







LICOR: Beyond the Design System. A Proposal to Empower Teams to Develop Software in Compliance with the Principles of Accessibility, Usability, and Privacy by Design in the Extreme Contexts and Challenging Domains Post-COVID-19

Deivith Silva Matias de Oliveira¹ , Francisco C. M. B. Oliveira¹ ,
Cláudia A. C. Pernencar² , Breno S. de Moraes¹, João W. Silva¹,
Antonio R. B. Costa¹, Joaquim B. C. Pereira³, and Inga F. Saboia⁴ 

¹ Department of Computer Science, State University of Ceará, Fortaleza, Ceará, Brazil
deivith.oliveira@aluno.uece.br, fran.oliveira@uece.br,
{breno.morais,weslley.silva,rootedy}@ffit.com.br

² NOVA Institute of Communication, Nova University Lisbon, Lisbon, Portugal
claudiapernencar@fcsh.unl.pt

³ Institute UFC Virtual, Federal University of Ceará, Fortaleza, Ceará, Brazil

⁴ Department of Communication and Art, Aveiro University, Aveiro, Portugal
inga@virtual.ufc.br

Abstract. During the pandemic, people were asked to stay at home, which increased the demand for software. The quality of the software is improving as a result of this trend. In response, companies have accelerated their digitization processes to provide better quality software that is more accessible, user-friendly, and secure. Since the pandemic, software development teams around the world have struggled to meet deadlines during uncertain times. This poster addresses the difficulties developers and designers face in developing and managing digital health software. This is due to the growing appeal of low and no-code platforms that are becoming more accessible and user-friendly to non-programmers. The goal of this project is to develop and evaluate a prototype digital health component library known as LICOR. This software development kit contains integrated digital components designed specifically for the digital health industry. It helps even non-experts create effective digital health products. The design approach was developed by combining various theories and techniques such as design system principles, domain-oriented design, micro front-ends, and microservices. The methodology is based on the design-based research approach, which combines theoretical research with working and coded software artifacts to enable real-world testing. We are interested in contributions from the community and would be happy to discuss ethical concerns, bias, and illiteracy in digital health with academics. Finally, we discuss future research opportunities and the difficulties of connecting the design phase with the development phase in digital health software development.

Keywords: Digital health · Software development · HCI

1 Introduction and Context

With post-Covid-19, the world as we knew it has collapsed in a matter of weeks. Reality presents itself as a fragile, anxious, non-linear and incomprehensible world - the so-called BANI world [1]. The current crisis has reached proportions not seen since the Great Depression [2].

At the same time, the pandemic has increased demand for software products as people have been encouraged to stay home. People are not just demanding more software - they are demanding better software. Accessibility, usability, and privacy have become key requirements in today's world.

The pandemic also had a severe economic impact on businesses in general, forcing many companies to close their doors. The context COVID -19 has accelerated the digital transformation of businesses and entire industries such as retail, hospitality, and education [2]. Other businesses are increasingly impacted by digital technologies such as the Internet, mobile connectivity, cloud computing, Big Data, machine learning, artificial intelligence (AI), blockchain, and the Internet of Things (IoT) [2].

At the same time, consumers now have many more choices [2]. Customers are increasingly intrigued by digital technologies, forcing traditional businesses to seek digital transformation. With new or renewed value propositions based on digital technologies, other companies are moving from one sector to another. As a result, IT teams are under significant pressure to develop increasingly advanced applications with shorter deadlines.

Software engineering teams should be adaptable, especially when it comes to structuring the development and design phases [3]. In recent years, large companies have developed and implemented their own design systems to solve this problem [4, 5].

Design systems became popular because they focus on scaling design decisions and accelerating code implementation while ensuring software quality and code reuse. [6, 7]. However, unfortunately, because they became so popular as simple UI example libraries, they are generally misused in this way. [8]. The goal of a design system is to support the development of software by providing guidelines that impact the overall user experience and software development process [9]. Therefore, it is important that everyone on the team is familiar with these ideas and principles.

To make developers' jobs easier, libraries such as PrimeFaces [10] focus on existing, reusable microservices that are connected to other services, such as an API. This process promotes more efficient long-term development [11].

On the other hand, most of these libraries are generalists. There are no specific standards for certain domains such as digital health, which is a major gap. This is a critical issue since the main role of software is to solve problems relevant to the user's domain and situation.

If these fundamental concepts, such as resistance to adoption or even outright exclusion of people due to non-compliance with accessibility standards, are not recognized by users [12, 13], it can affect their experience.

The rapid pace at which developers and designers are required to become near experts in such specialized areas has led to a lack of competent responses [12]. However, design techniques can help teams organize their products during development [12].

In the next section, we will detail the primary and secondary objectives of this research. Just reinforcing that this is still an ongoing project and more detailed results (as well with your dear participation) will be available soon.

1.1 Objectives

The goal of this study is to develop and evaluate a prototype LICOR component library that combines theoretical and practical aspects of Design Systems [4, 6], Domain-Driven Design (DDD) [12, 14], Microservices [15] and Micro-Frontends [16] for building digital health applications..

The specific objectives are:

1. Learn how to use micro front-end approaches with Design Systems, DDD, and other methodologies.
 - a. Identify the most common micro-scenarios of interaction in the health industry;
 - b. Design and produce the micro front-ends.
 - c. Create two different apps that utilize the same domain, one for a medical consulting and one for medical auditing, utilizing these tiny front-ends;
 - d. Analyze the characteristics of these applications with consumers;
 - e. With this new approach, discuss the building procedure with designers and developers.
2. Propose other courses of action based on the study's findings.
3. Investigate how accessibility, usability, and privacy can be integrated into library components in the user domain.
4. Examine the software to determine whether domain information affects its compliance with code reuse, user experience, usability, accessibility, and privacy.

In the following section, we'll describe the LICOR library prototype.

1.2 Introducing the LICOR Library Prototype

Our prototype is based on the Material Design System [17]. Due to copyright compliance, it will be used only for study and internal experimentation only.

The software development process will be the focus of our study, with designers, coders, clients, and other stakeholders all included. We'll be able to compare the various viewpoints like this.

We will create all artifacts, phases, and necessary activities based on Driven Domain Design [18], which is focused on the digital health sector. Mapping and construction of the domain, including bounded context, general language, context maps, context architecture, and domain model patterns are a few examples.

All about our day-to-day equipment now. The Figma [19] program is being used to create and improve our design system. We're utilizing Storybook [20] for the code components library. We're always trying out new tools, like as Anima [21] or Zeroheight [22], and seeing how they impact the way we do research.

We're also inviting seasoned experts from areas like privacy, accessibility, usability, and ethics to review and offer constructive criticism on our theoretical framework as well as the quality of our pre-formatted beta goods.

We must declare that failing quickly is an important goal for us, too. As everyone who faces the risks of innovation understands, we aim to fail fast in order to learn more quickly. You will find out what research methods we are utilizing in certain parts of this paper.

In the following section, we'll show you a tiny but carefully chosen sample of what libraries of digital interface components can do for you.

2 Related Works

There are some open resources that companies and professionals are using today to achieve some of these goals, such as scalability and product consistency.

Popular design systems such as Google's Material Design [23] and Ant Design [24] provide a huge repository of design guidelines and materials, components, and templates, complemented by solid documentation that guides both designers and developers in their use and implementation.

Moreover, these design systems support various device types, not only the most popular mobile and desktop devices but also wearable devices and some IoT products.

In addition, these design systems support some widely used technologies used in digital products, such as popular JavaScript frameworks like React and React Native.

In some cases, such as digital health, the requirements are so stringent that these resources are not sufficient to enable the development of digital products that meet users' needs.

In the following section, we'll give a quick rundown of our prominent theoretical references as they assist us in our study.

3 Theoretical Reference

3.1 Design Systems

A design system is a set of interrelated elements that are organized in such a way as to assist the objectives of digital products [5]. The components we use to construct an interface are known as patterns: Processes, interactions, buttons, text boxes, icons, colors, typography, and so on...

One of the most popular design systems today is Material Design, maintained by Google - currently in its third version [23]. Other examples include Microsoft's Fluent [25], IBM's Carbon [27] and Ant Design, from the Alibaba Group [24].

Digital Health

According to [28], Digital Health is the combination of information and communication technologies (ICT) to generate and deliver trusted information about a person's health when it is needed.

The term “digital health” is broader than e-health and also includes more recent technological advances such as social media concepts and applications, the Internet of Things (IoT), artificial intelligence (AI), and others. Health IT projects aim to identify, prioritize, and integrate health programs, projects, and policies, as well as information and communication services and systems. The goal is to turn the vision of ESD into reality by using digital health for its integration [28].

The Conecte SUS program is a Brazilian federal government program that supports access to health information and promotes digital engagement between patients and health professionals. Despite the detailed program and action level, the document itself notes the need to consider the integration of new information services such as artificial intelligence (AI), analytics, Big Data, IoT, and other emerging technologies for knowledge discovery in healthcare. In addition to providing services such as expanding telemedicine and consolidating clinical terminology, the document also addresses improving telemedicine interoperability and regulating smart healthcare [28].

Looking at the Brazilian scenario, the question is whether all these efforts are compatible with a global strategy for digital health, as promoted by The World Health Organization (WHO). An example of this is the WHO ‘Recommendations on digital interventions for health system strengthening’ to promote the integration of technology worldwide for advances in health [29].

The WHO promotes the use of digital learning and training methods in the health professions so that they can be used alongside, rather than in place of, traditional training methods. WHO [30] advocates the use of cell phones for births, deaths, inventory reporting, commodity management, telemedicine, targeted patient communication, decision support for health workers, and digital tracking of health status and performance in specific situations.

According to WHO, any digital health tool or technology should be developed and implemented according to digital development principles. It also provides advice on how to create an enabling environment to promote the adoption of digital healthcare.

There are many issues in the digital health space that can be addressed, as you can see. We'll discuss how microservices and domain-driven design connect in the following section.

3.2 Microservices and Domain-Driven Design

A microservice is a small service that provides a tiny number of functions. Microservice designs are gaining popularity as a means of building cross-platform applications based on service-oriented architectures [31].

The Domain-Driven Design pattern provides concepts for subdividing microservices. Domain-Driven Design (DDD) is a software development methodology that focuses on modeling a domain based on contributions from domain experts [32].

Microservices use bounded contexts to identify microservices. Domain-driven design provides concepts, patterns, and activities for building a domain model in its pure form. However, it does not provide a step-by-step development approach. [31].

There is a lot of uncertainty about how to create programming interfaces for web applications for microservices. Microservices effectiveness requires application participation. [31].

The domain-driven design approach emphasizes that the core domain logic of microservices should be determined by the application, while the user interface is considered when creating specific web APIs. In a microservice, there is no graphical user interface, although there is one for a technical purpose. For a domain, you need to consider a variety of use cases. [31].

Although the use of Domain-Driven Design may vary in practice, there are several advantages. The layered architecture required by Domain-Driven Design distinguishes the domain from other concerns. DDD is a framework that provides fundamental ideas and processes for developing applications based on a microservice architecture. DDD is about breaking down a software system into smaller components, and microservices is about how they are divided. One of the main advantages of DDD and microservices is the ability to reuse existing functionality. With the Web API, you can find and reuse microservices [31].

You saw why we decided to try combining Domain-Driven Design with Microservices to provide a new way for digital healthcare. In the section that follows, we'll go over how design systems and micro front-ends are related.

3.3 Micro Front-Ends and Design System

It is critical to understand the LICOR experiment's micro frontend approach in light of its definition and benefits within the software development process.

Many projects are divided into numerous components, team structures, and technologies to mitigate the problem of collaboration between teams and coworkers growing more complicated and difficult as the size of a project and team membership grows, according to [11]. A front-end team and one or more back-end teams may be used to build horizontal layers.

Micro frontends provide a unique viewpoint. The application is sliced into vertical sections. Each slice is developed from the database to the user interface and carried out by a separate team. The client's browser is used to link the various team front-ends together [11]. Individual backend teams can now work on and modify system elements independently, which enhances their long-term development prospects. Also, because small services are more manageable than monolithic systems, their effective development in growing teams becomes more likely in the long run [9].

The service, as you may recall, is made up of a number of smaller services. It also includes several advantages that are particularly useful for team growth and product development [9]. This method is connected to microservices architecture. The major distinction is that the service now includes a user interface as well. This addition to the service eliminates the need for a central front-end staff. Micro front-ends attempt to keep things simple by lowering the complexity of each front-end while also adding complexity to the architecture's organizational layer, ensuring that subsystems are linked

to the overall system. This entails activities such as autonomous deployment methods for continuously integrating and testing new micro frontend systems, or establishing failover procedures in case a piece becomes unavailable for any reason. Improve customer satisfaction. Customers receive everything they need directly from the vendor. There are no API specialists or operational staff involved.

Let's look at some of the organizational and technological advantages of this design after you've learned what micro front-ends are [9]. Optimize for the creation of new features. The primary motivation for employing a micro front-end is to improve development speed. In addition, everyone working on a feature in the micro-front-end approach works on the same team and communication is easier. The unified front end will no longer be present. Modern architectures don't have a concept of scaling front-end development, as far as I know. The application is divided into smaller vertical systems with micro frontends.

As a result, a micro frontend is [9]: Concentrates risk in a smaller area, making it easier to understand; Has a smaller codebase that is easier to refactor or replace when necessary.

Let's look at the design of systems in the micro front-end environment based on their features and a stakeholder analysis [9]: The connection between design systems and frameworks. Frameworks are frequently used in web development to embed an application within a supporting framework that aids the software's growth. A design system is a framework for creating user interfaces that adhere to principles, patterns, and a design language. Because a compatible design system is required for the development of a frontend, its framework has an impact on how it develops. However, because the design system was built for a specific technology stack and is not framework-agnostic, it would be unusable for various stacks.

4 Materials and Methodology

4.1 Evaluation Methodology Plan

We will hold a workshop with representatives of a software development team to learn about their reactions during the activities and receive anonymous evaluations at the end.

The researchers will conduct this workshop in different steps following the evaluation strategy used by Oran et al. [26].

The study will randomly divide participants into two groups and have them use LICOR components in real-time development.

For this task, groups are to create an interactive UX / UI prototype for a local health problem. The two groups will need to create a prototype in one hour, one with access to LICOR UI components and the other with access to a free library.

After the time is up, 10 people will be randomly selected to use a prototype and evaluate it based on SUS (System Usability Scale) (Table 1).

Table 1. Distribution of groups, scenarios, and specifications

Group	Scenario	Library
Group A	A simple digital health specific domain app	LICOR library
Group B		Google material design library

5 Results and Discussion

This study is a work in progress. All results will be published in the full paper along the year. A long-term study will assess how adopting the LICOR library may influence the Brazilian software companie’s success in the second half of 2022.

6 Conclusion

A widespread problem is that digital health disparities exist and they must be addressed to provide consistent benefits [29]. To really identify genuine needs and introduce effective changes in the software development process to support a constant, adaptable, and successful digital health solution, we’ve established a scientific production calendar for 2022/2023. Researchers are invited to submit a Systematic Literature Review, A/B testing results, and software development process analysis.

Acknowledgement. To all of the HCII 2022 personnel, my Master Professor Dr. Fran Oliveira and my family who encouraged me while I was writing this study.

References

1. de Godoy, M.F., Filho, D.R.: Facing the BANI World. *Int. J. Nutrology* **14**, e33–e33 (2021). <https://doi.org/10.1055/s-0041-1735848>

2. Soto-Acosta, P.: COVID-19 pandemic: shifting digital transformation to a high-speed gear. *Inf. Syst. Manag.* **37**, 260–266 (2020). <https://doi.org/10.1080/10580530.2020.1814461>

3. Varajão, J.: Software development in disruptive times. *Queueing Syst.* **19**, 94–103 (2021). <https://doi.org/10.1145/3454122.3458743>

4. Abdi, M.: Fundamentals of design systems. <https://iconline.ipleiria.pt/handle/10400.8/5839> (2021)

5. Kholmatova, A., Magazine, S.: *Design Systems* (Smashing eBooks). Smashing Media AG (2017)

6. Fessenden, T.: Design Systems 101. <https://www.nngroup.com/articles/design-systems-101/> (2021)

7. Gu, Q.: Design system as a service. <http://urn.fi/URN:NBN:fi:aalto-202106217676> (2021)

8. Nguyen, D.: Why design systems are a single point of failure. <https://www.chromatic.com/blog/why-design-systems-are-a-single-point-of-failure/> (2020)

9. Klimm, M.C.: Design systems for micro frontends – an investigation into the development of framework-agnostic design systems using Svelte and Tailwind CSS. <https://epb.bibl.th-koeln.de/frontdoor/index/index/docId/1666> (2021)

10. PrimeTek: PrimeFaces.org, www.primefaces.org (2021). Accessed 21 Oct 2021
11. Geers, M.: *Micro Frontends in Action*. Manning Publications, New York, NY (2020)
12. Evans, E.: *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional (2003)
13. Fowler, M.: *Domain Driven Design*. <https://martinfowler.com/tags/domain%20driven%20design.html> (2021). Accessed 21 Oct 2021
14. Cycle, F.: Aprenda DDD (Domain Driven Design) do jeito certo. <https://www.youtube.com/watch?v=eUf5rhBGLAk> (2021). Accessed 21 Oct 2021
15. Richardson, C.: *Microservices From Design to Deployment* (2021). Accessed 21 Oct 2021
16. Geers, M.: *Micro Frontends*, micro-frontends.org (2021). Accessed 21 Oct 2021
17. Google: *Material Design* (2021). Accessed 21 Oct 2021
18. Evans, E., Evans, E.J.: *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional (2004)
19. Figma: <https://www.figma.com/contact/> (2022). Accessed 4 Jun 2022
20. Storybook: UI component explorer for frontend developers. <https://storybook.js.org/> (2022). Accessed 4 Jun 2022
21. Anima: Design to code: <https://www.animaapp.com/>. Accessed 4 June 2022
22. Zeroheight: document your design systems, together: <https://zeroheight.com/> (2022). Accessed 4 Jun 2022
23. Material Design: <https://material.io/> (2022). Accessed 3 Jun 2022
24. Ant Design: <https://ant.design/> (2022). Accessed 3 Jun 2022
25. Microsoft Design: <https://www.microsoft.com/design/fluent/#/> (2022). Accessed 3 Jun 2022
26. Oran, A.C., Nascimento, E., Santos, G., Conte, T.: Analysing requirements communication using use case specification and user stories. In: *Proceedings of the XXXI Brazilian Symposium on Software Engineering (SBES 2017)*, pp. 214–223. Association for Computing Machinery, New York, NY, USA (2017). <https://doi.org/10.1145/3131151.3131166>
27. Carbon Design System: <https://carbondesignsystem.com/> (2022). Accessed 3 Jun 2022
28. de Oliveira, A.C.S.F.M.G.N.N.J.P. de S.Z.M.L. dos S.C.M.E.M. da S.M.L.D.: Brazil's 2019–2023 National digital health strategy action, monitoring and evaluation plan. SUS (Executive Secretary of the Ministry of Health) IT Department (2019)
29. Jandoo, T.: WHO guidance for digital health: What it means for researchers. *Digit Health*. **6**, 2055207619898984 (2020). <https://doi.org/10.1177/2055207619898984>
30. Labrique, A., Agarwal, S., Tamrat, T., Mehl, G.: WHO digital health guidelines: a milestone for global health. *NPJ Digit Med*. **3**, 120 (2020). <https://doi.org/10.1038/s41746-020-00330-2>
31. Steinegger, R.H., Giessler, P., Hippchen, B., Abeck, S.: Overview of a domain-driven design approach to build microservice-based applications. In: *The Thrid International Conference on Advances and Trends in Software Engineering*. Unknown (2017)
32. Vernon, V.: *Implementing Domain-Driven Design*. Addison-Wesley (2013)