Pavlo Leinonen

Design System in a large organization: Posti Design System as a case study

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and Communication Technology

Bachelor's Thesis

29 April 2025

Abstract

Author:	Pavlo Leinonen
Title:	Design System in a large organization: Posti Design System as a case study
Number of Pages:	24 pages
Date:	29 April 2025
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Mobile Solutions
Supervisors:	Ulla Sederlöf, Senior Lecturer

The Posti Design System was used as a case study to understand the impact design systems have on large organizations through this thesis. The research looks into how design systems internally enable the efficient, consistent, accessible, and user-friendly digital products at the largest postal service provider in Finland.

The study addresses the basic elements of design systems and follows their growth from a simple shared component library to a complete design system. It looks at the system's technical structure based on React and TypeScript, and how teams work together to maintain and grow it.

Through a practical example - the StatusIndicator component - the thesis shows the process of creating a reusable UI element from start to finish. This includes identifying needs across multiple products, defining requirements, ensuring accessibility, and fitting with brand identity.

The research also addresses challenges faced by the Design System team, such as handling too many requests and avoiding unnecessary components. Current improvements include creating a structured release schedule and better prioritizing which components to build based on wider organizational needs.

By integrating user and maintainer perspectives of the system, this thesis offers valuable lessons for other organizations that are planning to build or are in the process of refining their design systems. It highlights the importance of effective communication, alignment of interests, and governance far beyond just technical solutions.

Keywords: design system, Posti, component library, accessibility

---

Tiivistelmä

Tekijä:	Pavlo Leinonen
Otsikko:	Design system suuressa organisaatiossa: Postin Design System tapaustutkimuksena
Sivumäärä:	24 sivua
Aika:	29.4.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mobile Solutions
Ohjaajat:	Lehtori Ulla Sederlöf

Design systemin merkitys yritysten digitaalisissa palveluissa on kasvava. Tässä opinnäytetyössä tutkittiin design systemin roolia ja kehittämistä hyödyntäen Postin omaa Postin Design Systemiä esimerkkinä. Suomen suurin postipalvelu on kehittänyt design systemin yksinkertaisesta komponenttikirjastosta kokonaisvaltaiseksi järjestelmäksi. Tekninen toteutus perustuu React- ja TypeScript-teknologioihin, jotka mahdollistavat tehokkaan yhteistyön eri tiimien välillä.

Hyvän viestinnän, yhteisten tavoitteiden ja selkeän hallinnan merkitys korostuu design systemin kehityksessä. Nämä tekijät ovat teknisten ratkaisujen ohella avainasemassa, kuten Postin tapaus osoittaa.

Työssä tarkastellaan käytännön esimerkkinä StatusIndicator-komponentin luomista alusta loppuun. Tämä prosessi havainnollistaa, miten design systemissä tunnistetaan tarpeita, määritellään vaatimuksia ja varmistetaan brändin mukainen toteutus ja saavutettavuus.

Design system -tiimin keskeisiä haasteita ovat lukuisten kehityspyyntöjen hallinta ja tarpeettomien komponenttien välttäminen. Posti on vastannut näihin haasteisiin kehittämällä aiempaa selkeämmän julkaisuaikataulun ja parantamalla komponenttien priorisointia.

Tämä tutkimus tarjoaa käytännönläheisiä tietoja organisaatioille, jotka harkitsevat oman design systemin rakentamista tai olemassa olevan kehittämistä. Postin kokemusten perusteella voidaan todeta, että hyvin toteutettu design system tehostaa tuotekehitystä ja parantaa digitaalisten palveluiden laatua.

Avainsanat: design system, Posti, komponenttikirjasto, saavutettavuus

List of Abbreviations

UX:	User Experience
UI:	User Interface
WCAG:	Web Content Accessibility Guidelines
W3C:	World Wide Web Consortium
NPM:	Node Package Manager
yarn:	Yet Another Resource Negotiator
UMD:	Universal Module Definition
API:	Application Programming Interface
CSS:	Cascading Style Sheets
DS:	Design System

1 Introduction

Design systems have become essential tools for organizations creating consistent digital experiences across multiple products. As companies build more websites and apps, maintaining a unified look and feel becomes increasingly challenging. Design systems address this by providing standardized components, clear rules, and guidelines that serve as a shared language for teams. This framework enables faster development while ensuring all digital products work together cohesively and reflect the same brand identity. For large companies, especially, these systems have become a critical part of their digital infrastructure, allowing them to deliver quality experiences at scale.

As Posti Design System takes shape, this thesis focuses on its implementation, maintenance, and design iterations from a developer's standpoint. Posti is the largest postal and logistics service provider in Finland. At the beginning of this thesis, the author was part of a product team that utilized the design system as a consumer. Along the way, the author switched to the design system team and gained direct experience on internal processes and their decision-making structure. This change offered an additional viewpoint that enhanced the practical analysis presented in this study.

The design system implementation, its technical structure, and its governance model are discussed within the context of the challenges the system faced during its lifecycle. It examines how the system has evolved in response to organizational needs and identifies the improvements being implemented to address identified limitations. Through this analysis, the work aims to contribute to the understanding of how design systems function within complex organizations and the factors that influence their success.

2 The role of design systems in large organizations

The increasing use of digital interaction channels, applications, services, and platforms by companies poses new challenges in delivering a seamless user experience. This is why design systems are gaining more influence and strategic importance in the digital landscape.

Design systems bridge the gap between design and development by encapsulating the specialized work of both frontend developers and designers. This encapsulation becomes particularly valuable in today's business environment, compelling businesses to adopt a full-stack solution. Rather than sacrificing speed for quality, design systems enable organizations to deliver solutions that are not only visually appealing but also designed and compatible with existing products, to all users.

This section looks at how design systems help solve company challenges by creating standard ways to design and build digital products. They help teams work faster while keeping quality high across all products.

2.1 What is a Design System?

A design system is a collection of reusable components, rules, and guidelines that help companies create consistent designs across all their products. It includes more than just visual elements—it provides a complete framework for building digital products that look and work the same way [1].

Essentially, a design system is a result of close collaboration between developers and designers, creating a unified resource that guides designer-users and developer-users in their respective workflows. It is a cross-functional tool that provides comprehensive resources within a cohesive package - including code snippets, front-end components, and technical documentation alongside design guidelines, patterns, style definitions, and reusable components [2]. This collaborative foundation ensures that the system covers the entire product development cycle, catering to the requirements of all parties involved in digital product development.

Design systems fundamentally solve the problem of gaps in brand coherence because of increased complexity in products. By providing a common design language that can be used in a consistent manner across various channels, platforms, and touchpoints, they eliminate redundancy [1]. This uniformity ensures that the brand maintains its design look while development time is improved

A design system’s greatest strength is that the sole proof of truth, which is the gap between design and implementation, is provided in one source. [1] With ready-to-use and tested parts with a guide on how they should be used, design systems encourage smoother collaboration between the various stakeholders which including the designers, the developers, the product manager, and others who create the digital product [3].

Design systems have evolved significantly from their earlier versions - static style guides and pattern libraries, which became common practice in the late 20th century [3]. With the growth of technology, reach from websites to various platforms and devices, organizations saw the necessity for more flexible, holistic systems that could grow along with their products.

Design systems offer several advantages in modern organizational settings:

- They streamline workflows by eliminating repetitive design and development tasks
- They enhance cross-functional collaboration through shared technology and resources
- They enable rapid scaling of design implementation across expanding product portfolios
- They preserve brand consistency even as teams expand or change over time
- They improve accessibility and usability by standardizing best practices

For large organizations that have numerous digital points of contact, a well-built design system is fundamentally beneficial. Different teams are able to work autonomously, and at the same time, a unified experience is ensured that strengthens brand identity and fulfils user expectations for every interaction

Design systems are an essential framework for organizations seeking to deliver improved user experiences efficiently at scale while maintaining an innovative balance in the face of advances in technology in every field.

2.2 The essential elements and architecture

A design system includes a blend of standards, tools and best practices. It shapes the way teams build and maintain their digital presence. While specific implementations may vary across organizations, effective design systems typically incorporate nine essential elements:

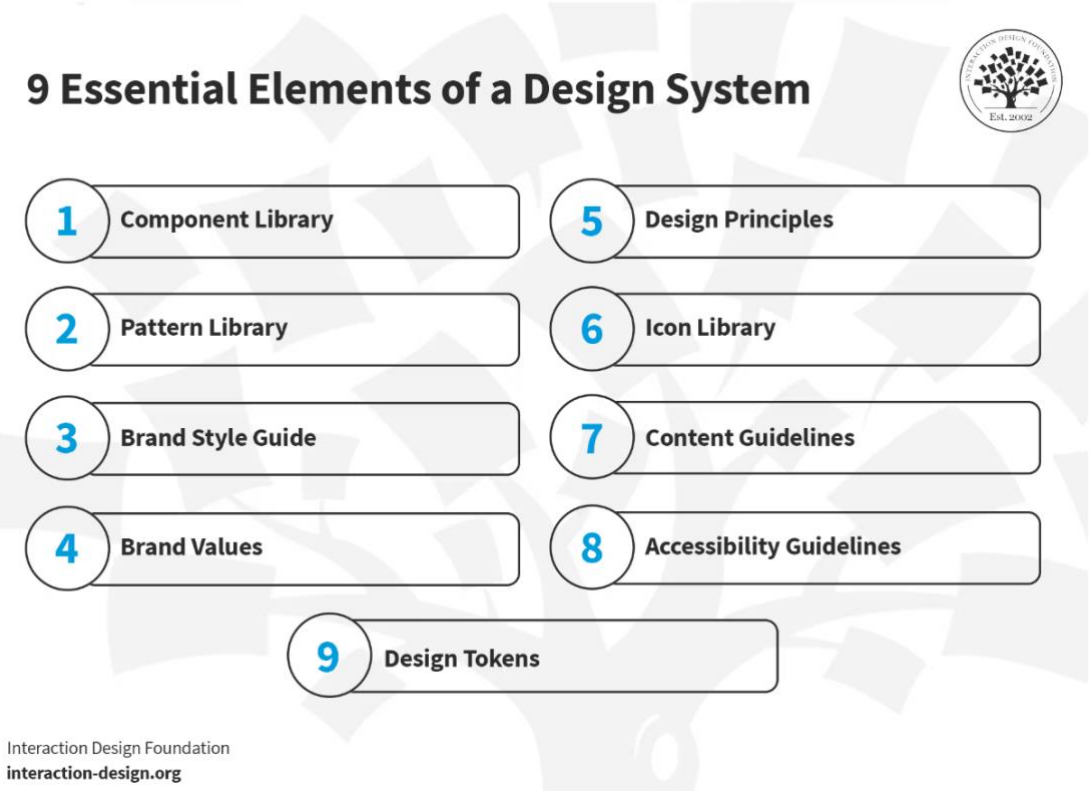

Figure 1. Interaction Design Foundation, CC BY-SA 4.0 [1]

These elements form a structured framework that ensures consistency, usability, and scalability in digital products. The component library contains reusable UI elements like buttons and menus, streamlining front-end development. The pattern library offers standardized solutions for common design challenges, such as login flows or contact forms, ensuring intuitive navigation. The brand style guide defines the visual identity, including

typography, color schemes, and tone of voice, fostering consistency across mediums.

At the same time, solid foundations like brand values, which shape a brand's identity, and design principles guide decisions for functional, aesthetically pleasing designs. Supporting elements like the icon library (visual symbols for navigation), content guidelines (rules for tone and style), and accessibility guidelines (ensuring inclusivity for all users, addressing aspects like color contrast and typography) enhance usability and communication. Finally, design tokens (variables for colors, dimensions, etc.) provide consistency and scalability, allowing seamless updates across the system. Together, these elements create user-centered digital experiences while maintaining efficiency and scalability

2.3 Value of Design Systems in organizations

The importance of design systems is particularly notable in large organizations with numerous products, multiple teams, and intricate digital ecosystems. The lack of systematic methods to address these challenges grows significantly as companies encounter difficulties maintaining consistency, quality, and efficiency across portfolios.

In these environments, Design Systems provide significant advantages across the entire product development cycle. Developers benefit from well-documented and maintained component libraries that speed up development and reduce technical debt. Designers work more efficiently with unified principles and standardized resources, which guarantee consistent creative output across various projects and teams.

The systematic organization of design tokens and icon libraries in a central repository, which makes asset management easier for developers and designers and promotes smooth communication. For business stakeholders, the consistent application of brand values and following style guides strengthens brand identity and builds market recognition across all product interactions

For organizations with regulatory requirements, integrated accessibility guidelines ensure products meet legal standards while simultaneously expanding market reach. Having these factors incorporated into the design system becomes not just advantageous but also necessary in 2025, as accessibility becomes more and more required by law in many areas.

When effectively adopted by large corporates, design systems evolve from being mere productivity-enhancers to shift value generators that manage the juxtaposed dual forces of creativity and consistency, innovation and reliability, as well as quality and efficiency. The upfront cost incurred in developing a complete design system is recouped over time as the organization grows, which further strengthens its design portfolios. In turn, the system becomes especially

useful for large-scale business entities that have sophisticated digital estates spanning across various markets and products.

3 Roadmap of Posti Design System

This section tracks the development of Posti's Design System over time. The progression from basic component sharing to implementing a complete design system represents a significant organizational evolution.

The section explores the challenges faced without a formal system, how teams operated independently, and the steps taken to create more structure. The journey illustrates difficulties that occurred when scaling design and development practices within the organization.

Examining Posti's evolution from scattered components to an organized system provides valuable insights into the development of design systems in organizations. This development reflects the growth of design governance in parallel with organizational development and the foundational approaches to design, which increase effectiveness.

3.1 Before the Design System

Before implementing a formal design system, Posti's UI component sharing strategy, like most growing organizations, went through various phases before a formal design system was put in place. At first, it started with a very primitive reusable library approach which unfortunately lacked the necessary structure and policies for scaling design operations. The basic shared library gave an instruction but not a theory.

The first attempt at reusing components in the company resulted in a very basic JavaScript bundle which was later published to (Node Package Manager) NPM. Although it provided some level of recognition as using shared materials, the execution was far too oversimplified. The library was mainly a stack of styled components, buttons and input elements, etc. all incrementally exported. This tried to implement controllable standards to support some level of unity but still, remained too crude and tailored to developer needs.

This preliminary shared library ran completely voluntarily of the participants at work and with no control whatsoever. Every team was free to add components as they deemed fit, and other teams could independently use those if they wished. As one developer from this period recalls: "At some point, each team was adding their own custom components, which bloated the library. There was little control over what elements were being added." Without consistent standards or review processes, the quality and consistency of components varied significantly.

In the initial stages, product teams typically functioned independently, implementing individual designs of UI components or modifying an open-source library to fit Posti's branding. This led to considerable organizational inefficiencies as different teams tried solving the same design problems in distinct ways.

The voluntary nature of this approach created several significant challenges:

- Inconsistent implementation: Without design guidelines, similar components developed by different teams often had conflicting behaviours and visual styles, creating a fragmented user experience across products
- Limited reusability: Components built for specific use cases lacked the flexibility needed for broader application, reducing their value to other teams
- Maintenance complexity: As the library grew without proper governance, the accumulation of one-off components increased maintenance overhead without proportional value
- Weak collaboration between design and development: Design, if present, was typically not involved until the later stages of the more execution-focused approach. This lack of collaboration further emphasized the perpetuation of inconsistencies.

3.2 Transition to a dedicated Design System team

The growing problems with the informal shared library approach became increasingly clear as Posti's digital presence expanded. The absence of an administrative framework led to a scenario where the internal shared library was either flooded with one-off components or completely ignored by teams who needed more dependable alternatives. Quality was inconsistent, and there was no way to ensure components followed accessibility standards or UX best practices.

A turning point came with a key digital transformation initiative, which highlighted the need for change. This initiative showed the critical need for better collaboration between design and development teams across the organization. It was clear to leadership that solving these problems would not work with tweaks made to the approach that was already there, and large changes were needed to how design assets were created, shared, and maintained.

The change from informal sharing to more structured professional design operations happened with the formation of a dedicated Design System team. This team, made up of both developers and designers working full-time on the system, began with several key tasks:

1. Cleaning up the existing shared library by removing redundant or low-quality components
2. Introducing proper version control with managed NPM releases and clear versioning
3. Creating clear contribution guidelines and quality standards
4. Setting up formal review processes to ensure component quality and consistency
5. Developing a management approach that balances central control with team input.

A formal component development process was implemented that poured effort into new components for multiple teams or cross-organizational use. This change was a fundamental shift from focusing on the individual teams to serving the entire organization.

As a result of the team's work, Design System version 1.0.0 was published which marked the transition from merely a shared library to a fully functional design system with well-defined processes, guidelines, and governance in place. This more professional approach laid the groundwork for scaling design efficiently across Posti's growing digital ecosystem.

3.3 Posti Design System today

The Posti Design System has grown from a partial React UI component library into a digital service supporting the entire company's digital infrastructure. Today's system is a strategic asset that connects teams, standardizes processes, and reflects the brand identity across all digital touchpoints. Throughout its life, the system has proven flexible and consistent, successfully handling a complete relaunch of branding while maintaining consistency across digital products.

The Posti Design System includes several connected elements:

- React Component Library: The @posti-web-components project forms the technical core of the system, offering production-ready React components showcased and documented in Storybook. These components implement not only visual styling but also accessibility features, interaction patterns, and responsiveness.

- Documentation Platform: This is stored on Zeroheight, a centralized documentation serves as the single source of truth across the organization detailing guidelines, best practices, and implementation details in an accessible format.

- UI Kit: Built and maintained in Figma, the UI Kit gives designers ready-to-use components and templates that match the coded implementation.

Storybook is an open-source frontend workshop tool that allows developers to build, test, and document UI components in isolation. It provides a dedicated environment where components can be crafted and refined independently from the main application, making it easier to develop and showcase hard-to-reach states and edge cases.

Storybook automatically works with various JavaScript supporting frontends such as React, Angular, and Vue. Its automation features include component and visual testing, which prevents test case redundancy, automated documentation data management where all components are accessible by team members, boosting teamwork, and components regaining trust during capture-the-flag sessions. Storybook works together with design systems and serves as a UI development workshop, enabling testing of components pre-implementation[20]

Zeroheight is a specialized documentation platform for design systems that bridges the gap between designers and developers by creating a centralized hub for design assets, code components, and guidelines. The platform enables design system teams to maintain comprehensive documentation that remains synchronized with the latest component implementations [13].

From a developer perspective, the Design System provides significant value by reducing duplicate work related to products within different teams. Instead of rewriting the same UI components of different quality, teams can utilize already built, tested components that are accessible (meet WCAG 2.2 level A & AA) and follow industry standards. This allows development teams to focus on creating unique value in their products rather than focusing on basic repetitive UI problems.

Despite having a dedicated team, the Design System maintains an open contribution model that encourages participation from across the organization. Recognizing that product teams often develop components that could benefit others, the Design System team welcomes contributions through a structured process:

1. Opening a ticket in the Design System's Jira board
2. Submitting a pull request to the repository
3. Undergoing code review by the Design System team
4. Testing the component for quality and compatibility
5. Documentation and release

This collaborative approach ensures that wider organizational problems are still addressed alongside quality standards. As will be demonstrated through a practical case study in Chapter 4, this model enables developers to create components that fulfil cross-team requirements and needs, even if the final implementations evolve significantly from the original contributions.

The system has also bridged the gap between design and development by creating a common language and shared assets. Designers work with the same component models that developers implement, reducing errors and inconsistencies. This collaborative approach brings together designers, developers, product owners, and business stakeholders around a shared vision for the user experience.

Posti today recognizes the Design System as an enabler of business speed through faster feature deployment, operational efficiency by reducing duplicate work, consistent brand expression across digital channels, and universal experience accessibility. The evolution from a basic shared library to today's comprehensive design system reflects Posti's commitment to treating design not as just a visual layer, but as a strategic business capability that improves both customer experience and the organization's ability to deliver digital products efficiently at scale.

4 Adding a component to Posti Design System

The process of adding a new component to a design system begins with designs in tools like Figma, followed by implementation using consistent design tokens to ensure visual coherence. The team must carefully evaluate which components to develop, as each addition requires resources for development and maintenance. Components that meet broad demands across several teams are usually prioritized in order to maximize return on investment.

This chapter presents a real-world case study of the component development process in a large organization, illustrating the journey from need identification through implementation and refinement.

4.1 Defining the need for component

During the rebranding project for the Customer Care feature in one of Posti's digital services, a visual element illustrated in figure 2 was introduced to indicate the availability status of live chat support. The design, created by a team of designer, was adapted from another Posti product and integrated into the new user interface without standardization.

While implementing this feature, the author recognized that similar patterns existed in multiple applications across the organization, yet no standardized component was available in the official Posti Design System. This observation highlighted an opportunity to improve design consistency and reduce redundant development work through the creation of a shared component.

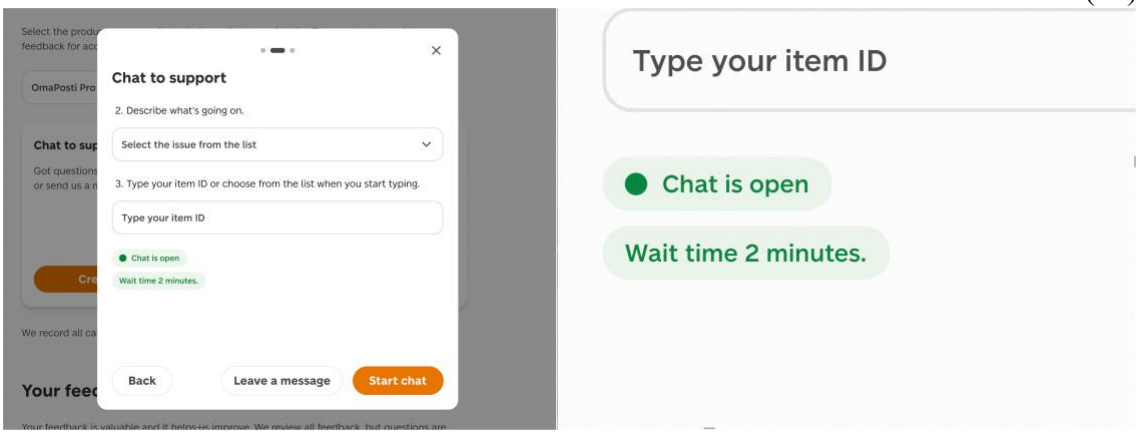

Figure 2. Design of the initial PillBadge component.

When introducing the standardization possibility to the Design System team, the author suggested the working name "PillBadge" for this component, referring to its pill-shaped appearance.

Function and design requirements

The component's main purpose is to give visual feedback on the state of service availability in real-time. In the context of the Customer Care feature, it communicates chat support availability through an intuitive color-coding system:

- Green indicates open chat services
- Red showing busy status with an estimated wait time
- Gray represents closed or unavailable services.

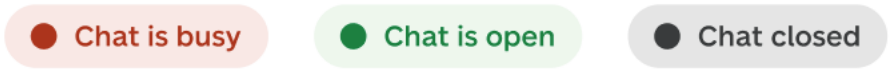

Figure 3. PillBadge component states.

This design approach increases transparency and improves user experience by establishing clear expectations for service availability. The component's visual presentation provides high visibility while maintaining design harmony with other interface elements.

To demonstrate the component's potential, the author developed a baseline implementation including Storybook documentation and initial test coverage. This implementation was reviewed by a mentoring lead developer from the author's team before submitting a pull request to the Design System repository, ensuring a higher quality contribution.

The submitted work served as a catalyst for further cross-team discussion and refinement. Following the contribution, the Design System team collaborated with other product teams to evaluate how this component might serve their needs as well. After evaluating it with various teams, the part was reworked and implemented as ‘StatusIndicator’ to fit with the scope and naming conventions of the organization. Though replaced by a more robust solution, the author's initial version identified a significant cross-team requirement that formed the foundation for a standardized component now widely used throughout the organization

4.2 Design considerations and accessibility

Digital accessibility means designing websites, apps, and interfaces so everyone, including people with disabilities, can use them. This is not only about meeting legal requirements but also about providing a consistent, positive experience for all users. For example, keyboard-friendly navigation not only helps users with motor impairments but also benefits power users.

Additionally, prioritizing UX accessibility is a critical strategy for innovative web design. Because it guarantees that all users, regardless of their cognitive or physical abilities, can effectively interact with digital content. By making UX accessibility a top priority, organizations can create a more inclusive user environment while positioning themselves as leader in ethical a customer-focused business practices. With increasing compliance and brand perception, this strategy not only improves user experience but also fortifies market position [4].

By following accessibility guidelines, products become not only accessible but also more efficient.

4.2.1 Accessibility Requirements (WCAG Standards)

As web technologies expanded and diversified in response to the need, the clear guidelines for providing improved accessibility became apparent. The Web Content Accessibility Guidelines (WCAG) were subsequently established by the World Wide Web Consortium (W3C) under the direction of the internet inventor Tim Berners-Lee. WCAG defines three levels of compliance: A (minimum), AA (commonly required by law), and AAA (most stringent) [15].

The guidelines are structured around four key principles, encapsulated by the acronym **POUR**:

Perceivable - Users must be able to perceive the content.
Operable - User interface controls must be operable.
Understandable - Both content and controls must be understandable.

Robust - Content must be robust enough to be interpreted reliably by various user agents [15].

Figure 4. The structure of WCAG [5].

The World Content Accessibility Guidelines (WCAG) were created to enhance web accessibility for individuals with disabilities. WCAG 2.2 was officially published in October 2023, representing the latest stable version of the guidelines [14]. The W3C has indicated that there are no plans for a WCAG 2.3 release, with development efforts now focused on WCAG 3.0 (codenamed 'Silver'), which is currently in draft form as of late 2024 [6], which has been revised over time to consider developments in assistive technology, web-based digital technology, and design and development trends. WCAG is utilized worldwide and was designed for content creators, developers, and anyone who wants to learn how to make digital experiences accessible.

4.2.2 Accessibility on the legislative level

Posti is operated by the Finnish State through the decision-making power of the shareholder. The State maintains 100 percent direct ownership of Posti Group Corporation. The government's role as owner, however, entails responsibility in meeting societal needs and ensuring postal services remain accessible to all citizens, regardless of market competition dynamics. This responsibility extends to Posti's digital services and products.

The fact that Posti operates under state ownership significantly influences its accessibility standards and commitments to public service. Due to this ownership structure, Posti must comply with legislative requirements aimed at

guaranteeing accessibility for people with disabilities. As a state-owned company, it is legally obligated to follow accessibility regulations by the Act on the Provision of Digital Services (306/2019) [7], which implements key technical standards. This includes the EU standard EN 301 549 for websites, mobile applications, and self-service kiosks, which currently references the Web Content Accessibility Guidelines (WCAG) 2.1 at conformance levels A and AA. With WCAG 2.2's publication in October 2023, regulatory standards are expected to update their references to incorporate these enhanced accessibility requirements in the near future [8]. These legislative requirements set mandatory accessibility rules for Posti's digital presence and service delivery.

Furthermore, the upcoming European Accessibility Act (EAA), which goes into effect on June 28, 2025, will expand accessibility standards beyond government-owned organizations to include private companies across a variety of sectors. This legislation will improve the accessibility of products and services throughout the European Union (EU), creating a more consistent approach to digital inclusion.

The WCAG 3.0 development marks a new step forward in digital accessibility standards, which continues to evolve. This future framework will likely use a more rounded approach that is centered on ‘outcomes’ when assessing accessibility. Systems designed with an accessibility compliance baseline grounded in WCAG 2.2 standards will be more adaptable to these upcoming shifts [9].

4.3 Development process

Posti Design System, also referred to as the @posti-web-components library, is constructed using a modern front-end technology stack that emphasizes maintainability, performance, and developer experience.

4.3.1 Technology stack

Core Technologies:

- React (v18+) serves as the primary UI library
- TypeScript provides type safety and enhances the developer experience
- Styled Components (v6.x) enables component styling and theming
- React Aria implements accessibility-first component patterns

There are several benefits to using TypeScript and React when implementing a project, especially one that can be used as a library. TypeScript is highly effective for collaborative development. Its static typing reduces runtime errors, while clear interfaces improve code readability and documentation [10].
The strong typing system enables developers to catch type-related errors at compile time, rather than at runtime, significantly improving code quality and reliability. TypeScript enhances development productivity through superior IDE support, providing developers with robust autocompletion, inline documentation,

and real-time error checking. This capability allows team members to code with greater efficiency and accuracy.
TypeScript simplifies the refactoring process by ensuring that all necessary changes propagate throughout the codebase. This feature allows developers to modify and improve the code without fear of introducing errors. As projects grow, TypeScript maintains code quality, enabling developers to manage larger and more complex codebases efficiently. This scaling ability, paired with TypeScript's predictability, translates into reduced maintenance costs and effort over time [11].

Additionally, React's virtual DOM optimization contributes to performance efficiency, particularly important when components are used in several applications at once. By leveraging TypeScript in conjunction with React, teams can build more maintainable, scalable, and robust design systems that stand up to the demands of modern web applications.

Build Toolchain:

- Webpack (v5) handles bundling for multiple output formats (CommonJS, UMD)
- Babel manages JavaScript transpilation
- TypeScript compiler performs type checking and generates declaration files
- npm/yarn facilitates package management

The project utilizes the benefits of a thorough build toolchain that ensures compatibility and peak performance. Webpack (v5) handles the sophisticated bundling process, supporting multiple output formats including CommonJS and UMD to maximize compatibility with various consumption patterns. Babel efficiently manages JavaScript transpilation, ensuring code compatibility across diverse browser environments.

The TypeScript compiler plays a dual role by performing rigorous type-checking during development while also generating declaration files that enhance the library's consumption experience. Package management is streamlined through npm/yarn, facilitating dependency handling and versioning control.

This technology stack, which is in line with the main front-end development technologies used across Posti's digital product portfolio, guarantees compatibility across several internal products.

4.3.2 Testing and deployment

After component implementation, a pull request must be approved by at least one core developer (maintainer) on the Design System team. Before being deployed, the component is thoroughly tested. (See figures 5 and 6)

Unit Testing

- Jest serves as the primary test runner for executing test suites
- React Testing Library enables component testing through user-centric scenarios
- Tests specifically focus on functionality and accessibility compliance
- Coverage extends to interaction testing and state management validation

Visual Testing

- Storybook provides an environment for visual development and testing
- Chromatic performs visual regression testing and facilitates UI review
- Snapshot tests automatically detect unexpected UI changes

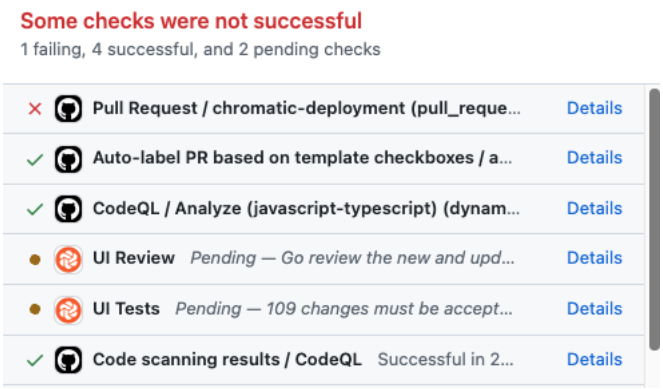

Figure 5. GitHub actions checks failed

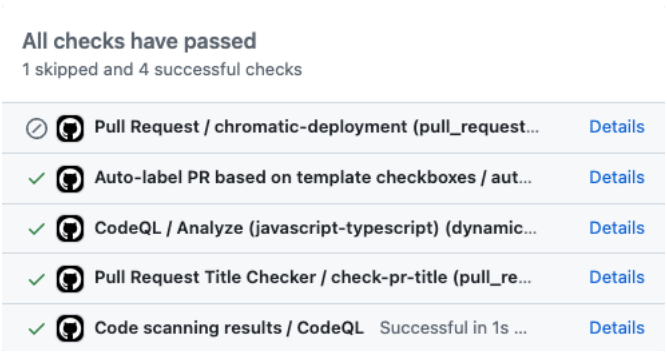

Figure 6. GitHub action check passed

Visual tests, also called visual regression tests, catch bugs in UI appearance. They work by taking screenshots of every story and comparing them commit-to-commit to identify changes. Chromatic is a visual testing service made by Storybook maintainers. Besides gathering UI feedback and visual testing, Chromatic publishes a secure Storybook to review. Therefore, a designer, tester, or product owner can review the changes in the feature branch before it is merged [12].

Deployment and Release Process

Following the required changes and pull request creation, an additional status check called UI Review takes place. If designer input on UI changes is required, the task can be assigned accordingly. Otherwise, reviewers approve UI changes in Chromatic.

The continuous integration process executes automated testing and code quality checks on all pull requests via GitHub Actions. Release automation is managed by Semantic Release (Figure 7), which handles versioning while automated processes perform changelog generation and package publication. In order to guarantee consistent updates with explicit guidelines for version increments, the project uses semantic versioning (MAJOR.MINOR.PATCH).

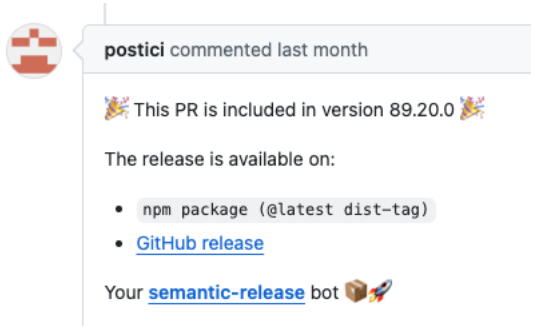

Figure 7. Semantic-release bot message in PR comments

This comprehensive approach ensures all components maintain high quality while allowing for efficient cooperation among developers, designers, and stakeholders throughout the development process.

4.3.3 Definition of done

The Design System team follows a comprehensive definition of done that includes three primary phases: design, development, and documentation. This rigorous approach ensures all components meet quality standards before release.

Design Phase

Components must be designed or redesigned according to the latest brand guidelines. The design process includes:

- UX research or at minimum a validation assessment to determine current usage patterns and necessity
- Component audit to evaluate existing implementations

- Accessibility compliance checks against WCAG 2.2 level AA standards.

Development Phase

The development phase requires component implementation and publication in the Design System code repository. Completed components must:

- Be available for viewing and testing within the Storybook UI component library
- Be accessible to developers as React components via the GitHub repository
- Pass all accessibility tests, including screen reader compatibility and cross-browser functionality.

Documentation Phase

All components require thorough documentation in the Design System documentation site Zeroheight. The documentation must include guidelines covering:

- Overview and principles explaining component purpose and function
- Design and anatomy details outlining visual elements and structure
- Links to relevant additional resources
- Usage guidance indicating appropriate and inappropriate implementation contexts
- Accessibility considerations that address inclusive design principles

This structured approach ensures consistency, quality, and accessibility across the Design System, facilitating effective implementation by product teams while maintaining brand integrity.

4.4 Implementing the component

Posti Design System maintains comprehensive guidelines and conventions for contributing to the react-components package. These established patterns—covering CSS implementation, React component structure, TypeScript usage, and pull request processes—ensure consistency, maintainability, and clarity across both internal development and external contributions.

When adding a new component to the Posti Design System, the process begins with clear requirements derived from both design specifications and functional needs. The StatusIndicator component (Figure 8) serves as an excellent case study of this process.

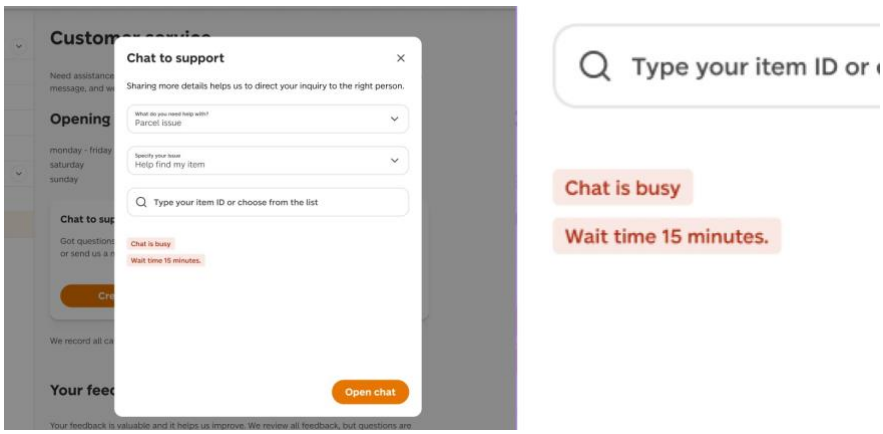

Figure 8. Implementation of the final StatusIndicator component from the DS team

4.4.3 Component structure and API design

The StatusIndicator component demonstrates how requirements are translated into a concrete implementation:

- Visual variants: alert, error, info, neutral, ok, success, warning
- Optional icon support
- Responsive text wrapping
- Semantic color coding for different statuses

As shown in Listing 1, the StatusIndicator demonstrates the design system's approach to component API design.

```
export const statusIndicatorTypes = ['alert', 'error', 'info',
'neutral', 'ok', 'success', 'warning'] as const
export type StatusIndicatorType = (typeof
statusIndicatorTypes)[number]

export interface StatusIndicatorProps {
  icon?: ComponentType<DefaultIconProps>
  label: string
  type: StatusIndicatorType
}
```

Listing 1. StatusIndicator API example.

The interface design follows Posti's stated guidelines for generating consistent, developer-friendly component APIs. By exposing only the necessary properties - icon, label, and type - the component stays straightforward to use while giving enough flexibility for a variety of scenarios.

TypeScript's union types are used to enforce type safety by limiting the type prop to a specified set of valid values corresponding to the component's visual variants. This approach avoids runtime issues by detecting incorrect usage during development.

The component's API demonstrates effective flexibility by providing optional parameters, such as icon, which allow developers to use the component with or without accompanying visual components. Each prop is properly titled to describe its function, making the component's usage intuitive even for developers who have never worked with it previously.

4.4.4 Implementation pattern

The StatusIndicator follows a consistent implementation pattern used throughout the design system:

- Component Declaration: Clean functional component with destructured props
- Markup Structure: Semantic HTML with accessibility considerations
- Styled Components: Separation of styling from logic
- Theme Integration: Direct usage of design tokens

```
export function StatusIndicator({ icon: Icon, label, type }:
StatusIndicatorProps) {
  return (
    <StatusIndicatorWrapper $type={type}>
      {Icon && (
        <StatusIndicatorIconWrapper>
          <Icon aria-hidden="true" width="12" height="12"
color="currentColor" />
        </StatusIndicatorIconWrapper>
      )}
      {label}
    </StatusIndicatorWrapper>
  )
}
```

Listing 2. Component usage implementation

4.4.5 Styling approach

The StatusIndicator's implementation reveals how styling and system integration work together in the Posti Design System. The component employs styled-components to maintain a clear separation between business logic and visual presentation, while simultaneously leveraging the design system's established tokens and patterns.

The StatusIndicator demonstrates several key styling principles used throughout the Posti Design System:

- Dynamic styling based on component state through custom properties
- Consistent application of design system theme tokens for colors and spacing
- Flexible layout that accommodates various content lengths and optional elements
- Visual variants are mapped directly to established status color schemes.

Accessibility and integration considerations include:

- Appropriate ARIA attributes ensuring screen reader compatibility
- Seamless integration with the icon component library
- Direct consumption of design tokens maintaining visual consistency
- Support for different status types while preserving a cohesive appearance.

This integrated approach to styling and system connectivity ensures that the StatusIndicator maintains visual consistency across all Posti digital products while remaining flexible enough to accommodate specific implementation requirements. By adhering to established design patterns and leveraging shared resources, the component contributes to a cohesive user experience that reinforces the Posti brand identity.

5 Challenges and improvements

5.1 Challenges in maintaining Design System

The primary challenge Posti Design System team has faced stems from the initial approach of reactively responding to every component request from product teams. This approach, while intended to help all teams, created several important problems.

The small team's resources became limited as requests came in faster than they could be properly handled. This created a difficult choice - either delay some requests (potentially slowing down product teams) or implement components with limited opportunity for thorough research.

Over time, many components were developed based on requests from individual teams without broader validation. As a result, the component library grew to include components with limited usage.

This situation is similar to what happened in the pre-design system phase, when there was a shared component library – teams sometimes lost interest in the official components and created their own custom versions. The difference now

is that these custom implementations exist alongside official components that aren't widely used.

The reality discovered is that it's not feasible to prevent teams from developing custom components by simply trying to fulfill every request. This approach is difficult to maintain and doesn't fully support the design system's goals. The most effective role for the design system is not to try to prevent custom components, but to identify patterns that naturally develop across teams and create standardized solutions.

5.2 Improvements

The Design System, like any product with active users, needs ongoing improvements. The team is implementing several significant changes to address these challenges:

Version 90.0.0 will be the last release under the current approach. Moving forward, the company is switching to a 6-month release cycle. This form of system will help in planning, researching, and testing, while providing product teams with a predictable cadence for updates. Every release will come with a detailed set of deliverables that includes a roadmap of new features, , and scheduled breaking changes which will aid in the work planning for the teams to ensure seamless workflow.

Additionally, in terms of technical architecture improvements, the design system is transitioning from styled-components to CSS modules. This is in part due to styled-components entering maintenance mode in April 2025 [16], but also has advantages in bundle size and performance. This shift creates possibilities to:

- Implement better bundle optimization through tree-shaking
- Review and potentially consolidate existing components
- Update the styling approach for easier maintenance
- Build a more sustainable foundation for future development.

The strategy toward component development is being changed from multiple directions at once. The team moves beyond fulfilling individual requests and starts to identify patterns across multiple products before new components are created. New potential components are evaluated using a framework that considers evidence of need across multiple products or teams, fit with the overall design language, maintenance considerations, and accessibility requirements[17]. To support this approach, communication channels have been redefined for release information, bug reports, and general communication. Monthly information sessions and sync meetings with all product teams help identify common patterns, gather feedback on existing components, and communicate

upcoming changes. This reflects a broader industry practice of fostering collaboration to surface recurring needs and prioritize solutions that benefit multiple teams [18]. For all approved components, the team is now actively collaborating with other stakeholders across multiple teams to ensure the proposed solution is optimal across different products.

6 Conclusion

This thesis has explored the implementation, challenges, and ongoing improvements of Posti Design System, offering insights into how design systems function within large organizations with diverse digital products. The work followed traces the journey from the implementation to the strategical refinement phase development.

The design system has successfully established a foundation for consistent UX across Posti's digital landscape, while also revealing important lessons about sustainability, component adoption, and cross-team collaboration. Key challenges identified include the difficulties in maintaining a reactive component development approach and ensuring consistent adoption across teams.

The experience at Posti demonstrates that design systems are not static products but evolving ecosystems that require continuous adaptation, clear governance, and strong collaborative practices. The most valuable role of a design system is not necessarily to prevent all custom implementations but to identify and standardize patterns that naturally emerge across teams.

Those changes will apply the components to align better with organizational goals and purpose, and particular practices and standards common for the industry. Focused on technical component standardization and integrated release rationalization, agile release tactics, and strategic component building.

The author's perspective evolved significantly throughout the research process. The transition from being a consumer of the design system to becoming a member of the team maintaining it provided a comprehensive view of the ecosystem from both sides. This dual role directly informed the proposed improvements, particularly around establishing stronger communication channels and ensuring new components address verified needs across multiple teams rather than isolated requests. This perspective revealed that successful design system improvements stem from understanding how the system is actually used in practice rather than how it is theoretically intended to function.

Moreover, the importance of starting with a clear understanding of user needs, pain points, and organizational goals before initiating technical solutions was highlighted during the research process. Prematurely focusing on components or tokens without a broader contextual understanding can lead to solving the wrong problems [19].

The opportunity to experience both the consumer and maintainer roles within the design system allowed for a more comprehensive understanding of the challenges faced by both sides. It highlighted the critical importance of communication, early needs analysis, and collaborative trust frameworks that secure the sustained use and acceptance of a design system. As was made clear during the investigation, creating a design system involves much more than building UI components, it is also about trust and collaboration across teams in order to develop system and user engagement [19].

In Posti's case, for organizations having their own design systems to implement or enhance, it is important to center around demonstrated needs instead of theoretical comprehensiveness, clear evaluation criteria for new components should be set, and collaborative processes that foster trust and enhance system acceptance should be prioritized across teams.

References

1. What are Design Systems? n.d. The Interaction Design Foundation. <https://www.interaction-design.org/literature/topics/design-systems> Accessed 24.4.2025.

2. UXPin 2025. 13 Best Design System Examples in 2025. Studio by UXPin. <https://www.uxpin.com/studio/blog/best-design-system-examples/> Accessed 12.4.2025.
3. What Is a Design System: An Introductory Guide for 2025. 2024. <https://uithings.com/blog/what-is-a-design-system/> Accessed 10.4.2025.

4. Accessibility Spark. n.d. Why UX Accessibility Should Be Your Top Priority in Web Design. Online material. https://accessibilityspark.com/ux-accessibility/ Accessed 15.3.2025.

5. Kalinkina, Iuliia. n.d. Creation of Web Accessibility Guidelines for an organization and analyzing their impact on the workflow. Master's Thesis 2024. <https://www.theseus.fi/bitstream/handle/10024/864474/Kalinkina_Iuliia.pdf?sequence=2&isAllowed=y>. Accessed 17.3.2025

6. Firth, A. 2024. Practical Web Accessibility: A Comprehensive Guide to Digital Inclusion. Online material. https://learning.oreilly.com/library/view/practical-web-accessibility/9798868801525/ Accessed 15.3.2025.

7. 306/2019 | Suomen säädöskokoelma | Finlex. n.d. <https://www.finlex.fi/fi/lainsaadanto/saadoskokoelma/2019/306> Accessed 5.4.2025.

8. Accessibility of Posti services. n.d. <https://www.posti.fi/en/customer-service/terms-and-statements/accessibility#accessibility-and-posti> Accessed 5.4.2025.

9. Sapega, Marissa 2020a. The History of Digital Accessibility and Why It Matters. TPGi. <https://www.tpgi.com/the-history-of-digital-accessibility-and-why-it-matters/> Accessed 2.4.2025.

10. TypeScript Documentation - The Basics. n.d. <https://www.typescriptlang.org/docs/handbook/2/basic-types.html#static-type-checking> Accessed 12.4.2025.

11. Dave, Chirag 2024. 10 Compelling Reasons to Use TypeScript with React in 2024. Medium. <https://medium.com/@chirag.dave/10-compelling-reasons-to-use-typescript-with-react-in-2024-cc43a41d97ca> Accessed 12.4.2025.

12. Posti Design System Development Guidelines. Internal document. Posti Group Oyj. Accessed 9.4.2025.

13. Mahe, Jules 2023. Which documentation tool for your design system? zeroheight. <https://wordpress-staging.zeroheight.com/blog/which-documentation-tool-for-your-design-system/> Accessed 12.4.2025.

14. W3C, 2024. Web Content Accessibility Guidelines (WCAG) 2.2: W3C Recommendation 12 December 2024. Available at: https://www.w3.org/TR/WCAG22/ Accessed 15.3.2025

15. Initiative (WAI), W3C Web Accessibility n.d. WCAG 2 FAQ. Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/standards-guidelines/wcag/faq/> Accessed 27.4.2025.

16. Stoiber, Max [@mxstbr] 2025. styled-components is entering maintenance mode. Tweet. Twitter. <https://x.com/mxstbr/status/1908201327811059926>. Accessed 24.4.2025.

17. Engineering, Agoda 2024. How We Prioritize Requests in Our Design System. Agoda Engineering & Design. <https://medium.com/agoda-engineering/how-we-prioritize-requests-in-our-design-system-c41c2dc02590> Accessed 22.4.2025.

18. Le, Nadia 2020. The role of research in Design Systems. Medium. <https://medium.com/@nadiahle/the-role-of-research-in-design-systems-dbe80e0b1618> Accessed 27.4.2025.

19. Design Systems with Daniel Yuschick. 2025. <https://www.youtube.com/watch?v=RslxDgFy3gQ>. Watched 25.4.2025

20. Dakowicz, Jakub 2024. What is Storybook and Why It's Worth Using. Pagepro. <https://pagepro.co/blog/what-is-storybook/> Accessed 1.5.2025.