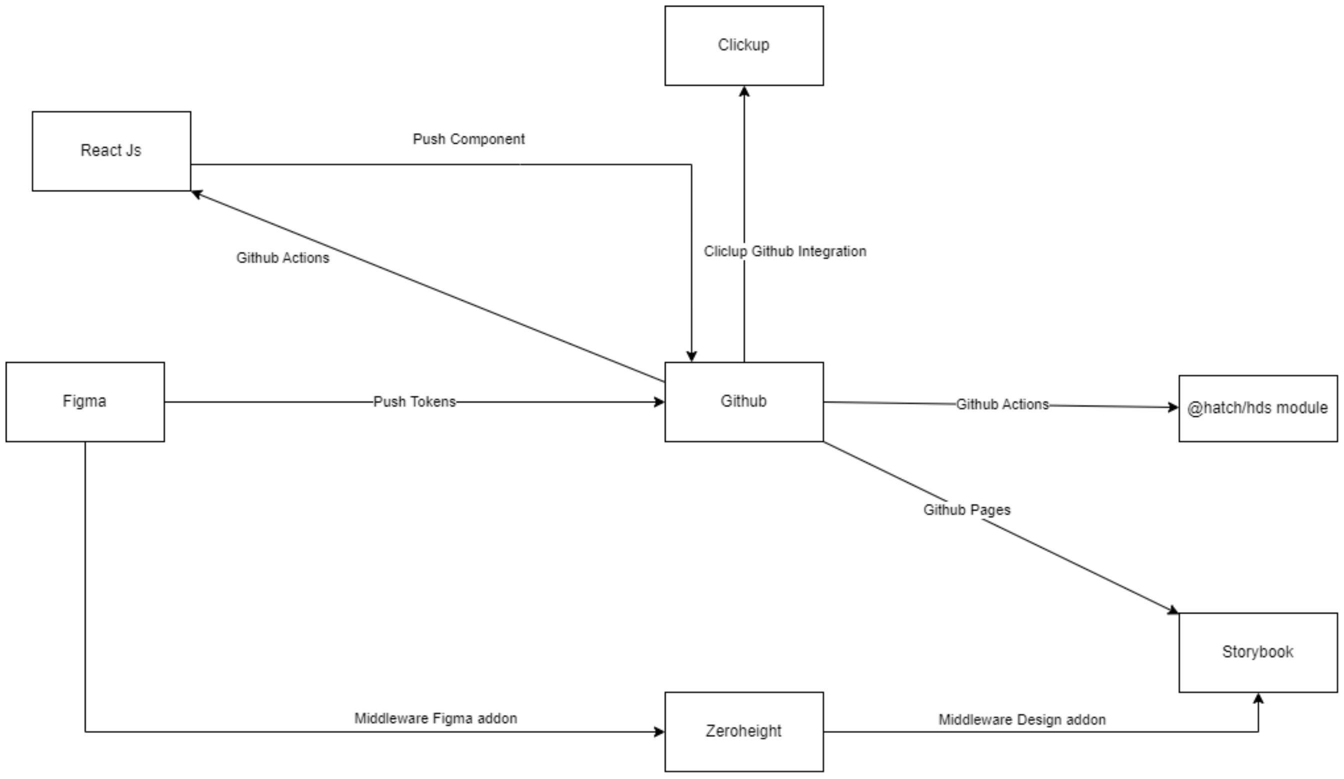


Figure 3.3: Logic architecture

Like discussed earlier, the solution we proposed consists of an automated system to encourage both design and development teams to contribute to the project and the output of this system is the component library and its documentation; the storybook. for this purpose we use the CI/DI practices to assure the continuous integration, delivery and deployment of our system.

3.3.1 CI/DI

Continuous integration, delivery and deployment are software development practices born out of the DevOps movement. They make the process of building, testing and releasing software more efficient and get working software into the hands of users more quickly than traditional methods. Done well, a CI/CD pipeline enables teams to deliver working software at pace and get timely feedback on their latest changes.

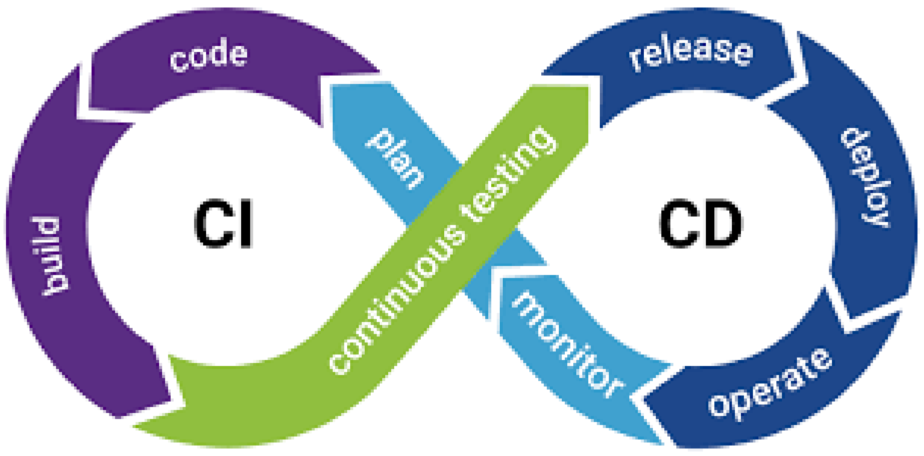


Figure 3.4: CI/DI logo

To perform this practices we used GitHub actions and GitHub pages as features from the version control we used which is GitHub.as the figure 3.6 shows this is an example of pipeline using these tools.

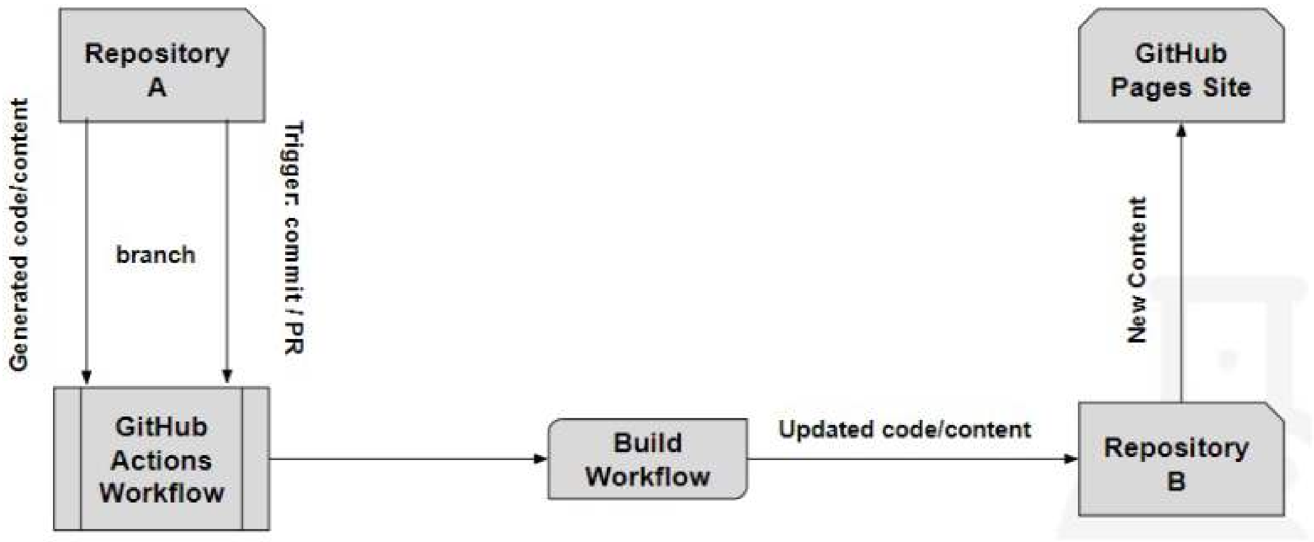


Figure 3.5: CI/DI Pipeline

Figure 3.7 shows a more detailed pipeline using GitHub actions.

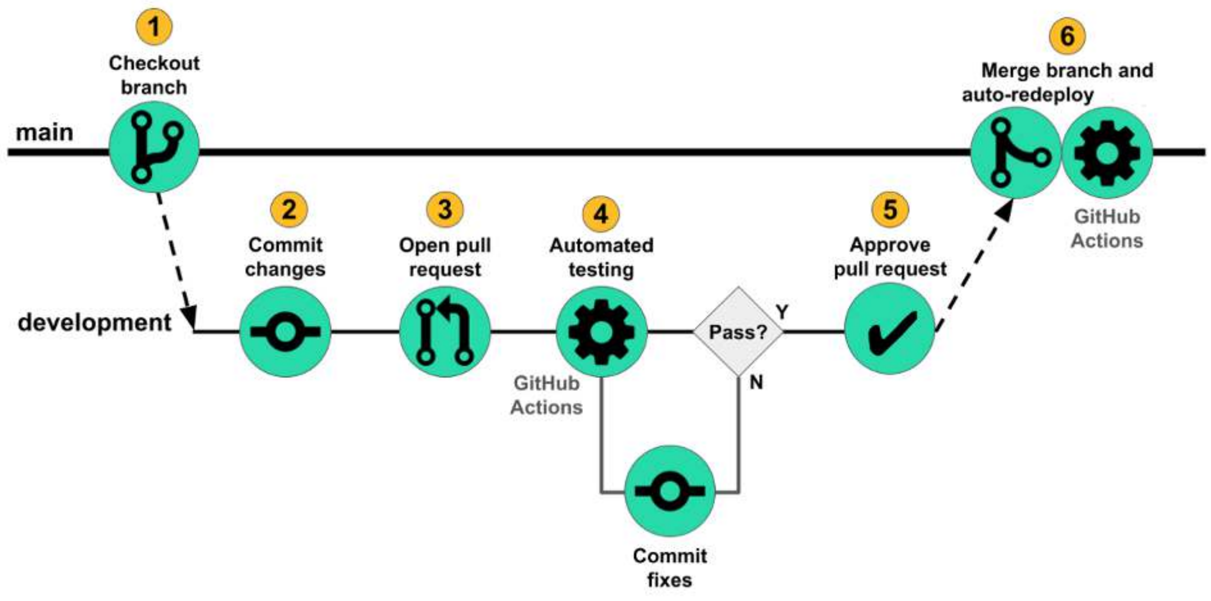


Figure 3.6: GitHub Actions Pipeline

After explaining our system principals, we will return to explain our system architecture and how the process work. we will divide the process into two phases : the Integration phase and the deployment phase.

3.3.2 Integration Phase

The overall design of the Integration phase is shown in figure 3.8.

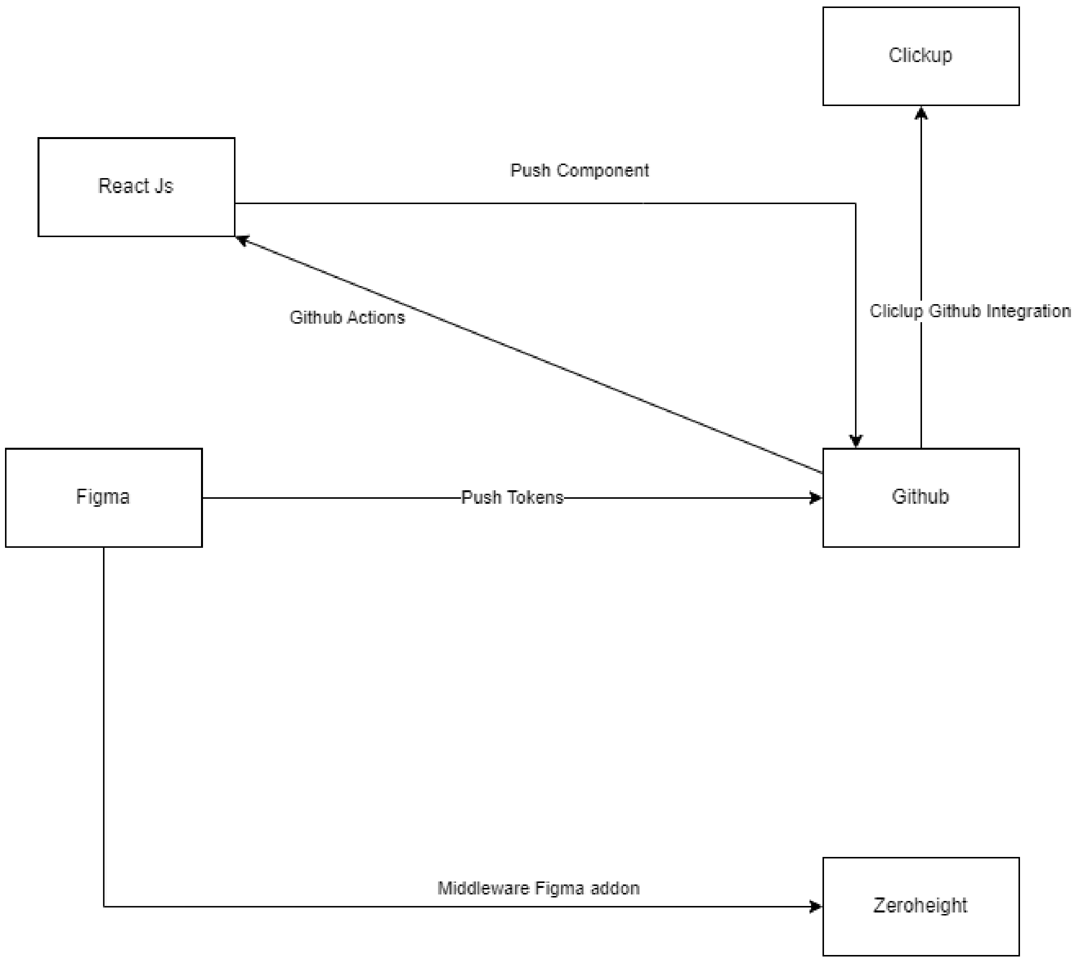


Figure 3.7: Integration phase

Designer contribute new UI:

Figma is the tool that our design team use to design, using Figma tokens,after finishing every new user interface or theme, the designer will push the tokens that he used to our GitHub repository,that push will trigger a workflow that will start running a JS script to convert the tokens to a CSS variables using two libraries(Figmatokensconverter [7] and style dictionary [8]),the the workflow will push the CSS variables to our repository.

The designer could also deploy his user interface documentation to the Zeroheight using the Figma addon middleware.

Developer Contribute new component: as we discussed how our GitHub actions pipeline work, when the developer finishes integrating a component using the design documentation from the Zeroheight and the CSS variables, he will push his work on his new branch and start a pull request, this PR will trigger several workflows :

- **ClickUp GitHub Integration**: Using the ClickUp public API, we will see all Github activity related to our task directly in ClickUp to help us Automatically track commits, merges, and pull request–all within task activity feeds [9].
- **Branch Storybook deployment**: Using the Github Pages and Storybook we will deploy a PR instance storybook to let both the reviewer and designer review the component working.
- **Eslint Review Action**: Using ReviewDog checker you have the ability to check the code quality with the option of reporting the code issues and suggesting the changes that you need to implement.
- **Visual Regression testing action**: This action uploads images and report as workflow artifact. The HTML report is generated by downloading and comparing the artifacts from the branch where the pull request will be merged. and it is deployed using the GitHub pages.

3.3.3 Deployment Phase

The overall design of the Deployment phase is shown in figure 3.9.

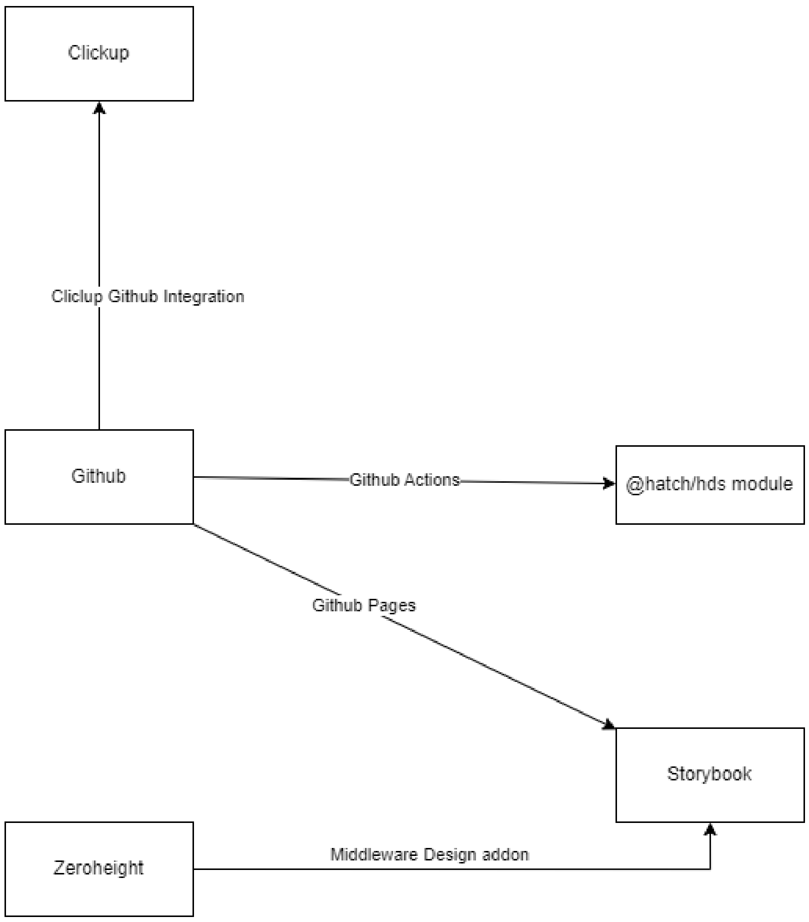


Figure 3.8: Deployment phase

After approving the pull request the development component branch will be merged to the main branch and the ClickUp task status will be changed to done,this merge will trigger the final workflow of our pipeline:

- **Remove Branch instance Action**: this action will remove the storybook branch instance.
- **Deploy General instance Action**: this action will deploy the new version of the storybook general instance using the GitHubPages.
- **Publish package Action**: this action will publish a new release of the npm package @hatch/hds using the Verdaccio plugins.

Conclusion

All throughout this chapter, We have clarified the functional, non-functional, and technical requirements of the basic concepts related to headless design system and to our project in general and we gained a deeper understanding of how the software components interact with each other. In the following and last chapter, we will finish up by talking about the implementation phase.

CHAPTER 4

IMPLEMENTATION AND WORKING ENVIRONMENT

Introduction

This chapter deals with the technical aspects related to the implementation of our system. We start by specifying the different tools, software, technologies and hardware used and adopted during the development process and the implementation of our system. We will then approach the main functionalities offered by our system through the illustration of the different interfaces that will summarize the global functioning of our system.

4.1 Working Environment

4.1.1 Hardware

During this project, we used an ASUS workstation with the following characteristics:

- **Processor:** Intel I5 7th Gen
- **RAM:** Intel 16 GB
- **Storage:** 512 GB SSD
- **Operating system:** Windows 64bit

4.1.2 Software

4.1.2.1 Clickup:

ClickUp is a project management tool that we use in our everyday life.It integrates flawlessly with our working methodologies.Providing a ticketing system and progress tracking with feedback return from product owners.All in one super easy and intuitive application.

Figure 4.1: ClickUp logo

4.1.2.2 VS Code:

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C, Java, Python, PHP, Go) and runtimes (such as .NET and Unity)[10].

Figure 4.2: Vs Code logo

4.2 Technologies

The rise of web development, gave us the opportunity to have several frameworks and technologies. In this section, we will be presenting the technologies used to elaborate our design system.

4.2.1 ReactJS

The React js framework is an open source JavaScript framework and library developed by Facebook. It is used to quickly and efficiently build interactive user interfaces and web applications with far less code than vanilla JavaScript.

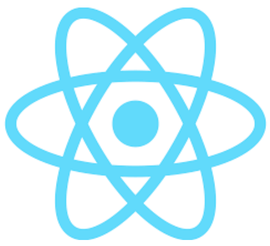


Figure 4.3: React logo

4.2.2 Storybook

Storybook is an open source tool that aids in the development of UI components, providing isolation and documentation, and the ability to create components in live preview with the ability for users to interact.

Figure 4.4: Storybook logo

4.2.3 HTML

HTML (HyperText Markup Language) is the code used to structure web pages and their content. For example, content can be organized using a series of paragraphs, bulleted lists, or tables of images and data.

Figure 4.5: HTML logo

4.2.4 CSS

Cascading Style Sheets, also known as CSS, is a style sheet language used to describe how markup language documents should be rendered in a web browser. Designed to separate presentation and content by incorporating layout, colors and fonts.

Figure 4.6: CSS logo

4.2.5 Javascript

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. It is high-level and often just-in-time compiled.

Figure 4.7: Javascript logo

4.2.6 Verdaccio

Verdaccio is a simple local private NPM registry that requires no configuration. You don't need an entire database just to get started. Verdaccio comes with its own small database out of the box, and the ability to proxy other registries (such as npmjs.org) also introduces caching of downloaded modules on the side. For those who want to increase their storage capacity.[11].

Figure 4.8: Verdaccio logo

4.2.7 Zeroheight

Zeroheight is a web-based application which allows companies to prepare and maintain a design system to have uniform design models in their products. The collaboration feature allows team members to maintain style guides and edit design principles, style elements, and best practices to be followed by the whole team. The company provides a plugin for Sketch, which can upload components to the Zeroheight styleguides. Zeroheight then extracts elements like color, typography, and icons to be used in the design system document. Zeroheight can also import design elements from Figma web application[12].

Figure 4.9: Zeroheight logo

4.3 Version Control

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

4.3.1 Git

In 2005, Linus Torvalds, who is also recognized for inventing the Linux kernel, created and developed Git, a version control system. It's a development tool that enables a group of developers to keep track of changes to source code over time. In a special form of database, version control software keeps track of every modification made to code.

Figure 4.10: Git logo

4.3.2 Github

Github is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

Figure 4.11: GitHub logo

4.3.2.1 GitHub Pages

GitHub Pages is a static site hosting service that takes HTML, CSS, and JavaScript files straight from a repository on GitHub, optionally runs the files through a build process, and publishes a website.

Figure 4.12: GitHub Pages

4.3.2.2 GitHub Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.

Figure 4.13: GitHub Actions

4.4 Implementation

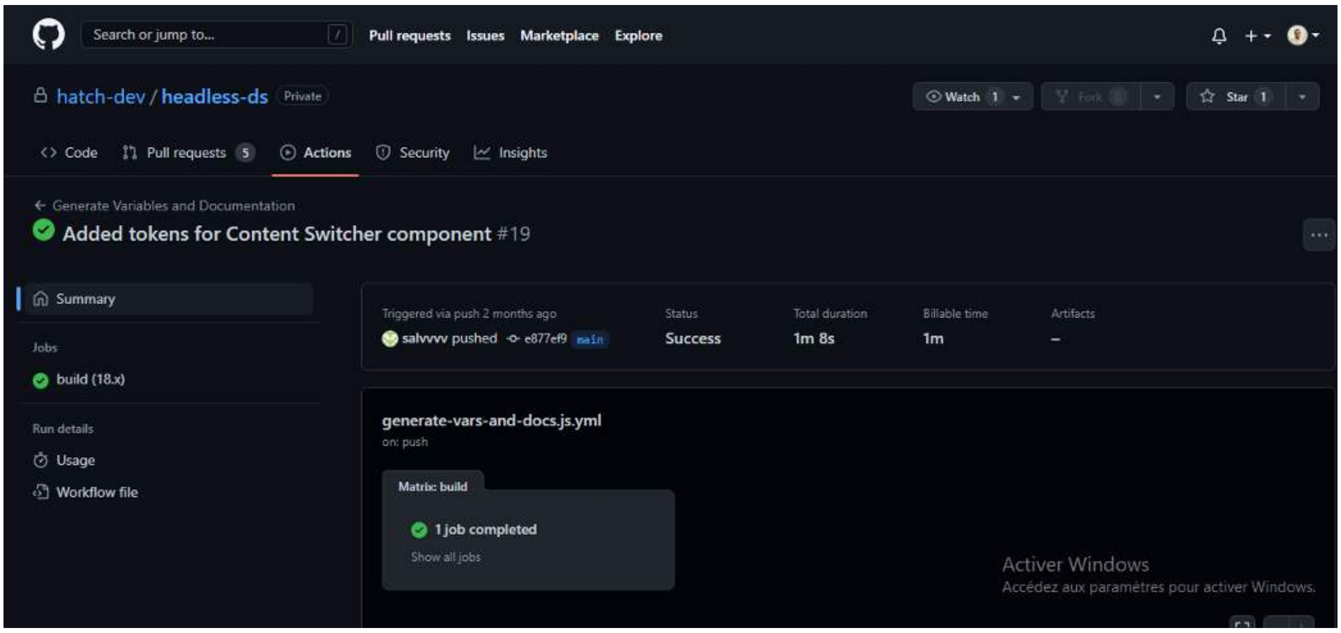
The implementation phase is mainly about showing images of the system and walking the reader through it without actually having to run the whole System. we will divide our section to two main subsections:

- The first subsection we will show the system workflows concerning the contributing process.
- The second subsection we will show the storybook and the component library implementation.

4.4.1 Contributing Process

4.4.1.1 Designer contribute new UI

Starting with the Figue 4.14, representing the GitHub actions triggered,when the designer pushes new token.



Added tokens for Content Switcher component #19

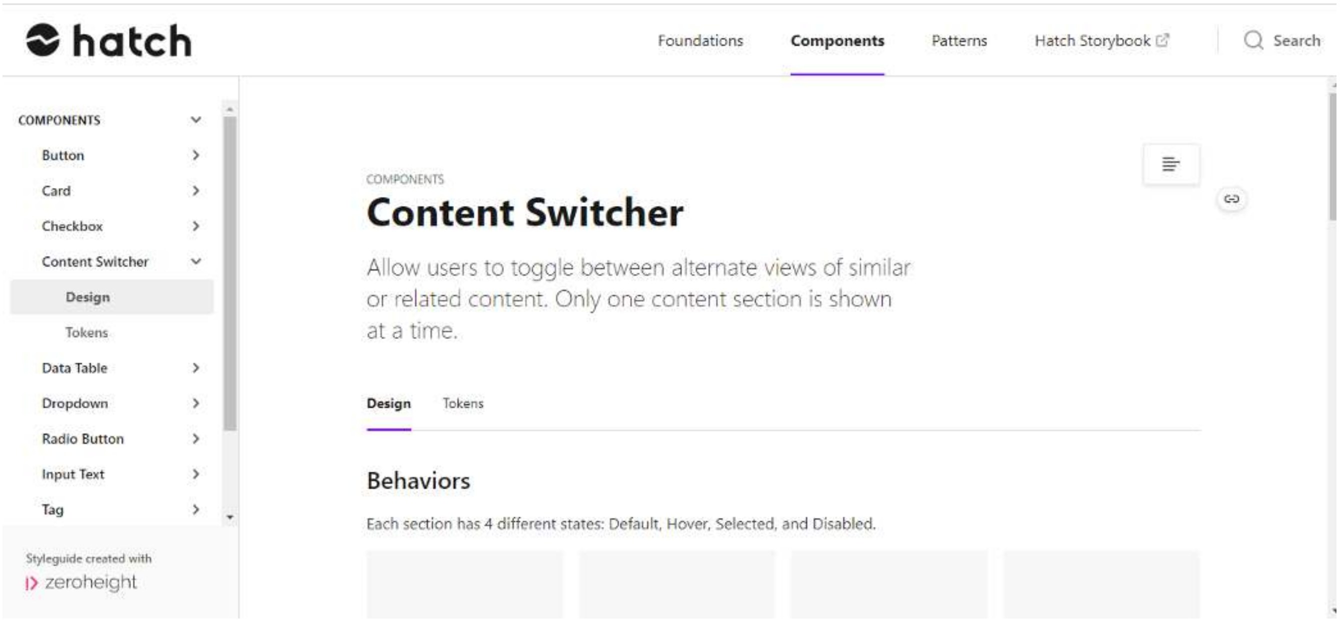
Triggered via push 2 months ago	Status	Total duration	Billable time	Artifacts
salvvvv pushed e877ef9 main	Success	1m 8s	1m	–

generate-vars-and-docs.js.yml
on: push

1 job completed

Figure 4.14: Push Token WorkFlow

Figure 4.15 shows the ZeroHeight design documentation that provides the developer with all the design behavior and its tokens.


hatch

Foundations | Components | Patterns | Hatch Storybook | Search

COMPONENTS
- Button
- Card
- Checkbox
- Content Switcher
  - Design
  - Tokens
- Data Table
- Dropdown
- Radio Button
- Input Text
- Tag

Styleguide created with zeroheight

COMPONENTS

Content Switcher

Allow users to toggle between alternate views of similar or related content. Only one content section is shown at a time.

Design | Tokens

Behaviors

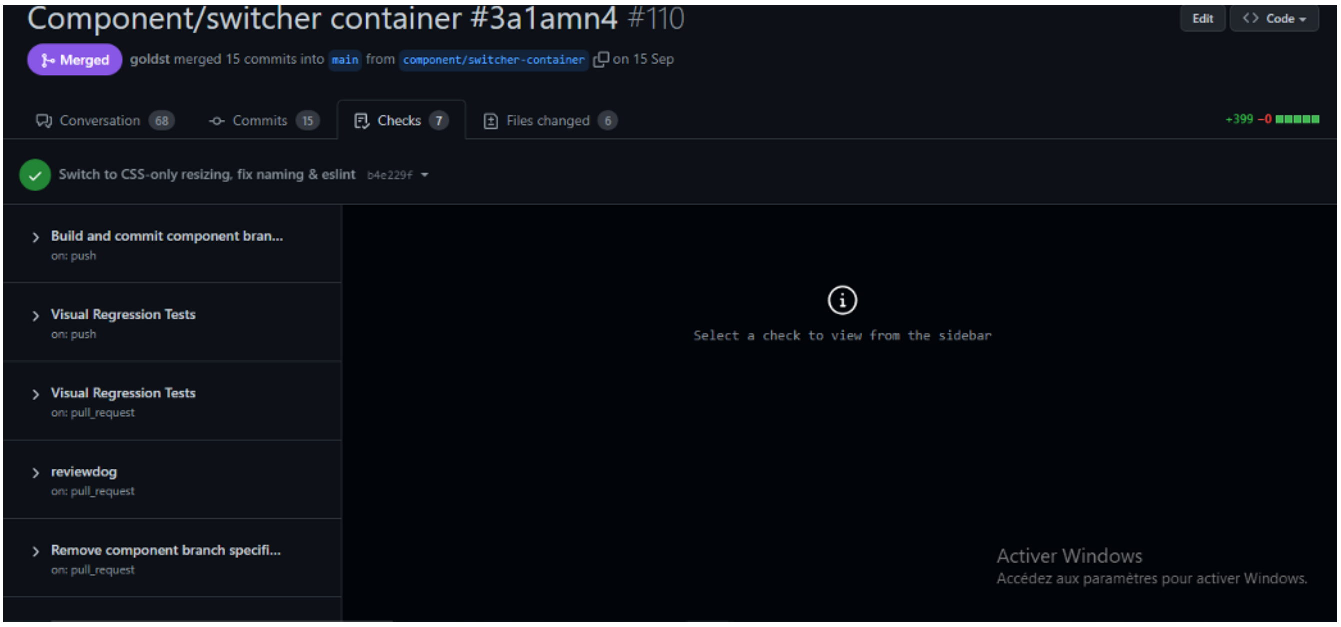
Each section has 4 different states: Default, Hover, Selected, and Disabled.

Figure 4.15: Zeroheight Design documentation

4.4.1.2 Developer Contribute new component

When the Developer Contributes a new component he will first make a pull request for the new component to be reviewed before merging it to the main branch.

Figure 4.16 shows the actions triggered by making this new pull request.


Component/switcher container #3a1amn4 #110

Merged goldst merged 15 commits into main from component/switcher-container on 15 Sep

Edit | Code

Conversation 68 | Commits 15 | Checks 7 | Files changed 6 | +399 −0

Switch to CSS-only resizing, fix naming & eslint b4e229f

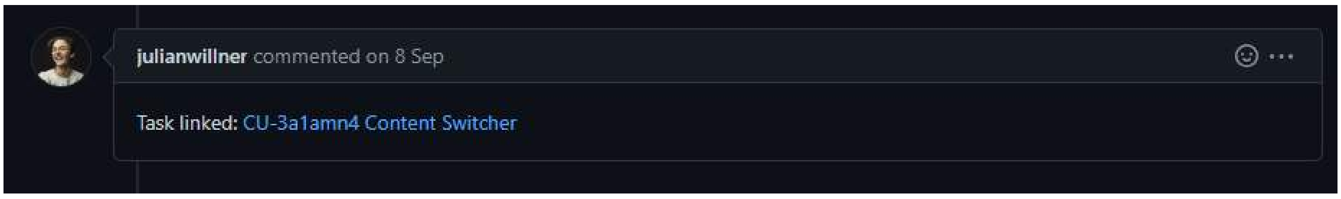
- Build and commit component bran... on: push
- Visual Regression Tests on: push
- Visual Regression Tests on: pull_request
- reviewdog on: pull_request
- Remove component branch specifi... on: pull_request

Select a check to view from the sidebar

Activer Windows
Accédez aux paramètres pour activer Windows.

Figure 4.16: Branch Work Flow

Starting with the Figure 4.17 that shows the Click GitHub integration that will provide a management role for tracking the task.

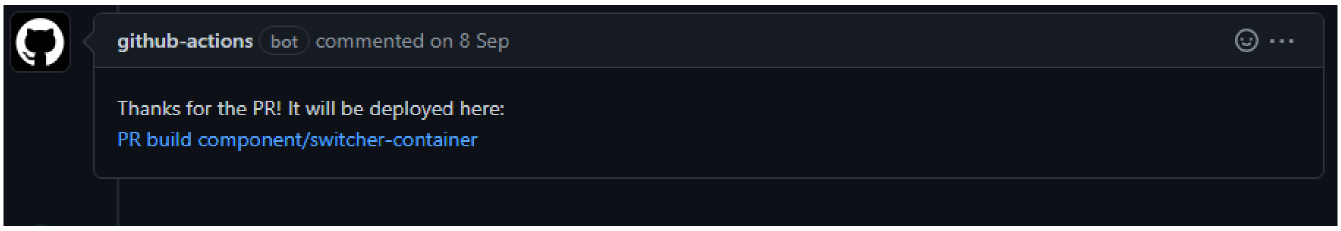


julianwillner commented on 8 Sep

Task linked: CU-3a1amn4 Content Switcher

Figure 4.17: ClickUp GitHub Integration

Figure 4.18 shows the Storybook branch instance that will be deployed to facilitate the review of the component.

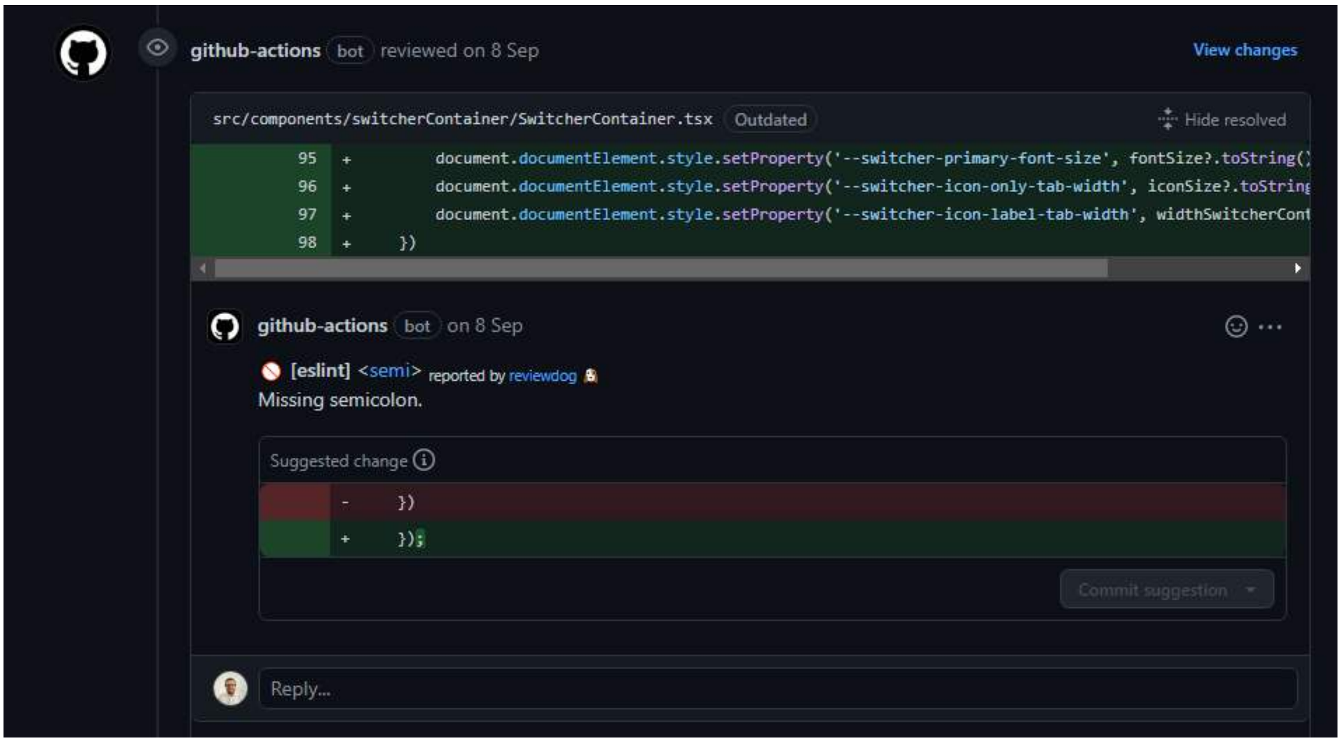


github-actions bot commented on 8 Sep

Thanks for the PR! It will be deployed here:
PR build component/switcher-container

Figure 4.18: Branch Storybook deployment

Figure 4.19 shows the Eslint Review action that will ensure the quality of the code.



github-actions bot reviewed on 8 Sep

src/components/switcherContainer/SwitcherContainer.tsx Outdated

```
+         document.documentElement.style.setProperty('--switcher-primary-font-size', fontSize?.toString()
+         document.documentElement.style.setProperty('--switcher-icon-only-tab-width', iconSize?.toString
+         document.documentElement.style.setProperty('--switcher-icon-label-tab-width', widthSwitcherCont
+     })
```

github-actions bot on 8 Sep

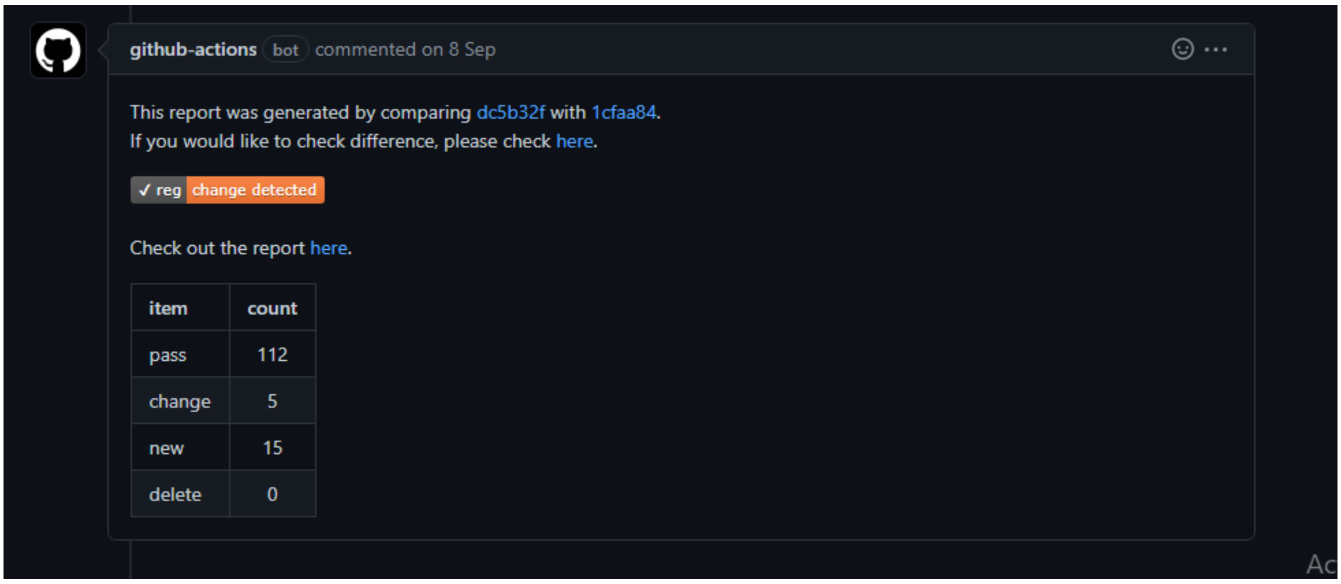
[eslint] <semi> reported by reviewdog
Missing semicolon.

Suggested change

```
-     })
+     });
```

Figure 4.19: Eslint Review Action

Figure 4.20 shows the Visual Regression testing action and its report that will provide the changes detected.



github-actions bot commented on 8 Sep

This report was generated by comparing dc5b32f with 1cfaa84.
If you would like to check difference, please check here.

✓ reg change detected

Check out the report here.

item	count
pass	112
change	5
new	15
delete	0

Figure 4.20: Visual Regression testing action

Figure 4.21 shows the Visual Regression testing report deployed.

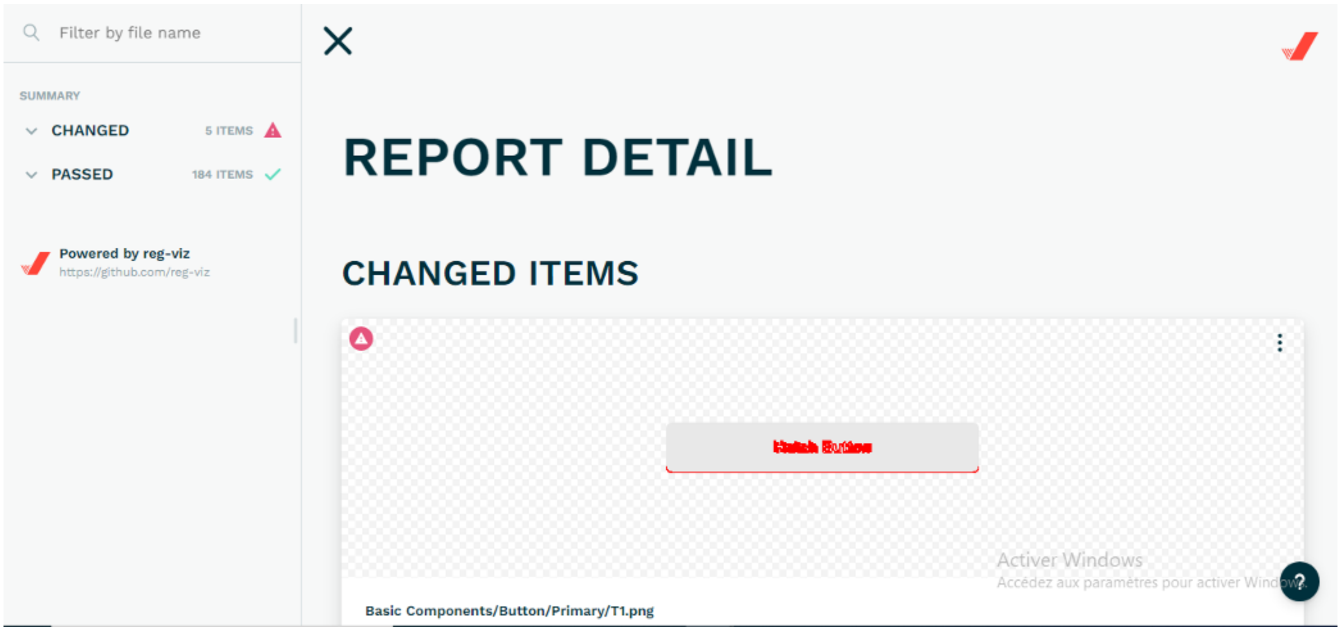
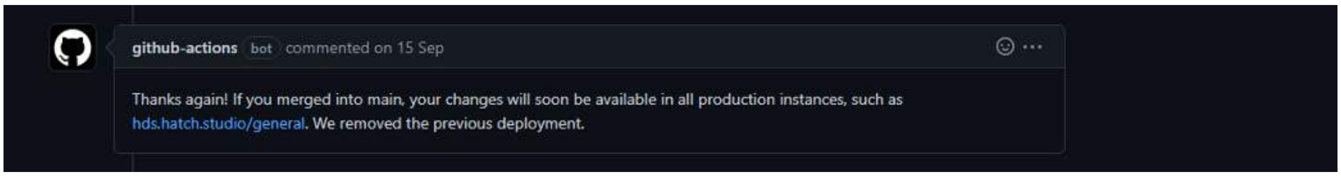


Figure 4.21: Visual Regression testing Page

After approving the pull request,the new component branch will be merged with the main branch and new actions will be triggered.

Figure 4.22 shows the removal of the Storybook branch instance and the deploying of a new version of the general instance.

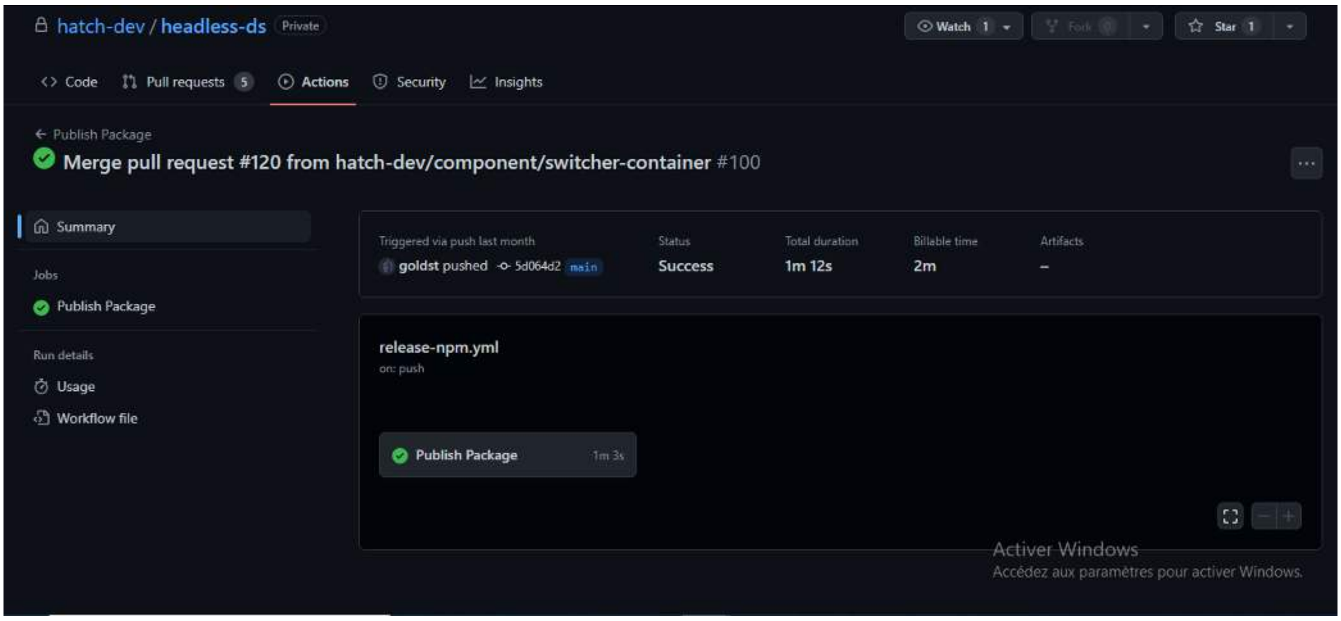


github-actions bot commented on 15 Sep

Thanks again! If you merged into main, your changes will soon be available in all production instances, such as hds.hatch.studio/general. We removed the previous deployment.

Figure 4.22: Storybook General instance deployed

Figure 4.23 shows the Publish Package action which will deploy a new version of our component library



Merge pull request #120 from hatch-dev/component/switcher-container #100

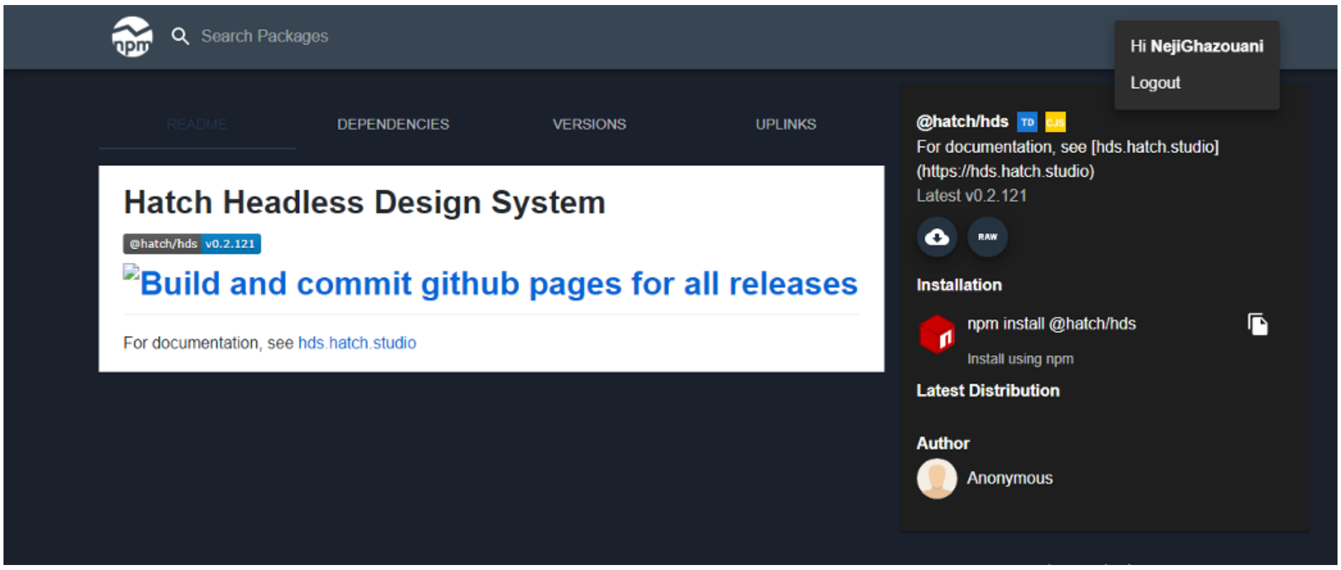
Triggered via push last month	Status	Total duration	Billable time	Artifacts
goldst pushed -o- 5d064d2 main	Success	1m 12s	2m	–

Figure 4.23: Publish Package action

4.4.2 Using System

In the following subsection, we will present the graphic interfaces showcasing the implementation of our System.

Starting with the first interface, representing our Hatch Headless design system module deployed as npm package.



Hatch Headless Design System

@hatch/hds v0.2.121

Build and commit github pages for all releases

For documentation, see hds.hatch.studio

@hatch/hds TD CJS

For documentation, see hds.hatch.studio

Latest v0.2.121

Installation

npm install @hatch/hds

Install using npm

Latest Distribution

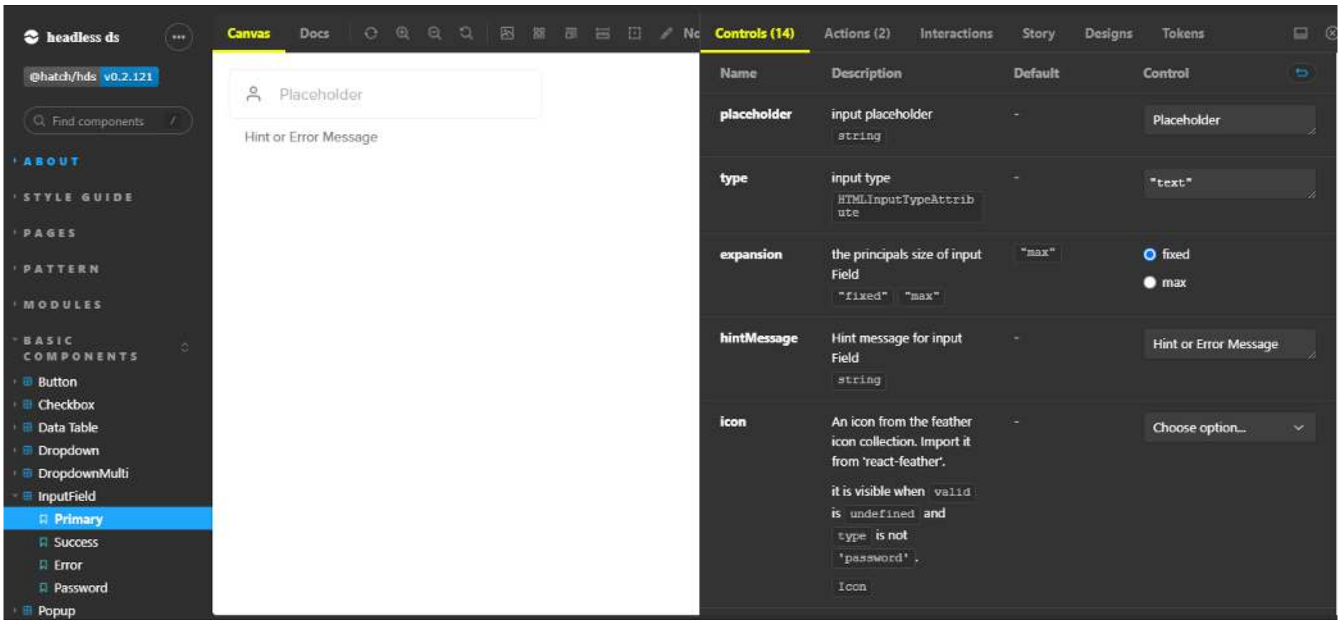
Author

Anonymous

Figure 4.24: @HatchHds npm package

In the following interfaces, we will present the implementation of the documentation which is our Hatch storybook. Storybook provides a preview option with functionality for the user to interact with the component, change the values of its properties and run some tests.

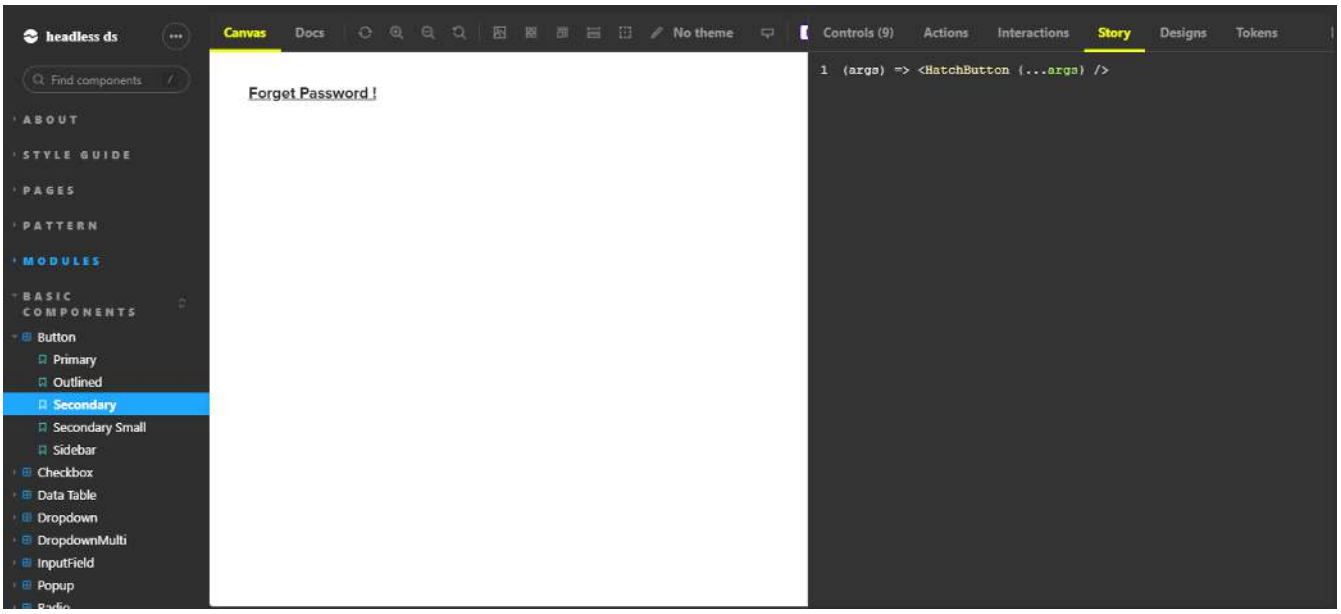
Figure 4.25 shows the input component with its documentation and properties.



Name	Description	Default	Control
placeholder	input placeholder `string`	-	Placeholder
type	input type `HTMLInputTypeAttribute`	-	"text"
expansion	the principals size of input Field `"fixed"` `"max"`	"max"	[✓ fixed] max
hintMessage	Hint message for input Field `string`	-	Hint or Error Message
icon	An icon from the feather icon collection. Import it from 'react-feather'. it is visible when `valid` is `undefined` and `type` is not `'password'`. `Icon`	-	Choose option...

Figure 4.25: Input component

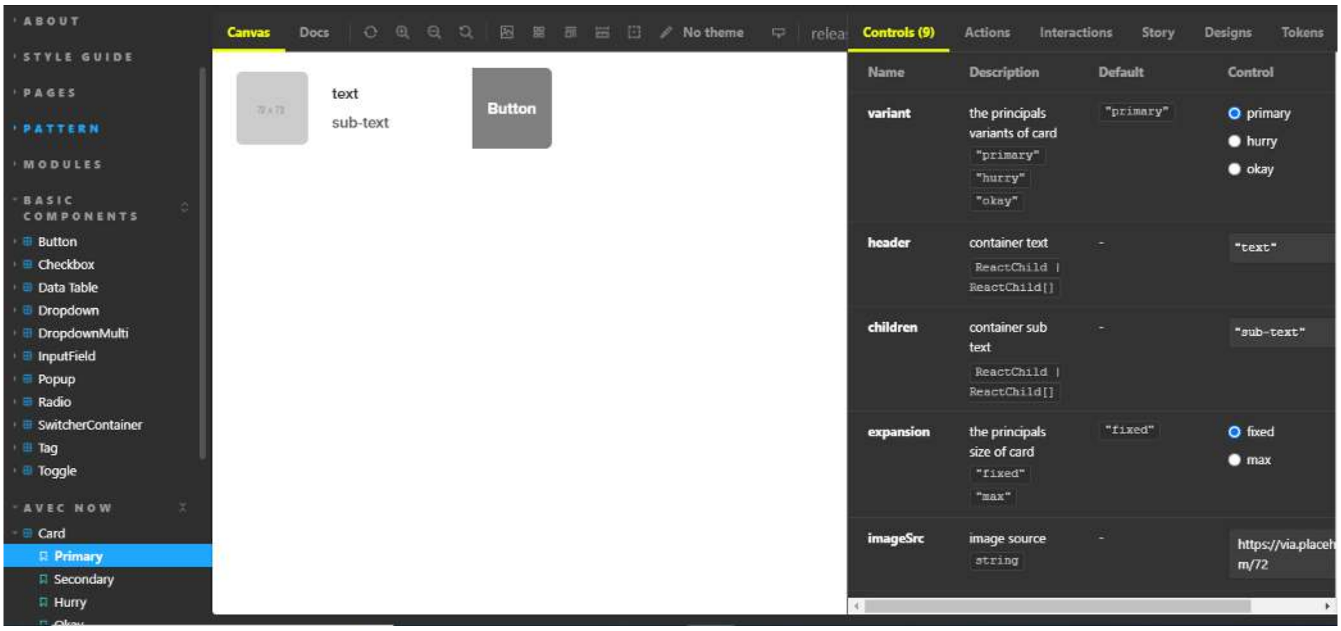
Figure 4.26 shows the button component with example of the code that you will implement to use this component.



```
(args) => <HatchButton {...args} />
```

Figure 4.26: button component

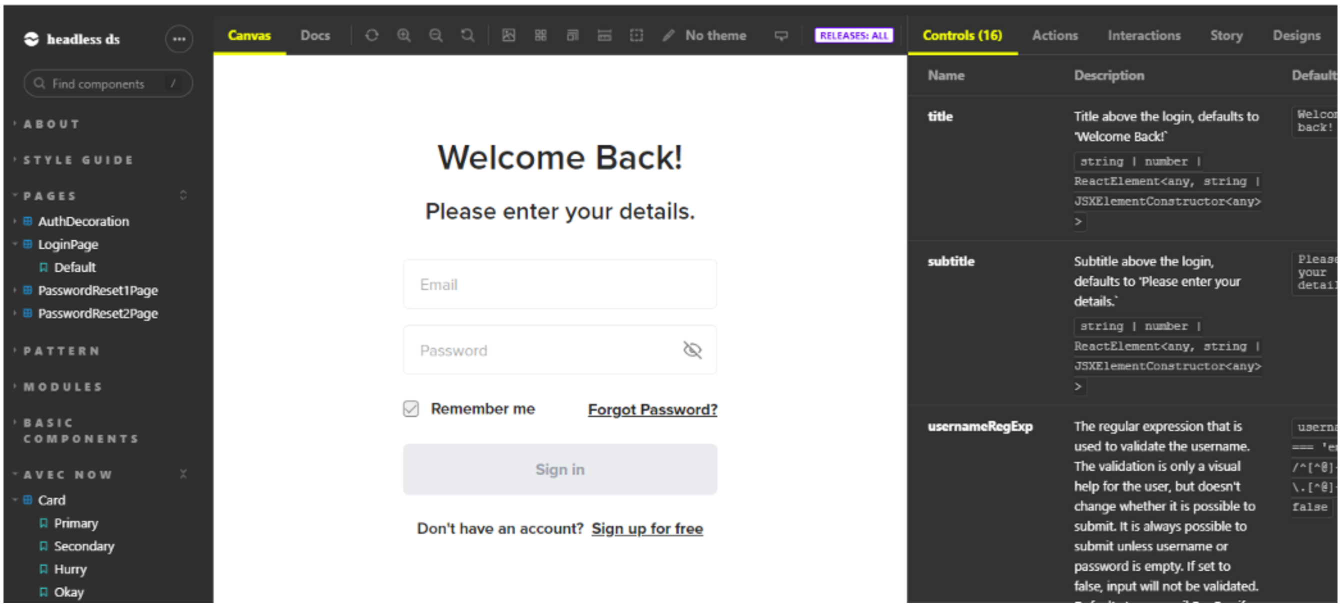
Figure 4.27 shows the card component which is a more complex component where we use our hatch Button component to build it .



Name	Description	Default	Control
variant	the principals variants of card "primary" "hurry" "okay"	"primary"	[✓ primary] hurry okay
header	container text ReactChild | ReactChild[]	-	"text"
children	container sub text ReactChild | ReactChild[]	-	"sub-text"
expansion	the principals size of card "fixed" "max"	"fixed"	[✓ fixed] max
imageSrc	image source string	-	https://via.placeh m/72

Figure 4.27: card component

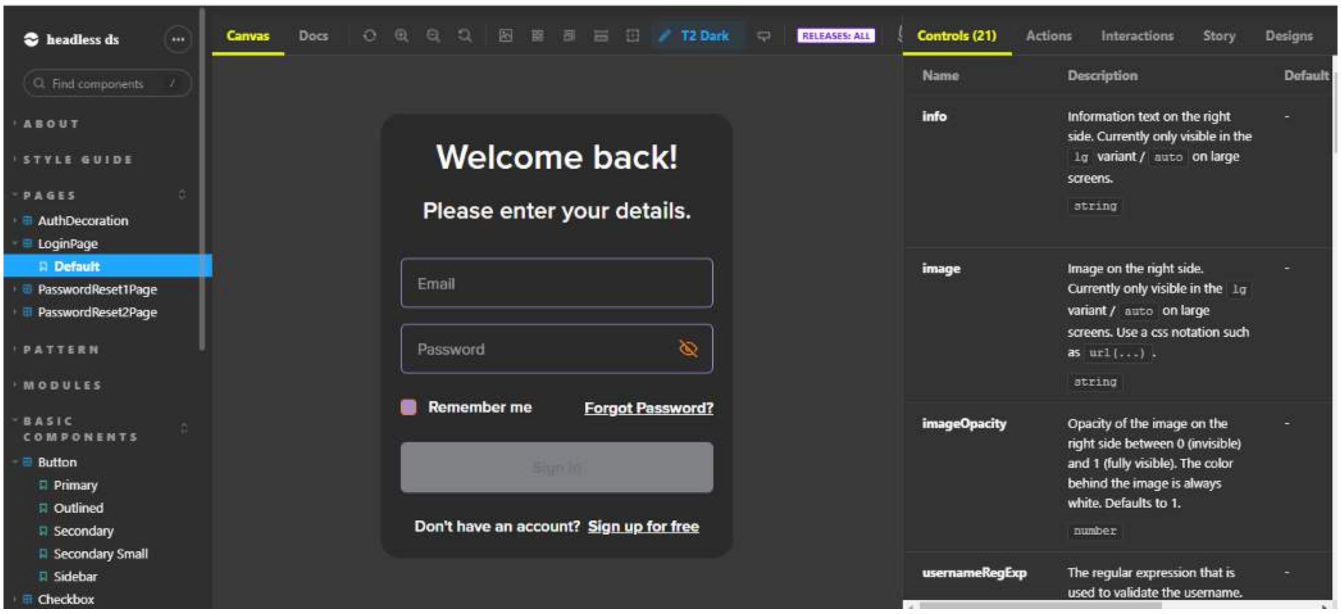
Figure 4.28 shows the login page. This page provides more context for the components to work together and work together to accomplish their goals.



Name	Description	Default
title	Title above the login, defaults to 'Welcome Back!' `string | number | ReactElement<any, string | JSXElementConstructor<any> >`	Welco back!
subtitle	Subtitle above the login, defaults to 'Please enter your details.' `string | number | ReactElement<any, string | JSXElementConstructor<any> >`	Pleas your detai
usernameRegExp	The regular expression that is used to validate the username. The validation is only a visual help for the user, but doesn't change whether it is possible to submit. It is always possible to submit unless username or password is empty. If set to false, input will not be validated.	userna === 'e /^[^@]- \.[^@]- false

Figure 4.28: Login Page component

Figure 4.29 shows the theme feature that provides the ability to check any theme that the designer adds.



Name	Description	Default
info	Information text on the right side. Currently only visible in the `lg` variant / `auto` on large screens. `string`	-
image	Image on the right side. Currently only visible in the `lg` variant / `auto` on large screens. Use a css notation such as `url(...)`. `string`	-
imageOpacity	Opacity of the image on the right side between 0 (invisible) and 1 (fully visible). The color behind the image is always white. Defaults to 1. `number`	-
usernameRegExp	The regular expression that is used to validate the username.	-

Figure 4.29: Login Page component

Conclusion

To sum up this last chapter, we dedicated it to the implementation phase. We talked about the hardware and software used all along this internship and we finished up by following it with our system interfaces.

GENERAL CONCLUSION

To conclude this report, we will briefly walk you through the chapters. Accordingly, the first chapter tackles our project scope, in which we presented our host company "Hatch", focused on the problematic, conducted a study of the existing, and finally we provided our proposed solution the "Headless Design System", accompanied by an explanation of the choice of the methodology applied. In the second chapter, we proceeded by analyzing the state of the art of the proposed solution, dissecting it into three key sub parts that comprise of: the atomic design methodology, the design system principles and the UI patterns. Afterwards, Chapter Three entails a Requirements analysis and specification. It distinguishes the Functional and the Non-functional specifications. Then, we showcased the main application with a global use case diagram and the deployment diagram. We also analyzed the architecture's functionality and explained its different components. The fourth and final chapter of this report specifies the technologies and the working environment harnessed to implement the proposed solution. As of now, Hatch has implemented the Headless Design system in their workflow on daily basis, which solves the issues mentioned in the problematic; time waste, money waste and quality. The implementation of the Headless Design system is on-going and ever-growing project, as we are seeking to expand it to other areas by creating version in other frameworks. This can be highly beneficial for both the company, by attracting clients and for the teams in their work, by helping them to overcome the difficulties they might face.

During my internship at Hatch, I had the opportunity to improve my web development skills in a real life company.I consider this internship an important milestone, helping me to launch my career in this area of expertise. On the personal level, it was an extremely interesting experience, that allowed me to make amazing contacts inside contentsquare from several nationalities that will stay for life.

WEBOGRAPHY

[1] Donella H. Meadows, Thinking in Systems , London, UK, 2009. [Accessed 10-May-2022].

[2] Brad Frost, Atomic Design, Pittsbugh, Pennsylvania, USA, 2016. [Accessed 10-May-2022].

[3] Marco Suarez, Jina Anne, Katie Sylor-Mille, Diana Mounte, Roy Stanfield, Design Systems Handbook, New York, NY, USA, 2017. [Accessed 15-May-2022].

[4] Bootstrap, Beagle – Responsive Admin Template [Online]. Available: `https://themes.getbootstrap.com/product/beagle-responsive-admin-template`. [Accessed 25-May-2022].

[5] Diana MacDonald, Practical UI Patterns for Design System Fast-Track Interaction Design for a Seamless User Experience. Victoria,VIC, Australia, 2019. [Accessed 25-May-2022].

[6] Micah Godbolt, Frontend Architecture for Design Systems, Newton, Massachusetts, USA, 2016. [Accessed 10-Juin-2022].

[7] `https://www.figma.com/community/plugin/759651077059504375`. [Accessed 17-September-2022].

[8] `https://amzn.github.io/style-dictionary/#/`. [Accessed 17-September-2022].

[9] `https://clickup.com/integrations/github`. [Accessed 30-September-2022].

[10] `https://code.visualstudio.com/docs`. [Accessed 10-September-2022].

[11] `https://verdaccio.org/`. [Accessed 10-September-2022].

[12] `https://golden.com/wiki/Zeroheight-8AAG5RX`. [Accessed 10-September-2022].