



Reimagining the Shuup Admin Dashboard

The Complex World of Frontend Design

Tamás Kertész

BACHELOR'S THESIS
May 2019

Media and Arts
Interactive Media

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Media and Arts
Interactive Media

TAMÁS KERTÉSZ
Reimagining the Shuup Admin Dashboard

Bachelor's thesis 66 pages, appendices 16 pages
May 2019

Web design and development is a complex process that has many considerations that affect the results. Users lie at the centre of these reflections as they are the ones who interact with the products developers and designers create. There are tried and true principles and methodologies designers can employ to ensure a baseline quality that boils down to user experience.

The aim of this thesis is to find out what the guiding principles of digital product design are and how to apply them to a real-world project. The project in question is the improvement of the admin dashboard of the e-commerce software, Shuup. It is a relevant use-case and an accurate representation of the average work a designer or developer toils on in the software industry.

The utilised methodologies deal with the theoretical practices that are accomplished by user experience researchers. This sheds light on aspects that deal with the underlying and unseen, such as information architecture, navigation, user research, user testing, iteration and so forth. These areas of research are the foundation for apps and websites, they answer the How, Why and Where of development.

The work which this thesis is based on was executed in the summer of 2018 as part of the Junior Developer role at Anders Innovations in Turku. The results were shipped in September of the same year and is available in the latest release of Shuup e-commerce software.

Key words: web development, UI, UX, JavaScript, design, Shuup

CONTENTS

1	INTRODUCTION	7
2	FROM IDEA TO PRODUCT: WEB DEVELOPMENT METHODOLOGIES.....	8
	2.1 Project management and planning	8
	2.2 Strategy, goals and scope.....	10
	2.3 Understanding the end users	11
	2.4 Testing with users in mind.....	13
	2.5 Mapping ideas to structures	14
	2.6 Design patterns and systems.....	18
	2.7 Prototypes, wireframes and high-fidelity mockups.....	22
	2.8 Development & Implementation.....	25
3	REIMAGINE AND REDESIGN: THE SHUUP ADMIN DASHBOARD .	28
	3.1 Background information	28
	3.2 About the project	29
	3.3 Redesign rationale.....	30
	3.4 Project planning and identifying objectives.....	31
	3.5 Research & Mockups	34
	3.6 Project management & collaboration	37
	3.7 Technical implementation.....	38
	3.8 User testing & project results	45
4	CONCLUSION.....	47
	REFERENCES	49
	APPENDICES.....	51
	Appendix 1. Shuup settings panel	51
	Appendix 2. Shuup contacts panel	52
	Appendix 3. New product page	53
	Appendix 4. Shuup mood board	54
	Appendix 5. Comparison of Shopify and Shuup product filtering system 55	
	Appendix 6. Admin panel dashboard mockup	56
	Appendix 7. Admin panel dashboard mockup	58
	Appendix 8. Shuup product table before and after.....	59
	Appendix 9. Table empty state.....	61
	Appendix 10. Dashboard main menu in open and closed state	62
	Appendix 11. Mockup settings page.....	63
	Appendix 12. Shuup setup wizard	64

Appendix 13. Login page subtle animation..... 65
Appendix 14. Order details page 66

ABBREVIATIONS AND TERMS

Adobe	Developer of the industry standard creativity suite of applications
Adobe Xd	Adobe Experience Design is a design and prototyping application
Bootstrap	CSS UI component library that makes it easy to build website layouts and designs
CMS	Content Management System
E-Commerce	Online web store
Empty State	A user interface term that refers to placeholder content when there is none available
Figma	Digital design tool
Gestalt	Principles of grouping objects in patterns
Git	Version control change tracking system used to collaborate in software development
GitHub	Web based interface for managing projects that is based on the Git version control system
HTTP	Hypertext Transfer Protocol, the main protocol through which the Internet is browsed
XHR	XMLHttpRequest is a browser API available to JavaScript which can send requests to web servers
IA	Information Architecture
Invision	Or InvisionApp, a digital prototyping software
IPO	Initial Public Offering
Jira	Project management and bug tracking software developed by Atlassian
JSON	JavaScript Object Notation is a data format standard used in the communication of data between machines
LESS	It is a pre-processor language that compiles to CSS and offers programming-like dynamic features
macOS	An operating system developed by Apple
Marvel	Digital prototyping software
MVP	Minimum Viable Product
NPM	Node Package Manager

Open Source	It is a general term under which software source code is distributed to anyone to modify and use according to the OSS license, usually, free of charge
Plugins	Software written for specific applications that extend its features
POC	Proof of Concept
SaaS	Software as a service
SCSS	Superset of CSS which, like LESS, compiles down to CSS and includes dynamic features
SEO	Search Engine Optimisation
Shuup	Open Source e-commerce software
Slack	Online communication platform for teams and companies
SketchApp	Digital design tool
UI	User Interface
UX	User Experience
WordPress	Open Source Content Management System (CMS) that powers many blogs and website on the Internet
YouTube	Video streaming site

1 INTRODUCTION

The e-commerce industry is a competitive and highly profitable market with many well-known players in the space. Amazon, Alibaba and eBay are just a few websites that consistently post some of the highest profits in the online retail industry. Riding on the coat of the success of these brands are small independent stores that operate in niche international or local markets. These webstores have special needs that the previously mentioned platforms do not cover. This includes necessities such as cost saving measures, own branding, own online presence, etc.

The burgeoning landscape of hosted e-commerce software are just clicks away. Companies, such as Shopify, offer this in the form of a service where the heavy lifting is taken care of. Like the big retailers, this is a highly competitive space where companies vie for a share of the profits generated from the mom and pop shops. Competitive edge is key and offering the best user experience for the money is the focus of the e-commerce software providers. Be it ease of use, cheap subscriptions or full custodial service, these companies want a piece of the market share. The big players such as WooCommerce are offered as free extension to other popular solutions, however, the giant Magento, was recently acquired by productivity giant, Adobe (Grant, 2018). The incentive for e-commerce software companies is clear, market dominance can be clearly translated into profits.

Within this industry Shuup provides both an open source and hosted alternatives for its products. As it is competing with the big players in the space, it is also looking for ways to stand out from the pack. The stale design of the key elements of Shuup were identified as a drawback by the team. This thesis relates the work and decision making that went into the improvements to this aspect in the admin panel of Shuup. These decisions are based on the principles and methodologies outlined in the second chapter. The methodologies in question deal with all aspects of frontend engineering. This includes the disciplines of user experience design, user interface design, researching, planning and user testing. While it does not offer a detailed glimpse into technical development, it is a launch point for tackling approaches to product redesign.

2 FROM IDEA TO PRODUCT: WEB DEVELOPMENT METHODOLOGIES

Software development is a complex process by which individuals or professional companies turn ideas into working products. The nature of software development is of a constant change as it is tied to the progress of technology. Advancements in hardware require advancements in the software that lays on top of it. The result is reflected in the innovations that strive to provide users with products that improve their lives.

Web development, much like general software development, is an ever-evolving field that creates products to be consumed on the World Wide Web in the form of web pages and web applications. It is spurred by the evolution of the web as a whole and web technologies specifically that make it possible for anyone in the world to participate. In 2005, there were over one billion internet users worldwide. Since then, that number has grown by three and a half times as more parts of Africa and China are coming online (Statista, 2017). More people translates to more business opportunities as well as innovation which, in turn, feeds on this cycle. These conditions make it possible for rapid progression of new ideas in the space. That said, ideas are plentiful, and it takes a considerable amount of effort to bring an idea to life. Web development houses and agencies are booming across the world as traditional businesses switch their focus onto the digital arena (VentureBeat, 2017).

2.1 Project management and planning

The number of projects that end on the plates of development companies demands a standardized framework through which it is possible to both offer the clients acceptable delivery time and quality. Furthermore, having an established practice when it comes to managing projects can make or break them. It is akin to building real world structures. In this scenario, the architects and engineers come up with very detailed plans and specifications which are passed on to the builders who execute it. The builders cannot do their jobs without blueprints and in the same way, architects cannot deliver the results to customers. In the real world, there is an established flow from idea to finished product, the principles of which are carried over into the software world.

The Project Manager role is essential for ensuring that all the moving parts of a work line up to meet expectations. In the book, *Real Web Project Management*, the authors make the case that efficient handling of every phase of a project will create the foundation on top of which the relationships between client and company sits (Shelford & Remillard 2003, 18-19). The project manager should take care to ensure that a given project is following a standardized flow and that every stakeholder is on the same page. The methodologies employed should be the same no matter the project's scope or goals to attain an accurate bird's eye view of its evolution. The work methodologies should be user-centred with a clear plan of action and end goals.

The problem with writing software is that often the agency, or the person charged with completing the assignment ends up winging it. If builders cannot afford the luxury of coming up with the plans on the spot, why should software be any different? (Janetak, 2018.) The reasons for this phenomenon can vary, one reason is budgeting does not allot enough time for exhaustive planning. Another explanation is inexperience on part of the developer where they make assumptions about the work they are about to embark on.

An answer to budgetary constraints is organizing development and design sprints. This is called agile work method and it has become the favoured way to organize work in software projects. The idea behind it is simple, plan a sprint for several days (usually five) that are laser focused in getting one aspect of the work done. This is decided on a needs basis, however, the best results are had whenever the task at hand is straightforward. For example, build a user onboarding process for a mobile application. This encompasses user registration, login and instruction, amongst other things. The target is to get these implemented within five days. The success metric is measured based on the work tickets accomplished within each respective area. At the end of the week, the team can go over what they achieved and if testing results unfavourably, the sprint can be restarted. Each iteration builds on knowledge from the last and the idea is that because these last for such a short span, developer time can be safely wasted in pursuit of better results. The knowledge and experience accumulated at the end of the sprint can be more insightful than a drawn-out process of perfecting one set of results. Iteration is a key principle when it comes to

producing meaningful work. The cycles give enough time for the software team to cement the scope and goals of the work they are producing without losing themselves in the implementation details. Essentially, the agile environment forces workers on moving fast and hitting goals. There is no time to develop auxiliary facets, therefore, common issues such as bike shedding never come about.

Jesse James Garrett wrote what is essentially the bible of meaningful design and development, *The Elements of User Experience*. He elaborates on five different planes that rely on each other and improve on what the previous plane brings. Garrett's methods are a tried and true way of organizing the phases of development to maximize production value.

2.2 Strategy, goals and scope

A crucial first step in the lifecycle of web development is doing the necessary prep work to understand the end users, the product and the niche in which it is being launched in. This involves identifying the problems of the chosen niche, the demographic in general, and the individual personas specifically of the target market. Furthermore, elaborate on the challenges and potential difficulties, risk analysis, creation of the business plan and agreeing to achievable goals that will drive the work forward. Garrett emphasizes a need for creating specific measurements of success and failure. These metrics will make it possible to identify problems and mend them in a timely manner. To tack on that, the author argues that ambiguous specifications lead to moving goal post effect where there is no conclusion to a project. That is a dangerous predicament for the company to be in which will cause a dip in morale among the developers, thus entering a vicious cycle of negativity. In the end, the product's chances of success diminish (Garrett 2011, 39-41, 78). The strategy can be distilled into two schools of thought, understand what users want and what clients want. Marrying the answers will provide the necessary scaffolding to base any further inquiry into how to conduct further investigations.

Putting together a comprehensive strategy will be the guiding light for a product's lifecycle. The key areas that need to be considered are the value proposition, i.e.

how does this solution improve the user's life? How will that generate profit for the company providing the solution? By answering these two questions, you open paths that will eventually define what the product is and, most importantly, what it is not. The latter of which is also important to identify as early as possible. Specifying in concrete terms what it should not be will impose clear limitations that simplify the result down to a Proof of Concept, or POC. At its heart, the product should do one thing and do it well enough for it to still make business sense. Bells and whistles can be added afterwards, however, a Swiss Army Knife solution will do little to improve on the core concept.

Garrett, calls out the lack of strategy in a product development cycle as the single cause of failure in the software world. A badly defined product will never stand a chance in the free market so long as the value proposition is not clear (Garrett 2011, 36). How do we prioritize our tasks and know what to target to maximize our chances of success? Go straight to the source, the users.

2.3 Understanding the end users

Before the Internet, researching users was a hectic endeavour. To get accurate results, an advertising agency, for example, had to employ hundreds of individuals for the task. These people would have to be a representative slice of the market, therefore, laboriously vetted. Afterwards, they would have to be organized into groups and given tasks. Not to mention the costs of paying them for the time they sacrifice. All but the biggest companies could afford to do extensive studies in this fashion. The findings and knowledge could end up saving these companies millions and would net them enough data to fix issues that may arise around their product. With the advent of the Internet, test groups can be spread out across the world and best of all, subjects do not have to leave the comfort of their home or even be paid. Methods such as focus groups still have their benefits, however, internet-based testing is considerably cheaper and has the possibility of aggregating data many times the size of traditional methods. Therefore, there really is no excuse not to test any new idea from even before the beginning of the product cycle.

Reaching users through the internet to glean information about a product idea is as easy as uploading a video to YouTube. What is usually the case when targeting a specific user problem, the users themselves do not know of it until it is pointed out. Once the users are conscious of the problem, you have their attention. As was the case with the founders of Dropbox, they knew that their product solves a real-world problem, but they were not the first company to market, they needed critical mass right off the bat to succeed. Drew Houston, one of the founders of Dropbox, created a short video that showed off what the service could do in 3 minutes. Essentially, overnight his video managed to create awareness and excitement that ended up validating the assumptions that were made (Reis 2011, 99-101). Today, Dropbox is worth more than 12 billion dollars (Tarver, 2018).

In *The Lean Startup*, Eric Reis argues for bringing this idea into the building phase as soon as possible in order to test the validity with actual users. Afterwards, collect data, analyse and iterate as soon as possible. This cycle should be repeated until the results point to something worthwhile (Reis 2011, 81-83). Success metrics should be agreed upon previously. Following this, developers should have a clear idea of what exactly they must build and, likewise, sales and marketing can come up with their own supporting materials. Thus, you end up with a unified vision that will guide the project to completion. However, user testing is not put on hold. User-centred design means that the users are kept in the loop throughout the lifecycle. Politics can get in the way of great ideas being fleshed out and subsequently diluted. User feedback is the single source of truth that can end disagreements over design and development decisions (Krug 2014, 108-109). Facebook synergized this best in their former motto, "Move fast and break things".

Other valid metrics that provide general insight into user behaviour are analytics. For instance, Google Analytics is a platform that, once installed on a website, can gather detailed information about each visitor's session. How much time they spent, which pages they preferred viewing, what website they originated from. All kinds of metadata are available at a glance that should inform any website project's strategy. Other tools such Hotjar, a heat map app, paints a clearer picture of what the users did on the website or mobile app. Which areas were

more favoured compared to others, based on mouse pointer activity. The strength of metadata collection is that it is done passively. The user does not necessarily have to be aware unless specifically approached. The sheer amount of data that is possible to collect can provide a general overview of not only the website's health but also how the users interact and experience it.

2.4 Testing with users in mind

Leading digital user experience experts, including Jesse James Garrett, Steve Krug and others, agree that user testing in the information age is cheap and valuable. How cheap? Steve Krug in his much-lauded work, *Don't Make Me Think*, recons that it can be as low as 10-dollar cents per day (Krug 2014, 113). How valuable? Testing one user is better than not testing at all (Krug 2014, 115). Clearly, the author is arguing that testing should never be an afterthought to any project. In this phase of the software development, we should be testing for errors in usability, as well as keeping an eye out for potential issues with the strategy.

Now, we have a target market and a good idea of who the users are, we move to test the concrete application itself. By continuously testing our ideas at every step, we ensure that we are staying true to user expectations and are providing the tools for them to complete their objectives. Every design decision will carry a weight in favour or against the users and there is no way to find out which case it is, unless the ideas are trialed. One way to think about it is a small MVP within the application itself. Steve Krug's theory describes something he calls the reservoir of goodwill. When presented with an interactive application, users have a certain amount of good disposition which can go up or go down based on the faced obstacles (Krug 2014, 167). By way of example, I would like to look at an imaginary user browsing an online clothing store and see what kind of user experience patterns can influence goodwill.

The objective in this scenario is to find the correct section, filter by colour and size, and finally initiate the checkout process. The user finds the women's section and that increases goodwill, though once there, she is bombarded by newsletter popups which lowers goodwill. After selecting a suitable option, she is taken to the product page. There, she is presented with relevant information about the

trench coat including size, manufacturer, colours, shipping details and price. Having all the data in one place and easily digestible increases her goodwill and she decides to go ahead with the checkout process. During this process, she finds that additional charges were tacked on to the final price and the selected colour is out of stock. Therefore, goodwill is lowered just above what she would consider unacceptable and, ultimately, decides to pick a different colour and continue the checkout process. The payment page consists of dozens of forms which proves too much for the user. Finally, the goodwill drops below the tipping point and the user abandons the cart. Ensuring a pleasant user experience goes hand in hand with establishing trust, credibility and loyalty with customers, particularly with e-commerce customers.

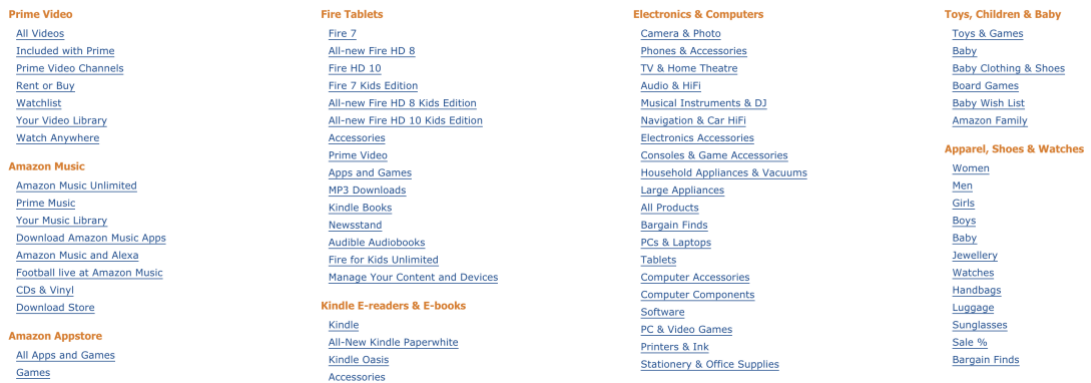
Setting up roadblocks for users in the name of satisfying a business requirement, in this case inflating the newsletter subscription count, may have consequences that are hard to determine without proper testing. Other problems, like hiding the full price and stock information, evokes the sense that the shop is employing dishonest tactics to trick users (Krug 2014, 168). On the one hand, testing flow with actual people would have uncovered these issues well before this scenario went into production. On the other hand, Jesse James Garrett views user testing as a part of a more complex investigation.

Garrett argues that users cannot be expected to provide insightful feedback during tests unless the user experience professionals know the ins and outs of their product. That is to say, asking the wrong questions will provide the wrong answers, thus, meaningless results. Knowing the problem before setting out to fix it is conducive to progress (Garrett 2011, 177).

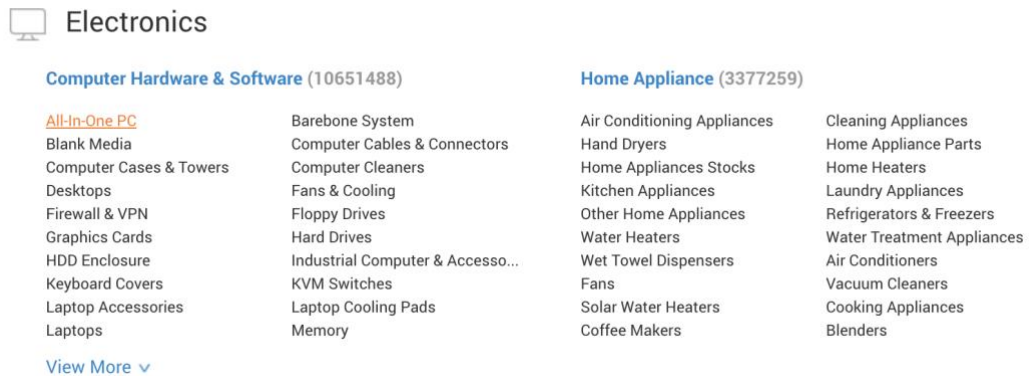
2.5 Mapping ideas to structures

Once our requirements and strategy are written out and we have a good idea about what our application should do, we must consider setting these notions into stone. Giving structure to supposition is the first actionable step in the development cycle that will have an immediate impact on the end results. The way information is grouped, ordered and organized is called information architecture. Applications or websites are, more often than not, text based. Users

with a goal or task in mind will want to sift through as much data as possible in the shortest time possible. The way information architecture helps users to achieve this purpose is by introducing clarity to otherwise disjointed data (Rosenfeld & Morville 2002, 16-17). A simple example to illustrate this concept is an e-commerce website’s categories. Large webstores such as Amazon or Alibaba have hundreds of thousands of items, each with its own page and descriptions. Consider the following two examples gathered from the aforementioned websites in their respective order (picture 1; picture 2).



PICTURE 1. A screen capture from Amazon’s department list.



PICTURE 2. Alibaba’s categories list.

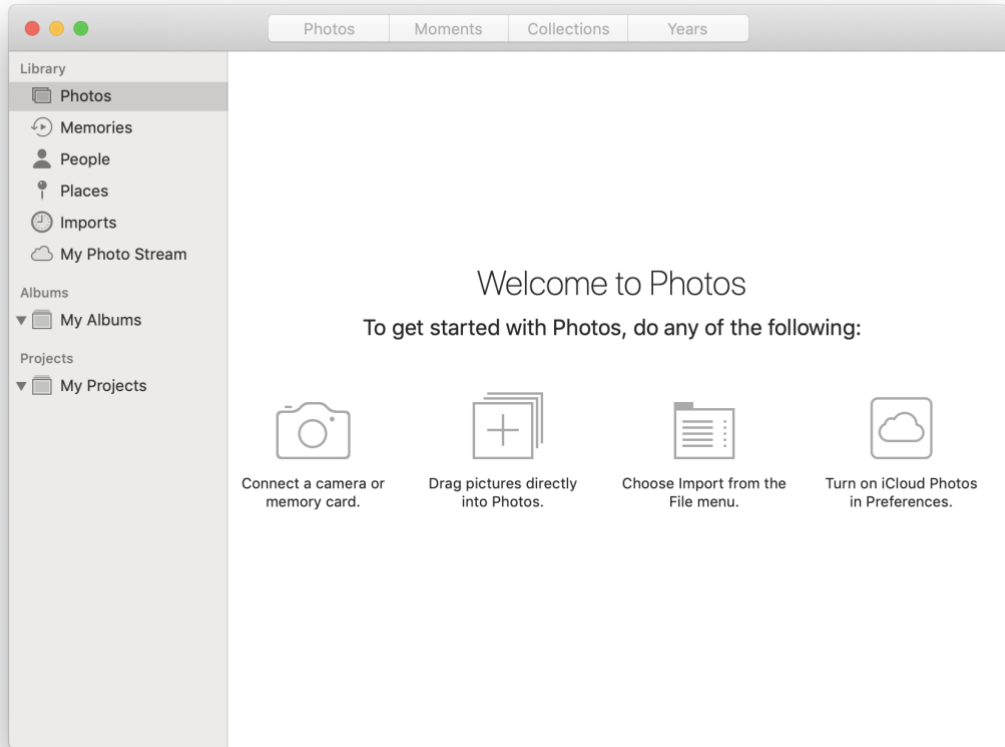
Note the phone book approach of ordering all the categories within each taxonomy for easier access. The screen caps only show a sample size, however, the taxonomies as well as the categories go from A through Z. As a result, items in these two stores must occupy one or many categories for ease of use by both users and internally by the system. In the real world, this is akin to browsing in a

bookstore and finding desired books on the shelves. These examples of information architecture abide by a universal ruleset which is expected by users through experience, namely the alphabetical order. Amazon's item categories are placed in alphabetical order within the taxonomies. However, information architecture is not always as cut and dry as in these cases. There are examples where grouping information in ascending order does not bring clarity about the content itself, as each section might have a contextual or a meaning-based relationship. For instance, how news sites group their articles by topics, foreign, business, politics, etc. (Garrett 2011, 97).

Classification of information should be an invisible aspect of a website to the end users. Proper IA is present below the content and it manifests itself in the form of usability (Rosenfeld & Morville 2002, 24-26). Rosenfeld, Morville and Garrett in their books reach an agreement that IA differs from graphics design in the way that graphics designers are not necessarily information architects. Graphics designers use their tools to decide how a given information is presented for visual consumption. Interfaces, for example, have the main role of suggesting to the user how they can interact with the given product. IA lies somewhere below the interface in a more abstract way. Rosenfeld and Morville propose three components that make up IA, Context, Content and Users. In essence, Context is an organization's philosophy, strategy, goals, resources, and so forth that are unique and lie at the foundation of the brand. Content is a broad term for whatever the organization outputs in the form of assets such as text, videos, brand, culture and other metadata that are in direct influence of the Context. Finally, Users are the ones that make use of the content for their own end (Rosenfeld & Morville 2002, 37-39). The authors theorize that these three components form the framework through which IA can be articulated.

Jacob Ruiz, a design consultant, takes a more semantic approach to defining IA, a collection of nouns and verbs. Nouns are how the information is organized within an application or website and the verbs relate to what the user can do, i.e. actions (Ruiz, 2017). He argues for a pattern-based design that organizes data in these two categories. Users have a clear picture of what the app can do for them based on the actions, while the nouns provide a top down view of structure.

Take for instance, the photos application on macOS (picture 3).



PICTURE 3. Photos application on macOS Mojave.

Right off the bat, we can ascertain all the sections of the application by looking for the nouns on the sidebar and in the application’s header. The actions are presented in the empty state in the main panel. Consequently, the user can make a mental map of the structure as shown in the following figure (figure 1).



FIGURE 1. Structure of the Photos applications.

In this example, the context of the application is photo management. The primary nomenclature is based around nouns such as images, photos, albums and collections. These are terms lifted from real world conventions and are used metaphorically to convey familiarity. These form the general level of information architecture that lays down the sections of the application and conditions the users to create mental paths. When completing tasks, users will go to the general section and look for a specific action (Garrett 2011, 88-90). For instance, a user wanting to edit a holiday picture from 2016 will go to the photos section and browse by year. Once the image is located, the user is presented with the list of verbs (show metadata, like, share, rotate, edit) where the subject picks edit. Thus, the user went from the generic level (photos section) to the specific action (edit) during the completion of their task. All the while information architecture was not playing an active role in guiding the user, however, it was present throughout the journey. Giving a form to the structure brings the product one step closer to production. At this point in the lifecycle, we have gathered market and user research, formed a strategy and limited our scope, tested the idea with users and iterated upon it and, finally, came up with a structure. The next phase in the process deals with everything that is visual and user facing.

2.6 Design patterns and systems

When facing the challenge of creating user interfaces, Jesse James Garrett insists that adhering to conventions is less risky than experimenting with new concepts (Garrett 2011, 111). Indeed, even Steve Krug's book on the subject is called "Don't Make Me Think". It is a cornerstone approach to successful application and web design because it eschews creativity in favour of tried and true methods. The gist of it is that the more time users must spend understanding and learning the controls, the more frustrated they become (Krug 2014, 24). As a result, usability takes a hit along with user goodwill. This leads to abandoned sessions and potential losses in profits, not to mention bad experience on part of users will dissuade them from returning in the future. From this point of view, following trends in user interface design cuts down on research time, as someone else already did the heavy lifting. Standards were put into place and thousands of web apps adopted these patterns and in turn taught their users. Consistent experience on different applications or websites creates a sense of security in

that users feel that they do not have to exert additional effort to use the service. By relying on past experiences, users can develop reflexes that save many hours of needless problem solving (Garrett 2011, 110).

Steve Krug explains that users do not read as much as they scan for familiar looking words that might lead them to what they are looking for. If that fails, users will then expand energy by reading content that is tangentially related to their goal. That occurs because users know that they do not have to comprehend everything about a certain page or application to complete their task. They can, as Krug calls it, “muddle” through until they get what they want (Krug 2014, 32-33). He rationalizes by pointing out that users in general do not educate themselves enough about technology and will elect to not read instructions, instead, find their own way. In this scenario, experience plays a central role in the way these types of people approach problem solving in the digital space. In this context, the role of the user interface designer and user experience engineer is to accommodate the muddler tendency. This is achievable by upholding industry standards and patterns set by the big players in the space such as Google, Apple, Microsoft, etc.

Digital design patterns can range from the high-level vague specifications to exact dos and don'ts. Garrett and Krug elaborate on the importance of navigation and giving visual cues to users about what the content is about (Garrett 2011, 118-123; Krug 2014, 64-65). Navigation design has a direct connection with the underlying information architecture and is a jumping point from which users orient themselves on a given web page or application. As a result, this is a highly recurring element in web design, to the tune that you would be hard pressed to find a popular website without some form of navigation. Other key patterns include heading and content hierarchy. Krug defines this as visual organization of content through the prism of Gestalt principles. For instance, highlighting key information in a body of text by increasing the size of the heading and adding more white space (Krug 2014, 50-51). Another example he submits is dividing a page into multiple sections that content or interaction wise accomplish different things. That way users can decide at a quick glance if that particular area is relevant to their needs. Website brand or logo is a key area that should make its appearance around or near the navigation. That identification marks sits at the

top of the information hierarchy which is why it makes sense to include it in the navigation (Krug 2014, 78). Other recurring elements include mission statements that lie generally above the fold and have the role of defining what the website is and what the very first step can be, i.e. a call to action area. Beyond these universal patterns there are specific standards that relate to usability and accessibility.

Big tech companies like Google, Shopify, IBM and others have through the years come up with a set of well-defined pattern and component libraries, otherwise known as design systems. While highly specific to each company's products, there is a lot to glean off these efforts. Design systems follow the atom, molecule, organism, templates and pages paradigm, where each element can live on its own (atoms), however, when composed together they form pages (Fanguy 2017). The value of a design system is a cohesive brand language which is the single source of truth for the organization. This consistency comes from a need to provide users with predictable experiences when they interact with the company's products. For the company, keeping a consistent face makes life easier when developing new products and pages. Designer's creative liberty is curtailed in favour of efficiency. The atomic design paradigm meshes well with modern web development techniques, whereby a web app or site is sectioned into building blocks that are made up of atomic pieces. Hence, it is designed to bring together designers and developers while putting the needs of the users to the fore.

Google's own Material Design was created to give their Android Operating System a common visual language and to unify the (at the time) highly disjointed user experience of third-party applications. Google gave developers a set of building blocks that allows design and UX disinclined developers with battle tested components to build with. The hope is that developers would not have to focus on reinventing the wheel, rather they can put all their energy into their ideas. Google relies on the metaphor as a driving force and explains that Material Design is a collection of principles based on science and technology, meaning through motion, and meaning through flexibility (Google 2018). Other design systems, such as Polaris by Shopify, Carbon Design System by IBM, follow the same principles that make up Material Design. The result of applying atomic

thinking to the web and app environment is an improved interface consolidation, noise reduction that stamps out inconsistencies. The end-users will receive a predictable experience each time they use applications that belong to the same suite.

Visual design is not strictly aesthetics in the sense of what looks pleasing. Garrett asserts that good design does not stand in the way of the site's stated objectives (Garrett 2011, 143). If the design does not support the underlying content, it should be discarded. That said, design is not an exact science, it is based on creativity. One of the pitfalls of approaching a website design is treating it as an art piece. The temptation is apparent as it is a visual medium without boundaries. Sometimes it is warranted, however, most times there are practical needs to consider. For instance, if a drawn-out animation of critical content is taking too long to finish, and thus be visible, that poses a hindrance to users who are forced to wait it out even if they are disinclined. The results can be measured in bounce rates from users who value their time more than what the perceived worth of the content may be. Final designs can and should be user tested well before taking it to production. That is the only way to objectively measure the success of not just aesthetics but also functionality.

Once again, user-centred design lies at the core of interfaces and visual communication. Design systems abstract away the research and testing efforts lead by the teams that create the frameworks, however, using something like Material Design in a project will not absolve the developer from testing. If a team elects not to use Material Design components in their application, but decide to engineer their own solution, the base principles that are presented by Google and others are still valid. The aim of creating custom brand design language is that it is more in tune with company core values and aesthetic choices than something that is generic and off the shelf, as is the case with Material Design.

Creating design systems from scratch at any level would mean a lot of time would be wasted on development and iteration cycles on top of that. What if there was a way for programming unaccustomed designers and user experience engineers to get a visually functioning product to the users faster? That would allow them to control all the testing variables without writing a functioning application and

wasting developer time. Prototyping application suites from Invision, Marvel, Adobe, Figma and others have introduced tooling that is meant to solve this problem niche.

2.7 Prototypes, wireframes and high-fidelity mockups

In recent years, the prototyping world has seen an explosion of programs meant to speed up design and testing even before committing any developer time. This takes the shape in a suite of applications developed by established companies and startups alike. In the past, prototyping applications did not necessarily include design tools and vice-versa, however, recently that line has been blurred. SketchApp, Figma, Adobe Xd, Framer, Invision Studio and others all ship design tools for creating mockups and wireframes. Additionally, prototyping functionalities baked into the same applications. With the first version of SketchApp which was released in 2013, designers switched en masse from the Adobe sphere of influence of Photoshop and Illustrator (bin Uzayr, 2016). This opened the door to competing design and prototyping software that brought innovation into the user testing realm. That leads to the question, what are prototypes and when to utilise them?

Ben Coleman and Dan Goodwin, authors of *Designing UX: Prototyping*, assert that prototyping is not limited to wireframes and high-fidelity mockups but it is more than that. Clickable prototypes are interactive mockups which help bring the end-users into the design process and shed more light on how the attributes of the final product will function (Coleman & Goodwin 2017, 1-2). More importantly, designers receive constant feedback on the quality and effectiveness of their ideas, while stakeholders can prevent straying from the overall scope and strategy of the project. In short, the sooner users can get their hands on the product, the faster designers can iterate.

A prototype can consist of wireframes, high-fidelity mockups or even pen and paper sketches. The beauty of these techniques is that valuable user data can be accessed regardless of the medium in which testing occurs; be it digital, in the case of clickable prototypes, or in the physical world, crudely drawn paper mockups. Coleman and Goodwin emphasise that more complete knowledge can

be learned if test subjects are presented with real looking data within the prototypes. The tasks that can be given to users should be achievable within the prototype itself. They go on to suggest that prototypes, that only offer limited functionality, will lead to feedback based on opinion that cannot be measured against a list of success metrics (Coleman & Goodwin 2017, 10-11).

Wireframes are made to the specifications laid out in the information architecture research. That can be a starting point where prototyping can be baked into the testing phase. Such mockups should, at this point in the process, offer a one-to-one picture of what the final product will look like. Prototyping should consider interaction design, animation and other finer details that will be present in the final product. These can be page or screen transitions, sliding navigation bars, carousels, and other elements that will find their way into the product and require user action. Ultimately, there is no one way to create prototypes, however, the value they offer is such that it cannot be avoided in modern web development projects. Tools, such as InvisionApp and Marvel, offer the possibility of testing individual screens with minimal interactive elements and moving parts, however, it is meant for simplicity and fast paced iteration cycles. Fully featured prototyping apps, like Framer X, offer deeper nuances. In this case, with Framer X, a designer can implement both mockups and interactions necessary for a realistic mimicry.

Ben Blumenfeld, in an opinion piece for Fast Company, writes that the new wave of prototyping applications is not necessarily bringing key innovations into the space. The learning curve is increasing and choosing one design tool over another is based on whim rather than features that best supports frictionless user interaction testing (Blumenfeld, 2018). Designers must effectively communicate their vision with project stakeholders and provide a visually functioning application schema. That is how modern digital development agencies tackle projects and concurs with agile work methodologies. Choosing the suitable design and prototyping app is a task in and of itself.

The landscape of prototyping is continuously shifting as the industry has not yet settled on a clear winner. Framer X is versatile but with a steep learning curve. InvisionApp and Marvel are not useful for prototyping complex interactions. Flint and Principle require too much time to set up motion prototyping. Adobe Xd and

Figma combine design and prototyping, however, do not yet make a compelling case for the design community to switch away from battle hardened SketchApp. The prevailing strategy is using a combination of applications to craft wireframes and organize them into prototypes. For example, the designs can be laid out in SketchApp and uploaded to the InvisionApp service where the screens can be combined into an interactive flow. Though, the details of choosing the most efficient combination of prototyping and design tools is very much up to the designer. It should be noted that the result should be the same: interactive mockup of the end-product for the purposes of user testing and dissemination to project stakeholders.

With high-fidelity mockups, the final visual design of an app or website can be brought to life. In the previous subchapter, I put forth the idea that visual design ideas should be user tested. Prototyping applications are a solution to this niche problem. Designs do not have to be implemented into the app codebase before it can be tested. Furthermore, the arguments surrounding aesthetics can be laid to rest once solid data can be gathered from users. Questions, such as a button's colour, can have a definitive answer. Moreover, through A/B testing designers can iterate as many times as necessary to meet stated project goals while making sure that the design does not block goals but supports it. In one fell swoop, interface and user experience designers can test interactivity and visuals to craft the most effective version of their product.

By way of example, designers can jump into SketchApp and have prototyping tools available as a secondary context of the application. With the goal of making an onboarding wizard, the designer creates an artboard per page with all the expected elements. Copy, buttons, decorative and helpful illustrations. Here, it is encouraged to include the brand's design guidelines or design system for the sake of realism. Colours, fonts, spacing and dimensions are atomic details that make up the application interface. After completing all artboards with the necessary information and suggestions for interactivity, the designer can create links in the prototyping context. Buttons can be hooked up to point to subsequent artboards that are effectively the reaction to the click action. A tree of choice can be designed with all artboards put into their logical place. Thus, SketchApp can generate a prototype application from the linked artboards that can be shared

and tested. Functionality wise, the prototype behaves like the subject website or application but with limitations. For example, button hover and pressed states cannot be represented, animations are limited to screen changes, UI sections cannot be overlaid, etc. The result can be likened to a slideshow that conveys the general aspects rather than the specific implementation details. That said, utilising these tools improves the eventual product as big iterations can be performed faster and easier than with already implemented code. This flexibility allows for the exploration of many creative paths that otherwise cannot be afforded due to the inherent risk of untestable ideas.

The value of prototyping is apparent during the implementation phase. Software developers can recreate the prototype into the accurate bona fide deliverables. The details in the blueprints are not left up to the developer for interpretation and, therefore, prototypes facilitate an improved hand-off experience over vague screen shots and spec sheets. That said, during the prototyping phase the designer can add trivial looking details on the surface that can prove challenging to implement. This brings up the question of should designers code? On the flipside, should developers know how to design? The idea here is that to increase empathy and knowledge both ways, designers who have a clue about programming can adjust their designs so that it does not pose an obstacle during development. Likewise, developers with some knowledge of design can understand the reasons behind choices made by designers (Follett 2017). Finally, prototypes strengthen designer and developer relationships as well as bring in the project stakeholders early in the game. It is a marked improvement over static images as well as leads to better user testing and faster iteration cycles.

2.8 Development & Implementation

The place in which the visible layer is implemented is referred to as the “frontend”. The implication of the term nods to the existence of a “backend”. In web development nomenclature, the backend comprises the unseen yet vital part of any useful web application, website or service. The business logic, database, API, cache, algorithms, etc. are invisible to the end user, however, this is where the backbone of a given product lays. It is where functionality is defined that allows for users to interact with the system and complete their objectives.

Therefore, backend development is a specialisation with its own world of complexities that, in some cases, are far removed from frontend development duties. In short, the backend provides the means on which the frontend relies on and sets the expectations of what a web page or application should contain.

On the flipside, the frontend is interlocked with the backend via APIs or server side rendered templates. The former is a popular paradigm through which applications, such as Facebook and Twitter, are built on. It emphasises reactivity to user input, efficient data loading, client side rendering, and more. The latter is the traditional server request-response approach. In short, as a user on a website clicks a link, the server responds with the fully rendered page with all the data in place, for example, a typical WordPress website. This happens for each subsequent interaction and each action is always broken up by the wait time of the server's response. Each time the server receives a request, it runs through its internal logic and gathers the necessary data, populates the pages, and sends it back to the user, including all JavaScript and CSS scripts.

The duty of the frontend developer is to create and manage the frontend assets and templates. Furthermore, the visual and user experience design is carried out on this end. The user will interact with the backend through the frontend, therefore, the product lives by what the frontend can accomplish. As such, the importance of the job rivals that of the backend engineer. The frontend developer's skillset includes HTML, CSS and JavaScript knowledge as often the case is that the designer's work is implemented with these technologies.

With API driven frontends, developers are increasingly expected to create full blown applications that not only sport good design, but must perform complex tasks. Web apps such as Google Docs and Dropbox are but few examples of how much is possible to accomplish with frontend technologies. As user expectation is constantly conditioned upwards by popular web apps, developers must rise to meet them. This is done by adopting cutting edge software that enables such apps to exist. As a result, to stay competitive, frontend developers should continuously learn and adapt to change. Whereas backend development is based around battle tested tried and true methods, the frontend is a struggle to

maintain relevance in a rapidly evolving digital space. The latter thought is a cornerstone theme of this thesis' project.

The methods for going about bringing prototypes into existence can vary depending on the scope of the project, the chosen technology stack and other factors. Broadly, programming and design are different roles that require different types of individuals with those skills. Since there is little overlap in these disciplines, the development process needs to be lead accurately by the prototypes or mockups. Communication and empathy takes on an even bigger importance as designers need to pass on their vision and make sure it lines up with the expectations set forth in the previous phases of the lifecycle. Consequently, development methodologies are not specific and cannot be pinned down. This thesis only explores the development path for the undertaken project — reimagine and redesign the Shuup Admin Dashboard.

3 REIMAGINE AND REDESIGN: THE SHUUP ADMIN DASHBOARD

In the previous chapter, I delved into the nature of web development projects. How to implement a plan and execute it while ensuring that the goal is always within sight. Steve Krug, Jesse James Garrett and others offer the advice and techniques. This chapter is about how that knowledge is applied to a real-world project. Real in the sense that it encompasses all aspects of professional web development. Namely, the frequent iterations, pivoting of ideas, scrapping and replanning, team work, communication, design and so forth.

3.1 Background information

Primarily, Shuup is a company that offers online ecommerce solutions for companies. More specifically, Shuup is a piece of software that powers online marketplaces. It features an inventory system, curation, customer facing website, tools for managing multiple shops, plugin system, campaign management, and many other components. Furthermore, the company's business model is based on offering two packages of Shuup, commercial and free. The former is supported and hosted by the company themselves, while the latter constitutes the open source project which does not offer any extra perks. In addition, the free version is offered as-is and thus maintenance and administration is left to the user. This echoes the way in which the company behind the wildly successful WordPress platform, Automattic, provides their CMS product to the world.

Some of Shuup's competitors in the open source space are Magento, OpenCart, WooCommerce and PrestaShop. On the commercial side, Shopify, BigCartel, Wix, BigCommerce and Volusion emerge as contenders. As can be seen, this niche of software products is highly competitive and highly lucrative at the same time. Statista suggests that worldwide e-commerce sales are climbing with no signs of slowing down in the next decade (Statista 2019). Hence, there is a lot of incentive to innovate and capture a share of the audience. As recently as 2018, Adobe, the maker of leading productivity applications, purchased Magento for \$1.68B (Grant, 2018). Magento is the most deployed open source e-commerce solution in the world which stands to reason that Adobe did not so much only pay for the software but for the users as well (Rogers 2018). Looking at these metrics

and the likely evolution of the industry in the years to come, standing out in the sea of marketplace SaaS solutions is paramount to success.

3.2 About the project

The circumstances of this project and how I personally got involved in it is straightforward. Shuup Inc., and by extension the software itself, is an offshoot of Anders Innovations. The latter is a company based in Turku, Finland whose bread and butter is web development for business clients. I had been employed at Anders for a year when I was approached with this task. Namely, the task was defined as a manifold redesign of customer facing areas of the application. Firstly, the admin dashboard which constitutes one of the more important sections when it comes to user interaction with the product. Here, the user can configure, update and establish shops and inventory. Second, the accompanying application of Point of Sale (PoS) terminals required a similar facelift to match the design direction we were about to embark on. These are separate, yet related applications with different contextual considerations which would later play a part in how the project would unfold. The PoS is a mobile application designed for tablet devices which would link Shuup shops with the physical world. This allows for inventory interaction, payment and processing of shipment right in the brick and mortar store.

The motivation behind the Shuup redesign is based on the wish that the product should keep pace with the evolving world of e-commerce. Innovations brought about by Shopify, Square and the like shape user expectations in the way that Steve Krug wrote about. Chiefly, if users are accustomed to Shopify's user interface and interaction flow they will, even if subconsciously mandated, expect the same from Shuup and others. Due to the popularity of Shopify worldwide, they are the trend setters that less popular solutions aim to imitate. According to the web statistics analysis tool, BuiltWith, Shopify usage represents 19% of all e-commerce solutions, second only to WooCommerce (BuiltWith 2019). The Canadian company boasts a \$1,3B valuation at IPO with thousands of employees around the world (crunchbase 2019). In short, they have the necessary weight for trend setting and innovation, especially when it comes to user research.

A second and more relevant motivation is that Shuup's user interface was not updated since its inception. Throughout the years, the constant evolution of the application's underlying functionalities did not keep pace with the UI and with time, cruft accumulated. This is evidenced by outdated graphics, layout and flow when contrasted with the competition's offerings. Whenever tests show that users have difficulties in accomplishing their objectives, it is time to think about your product critically.

3.3 Redesign rationale

What are the metrics by which we can decide that a redesign is warranted? I have mentioned difficulties expressed by users, however, what are some other reasons? Sandijs Ruluks is a San Francisco designer who shares a few arguments. According to him, you should consider rethinking your website whenever, your current design is not flexible anymore. When a considerable amount of time passed before your last redesign. When your product has changed and the current website does not represent it. When you want to stand out (Ruluks, 2016). Of course, this is only the tip of the iceberg when it comes to justifications. A more interesting way to look at this problem is to find reasons why you should not redesign a website and work your way backwards.

Brent Summers from InvisionApp goes over his reasoning. There is more to fixing apparent website problems than resorting to design overhaul. Incremental changes can go a long way to reaching objectives. Just by improving the existing website with small component changes, such as adjusting colours or writing new copy, it is possible to gain as much in value as with a redesign. Furthermore, he submits that increasing traffic, SEO rank or reducing bounce rate are not directly measurable results of redesign, as too many variables change to be able to draw conclusions. Again, these aspects can be enhanced without ripping out existing solutions (Summers, 2015). In short, large overhauls are risky as they require the intervention of many stakeholders. From developers to designers to management and copywriters, the process is long and fraught with the danger of missing objectives and worst of all potentially wasting time.

The arguments supporting a redesign for the Shuup admin dashboard are so far, to keep up with the competition, breathe new life into the product by refreshing the branding, and thus the design, and to fix user experience issues. This effort can be put under the umbrella of increasing Shuup's market share. At the same time, the PoS application needs to be considered as an integral part of the Shuup product. Therefore, the design must be suitable for both a web app and a mobile app. As these are slightly different contexts, with different technological backgrounds, practically these are considered as two separate yet related projects. One aspect to note is that this project is not a complete re-authoring of the Shuup software. Therefore, only the existing structure is improved. As a result, the planning phase will go through a few iterations as expectations are adjusted to realities.

3.4 Project planning and identifying objectives

The first iteration of the plan involved extensive backend re-engineering which would have allowed for a frontend rewrite. In software engineering, backend is used to describe the layer of software that deals with logic, data access and infrastructure. Frontend is a keyword reserved for delineating the presentation layer, usually a graphical user interface (GUI). Through the frontend, users can interact with the backend and carry out tasks and other activities. Collectively this is referred to as separation of concerns which is the philosophy of decoupling and modularisation for the sake of flexibility (Laplente 2007, 85). Shuup suffered, and to some extent still suffers, from a tightly coupled frontend code with the backend.

One of the big technical objectives early on was to solve the coupling issue and create the environment in which backend and frontend can co-exist somewhat independently. The immediate benefit would be faster iterations and feature development. Basing the new frontend work on modern technologies means that user experience can be elevated. Another side effect of decoupling is that the backend can be leaner, more focused without having any strong opinions on how the frontend should work. This paradigm shift would allow for writing the admin dashboard as component based single page application, or SPA. The strength of SPAs is speed and responsiveness. Traditionally, when a web browser makes a request for a page from a server, for example hs.fi, the server responds with the

page and content already rendered (figure 2). Figure 2 illustrates a typical browser request and server response. In this scenario, the server responds with the all necessary data and page elements for the browser to render. Every subsequent browsing action will prompt the server to construct the page and then transfer it across to the user.

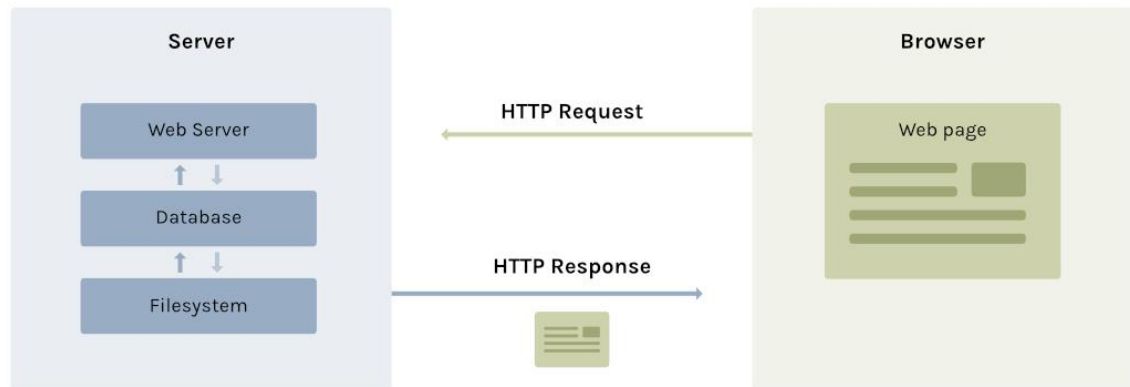


FIGURE 2. Simplistic background representation of a typical browser request.

SPAs, on the hand, once initially loaded do not reload in the same way. Instead, only parts of the application that receive new data re-render while other areas remain unchanged (figure 3). From the user's perspective, this translates to an unbroken and fluid experience.

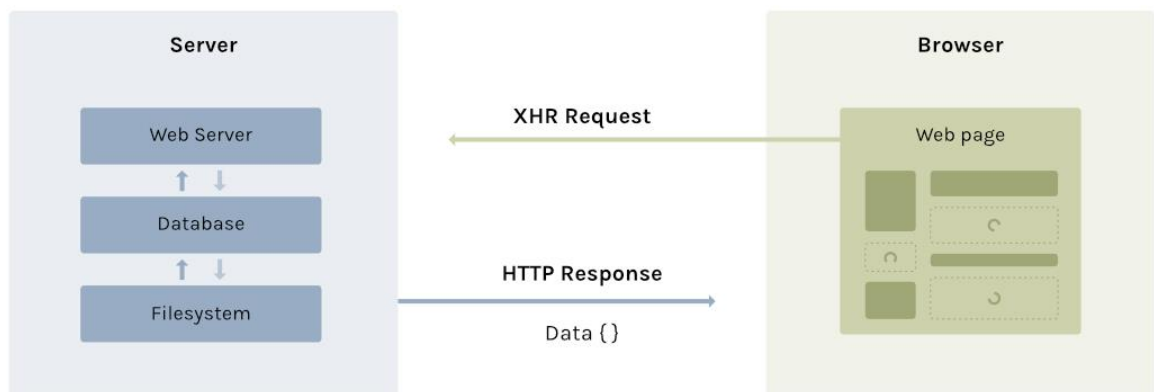


FIGURE 3. The web application sends requests whenever new data is needed.

In figure 3, the web applications upon needed new data opens requests through the XHR API. Upon the completion of the request cycle, new data is populated into the components where they are necessary. The server responds with data in

the shape of human readable text defined in a format, such as JSON, which then the SPA can easily reason about. Certainly, there are drawbacks when it comes using either paradigm, for instance, SPAs can weigh significantly more than normal web pages, the pros must be weighed against the cons. In Shuup's use case it would add extra steps in code maintainability, however, it would drastically simplify the backend logic as it will not have to consider the eventual form and design. The delegation of this task would need to be handled by the frontend.

A second and, in my opinion, more important objective is the re-organisation of the information architecture. The addition of new features over the years increased the number of items in the navigation and with it, cognitive load. Appendix 1 and 2 are screen caps from the old navigation panel which shows the contrast of available links. On the one hand, the contacts section sports only one link (appendix 2), whereas the shops area is chock full of various options that command the same level of importance (appendix 1). There are many such examples that viewed separately do not pose much of a concern but when viewed in concert create consistency issues. Moreover, each link in the list represents a page that content wise is superfluous and does not merit a separately defined area.

Moving along the list of grievances, the action button area that lies at the top right of most pages require extra attention. The non-existent categorisation of action types has led to a mishmash of meaning and layout (picture 4).



PICTURE 4. Action buttons in the product editor view.

Picture 4 is a representative case where the button layout can be confusing and unnecessarily complex. The delete and discard buttons are prominently placed next to a button with an opposite action (save). Furthermore, the difference between deleting and discarding changes can be misconstrued due to their ambiguity. On the far left, the button labelled "Actions" poses more questions as to its functionality.

A common theme throughout the Shuup admin panel is the number of text fields as can be seen in appendix 3. In this example, the text fields span throughout all sub-views listed on the left-hand side menu. This menu acts as a tabbed component which hides and shows the main content based on the category description. The weakness of tabs is that it also hides important information such as mandatory fields (appendix 3). The new entry cannot be saved until all marked fields are completed, however, this is impossible to tell without cycling through all the tabs. In addition, the form makes certain assumptions about the type of the entry and tries to box it into a model that may or may not be relevant in every scenario. Forcing users to acclimate to a wide range of suppositions leads to subpar experience that only serves as obstacles. An audit of all forms is necessary to optimize common tasks and fix issues that may bring about cognitive overload.

Finally, the design needs a refresh to align with the new branding and design system. Shuup has undergone several personality changes over the years and the admin panel has not been considered when the improvements were undertaken. Given the importance of the admin panel, the Shuup technical team wished that I take charge no holds barred and design as well as implement this vision. Along with this main goal, it became evident that the new design cannot be enacted without modernizing the frontend software that would drive this effort. Consequently, the skills required to tackle these tasks suppose design and programming experience.

3.5 Research & Mockups

With the major points of the objectives defined, I set aside a couple of days for researching the competition landscape. The problems that we are solving in the Shuup redesign are not unique and do crop up in any suitably large software products. What's more, I wanted to compile a resource for common patterns that especially e-commerce products follow. In the product editor, for instance, what kind of features are highlighted and what do those components achieve?

This line of questioning is related to the scope of rethinking the convoluted information architecture in Shuup. The simplification of the interaction flows can

be achieved by determining a common thread between all “create new” actions, such as creating new products, categories, contacts, orders, etc. I wanted to see how products like SquareSpace, Shopify or Square handle these aspects. At the same time, I paid special attention to the quality of the design and user experience of these wizards and components to spot trends. Similarity of user interfaces can suggest user experience patterns that are good to follow, once again pointing to Steve Krug’s Don’t Make Me Think mantra (Krug 2014, 11-12). In essence, familiar experience breeds comfort in users.

Mood board is a great tool for collecting and displaying data from which you can springboard your own ideas. I looked at four SaaS products that solve different pain points in e-commerce. Shopify for storefronts, Stripe for payments, SquareSpace for marketing site building, and Square for brick and mortar PoS interface. This collection of products, I picked consciously for their design, metaphor and business similarities to each other and to Shuup. Refer to appendix 4 to get a sense of how this mood board looks like.

In addition to identifying common traits, I wanted to have a well of material to draw inspiration from for the new Shuup design language. The key here is to target a general feel or emotion based on choice of colour palette, contrast, typography, layout, copy, and so forth. As no design happens in a vacuum, my approach down the line is to remix and create a spinoff of certain components for their UX and UI. This is later evidenced in the update of the table filter system in Shuup (appendix 5). Moreover, I needed to document how Shopify et al. handle common e-commerce tasks that are by nature repetitive. For example, fulfilling orders, editing product images, interacting with analytics and chart data, etc. By breaking down and analysing individual components with regards to how they work and look, I can draw conclusions that will influence the design down the line.

A strong wish early in the project was to create low fidelity mockups and test them thoroughly before committing a single line of code. As the original plan commanded for a clean slate for the admin dashboard, the design and layout can take a large departure from the existing solution. Therefore, breaking or neglecting existing features is undesired and the new dashboard should have feature parity with the old one. To this effect, we elected to use InvisionApp as

the team already had prior experience with it. This product is designed for quick and easy prototyping by simply collating static mockup images into a living device screen. The way it works is the user uploads ready-made images that describe every section, interface element and element state in the application. Within InvisionApp, the images can be ordered and connected by defining “hotspots” that are clickable areas to trigger the loading of the relevant images. This technique allows for building powerful prototypes that can mimic the ready-made application without writing any code. The time is cut between idea and testing, and we can get the prototypes into the hands of users at lightning speed.

For Shuup, I created two prototypes over the span of a few weeks. One for the PoS application and one for the dashboard. In appendix 6, you can compare the old dashboard design with the mockup. At this stage, I have collected enough design resources and inspiration to tackle the task of coming up with my own layouts. I felt confident that in the mockup stage I could go for more precise copywriting to give it a more real feel. Furthermore, it is generally encouraged to include real content as soon as possible in the design process (Prototypr 2018). This gives legitimacy for testers when interacting with the prototype, something that a standard Lorem Ipsum would not be able to accomplish.

With the PoS app, I went to the farthest possible length to capture every single user flow and present the exact way in which the UI would react to input. These include slide-out components that show and hide when accomplishing a wizard type action. For instance, when going through the checkout process there are certain steps that must be broken up into logical order. Scanning/adding items to the list, editing customer details and accepting payment were given a good amount of reasoning to make it seamless (appendix 7).

Feedback was received through InvisionApp from the team which made for frictionless collaboration. Working remotely, this allowed us to be on the same page at all times. Each morning I set aside time for catching up with messages and creating tickets to appease the feedback. As a result, iteration was a daily occurrence which kept me consistently looking out for problems and fixing them before they became a larger design issue.

3.6 Project management & collaboration

Shuup is composed of a local team in Vancouver, Canada, however, the rest are spread out globally. One of the biggest challenges in this project is coordination and real-time communication. The 10-hour time zone difference meant that most of the back and forth happened through Slack messages left during the previous day. The team lead suggested that I leave them a daily assessment at the end of the work day. This solution worked for the most part, however, any questions or feedback understandably could not be dealt with immediately.

On my part, the work required planning and foresight. At Anders we use Jira for project management and time logging. I was given carte blanche to write and break down my work into as many tickets and epics as was necessary. This gave me the possibility of employing Kanban principles and be able to plan and reason about my work. Furthermore, the Shuup team was also able to track what I was working on at any given point. I divided my work into Backlog, In Progress and Finished sections, the most basic Kanban setup. Every big feature entailed its own “epic” which, in turn, had its aspects broken down to tickets that are laser focused in nature.

Time wise we were constrained to three months from the start of the project to delivery, preferably, as soon as possible. However, the design and research were just one aspect of a larger body of work that accomplishing it within this period would not have been realistic. The Shuup team’s insistence on meticulous wireframing and prototyping ate a large portion of the allotted time and, thus, the project plan required revision. In software development, time is always the enemy and in this case, it was no different.

The original plan called for a clean slate redesign. Rewriting the dashboard from the ground up in modern frameworks and methodologies would have entailed rewriting of a large portion of the backend. As a result, this was quickly scrapped, and we settled on simply updating the existing stack and giving the old dashboard a paint job. Crucially, this would save a considerable amount of time and would put us back on track to meet the deadline. This cost us the chance to bring Shuup up to user experience parity with the competition and serves only to kick the can

farther down the road. It is not an ideal solution, however, in this business pivoting is an essential tool in the developer's arsenal. The PoS app work was rescheduled for a later date and, thus, all effort was focused on the dashboard. Consequently, a tentative deadline was set for late August. The dashboard rewrite project was pushed far into the future, however, this update could be a spring board for that cause.

3.7 Technical implementation

With the change in the plans, we elected to keep the existing system for the time being. This decision made a few ripples in the project. First, all existing JavaScript plugins, libraries and packages needed to be brought up to date. In some cases, such as the date picker component, alternatives had to be found as the existing solution had been abandoned by the author. The delicate balance of the more than 30 scripts was disturbed and invariably many broke. Furthermore, we moved away from using Gulp task runner to using Parcel. Gulp is a JavaScript task runner for which Shuup had custom functions written that dealt with minifying JavaScript files, compiling LESS markup into CSS, concatenating multiple files, etc. Parcel is a new asset bundler that is faster and does not require separate configuration. Second, the styles were moved from LESS to SCSS in order to make way for upgrading Bootstrap 3 to Bootstrap 4. The latter upgrade was challenging as the version 4 is a large enough departure from the previous that it broke most of the layouts across the board. The dashboard is composed of dozens of partial HTML templates and all of them needed to be combed through.

During this process, I removed a lot of cruft from deprecated software. I completely removed Bower package management software from the project as it is deprecated. JavaScript packages installed with Bower were reinstalled through NPM and configuration files were removed. This change simplifies package administration to the current standards. Next, Parcel was introduced and as a result, Gulp was eliminated as it was slow and clunky for the Shuup's use case. During Shuup's installation, a step in the CLI wizard is to compile and make the frontend code available for production. This involves creating bundled JavaScript files that are transpiled from ES6 to ES5 standard, this is minified and placed in the appropriate production directory. Previously, this stage took a considerable

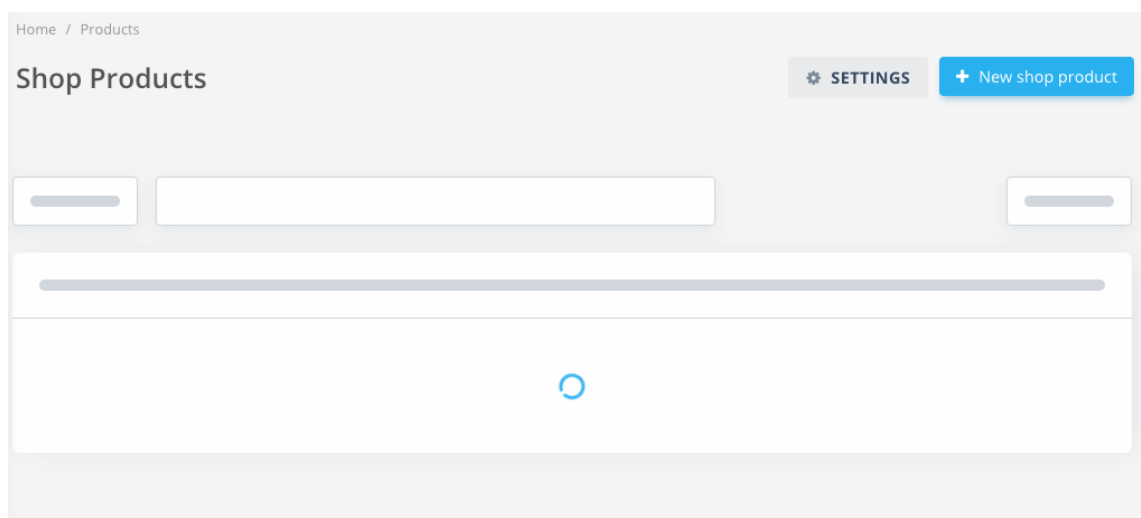
amount of time to complete using Gulp, with Parcel it was reduced to 7 minutes. The transpilation process involves a JavaScript tool called Babel that effectively translates new JavaScript (ES6) standard to the older and more supported ES5. Only newer browsers understand ES6, while older browsers, such as Internet Explorer 11, are missing language features which prevents code written in ES6 to function in such environments. Having Babel in the chain means that the transpiled code will be module scoped and many of the global methods will be unavailable for certain plugins and scripts to consume. This caused many of the JavaScript code to malfunction and prevented many of the UI components to work. These were manually fixed by giving access to the local scope as was necessary. Where that failed, I assigned the necessary functions to the global scope which is a Band-Aid solution at best.

Most of the legacy libraries and even new ones that we are using, e.g. JQuery, are written in such paradigms that it does not take advantage of ES6 features like module imports and tree shaking. Meaning to say, if only one method is necessary from jQuery to complete a task in a separate script, we must import the whole jQuery object to have it available. That is the case with older libraries, another example being Lodash. The result is a larger than necessary final (after transpilation) file size. Though, this is mitigated by chunking the code into individual files that are loaded only on pages that require them. The other side of the coin is increased difficulty in the maintainability of the scripts. In the future, Shuup's scripts will go through new modernization efforts and all the same issues will crop up in new ways.

Some dependencies could not be updated at all since they rely on old version of libraries whose APIs have changed in such significant ways that they do not offer backwards compatibility. Such is the case with Shuup's own table and file browser scripts. They are based on an old version of the Mithril JavaScript framework. Updating Mithril to the latest version was not an option, therefore, it had to be kept as it was. The table and file browser components are essential to the admin dashboard and would need to be rewritten from the ground up. That would have needed a decent amount of time for which there was no plan. Thus, edits to the functionality of the table were made within the constraints of the available version of Mithril.

The Bootstrap update task was straightforward as the tool made resources and guides available to make the transition as smooth as possible. That did not pose a challenge, what was more difficult was going through each class name and determining if it was dead or active. That is to say, if Bootstrap 3 class names were deprecated in the latest version. The sheer number of template files complicated the process and bogged it down. The redesign touched many of the template files and consequently I had to learn my way around relatively swiftly. With no previous experience developing in Shuup, this process felt like learning to swim in the deep end.

The most significant change I have contributed was the update in the UI and UX of the table component used across the admin panel. I identified a few UX problems that I wanted to rectify. In appendix 8 the old table design featured filtering text fields underneath the table headers. As far as information architecture goes, having them in that place links them with the headers and, therefore, by association it is obvious as to what they control. That said, functionality wise it immediately applied the filters as the fields received text which caused the table to freeze while the logic in the background worked. The non-responsive nature of this scenario creates the illusion of slowness and jank. I opted to move the filters into a separate menu as seen in appendix 5. Additionally, to solve the unresponsive issue I created a loading UI state to communicate to the user that there is something happening in the background (picture 5).



PICTURE 5. Table loading state.

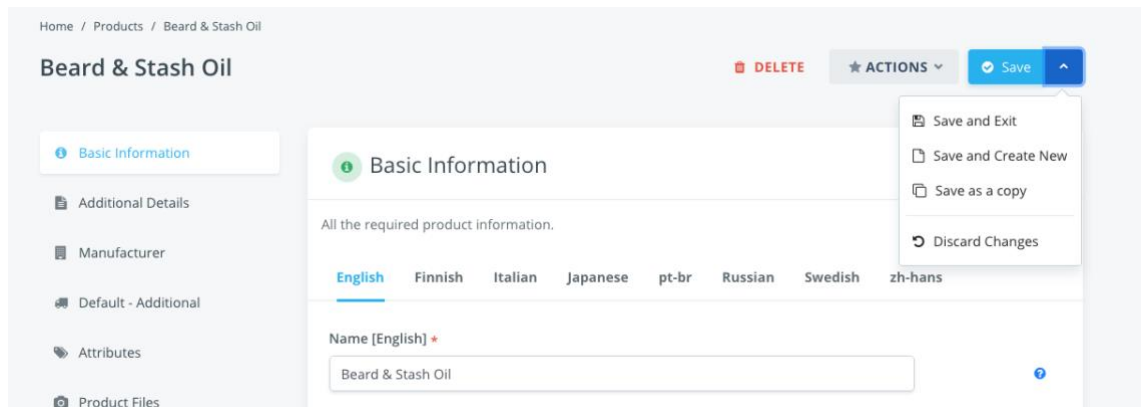
This shows up any time the table is loading new information from the Shuup backend, however, not when there is filtering in progress. This, I did not have time to properly address. Furthermore, design of the skeleton loading state is twofold, first as stated before, the progress indication for users. The time it takes for the data to load depends on the quality of the connection the user has to the server where Shuup is hosted on. The skeleton stays in the loading state up until the data is ready to be presented. Second, the skeleton takes up the shape that the eventual data will be in. As a result, the user has a good idea of what the eventual screen will contain. This serves as a role to prime or to condition users which lowers the perceived wait time of UI loading (Chung, 2018).

In the cases where there is no data to display, previously the table loaded without any content. I designed an empty state that instead of showing nothing, it rendered text that guides the user into committing an action through which data can be populated. This can be seen with the campaign section used as an example in appendix 9. The focus area is the message that is displayed where it prompts the user with a task to follow. The implication here is that by going through with the task, the user will see the result on that very page.

With regards to the filters, depending on the available table columns, the filters appear under the namesake menu. Name and general text filtering are extracted to the top of the table as those are frequently used. With the general layout of these elements, I took inspiration from Shopify and moulded the existing components and functionality accordingly. I also added a checkbox to each row as the original table design lacked a visual representation of the selection action. Before, the rows were highlighted whenever a user selection took place.

I detailed the page action buttons problem previously and in picture 4, the old button layout can be seen. In the next phase, I consolidated the button layout and brought clarity to the meaning based on context and proximity. Actions that deal with page state are grouped into one dropdown with clear delimitation of the type of action. In picture 6, the dropdown menu features a discard function that is grouped separately as it is a destructive option. All variants of the save are placed closely together. Button design wise, I elected to highlight the affirmative action

with the primary colour, whereas the delete button is deemphasized to avoid doubt about the nature of the action.

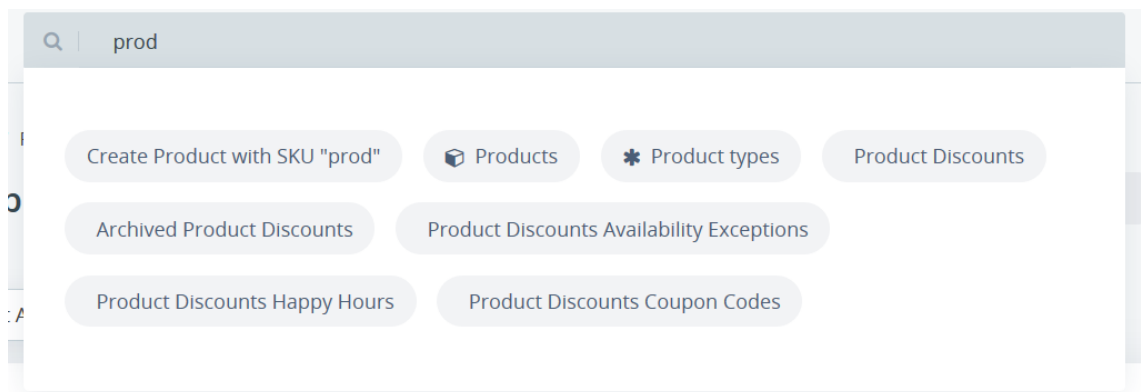


PICTURE 6. Product page action buttons.

Next, I focused my attention on laying down the ground work for a simplified navigation bar. The old navigation solution can be viewed in appendices 1 and 2. The slide out style stacked menu featured grouping that visually decluttered the links at the cost of taking up a great deal of horizontal space. My mission throughout the project that did not get realised is the restructuring of the information architecture. With this in mind, in the new menu I opted to go with a simpler more straightforward design. Each dropdown expands and reveals a set of links with minimal wait in between. Later, through testing I learned that users could navigate more rapidly than before, however, the number of links proved to be hindrance when searching for a specific page. The problem is illustrated in appendix 10 where the open and closed states of the menu can be seen. The open state with a dozen links can stretch the height of the menu below the viewport of the browser. With the number of pages all pushed into the main menu this is bound to happen. The solution is far more complex than adding more white space or sections to the menu. The underlying cause will persist until a large information architecture overhaul project can be launched. Appendix 11 offers a view into how I propose to move around the pages. Instead of having multiple pages with one option, it would be possible to combine related settings on a handful of pages. Each setting would receive its area in sequential order. Going further, to limit the amount of options on a page, we can categorise them into tabs

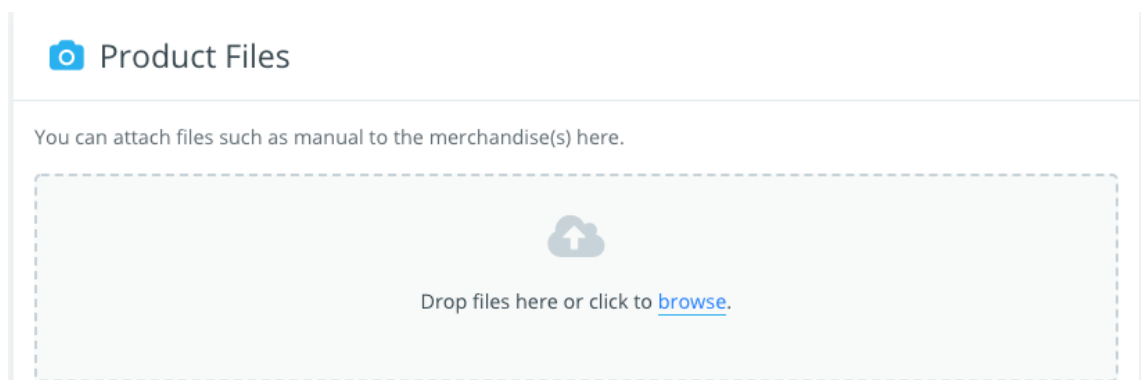
or accordion type UI elements. Thus, we limit the cognitive load that user face without straight up limiting the options for customisability.

The majority design changes that remained were quality of life fixes and details that brought the whole design up to the new standard. The last phase of the redesign took up much of the remainder of the time as there were dozens of areas that required fine tuning. The search bar received new animation additions that were meant to create a delightful experience.



PICTURE 7. Search bar.

What cannot be seen in the upper image is the animation when a search function is being undertaken. Moving along, I took care of the finer UI details wherever it made sense. For instance, for the upload area I reworded the prompt to communicate the available options more clearly to the user (picture 8).



Also, I redesigned the drag state to react when files are being moved with the pointer over the area to indicate that the UI is ready to receive the item.

I improved the visual style of the Shuup setup wizard. It is the area of the admin dashboard that guides new users through the necessary steps to successfully set up their shop (Appendix 12). The steps indicator, conveys the complete and unfinished tasks more prominently. Since it is the first screen the user sees after installing Shuup, the team felt that it needed to be eye catching and easy to follow.

For the login page, I brought the design in line with the rest of the dashboard and added an additional small animation. Usually, after the user enters their email and password and hits the login button there is a slight wait until the server responds with results. This period of time is undeterminable as it depends on the user's bandwidth among other things. Similar to the table UX solution, I added a loading state indicator that appears in place of the clicked button (Appendix 13). The immediacy of the UI response carries the natural meaning that the action is being processed. It is a trivial addition that can make a big difference in the perceived user experience.

I continued improving elements wherever I ran into them. This includes areas that are nested within pages, such as the individual tabbed content of the order creator. I focused on readability of tabular data within these areas and endeavoured to lift important information to the top (Appendix 14). Appendix 14 relates the order details page where the contextual information is at the top of the tabular data. Furthermore, I emphasized the data headings for improving scanability. I did not have the chance to rework the presented information, however, for future consideration I would omit unnecessary headings. I am referring to scenarios where the content's formatting already alludes to the type of data it belongs to. Email, phone number and date, as an example, headings are superfluous and can be removed. Such considerations would go a long way into cleaning up and reducing parsing fatigue.

Throughout the project, I synched my changes on a forked copy of the official Shuup repository hosted on GitHub. Occasionally, I rebased on top of the new changes that were added to the upstream repository. The rules in place for commit history for Shuup are strict in the way that it must be easy to view changes being made. The acceptable format is one commit per feature or change so that it can be easily reverted if necessary. I did not manage my own commit history

as carefully as was mandated and I had to go back and forth squashing together past commits and editing the general history. After I ordered my git history I submitted the final Pull Request, which is a terminology in git that refers to synching up my Shuup branch with the original. That is to say, I formally ask the maintainers of Shuup to accept the proposed incoming changes. In this phase, the project leaders can request changes to be made to individual files which falls on the PR submitter's shoulders to amend. Finally, the redesign was merged into the main source code which marked the end of the Shuup redesign. Additional changes and improvements will go through the same process. This practice is the standard way in which collaboration happens in the software development industry.

3.8 User testing & project results

There is not much said about user testing in this project as the nature of this endeavour was insisted to be closed. The project leader and Shuup CEO, described it as a sensitive matter that must not be shared even with co-workers within the mother company, Anders. As a result, broad user testing was not a viable option, however, I still recognized the need for validation. Moreover, there was no usage data collected by Shuup that I could have relied upon. Things like heat maps, page usage statistics or user feedback were unavailable to me. That said, I will re-iterate and paraphrase what Steve Krug champions in his book, testing with one user is better than not testing at all (Krug 2014, 115). In my case, it is more poignant and clearer than ever. I chose to test the bigger changes I introduced as those areas are the ones I can refine. I made use of two users, one who had prior knowledge of Shuup and another who did not.

The first user expressed uncertainty about the navigation menu's functionality where only one submenu is revealed at once. Upon opening a different item, the previous one closes. With regards to the table, the user had less trouble sorting through items and perusing the selection system than previously. The marked improvement seemed to be the easily available main sorting textbox, the prominently placed text field.

The second user, with no working knowledge of e-commerce, had difficulties with understanding the scope of the tasks I gave. For instance, when creating new products, the abundant options seemed to have had a confusing effect. The user indicated a lack of awareness as to how the varying options would affect the result. This points to ambiguity with the product creation wizard, therefore, my suggestion remains steadfast, Shuup requires an overhaul with its information architecture and information hierarchy. Furthermore, performing extensive user research on current usage would go a long way into understanding how the product is being used and in what circumstances.

The inability to test the redesign did not cripple the goal. After pivoting from the initial plan, the scope was adjusted to improving the design without resorting to backend disruptions in any significant way. There is little new user experience to fully test that was not there before the project was started. It made little sense to test these areas as the acquired data would be of little use in case of an extensive overhaul.

Large changes cannot be committed under such constraints as a lot of the issues can be traced back to decisions made during the planning of the underlying logic. Time constraints prevented meddling with the backend in a meaningful way, as such changes need to go through testing, review and iteration. The mockups made during the initial phase can be further refined in the future to shed new light in this regard. Moreover, the initial research is as relevant as ever if Shuup decides to fuel the rewriting process outlined in the first version of the plan.

4 CONCLUSION

Chapter 2 of this thesis provides an ideal scenario in which a software design proposal is turned into deliverables. From research, planning, mockups and testing to implementation there is a clear path informed by decades of collective experience outlined by established experts in the field. Steve Krug and Jesse James Garrett are prominent figures when it comes to User Experience and Information Architecture. The insight they provide is based on decades of hindsight that holds true regardless of technological evolution. The principles they delineate are as applicable in software as they are for other types of products physical or digital. That is because they consider the user first and foremost through what is known as user-centred design. It is the study of how users *use* the products, how it affects them and what can be done to improve these events. It is learning by observing to understand your work from multiple angles. This leads to incremental betterment which translates to satisfaction, increased statistics, customer loyalty and, at the end of the day, higher profits.

In Chapter 3, I sought to bring all this knowledge to Shuup where there was a lack of coherent user experience considerations. In the span of a handful of months, I contributed a redesign with a few user interface overhauls wherever necessary. The mockups and prototypes for the Point of Sale application and a new Shuup admin dashboard, even though not realised, remain as a source for future rumination. The lessons received from the performed initial competitor research are reflected in the mockups and should be taken into account for when the Shuup team decides to rework either interfaces, PoS and the admin dashboard.

In the wild, often the designer does not have complete control of the given project variables, time and budget. Every project has different constraints that the designer must adhere to in order to meet expectations. Accordingly, apart from technical skills, soft skills play a pivotal role. Communication, planning and team work take precedence. Despite, missing critical elements, such as extensive user testing, this thesis is very much a “do as I say and not as I do” situation. Understanding these principles while still being in the position to deliver makes for better hindsight in the future when faced with a similar scenario.

At the end of the day, the Shuup team was pleased with the work, albeit the general feeling was that it ran past the deadline. This project challenged my technical skills. With no prior knowledge of Shuup's source code I had to wade through a lot of code and learn many new paradigms to effectively accomplish my goals. However, this is the general case in software development. Developers are often thrown into unfamiliar projects and are expected to thrive and think on their feet. The time it takes to acquiesce to new surroundings separates the senior developers from the juniors. Experience speaks volumes and with enough projects I expect to follow the same path in my personal development.

REFERENCES

Blumenfeld, B. 2018. The explosion of design tools has come at a cost. Released on 7.13.18. Read on 27.11.2018. <https://goo.gl/a7G9F4>

BuiltWith. 2019. eCommerce Usage Distribution in the Top 1 Million Sites. Released 11.4.2019. Read on 11.4.2019. <https://trends.builtwith.com/shop>

Chung, B. 2018. Everything you need to know about skeleton screens. Released on 19.10.2018. Read on 20.4.2019. <https://uxdesign.cc/what-you-should-know-about-skeleton-screens-a820c45a571a>

Coleman, B. & Goodwin, D. 2017. Designing UX: Prototyping. Communicate and Test Design Ideas. Melbourne: SitePoint.

Crunchbase. 2019. Shopify. Released 11.4.2019. Read on 11.4.2019. <https://www.crunchbase.com/organization/shopify>

Fanguy, W. 2017. A comprehensive guide to design systems. Released on 1.12.2017. Read on 26.11.2018. <https://www.invisionapp.com/inside-design/guide-to-design-systems/>

Follett, J. 2017. The designer's perspective on prototyping. Released on 15.3.2017. Read on 28.11.2018. <https://goo.gl/njxatu>

Garrett, J. 2011. The Elements of User Experience. User-Centered Design for the Web and Beyond. Second Edition. Berkley: New Riders.

Google. 2018. Material Design. Read on 26.11.2018. <https://material.io/design/introduction/#principles>

Grant, N. 2018. Adobe Buys Magento for \$1.68 Billion to Target E-Commerce. Released on 21.5.2018. Read on 10.4.2019. <https://www.bloomberg.com/news/articles/2018-05-21/adobe-buys-magento-for-1-7-billion-to-boost-commerce-ambitions>

Janetak, N. 2018. How to Start and Finish Any Web App. Released on 6.3.2018. Read on 9.11.2018. <https://goo.gl/Zg5ENN>

Krug, S. 2014. Don't Make Me Think, Revisited. A Common Sense Approach to Web Usability. Third Edition. Berkley: New Riders.

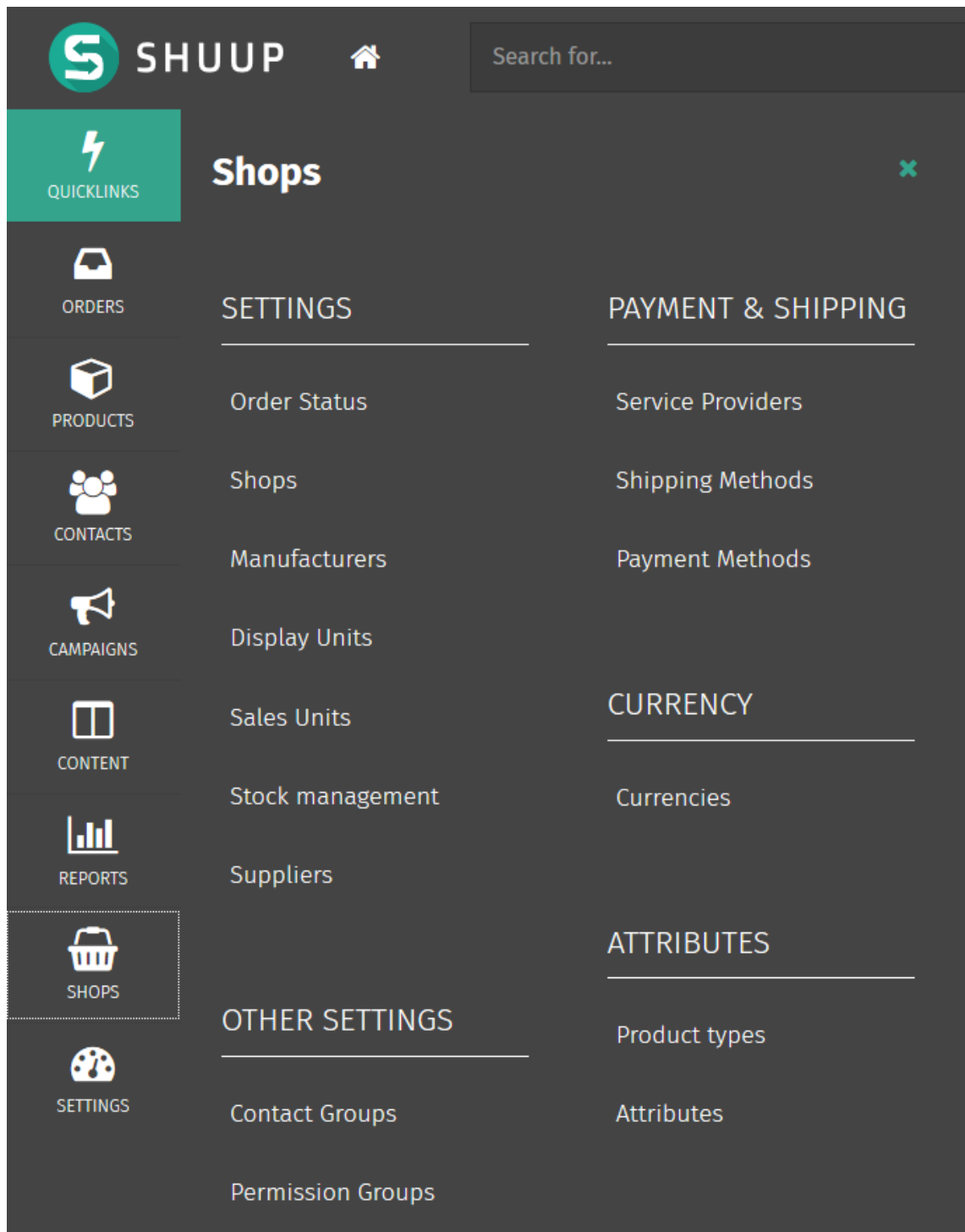
Laplente, P. A. 2007. What Every Engineer Should Know about Software Engineering. First edition. Abingdon: Routledge.

Mind Matters. 2018. Facebook's old motto was "Move Fast and Break Things". Released on 10.19.2018. Read on 9.4.2019. <https://mindmatters.ai/2018/10/facebooks-old-motto-was-move-fast-and-break-things/>

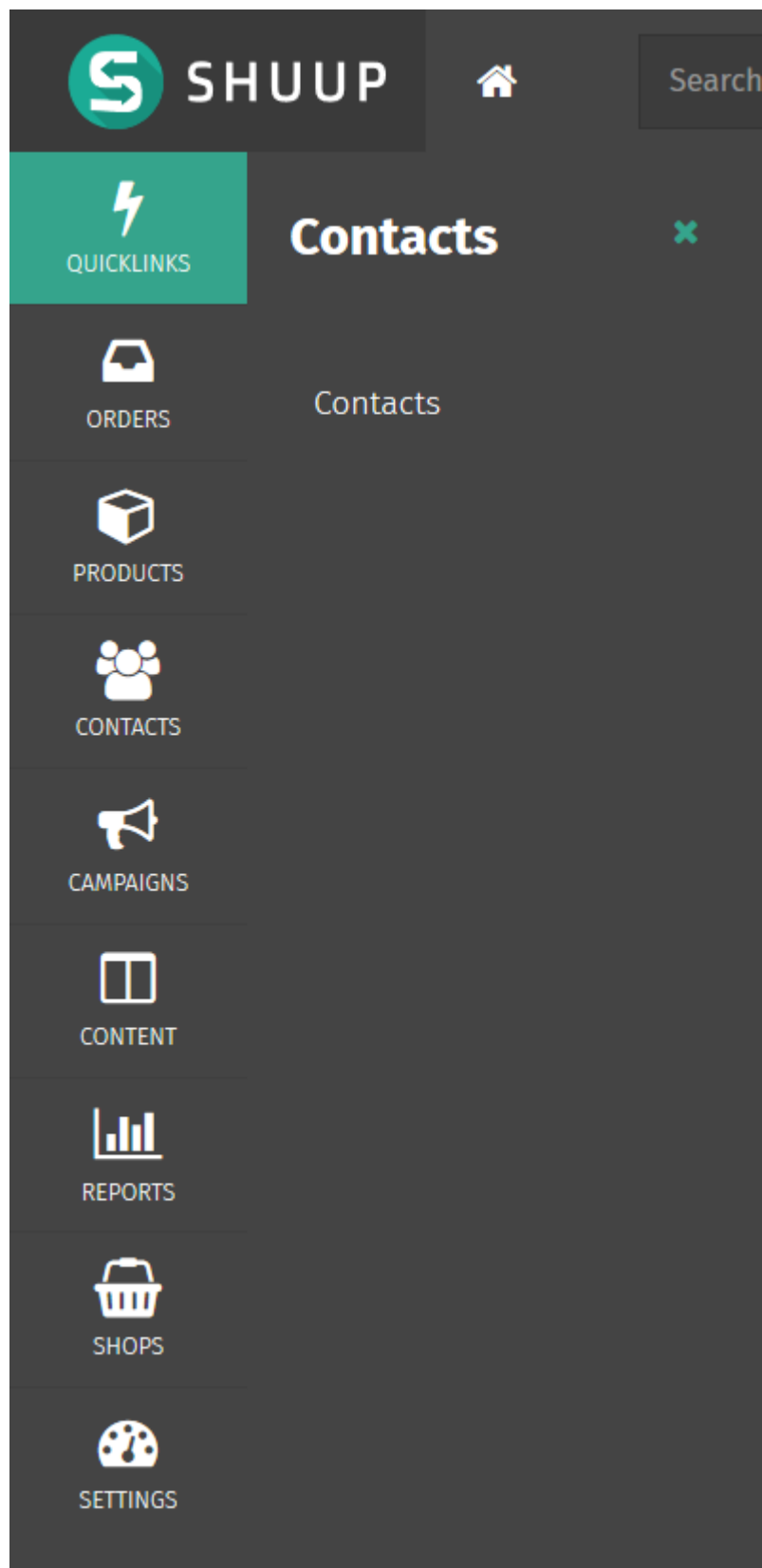
- Mulder, S. & Yaar Z. 2007. The User is Always Right. A Practical Guide to Creating and Using Personas for the Web. Berkley: New Riders.
- Prototypr. 2018. Content First, Design Second: Prototyping with Words and Adobe XD. Released on 19.09.2018. Read on 18.4.2019.
<https://blog.prototypr.io/content-first-design-second-prototyping-with-words-and-adobe-xd-c4c07cac21ef>
- Ries, E. 2011. The Lean Startup. How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Publishing Group.
- Rogers, P. 2018. Big Brands using Magento. Released on 20.1.2018. Read on 10.4.2019. <https://vervaunt.com/big-brands-using-magento/>
- Rosenfeld, L. & Morville, P. 2002. Information Architecture for the World Wide Web. Second Edition. Sebastopol: O'Reilly.
- Ruiz, J. 2017. Information Architecture. The Most Important Part of Design You're Probably Overlooking. Released on 16.04.2017. Read on 20.11.2018.
<https://goo.gl/ndNgNT>
- Ruluks, S. 2016. When to redesign your website. Released on 13.6.2016. Read on 12.4.2019. <http://blog.froont.com/when-to-redesign-your-website/>
- Shelford, T. & Remillard, A. 2003. Real Web Project Management. Case Studies and Best Practices from the Trenches. Boston: Addison-Wesley Professional.
- Statista. 2017. Number of internet users worldwide from 2005 to 2017 (in millions). Released on 1.7.2017. Read on 7.11.2018. <https://goo.gl/7JHuuW>
- Statista. 2019. Retail e-commerce sales worldwide from 2014 to 2021 (in billion U.S. dollars). 2019. Read on 10.4.2019.
<https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/>
- Summers, B. 2015. 9 bad reasons for a website redesign. Released on 29.1.2015. Read on 12.4.2019. <https://www.invisionapp.com/inside-design/9-bad-reasons-for-a-website-redesign/>
- Tarver, E. 2018. How Much is Dropbox Worth. Released on 14.10.2018. Read on 27.5.2019. <https://www.investopedia.com/articles/markets/082015/startup-analysis-how-much-dropbox-worth.asp>
- bin Uzayr, S. 2016. The Rise of Sketch App in UI Design. Released on 9.4.2016. Read on 27.11.2018. <https://envato.com/blog/sketch-app-popular/>
- VB Staff. 2017. Demand is booming for web developers — along with need for more tech skills. Released 15.08.2017. Read on 9.11.2018.
<https://goo.gl/ZTD5qS>

APPENDICES

Appendix 1. Shuup settings panel



Appendix 2. Shuup contacts panel



Appendix 3. New product page

New shop product

Save

Basic Information

Additional Details

Manufacturer

Shuup demo (Easylin) - Additional

Contact Group Pricing

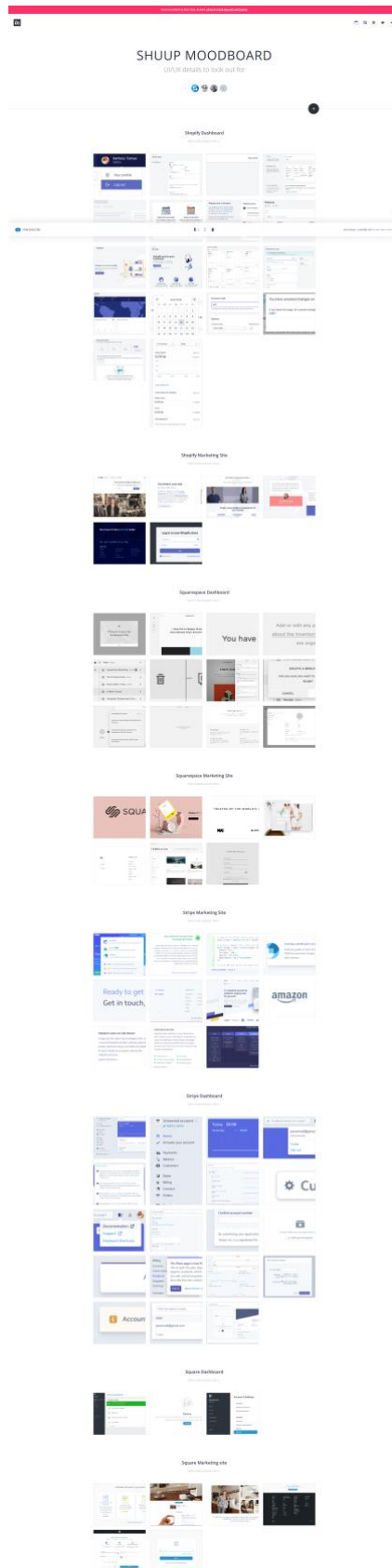
Contact Group Discount

Additional Details

Product features such as size, weight and keywords are located here.

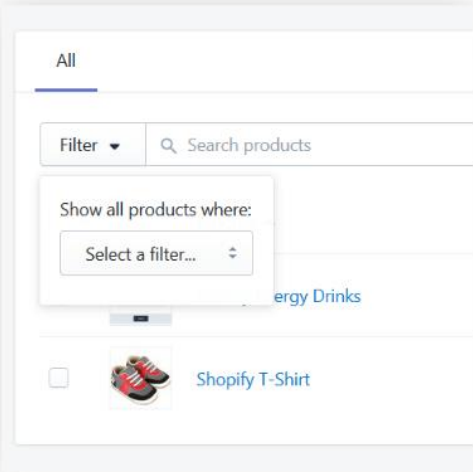
Stock *	unstocked	?
Shipping mode *	shipped	?
Tax class *	vero 1	?
Barcode	Barcode	?
GTIN	GTIN	?
Width (mm) *	0	?
Height (mm) *	0	?
Depth (mm) *	0	?
Sales unit *	Pieces	?
Net weight (g) *	0	?
Gross weight (g) *	0	?

Appendix 4. Shuup mood board

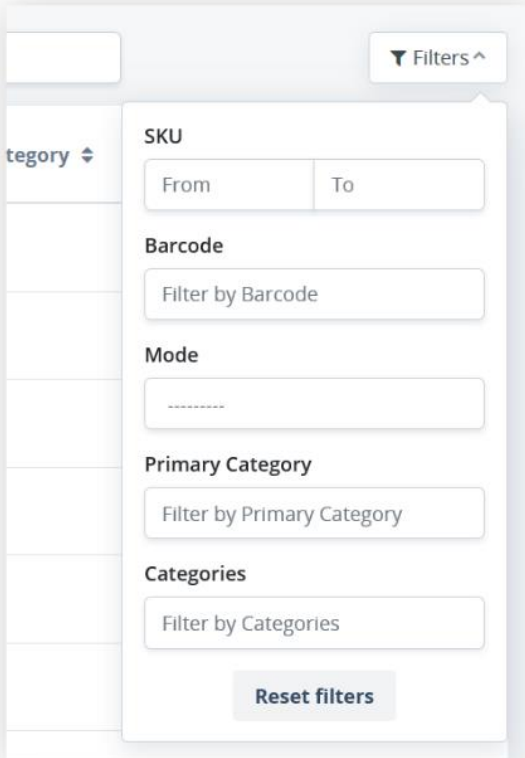


Appendix 5. Comparison of Shopify and Shuup product filtering system

Shopify

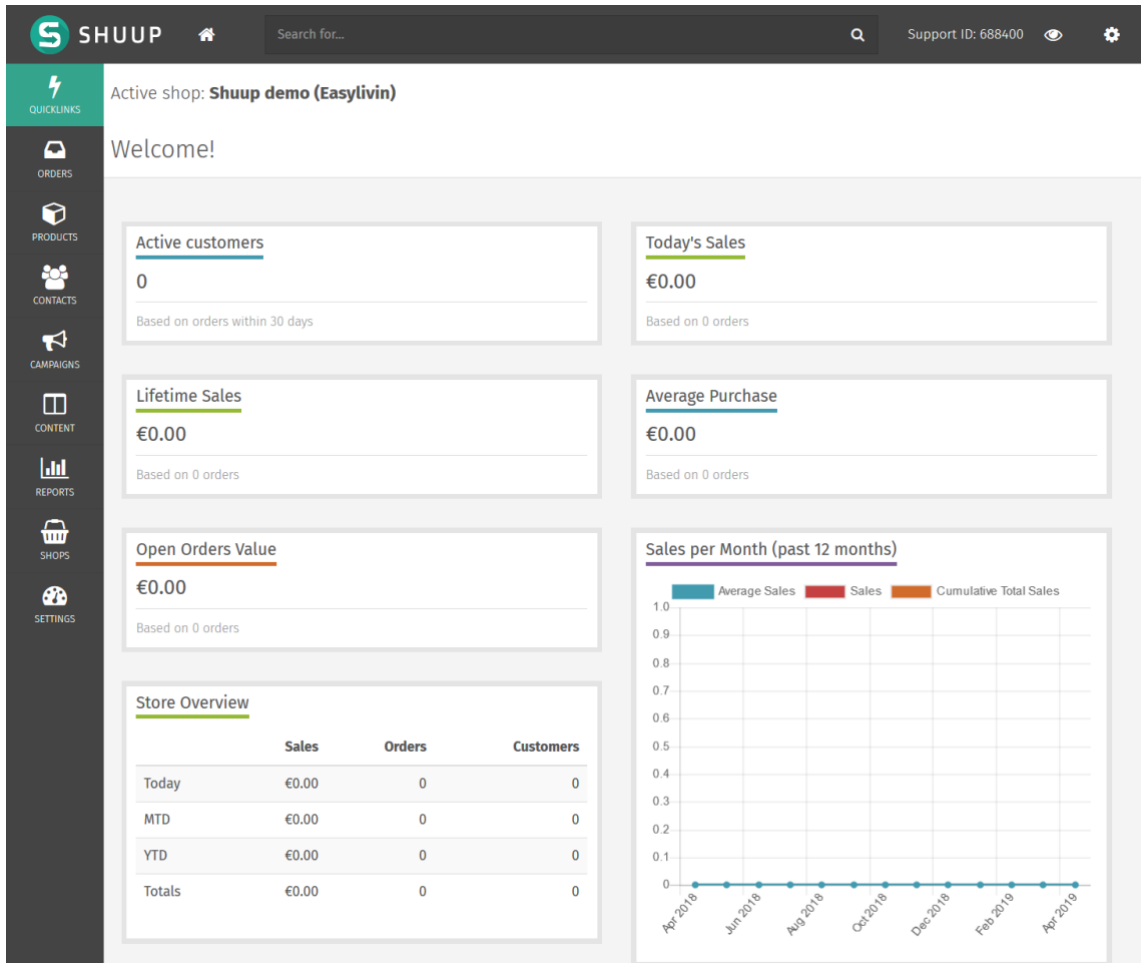



Shuup



Appendix 6. Admin panel dashboard mockup

1 (2)





TK

Hey Tamás, welcome back!
Here is how your store has been doing lately

MY SHOP

- Dashboard
- Orders 10
- Products
- Customers
- My Shop

APPS & SETTINGS


- Apps
- Settings

Dashboard
Configure Your Shop


● Daily Tip

Have you thought about connecting your PoS system to Shuup?


Visitors



Customers

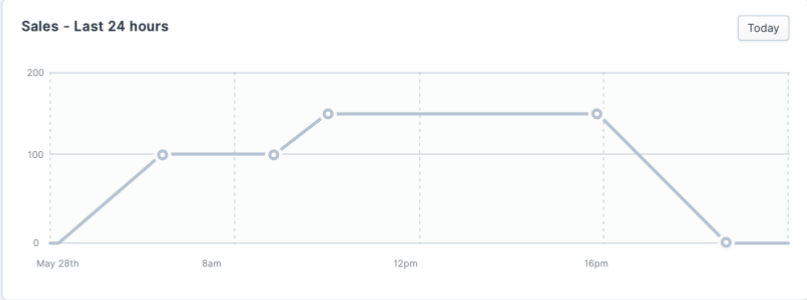


Orders



Sales - Last 24 hours

Today



Latest Orders

<input type="checkbox"/> Order	Date	Customer	Payment Status	Shipping Status	Total
<input type="checkbox"/> 5698069993	08-21-2018	Harry Day	pending	Not Shipped	\$26.58
<input type="checkbox"/> 0940103354	07-24-2018	Sophie Little	pending	Not Shipped	\$498.99
<input type="checkbox"/> 1349800157	08-21-2018	Christine Cunningham	pending	Not Shipped	\$55.38
<input type="checkbox"/> 1349800157	07-24-2018	Jeffrey Miles	pending	Not Shipped	\$7.59
<input type="checkbox"/> 1805991527	08-21-2018	Devin Walton	pending	Not Shipped	\$96.7
<input type="checkbox"/> 1805991527	07-24-2018	Iva Wood	paid	Shipped	\$19.07

See Orders

Appendix 7. Admin panel dashboard mockup

The dashboard is divided into several sections:

- Product Lists:** A sidebar on the left with options for 'All Products', 'List 1', and 'Custom List 2', along with a 'Create List' button.
- Product Grid:** A central area with a search bar and a list of products, each with a thumbnail, title, description, price, and a 'Remove' icon.

Product Name	Price
Multivitamins for Dogs	\$19.95
Natural Balance Pot...	\$82.29
Petpany Dog Rubber Toy	\$24.04
Dog Food	\$12.47
Welness CORE Natural...	\$59.06
Pedigree Food	\$87.18
- Basket:** A right-hand panel titled 'Order #001' showing customer information (Michael Powell), a list of items in the basket with quantity and price, a quantity selector, a 'Freebie' toggle, and a 'REMOVE' button.
- Payment:** A section with an 'Add Discount' button and a 'Discouts' summary of \$0.00.
- Order Summary:** A 'Cancel Order' button and a 'Total \$19.95' button.
- Navigation:** A bottom bar with icons and labels for 'Checkout', 'Orders', 'Customers', and 'Menu'.

Appendix 8. Shuup product table before and after



1 (2)

Home / Products

Shop Products

[Settings](#) [+ New shop product](#)

Select Action Items per page: 20 Showing 2 of 2 shop products [Reset filters](#)

Primary Image	Name	SKU	Default Price	Stock
	AXXIND ZIGBEE VALAISIMEN HIMMENNIN, VALKOINEN	höpö-llöpö	100.000000000	stocked
	Fixed ALR screen	2	3549.000000000	unstocked








Previous 1 Next

Home / Products

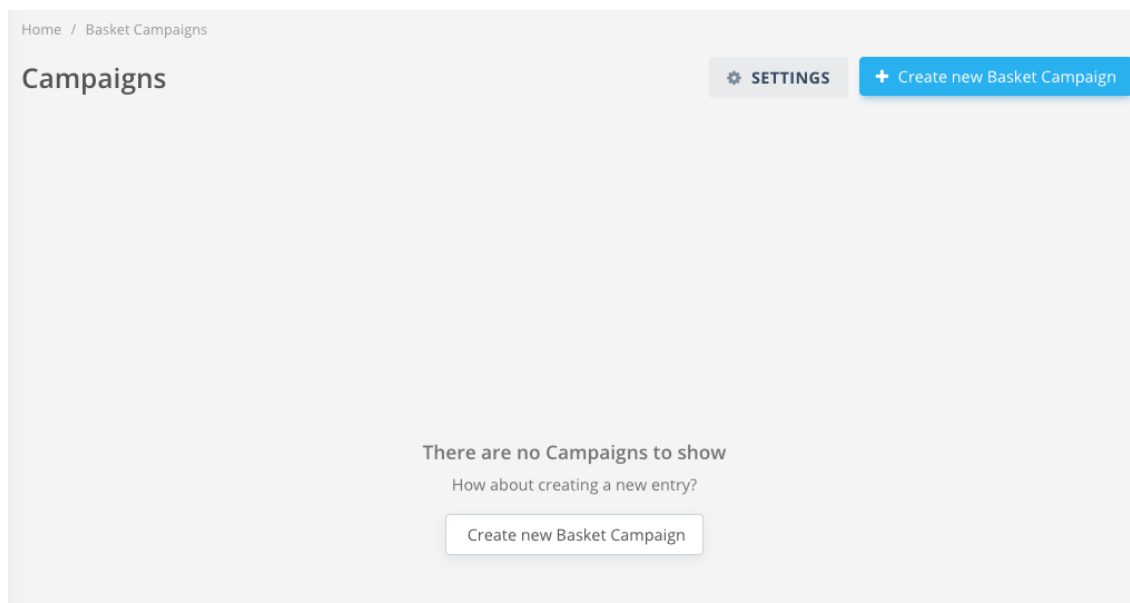
Shop Products

[Settings](#) [+ New shop product](#)

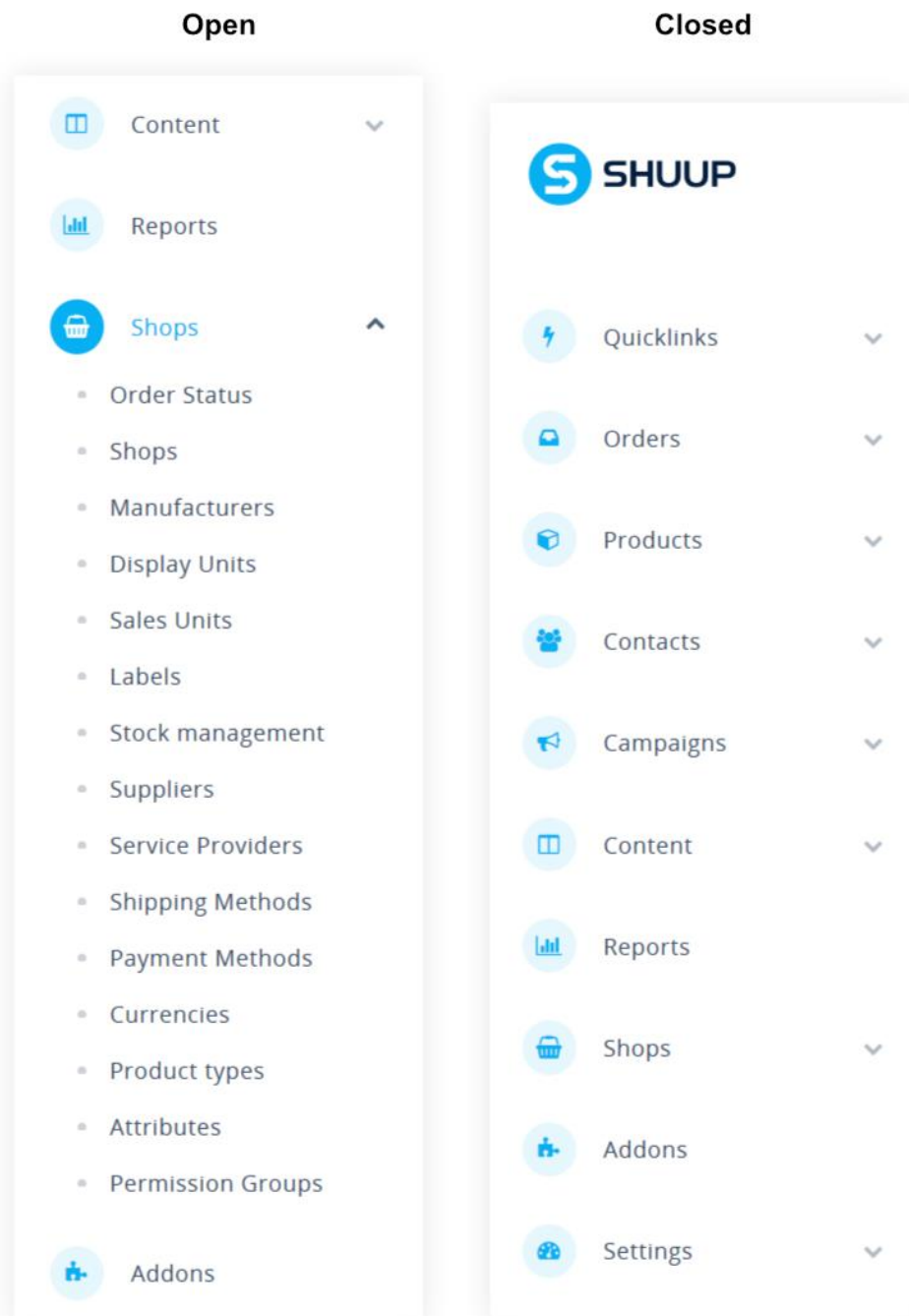
Select Action

	Name	SKU	Mode	Default Price
<input type="checkbox"/>	 Beard & Stash Oil	zratKIBOHB	normal	100.220000000
<input type="checkbox"/>	 Smooth Viking Beard Brush	TRNlloIsth	normal	221.850000000
<input type="checkbox"/>	 La Vie Est Belle	zrbtVAvQXP	normal	327.440000000
<input type="checkbox"/>	 Still Jennifer Lopez Perfume	ozuEaidnpr	normal	268.930000000
<input type="checkbox"/>	 Bvlgari Omnia Crystalline Eau de Toilette	pUnqXAcOfs	normal	34.730000000
<input type="checkbox"/>	 Tommy Gird by Tommy Hilfiger Perfume	KsitylgsNe	normal	14.540000000
<input type="checkbox"/>	 Clinique Happy	eVseiEcuft	normal	46.840000000


Appendix 9. Table empty state



Appendix 10. Dashboard main menu in open and closed state



Appendix 11. Mockup settings page

TK

MY SHOP

- Dashboard
- Orders 12
- Products
- Customers
- My Shop**

Preferences

- Appearance
- Pages
- Payment Methods

APPS & SETTINGS

- Apps
- Settings**

Pet Store

How Great Is The Strength Of Your Belief

Site details

Change your shop's customer facing information

Order Configuration

Free Beauty Samples What They Are And How To Find Them

Payment

How Does An Lcd Screen Work

Shipping

Today S Orthodontic Treatment Comfortable Convenient

PoS Configuration

The Perfect Pot For Every Preparation

Shop Name edit

12 characters left

Short Description

21 characters left

Contact Information edit

800 Wilshire Boulevard, Los Angeles, 90001, CA

Description

50 characters left

Rewards edit

Extra points per order **0.00**

Allow cashiers to give points manually?

Dollar points ratio **100.000**

Order percentage grant points (%) **0.00**

Order points payment threshold (%) **0.00**

Minimum Order Total edit

Order Minimum **\$0.00**

Order Minimum Active

Other Settings edit

Checkout required fields

Payment Methods Add

Name	Method	Price	Status
Manual	Cash	\$0.00	<input checked="" type="checkbox"/>
MasterCard	Card	\$0.00	<input type="checkbox"/>

Shipping methods Add

Name	Price	Status
As soon as possible	\$0.00	<input checked="" type="checkbox"/>
Pickup	\$0.00	<input type="checkbox"/>
Express Delivery	\$50.00	<input checked="" type="checkbox"/>

Delivery Service Area edit

Configure the delivery areas for your shop directly in the map.

Money Locations Add

Name	Status
Safe	<input checked="" type="checkbox"/>

Transaction Reasons Add

- Cash in (system)
- Cash out (system)
- Money lost between days (system)
- Order payment (system)
- Unkown (system)

Appendix 12. Shuup setup wizard

S SHUUP

Search for...

admin

Welcome to Shuup!

You're almost done! Complete the following steps to sell your products online

Complete the setup wizard

- Shop Details
- Payment Provider
- Carrier
- Theme
- Content & Behavior
- Sample Data

Add a product to see it in your store

- New product
- Import

Add a product category to organize your products

- New category

Add a logo to make your store stand out

- Add logo

Add some users to help manage your shop

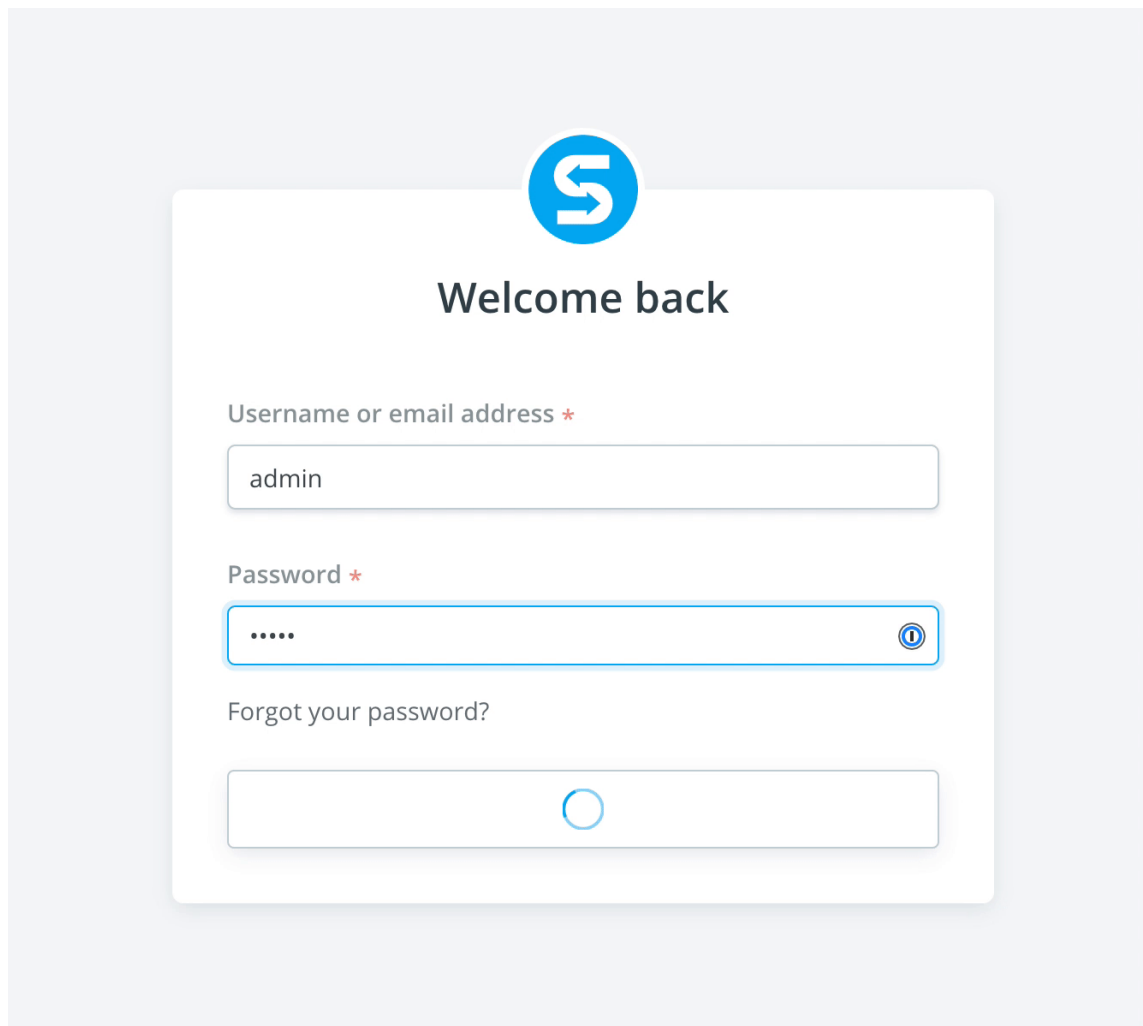
- New user

Publish your store


Let customers browse your store and make purchases

- Publish shop

Appendix 13. Login page subtle animation




The image shows a login page with a subtle animation. At the top center is a blue circular logo with a white 'S' and a right-pointing arrow. Below the logo is the heading "Welcome back". The form contains three main sections: a username field with the text "admin", a password field with four dots and a blue eye icon, and a "Forgot your password?" link. At the bottom is a large white button with a blue circular icon in the center. The entire form is set against a light gray background.



Welcome back

Username or email address *

Password *

[Forgot your password?](#)

Appendix 14. Order details page

Home / Orders / Order 1 (Jane Doe)

Order 1 (Jane Doe) Edit order Set Status Actions

Details

- Payments
- Shipments
- Log Entries
- Admin comment/notes
- Printouts

Order Details

Order Status: **In Progress** Payment Status: **fully paid** Shipping Status: **fully shipped**

Order Number	1	Customer	Jane Doe
Order Date	Aug 15, 2018, 1:31:40 PM	Creator	admin
Reference	180815021100800016	Phone	0407154511
Label	Default	Email	ll@google.com
Payment Method	PaymentMethod	Tax number	03265694
Shipping Method	ShippingMethod	Total Price (taxless)	€0.00
		Total Price	€0.00

Address Information

Billing address	Shipping address
Jane Doe	jane Doe
pansionkatu 4 a 32	pansionkatu 4 a 32
Testcity	pansionkatu 4 a 32
Andorra	Antigua & Barbuda