MIT INSTITUTE OF DESIGN | MIT-ADT UNIVERSITY PUNE, INDIA A Leap towards World Class Education

GRADUATION PROJECT REPORT
Academic Year 2024-25

Reducing friction between designers and developers

Sponsor

frog part of Capgemini Invent

Student

Kumar Satvik

MITU21BDES0104

Graphic Design

Industry Guide | Faculty Guide

Ishnishan Singh | Rohit Keluskar, Rikimi Madhukaillya

MIT INSTITUTE OF DESIGN | MIT-ADT UNIVERSITY PUNE, INDIA

GRADUATION PROJECT 2024-25

Programme: B.Des

The Graduation Project Evaluation Jury has examined the project report titled

Understanding Friction between Designer and Developers

Project Start Date: 16/01/2025 Project End Date: 03/06/2025

Submitted by

Kumar Satvik - MITU21BDES0104

in partial fulfilment for the award of the Bachelor of Design of the MIT Art, Design and Technology University, Pune

In Graphic Design and certifies that it is a record of bonafide work.

JURY MEMBERS

Role	Name	Organisation	Signature
Faculty Guide			
Internal Examiner-1			
Internal Examiner-2			
External Examiner (Industry Mentor)	Ishnishan Singh	frog part Capgemini Invent	

On fulfilling the further requirements by*

(*subsequent remarks regarding fulfilling the requirements)

Examined & Checked by

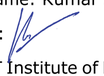
Role	Name	Organisation	Signature
Chairperson of the Jury			
Director:	Prof. Dr. Nachiket Thakur		
Dean:	Prof. Anand Belhe		
Place:	Date:		

Annexure 3 – IPR Declaration

IPR Declaration

I hereby declare that the thesis Understanding Friction between Designers and Developers is a result of my independent work and effort. I certify that to the best of my knowledge, it does not infringe upon any copyrights. Where other sources of information have been used, they have been acknowledged. This thesis has not been submitted anywhere for any other comparable academic degree.

Student name: Kumar Satvik

Signature: 

Place: MIT Institute of Design, Pune

Date: 03/06/2025

Annexure 4 – Faculty Guide NOC

Faculty Guide NOC

It is certified that the work contained in the thesis titled Understanding Friction between Designer and Developers has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Faculty Guide Name: Rohit Keluskar, Rikimi Madhukaillya

Signature:

Place: MIT Institute of Design, Pune

Date: 12/06/2025

Copyright © 2025
Student document publication, meant for
private circulation only. All rights reserved.
Graduate Degree Program in Design, Graphic Design 2020 - 2025
MIT Institute of Design, Pune, India.
This document or any portion of it must not be reproduced or used in any form or by any means whatsoever, electronically or manually, without written permission from the publisher, Kumar Satvik. All illustrations and images used in this document are Copyright 2025 by their respective people organizations.
The typefaces used for the documentation are frog Serif, BentonSansF and Inter.
MIT Institute of Design, Rajbaug, Loni- Kalbhor, Pune-412201.
June 2025

frog™ Part of Capgemini Invent

frog India Studios,
Tower 5, Candor SEZ,
Sector 48, Gurgaon

Annexure 5 – Completion Certificate

Date : 03/06/2025

Completion Certificate

This certifies that Kumar Satvik with Enrollment No. MITU21BDES0104 Program Bachelor's of Design in the Discipline of Graphic Design , has completed his Graduation project with frog part of Capgemini Invent from 16/01/2025 to 03/06/2025 under the guidance of Ishnishan Singh (frog part of Capgemini Invent).

As a part of his project study, he worked on Understanding the Friction between Designer and Developers and has completed the same.

It is our pleasure to congratulate Kumar Satvik on his accomplishments and extend our best wishes for his future endeavors.

Ishnishan Singh (Interaction Designer II)
frog part of Capgemini Invent

Acknowledgment

I extend my deepest gratitude to my parents, Mummy and Papa, for their unwavering love, endless support, and constant encouragement throughout my academic journey, this project would not have been possible without their belief in me.

My sincere thanks go to my academic mentors, Rohit Sir and Rikimi Ma'am, for entrusting me with the freedom to pursue this topic of personal interest and for their invaluable guidance and faith in my abilities.

I am profoundly grateful to my industry mentor, Ishnishan, whose insightful guidance and real-world perspectives were instrumental in shaping the practical application and relevance of this project.

Finally, a heartfelt thank you to the entire frog team, especially to BNP, Amit, Dhruv, Meilin, and all the Interns, for fostering a collaborative and inspiring environment that significantly enriched this project.

Content

frog Part of Capgemini Invent

Hartmut Esslinger frog's Founder Pioneers Emotion-Driven Design Shaping an Era with Iconic Projects

Early frog's Iconic Creations Shaping a Decade

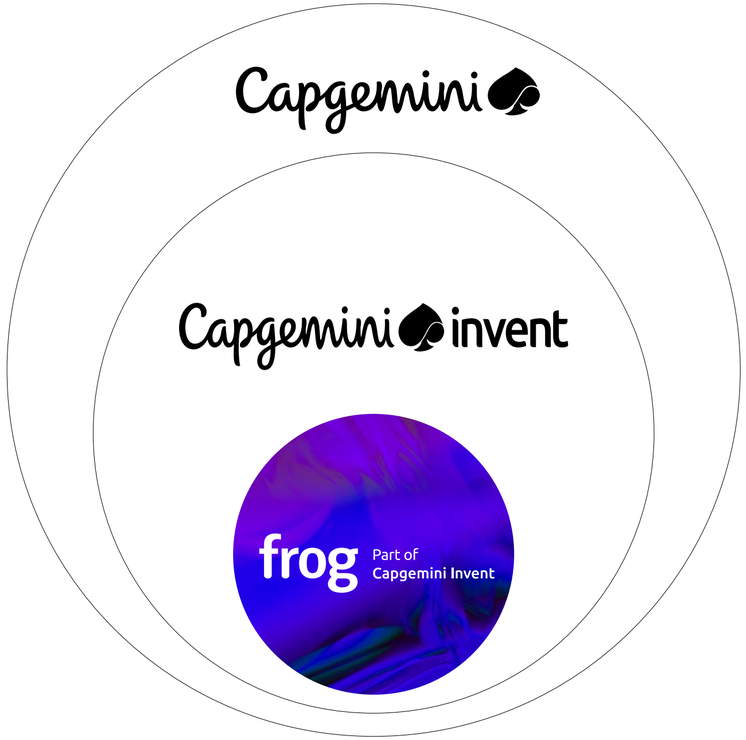
Work with Wega

Wega, a German innovator in high-fidelity audio and video, was known for well-designed consumer electronics long before Sony acquired the company in 1975. As a freelance industrial designer in the 1970s, Hartmut Esslinger began his career collaborating with Wega, marking a significant early chapter in his design journey. Esslinger's designs for Wega high-end stereo systems were not only functional but also highly desirable, helping Wega stand out by making technology emotionally resonant with consumers.

Work with Apple Macintosh

frog's most transformative collaboration began in 1982 with Apple, driven by Steve Jobs' admiration for Esslinger's "Form Follows Emotion" philosophy. The "Snow White" design language, born from this partnership, revolutionized Apple's product aesthetic. Characterized by a clean white palette, soft lines, and the distinctive "linea" element, Snow White debuted with the Apple IIc in 1984 and profoundly influenced the iconic design of the original Macintosh and subsequent Apple products, shaping the visual language of personal computing.

frog is part of the Capgemini network



Capgemini is a global leader in consulting, technology services and digital transformation.

Capgemini scale frog's ability to create social, environmental and economic value by leveraging deep expertise across technology, engineering, operations, supply chain and industry.

A proven end-to-end approach for imagining, making & scaling what's next.

Set the Vision

Understand context and gain empathy
Envision possible futures
Identify insights and opportunities
Ideate concepts to meet unmet needs
Explore and build quick proof of concepts

Shape the Product

Define product and roadmap
Design the user experience
Architect the technical solution
Develop and release the product
Validate business case and experience

Deepen the Impact

Continuous testing and learning with users
Monitor performance and track metrics
Operationalize business and services
Scale platform across ecosystem
Expand product features and business.

Reducing friction between designers and developers

frog Part of Capgemini Invent x MIT INSTITUTE OF DESIGN | MIT-ADT UNIVERSITY PUNE, INDIA

What does an Interaction Designer (IxD) do at frog?

- Understand the goals, business requirements and constraints for the project.
- Help to translate business requirements into user stories.
- Conduct secondary research, including trend benchmarking and competitor analysis.
- Create frameworks such as customer journeys, service blueprints or ecosystem maps based on inputs from research
- Define and prioritize opportunity areas and then translating them into concepts
- Design wireframes, user flow diagrams and application maps
- Create and test interactive prototypes
- Document interaction guidelines
- Prioritize MVP features and build a backlog
- Support the definition of roadmaps in collaboration with Strategy, Design Technologists and Visual Designers at frog

My role was Interaction Design Intern

Guide at frog

Ishnishan Singh

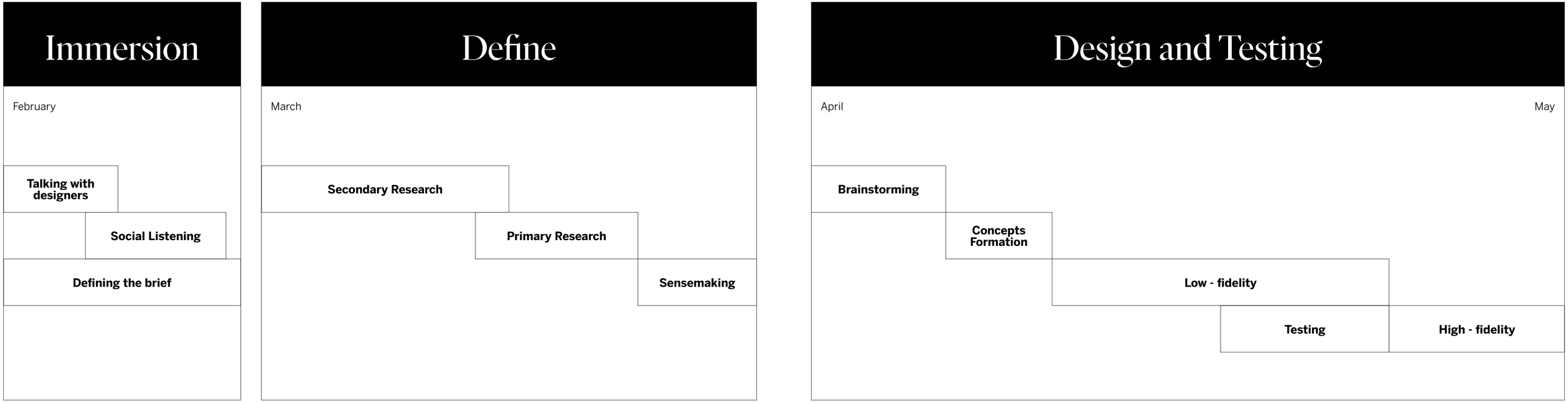
Interaction Designer II

A self-initiated & sponsored graduation project

frog provided the exceptional framework for a self-directed graduation thesis, sponsoring an initiative where the specific area of inquiry was determined by individual focus. Driven by a deep interest in the synergistic potential between design and technology, this project centered on exploring solutions to the persistent friction between designers and developers.

Guided by invaluable mentorship from the studio's experts, a systematic investigation culminated in the conceptualization and prototyping of a Figma plugin aimed at reducing repetitive design tasks and ensuring more complete UI specifications are delivered to developers.

Figuring out Brief


Immersion
February
Talking with designers
Social Listening
Defining the brief
Define
March
Secondary Research
Primary Research
Sensemaking
Design and Testing
April
May
Brainstorming
Concepts Formation
Low - fidelity
Testing
High - fidelity

Timeline

Validating a Recurrent Challenge in Designer-Developer Workflow

Initial observations during a prior summer internship highlighted potential difficulties in the seamless translation of design intent into executed developer output. To determine if these instances represented a broader, systemic issue, focused dialogues were initiated with experienced designers at frog.

These discussions provided critical qualitative insights and firsthand accounts, confirming that friction points within the designer-developer workflow are indeed a recurring and significant concern in product development, warranting deeper investigation.

"Hi my name is Satvik and i am interaction design intern over here.....I am looking for problems that designers faces when working with developer.... kya apne kabhi kuch issues face kiya hai?"

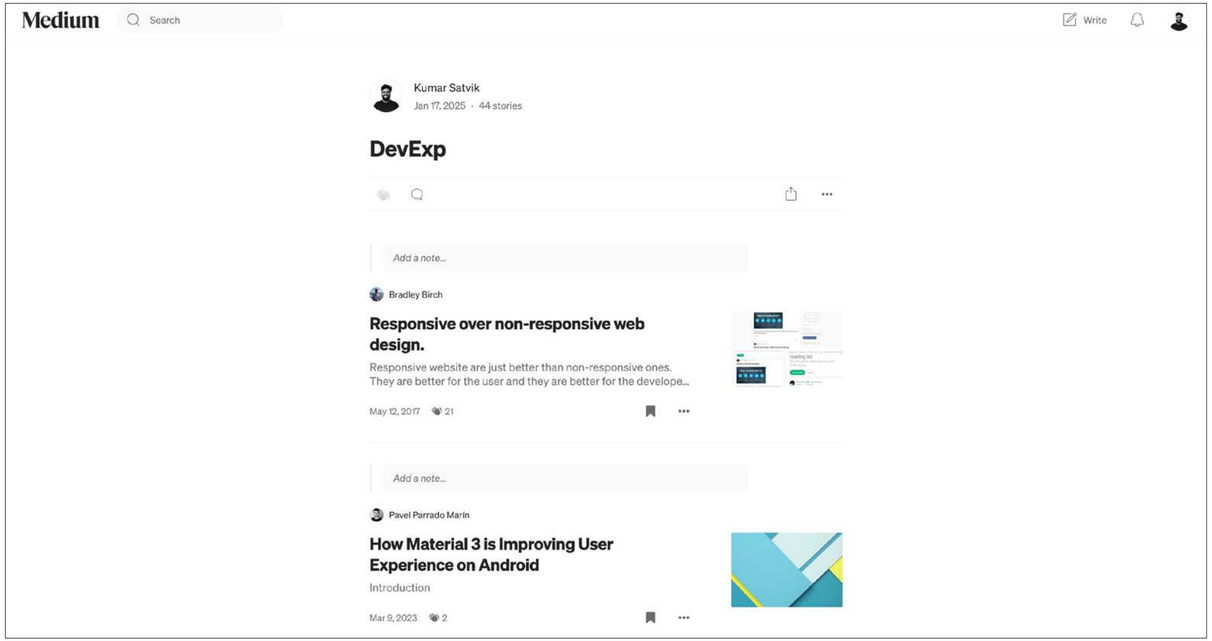
Discussion guide to start the conversation

Informal & Open-Ended

Active Listening & Empathy

Confidentiality & Comfort

Some of the quotes that designer's mentioned on their experience with collaboration with developers in frog Delhi



44 Medium articles to understand Developer Experience

Diving into articles on platforms like Medium, I encountered diverse perspectives

Why Developer Experience is Key to Productivity

Philips highlights SpaceX's 50x productivity, crediting superior Developer Experience (DX)—not just "10x developers." DX, encompassing developer productivity, impact, and satisfaction, stems from great tooling, architecture, and culture, proving a strategic imperative for innovation.

How Internal Developer Platforms Improve Developer Experience

Internal Developer Platforms (IDPs) enhance Developer Experience by automating tasks, improving collaboration, and offering self-service tooling access. IDPs streamline workflows across the software development lifecycle, empowering developers and boosting speed and quality.

Designing a developer portal experience

Designing developer portals requires applying UX principles for better engagement and community. Key considerations include content strategy, intuitive IA, guided user journeys, and community features, making the portal a user-centered experience beyond just documentation.

DX is UX for Developers

“Don’t Make Me Think” for Developers
Developer tools should be intuitive and require minimal cognitive load.

Zero Configuration (Ideal)
Aim for “zero configuration” setups to reduce friction and onboarding time.

Clear & Concise Documentation (Essential)
High-quality, comprehensive, and accessible documentation is non-negotiable. Documentation is the “Single Source of Truth.

Predictable Usage Patterns
APIs and components should have consistent and predictable usage patterns to enhance intuitiveness.

Functional Aesthetic
Prioritize functionality and clarity over purely visual aesthetics in developer UIs. “Functional aesthetic over visual aesthetic”.

Automate Repetitive Tasks
Prioritize functionality and clarity over purely visual aesthetics in developer UIs. “Functional aesthetic over visual aesthetic”.

Automate Repetitive Tasks
IDPs and tools should automate manual, repetitive tasks. capabilities.

Streamline Workflows
Optimize workflows to reduce friction and accelerate development cycles.

Reduce Dependencies on Operations
IDPs reduce developer reliance on operations teams for routine tasks.

Freedom to Choose Tools
Offer developers choices in tools and technologies where appropriate.

Human Interaction (IRL)
Recognize the importance of in-person human interaction alongside digital tools for building developer relationships.

Key Elements & Principles of Good DX

Good Developer Experience isn’t accidental. It’s built on key elements and design principles that prioritize Simplicity, Efficiency, and Developer Empowerment.

Developer Satisfaction Surveys
Use surveys (NPS-style, Likert scales) to directly measure developer satisfaction with tools and workflows.

Usage Metrics & Analytics
Track tool usage, feature adoption, workflow efficiency, and other metrics to quantify DX improvements

Qualitative Feedback (Essential)
Supplement quantitative data with qualitative feedback from developers through interviews, forums, and usability testing to understand the why behind the numbers.

Focus on Developer Productivity, Impact, & Satisfaction
These three elements are core to evaluating DX effectiveness.

Measuring and Evaluating Developer Experience

Measuring Developer Experience (DX) is crucial for understanding its impact and guiding targeted improvements. A combination of quantitative metrics, qualitative feedback, and a focus on key indicators provides a holistic view of DX effectiveness.

Designers as DX Advocates
Designers have a crucial role to play in shaping and improving DX, not just for end-users but also for developers.

Apply UX Principles to Dev Tools
UX methodologies and principles (user research, usability testing, user-centered design) are directly applicable to designing developer tools and platforms.

Empathy for Developers (Essential)
Designers need to develop empathy for developers, understanding workflows, pain points, technical constraints, and motivations.

Clear & Detailed Design Specs
Designers contribute to DX by providing comprehensive, technically sound, and dev-friendly design specs and assets for handoff.

“Editors” not just “Authors”
Design leaders should focus on empowering and enabling their teams, acting as editors and guides rather than sole authors of design solutions

Learn Basic Code (Designer Skill)
Basic coding knowledge enhances designers’ empathy for developers, and enables more technically informed design decisions.

Attend Developer Events & Understand Workflows
Designers should immerse themselves in the developer world

Include developer in design process
Involve developers in the design process to foster team work and collaboration

Designers’ Role in Developer Experience

Designers are essential DX contributors. Through collaboration and technical awareness, designers create developer-centric tools and workflows that boost team value.

Design Systems Enhance DX
Well-designed design systems (component libraries, style guides) are crucial for improving DX by promoting consistency, reusability, and efficiency in development

Atomic Design for DX
Atomic Design principles (atoms, molecules, organisms, templates) can improve DX in design systems by promoting modularity, readability, maintainability, and reusability.

Component Libraries & Style Guides
Design systems should include well-documented component libraries, UI kits, and comprehensive style guides to ensure developer understanding

“Zero Configuration” for Design Systems
Aim for minimal setup friction when adopting and using design systems to improve DX.

Automation in Design Systems
Automating design system build, release, and update workflows (e.g., with Lerna, Travis CI, Figma API) is crucial for maintainability.

Versioning & Update Guides
Design systems need clear versioning and update guides to manage breaking changes and minimize disruption for developers.

Figma for Design Systems
Figma is highlighted as a powerful tool for design systems due to its collaboration features, Styles feature for primitives, and API for automation.

Openness & Transparency
Building design systems in the open, sharing release planning, style guides, and seeking contributions fosters trust and collaboration.

Design Systems & Developer Experience

Design Systems are more than style guides. Well-designed systems directly improve DX by boosting consistency, efficiency, and developer autonomy.

Problem Statement

Poor designer-developer collaboration creates friction and impacts over all project experience. Streamlining workflows and fostering partnership are critical for success.

Plan of action for foundational research

1. Conduct literature review (Secondary Research)
2. Draft research proposal
3. Draft Interview Guide
4. Conduct Interviews (Primary Research)
5. Sensemaking
6. Identifying Opportunity Areas (How Might We...)

Defining Problem

Introduction to Literature Review

Scope
Peer-reviewed articles published 2010-2024, covering Tech, UX Design, and Software Engineering domains.

Database
Consensus App (consensus.app) - AI-powered literature search engine.

Keywords & Search Terms
Developer Experience, Developer Workflow, Design Tooling Integration, Performance Budgeting, A/B Testing Design, Technical Debt Management, Designer-Developer Collaboration

Inclusion Criteria

Peer-reviewed publications

Focus on empirical studies, systematic reviews, and meta-analyses.

Minimum 4-5 Citations

Relevance to DX, TDM, Design, and Developer Workflow.

Exclusion Criteria

Grey literature (blogs, white papers, non-peer reviewed).

Purely theoretical papers without empirical grounding.

Studies outside of Software Engineering & Design domains.

Domains

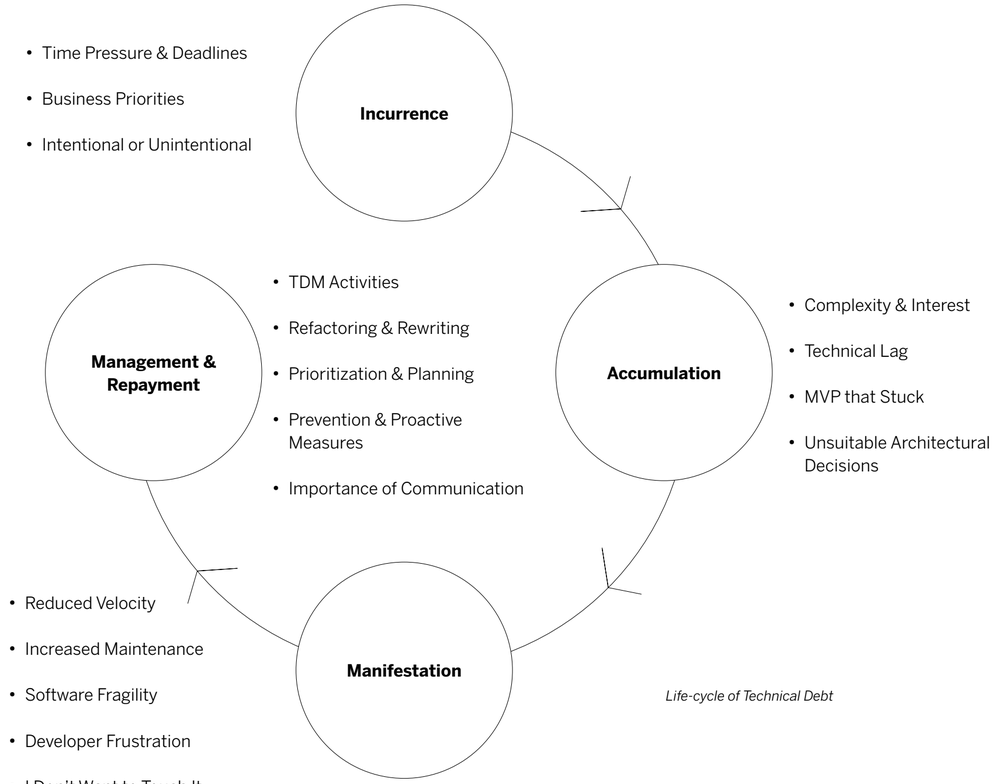
Technical Debt

Performance Budgeting for Design

Design Tooling Integration with Development Workflows

A/B Testing Design Implementations Workflows

Building a Culture of Empathy



Life-cycle of Technical Debt

TD Defined
Technical Debt, coined by Ward Cunningham, is more than just a code problem; it's a powerful metaphor for understanding the trade-offs inherent in software development.

Financial Analogy
Like financial debt, it represents a choice: a shortcut taken for short-term gain, with the understanding that 'interest' will accrue over time if the 'principal' is not repaid.

Fowler's Quadrant
The TD Quadrant (Fowler) helps categorize TD by intent and prudence, revealing that TD isn't always 'bad' – deliberate TD can be a strategic choice.

Risky TD
Often lead to unintended negative consequences and increased 'interest' payments in the form of escalating maintenance costs and reduced development velocity.

TD's Broad Scope
It encompasses architectural debt, design debt, test debt, documentation debt, and even infrastructure debt, highlighting the far-reaching impact

Intentional vs. Unintentional
TD can be intentional or unintentional. Intentional TD is a conscious decision for short-term expediency.

Unintentional TD Causes
Unintentional TD arises from lack of knowledge, evolving requirements, or technical debt accumulation over time, often being more challenging to manage due to its hidden nature.

Impact in Large Systems
TD is particularly impactful in large, long-lived systems, where accumulated ATD can lead to 'crystallized architectures' that are extremely difficult and costly to change.

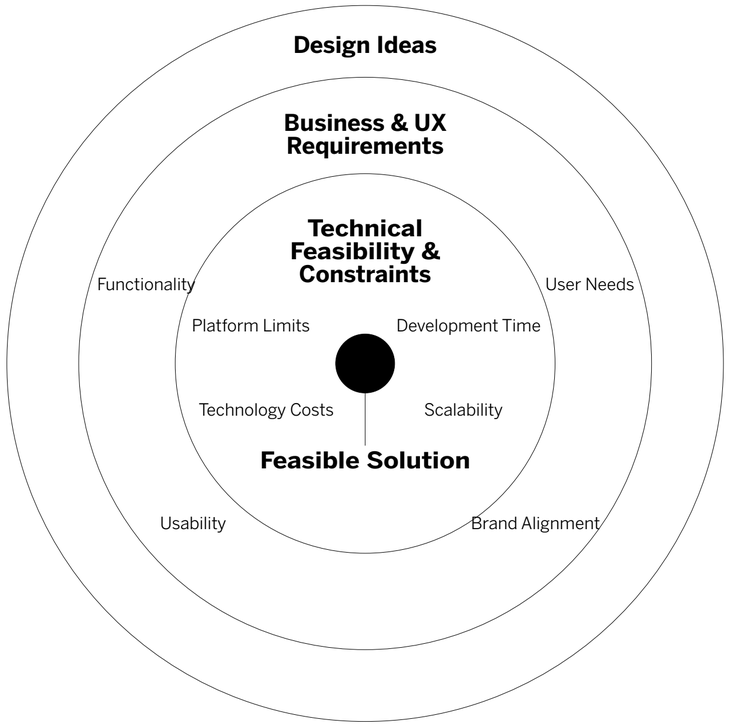
Proactive Management Critical
Understanding and managing ATD proactively is therefore critical for long-term project health and sustainability.

Holistic TDM
Activities across the entire Software Development Lifecycle – from prevention and identification to measurement, prioritization, repayment, documentation, and communication.

Architectural Debt (ATD)
The most insidious and costly form, representing deep-seated structural issues that are harder to detect and remediate, and have long-term consequences for system evolvability.

Beyond Code Fixes
It's not just about 'fixing code' but about managing a complex system.

Technical Debt



Funneling down of design ideas to Feasible solution

Value-Driven Design
Performance Budgeting (PB) is a value-driven approach to design, shifting the focus from purely aesthetic or feature-driven design to performance-conscious design.

Explicit Trade-offs
PB is about making explicit trade-offs between design choices and performance. It's not just about measuring performance after design.

Informed Decisions
PB helps designers and developers make informed decisions by quantifying the performance implications of different design options.

Flexible Framework
A framework for guiding design exploration. It allows for creative design solutions while ensuring that performance remains a key consideration throughout the design process.

Resource Allocation
By setting performance budgets and measuring design performance against those budgets, teams can optimize resource allocation and focus development efforts.

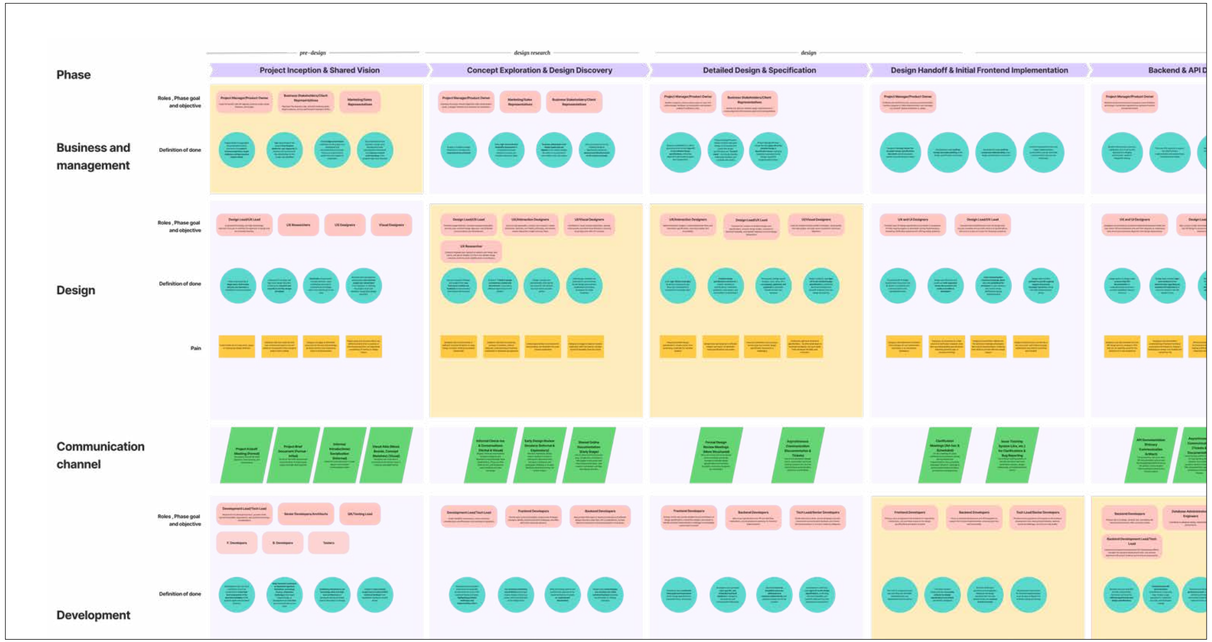
Data-Driven Culture
It encourages designers to use data and performance metrics to inform and validate their design decisions, moving beyond purely subjective or aesthetic considerations.

User-Centric Perf.
By prioritizing performance from the design stage, PB helps create user experiences that are not just visually appealing, but also fast, responsive, and enjoyable to use.

Prevents Bottlenecks
By considering performance early, designers can avoid architectural design decisions that lead to costly performance issues later in development.

Better Collaboration
It provides a shared framework for discussing performance trade-offs and making informed design decisions together, bridging the design-development gap.

Performance Budgeting for Design



Link to Figjam board of service blueprint showcasing project workflow and pain points from secondary research

Manual Handoffs
Time-consuming and error-prone manual handoff processes.

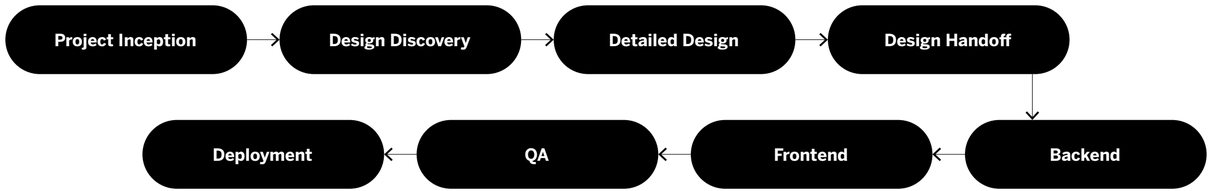
Data Silos
Lack of interoperability leads to data silos, inconsistencies between design and code, and rework.

Discarded Prototypes
Early prototypes created in isolated design tools are often discarded, leading to wasted effort and duplicated work.

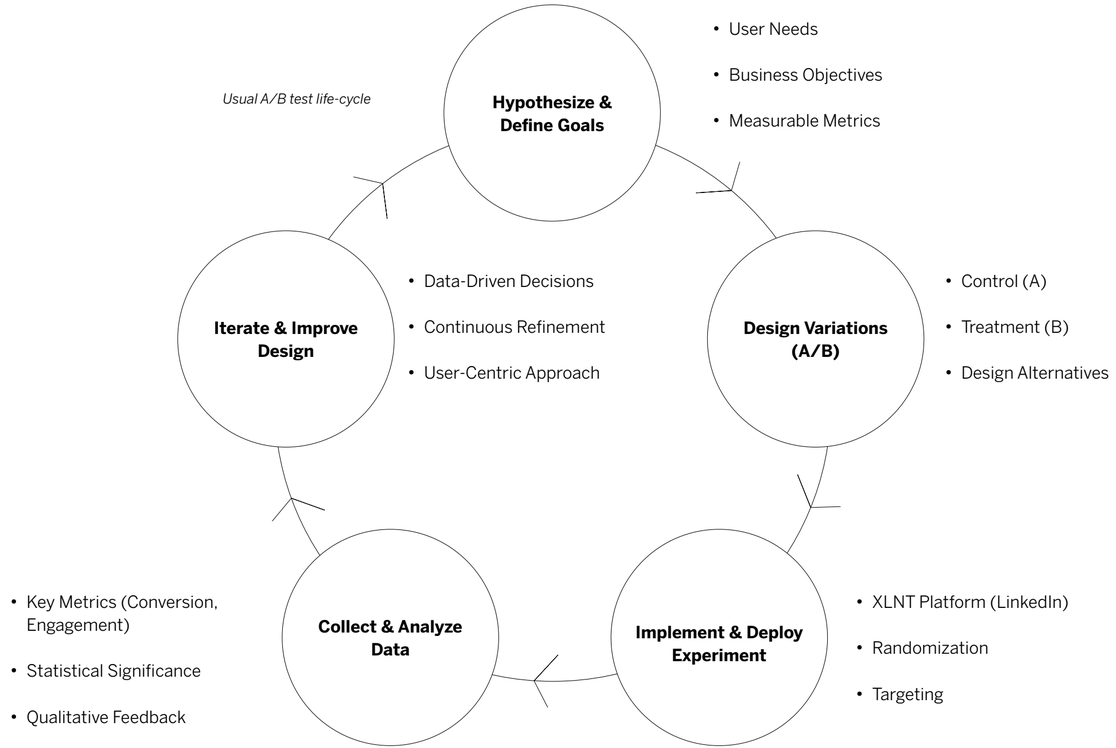
Inefficient Comms
Inefficient communication channels and lack of shared context lead to constant clarification requests and delays during handoff and implementation.

Tech Debt Impact
Workflow friction and miscommunication contribute to technical debt accumulation and can negatively impact software quality and DX.

Fosters Collaboration
Tool integration can foster a more collaborative environment, breaking down silos and promoting shared ownership between designers and developers.



Design Tooling Integration & Workflows



Beyond Assumptions
A/B testing design implementations is more than just validating assumptions; it's a powerful method for data-driven design iteration and continuous DX improvement.

Evidence-Based Validation
A/B testing provides evidence-based validation of design choices, moving beyond hunches or purely subjective opinions.

Measure Behavior
A/B testing allows for direct measurement of user behavior and preferences, enabling data-driven optimization of UX and usability.

Iterative Learning
A/B testing is not a one-time validation exercise, but an iterative process of learning and refinement.

Inform Iterations
Use test results to inform design iterations, refine hypotheses, and continuously improve the user experience.

Early Issue ID
Early A/B testing of design concepts can identify potential usability issues or technical feasibility problems early in the design process, reducing costly rework later in development.

Data-Driven Mindset
A/B testing promotes a data-driven mindset within design and development teams, encouraging a culture of experimentation, learning, and continuous improvement.

Meaningful Variations
Test distinct and meaningful design variations (A vs. B) that directly address the hypothesis, rather than subtle or cosmetic changes.

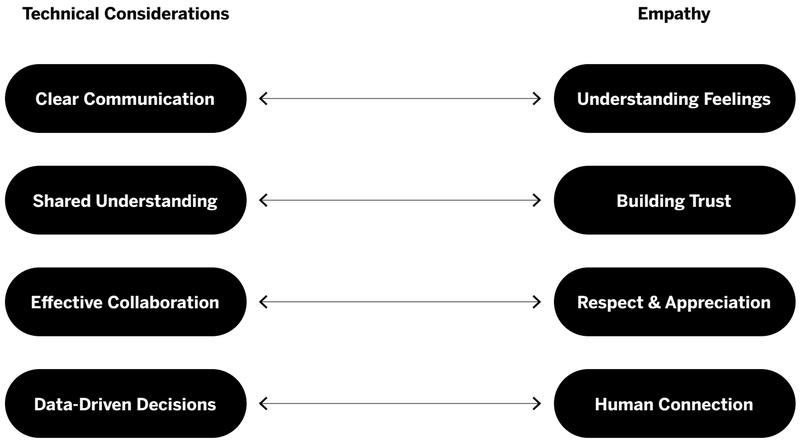
Isolate Variables
Isolate variables for clear results.

Sample & Duration
Pay attention to sample size and test duration to ensure statistically significant and reliable results.

Avoid Premature Conclusions
Avoid drawing conclusions from underpowered tests or premature data analysis.

Supplement with Qual
Supplement quantitative A/B test data with qualitative user feedback to understand the why behind the numbers and gain deeper insights into user behavior and preferences.

A/B Testing Implementations



Human Connection
Building a culture of empathy is about fostering the human connection within development teams and between designers and developers.

Prioritize Dialogue
Prioritize face-to-face communication, open dialogue, and active listening to build understanding and trust.

Foster Purpose
Encourage team building activities, cross-functional collaboration, and shared goals to foster a sense of common purpose and mutual understanding.

Mutual Respect
Cultivate a culture of mutual respect and appreciation between designers and developers, recognizing the value of both design and technical expertise.

Safe Environment
Create a safe and inclusive environment where team members feel comfortable sharing concerns, asking questions, and providing honest feedback without fear of blame or judgment.

Improved Comms
Improved communication, collaboration, and shared ownership fostered by empathy can indirectly lead to more proactive and effective Technical Debt Management.

Aligned Teams
As teams are more aligned and motivated to address quality concerns.

Open Communication
Empathy fosters more open, honest, and effective communication, reducing misunderstandings and streamlining workflows.

Understand Constraints
Empathy enables designers to better understand technical constraints and developer perspectives, leading to more technically feasible and robust design solutions.

Building a Culture of Empathy

Research Proposal

Core Investigation
This research investigates the critical intersection of designer-developer collaboration, Developer Experience (DX), and Technical Debt (TD) management within software development.

Problem Statement
Poor designer-developer collaboration creates friction and impacts over all project experience. Streamlining workflows and fostering partnership are critical for success.

Scope & Applicability
The study aims for broad applicability across diverse software development contexts.

Methodology
- An online survey will be distributed to designers and developers via LinkedIn messages, posts in relevant professional groups, and targeted Reddit advertisements. The survey will use ranking/rating scales and open-ended questions. It will also recruit for interviews.
- In-depth, semi-structured conversational interviews (45-60 minutes) will be conducted with 8-10 designers and developers.
- Interviews will incorporate Journey Mapping to detail workflow phases, actions, tools, pain points, and emotional states.
- Interviews will also use Ecosystem Mapping to illustrate stakeholders, tools, artifacts, and communication dependencies.
- All interviews will be recorded (with consent) and transcribed for detailed thematic analysis.

Overall Contribution
The findings will contribute to a deeper understanding of the socio-technical factors influencing these critical workflows and provide practical guidance for creating more efficient, aligned, and satisfying environments.

Gap
Lacks a direct comparison of designer and developer perspectives on these shared challenges and their impact on both TD and DX. This research seeks to fill that gap by deeply exploring both disciplines.

Hypotheses
- Structured frameworks and intelligent tools for comprehensive design specifications (states, interactions, rationale) will significantly reduce developer ambiguity, decrease rework, and improve implementation alignment.
- Robust systems for a single source of truth (SSoT) and clear design version control, integrated into workflows, will minimize errors from outdated information and enhance developer confidence.
- Integrating earlier, more frequent, and collaborative technical feasibility consultations throughout the design lifecycle will lead to more practical designs and fewer late-stage surprises.
- Providing developers with easy access to the “Why” (user needs, goals) behind designs will improve their technical problem-solving and lead to more value-aligned implementations.
- Fostering a culture of mutual respect, shared understanding, and proactive communication will significantly reduce mindset-driven conflicts and improve collaboration.

Objectives
1. To identify and categorize the most common and impactful pain points, unmet needs (functional and emotional), and workflow inefficiencies experienced by designers and developers.
2. To determine which phases, activities, and communication touchpoints within the designer-developer workflow are perceived as most critical for improvement.
3. To explore and analyze existing collaboration strategies, design/handoff tools, specification practices, and communication channels currently used, and to assess their perceived effectiveness.
4. To deeply understand the differing (and shared) attitudes, beliefs, and expectations of designers and developers regarding their roles and collaborative interactions.
5. To compare and contrast these perspectives to identify key areas of misalignment, miscommunication, and misunderstanding that contribute to friction, rework, and TD.
6. To develop a set of prioritized, actionable recommendations and high-level conceptual solutions for optimizing designer-developer collaboration

Research Questions
- What are the most significant pain points, unmet needs, and sources of friction in the current collaboration workflow?
- How do designers currently create and communicate design specifications, and how effective do developers find these?
- What are the primary challenges related to maintaining a single source of truth and managing design changes effectively?
- To what extent, and how effectively, are developers involved in early technical feasibility discussions, and its impact?
- How is design rationale currently communicated, and how does its availability impact developers?
- What are the key differences and similarities in how designers and developers perceive collaboration challenges and their impact?
- What specific tools, processes, templates, or cultural shifts could most effectively address identified pain points?
- How do factors like team size, organizational structure, remote work, and toolchains influence collaboration?

Introduction

What is your perspective on working with developers?

I am researching common issues designers face when working with developers to improve workflows.

(~ 2 min long)

1 What is your current role?*

2 For how long have you been working in the industry?*

in years

3 Have you ever felt any problem, when working with developers?*

Yes No

further discussion?

7 How can we connect?

Name

Where can we talk? (Phone/WhatsApp number, Linkedin/Insta/Reddit profile, Email)

8 On which day and time, you will be available for a call?

Date

00 : 00 am

Ranking the problems

4 Rank these pain points (1=Biggest Pain) you face when collaborating with developers:*

- Designs Not Implemented
- Slow Iteration/Feedback Cycles
- Lack Dev Input Early Design
- Tech Constraints Limit Design
- Time-Consuming Specs/Handoff
- Vague/Unclear Project Briefs

5 Any story about your developer-designer issues you've experienced OR any interesting solution you have in mind?

6 Will you be also interested in a 30-45 min, discussion on your answers?

Yes No

To sculpt impactful solutions for enhanced designer-developer synergy, this survey seeks to understand designers' direct experiences, workflow nuances, and key priorities for collaborative improvement.

Designer's Survey

2 For how long have you been working in the industry?

2.39	0	9	23
Average	Min	Max	Response count

3 Have you ever felt any problem, when working with developers?

65% (15) Yes

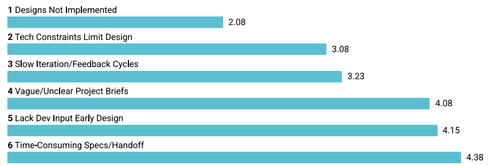
35% (8) No

(23) Responses

Ranking the problems

Enter some body text

4 Rank these pain points (1=Biggest Pain) you face when collaborating with developers:



5 Any story about your developer-designer issues you've experienced OR any interesting solution you have in mind?

Sometimes it gets difficult to communicate the responsive aspects of design. The current developers that I am working with does not make anything responsive by default and it is hard to tell them how to make it responsive. Then there are some common issues like tech constraints and library constraints

I have issues when devs don't code the design in the exact way I have given and give excuses that there are just minute changes no one will notice it.

As a UX Designer, I prioritize collaborative discussions with developers regarding the design system, specifications, and technology. This involves jointly building and maintaining a consistent, accessible design system (using tools like Figma/Storybook), providing detailed specifications for interactions and responsiveness, and collaboratively selecting appropriate technologies like front-end frameworks, while always emphasizing open communication and iterative design to ensure a user-centered and technically feasible product.

-

Can't disclose.

6 Will you be also interested in a 30-45 min, discussion on your answers?

33% (4) Yes

67% (8) No

(12) Responses

Designers pinpoint time-consuming handoffs and lack of early developer input as top frustrations, signaling a clear call for streamlined processes and more integrated, proactive collaboration.

Introduction

What is your perspective on working with designers?

I am researching common issues developers face when working with designers to improve workflows.

(~ 2 mins)

1. What is your current role?*

2. For how long have you been working in the industry?*

in years

3. Have you ever felt any problem, when working with designers?*

Yes No

Ranking the problems

4. Rank these pain points (1=Biggest Pain) you face when collaborating with designers:*

- [ ] Ambiguous Early Requirements
- [ ] Lack Design Rationale (The "Why")
- [ ] Handoff Process is Slow/Inefficient
- [ ] Dev Time Spent on Ambiguities
- [ ] Specs Lack Technical Detail
- [ ] Constant Clarification Needed

5. Any story about your developer-designer issues you've experienced OR any interesting solution you have in mind?

6. Will you be also interested in a 30-45 min, discussion on your answers?

Yes No

further discussion?

7. How can we connect?

Name

Where can we talk? (Phone/WhatsApp number, Linkedin/Insta/Reddit profile, Email)

8. On which day and time, you will be available for a call?

Date

00 : 00 am

This survey aims to capture developers' perspectives on collaborative workflows, seeking insights into current challenges, the value of technical expertise, and priorities for creating seamless design-to-development integration.

2 For how long have you been working in the industry?

4.29	1	20	24
Average	Min	Max	Response count

3 Have you ever felt any problem, when working with designers?

58% (14) Yes

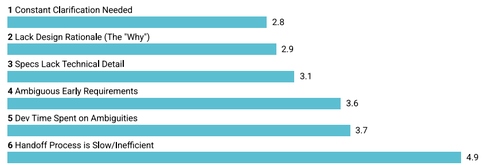
42% (10) No

(24) Responses

Ranking the problems

Enter some body text

4 Rank these pain points (1=Biggest Pain) you face when collaborating with designers:



5 Any story about your developer-designer issues you've experienced OR any interesting solution you have in mind?

Designers that i have worked with were highly experienced and worked on famous products. But one basic thing that they do is that they focus more on design and less on product. They don't gain product knowledge and constantly rely either on developers and product managers

Nope

They just send design docs and as developers we don't know what are the exact requirements and how things will look in end product

Sometimes designers create unrealistic design and expect the developers to copy as it is in limited time

6 Will you be also interested in a 30-45 min, discussion on your answers?

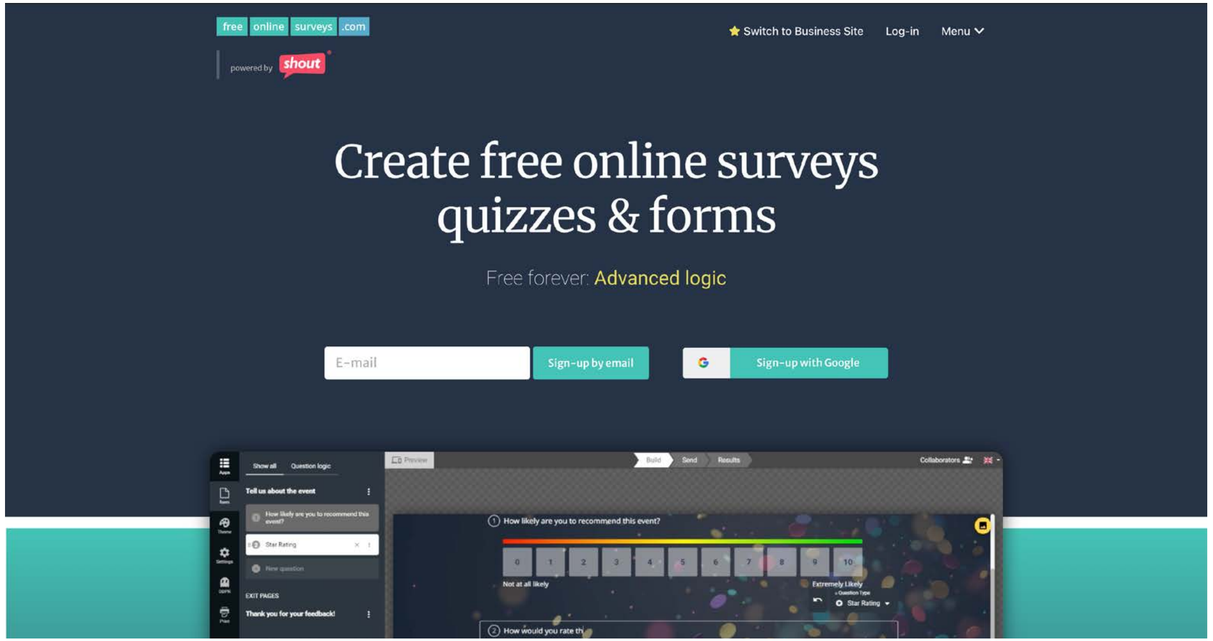
29% (2) Yes

71% (5) No

(7) Responses

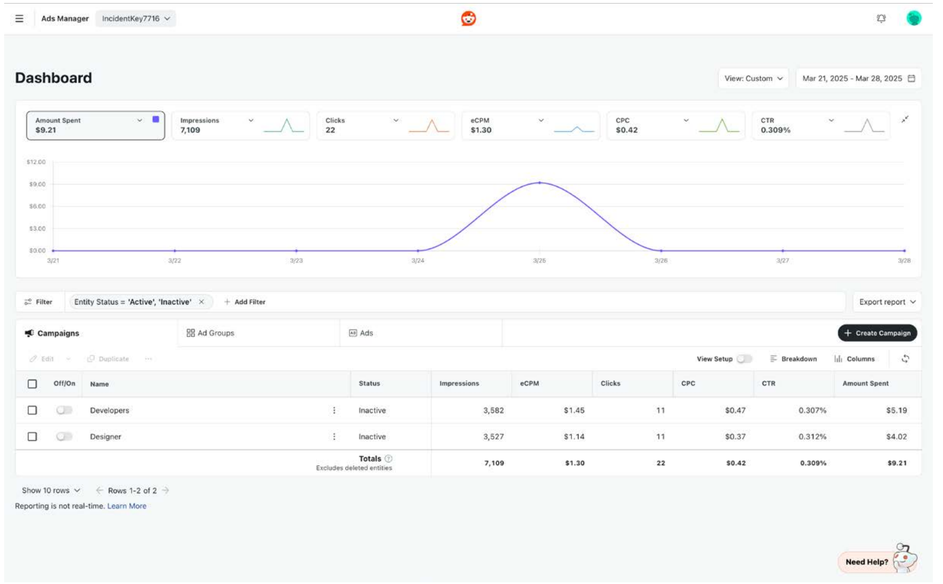
Devlopers's Survey

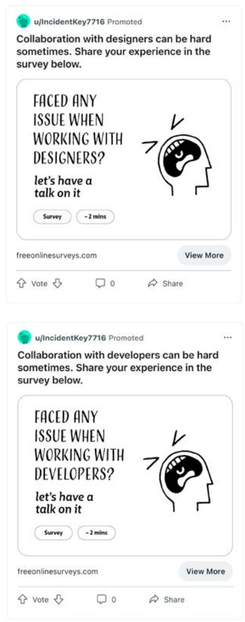
Developers critically highlight slow/inefficient handoff processes and ambiguous early requirements as major pains, underscoring the need for clearer initial specs and streamlined design delivery.



Survey created on freeonlinesurvey.com

Cheapest alternative of Typeform with Ranking style questions

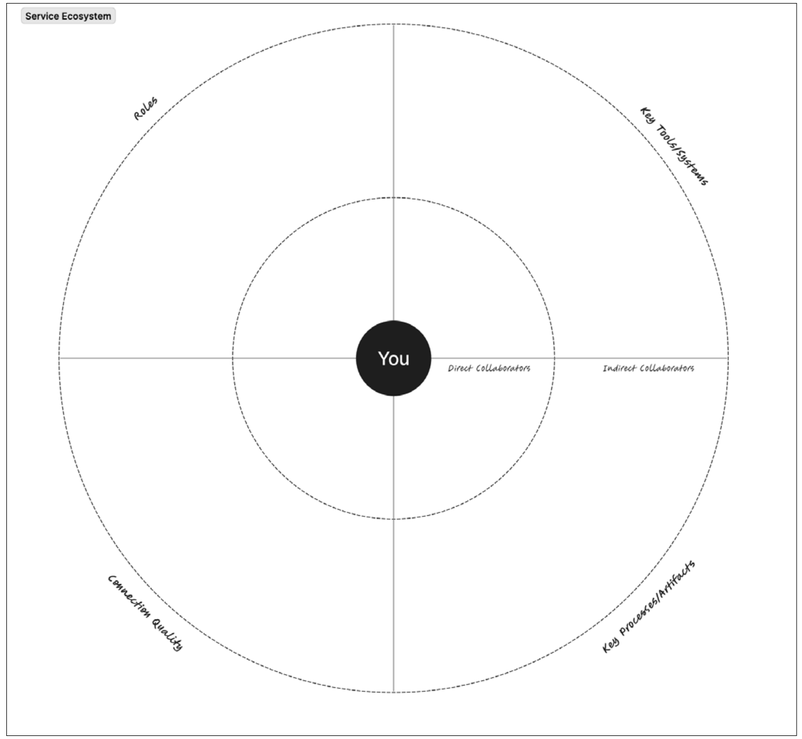




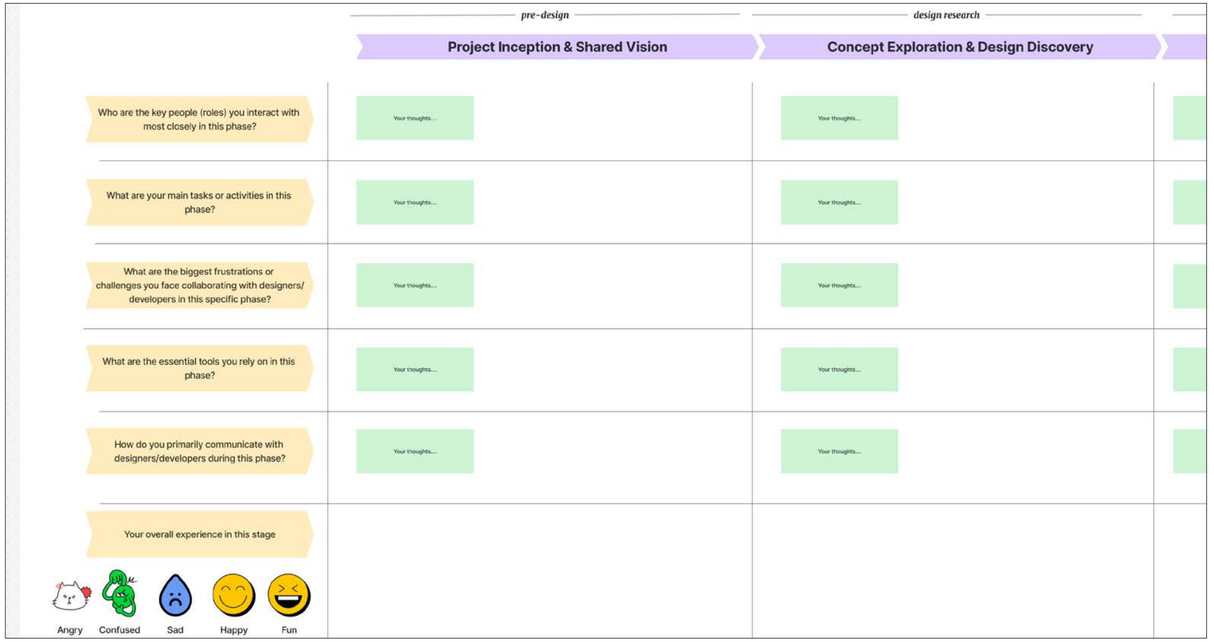
Reddit ads for distribution of survey

Created ad creatives and setup a targeted campaign to get more varied responses on survey

Ecosystem map to understand who are the direct and indirect collaborators and their details



Activity Design



Journey map to identify at which steps they face issues

Interview Guide

Introduction & Warm-up (5 minutes)

- Thank participant for their time.
- Briefly re-introduce the research project: "We're exploring ways to improve collaboration, workflows, and overall experience between designers and developers, aiming to make product creation smoother and more effective for everyone."
- Explain the interview format: "This will be a semi-structured conversation. I have some key areas I'd like to discuss, but please feel free to share any thoughts or experiences you feel are relevant. There are no right or wrong answers; we're interested in your honest perspective."
- Confidentiality reminder.
- Ask for consent to record.
- "To start, could you briefly tell me about your current role and the types of projects you typically work on that involve [designers/developers]?"
- "And roughly how long have you been in this role / working in this field?"

Current Workflow & Collaboration Experience (15-20 minutes)

- "Can you walk me through a typical project from when a new feature idea starts to when it's handed off to [development/design for implementation]? What are the main stages for you and where do you interact most with [designers/developers]?"
- "Thinking about the early stages – when ideas are just forming or initial concepts are being explored – how and when do you typically first engage with [developers/designers] about technical feasibility or design direction?"
- "What works well in these early interactions? What are the challenges?"
- "In an ideal world, when and how would you like these early feasibility/alignment discussions to happen?"
- (For Designers): "When you're preparing designs for handoff, what does your 'spec package' typically include? How do you ensure developers have all the information they need (states, interactions, edge cases, responsive behavior)?"
- (For Developers): "When you receive designs for implementation, what information is usually provided? What's typically clear, and what often requires further clarification or is missing (e.g., states, interactions, edge cases, responsive behavior)?"
- "What are the biggest challenges or frustrations you experience around the design specification and handoff process?"
- (For Designers): "How do you typically communicate the user problems, research insights, or strategic goals behind your design decisions to developers?"
- (For Developers): "How important is it for you to understand the 'why' – the user needs or business goals – behind a design? How often do you feel you get this context clearly?"
- "Can you give an example of when understanding (or not understanding) the 'why' significantly impacted your work?"
- "How does your team manage design versions and ensure everyone is working from the latest, correct 'source of truth' for designs?"
- "What challenges, if any, do you face with design versioning or knowing what has changed between iterations?"

Pain Points & Unmet Needs (10-15 minutes)

- "You mentioned [refer back to a challenge shared earlier, or use survey top pains]. Can you tell me more about that specific situation or type of frustration?"
- "If you could wave a magic wand and fix one or two things about how designers and developers work together on your team/projects, what would they be?"
- Functional Needs: "What specific information, tools, or process steps do you feel are currently missing or inadequate that would make your collaboration with [designers/developers] smoother or your work more effective?"
- Emotional Needs: "Thinking about your interactions with [designers/developers], what makes you feel most productive, valued, and understood? Conversely, what makes you feel most frustrated, disempowered, or disconnected?"

Attitudes, Beliefs & Cultural Aspects (5-10 minutes)

- "In your experience, what are some common assumptions or beliefs [designers have about developers / developers have about designers] that can sometimes cause friction?"
- "What does 'good collaboration' between designers and developers look like to you? What are the key ingredients?"
- "How would you describe the current culture of collaboration between design and development in your team/organization?"
- "What role do you think things like 'ego,' 'ownership,' or 'understanding each other's constraints' play in the success (or failure) of collaboration?"

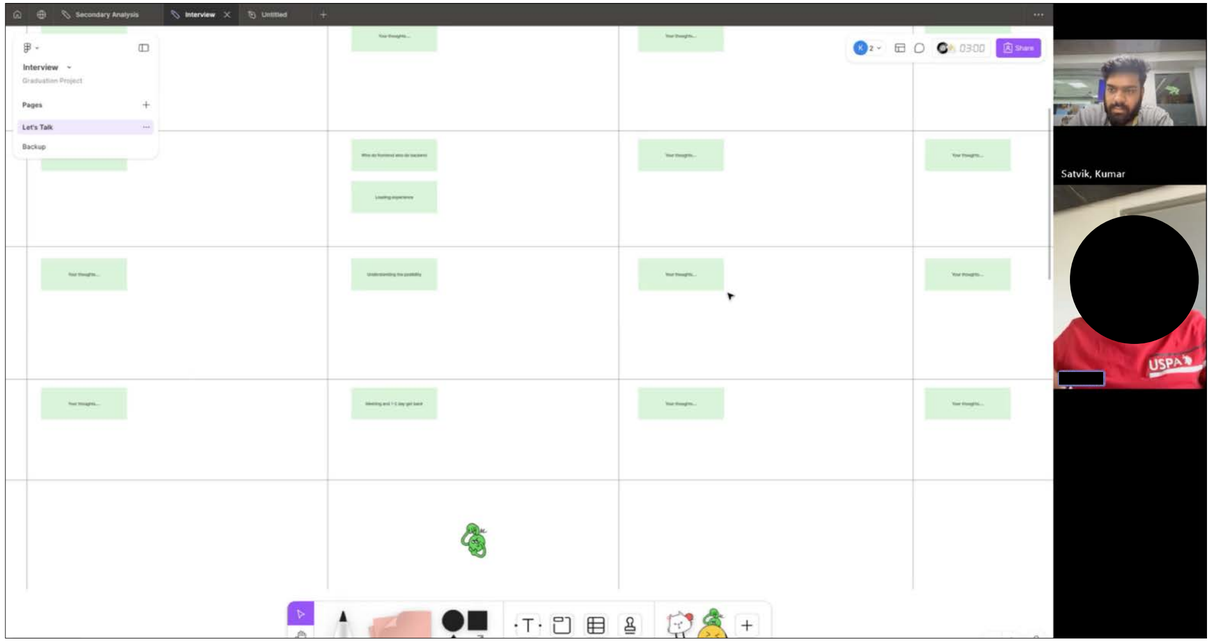
Ideal Future State & Potential Solutions (5-10 minutes)

- "We've talked a lot about challenges. Now, let's imagine an ideal future. If you had the perfect set of tools and processes for collaborating with [designers/developers], what would that look like or enable you to do?"
- "What's one change – big or small – that you believe would have the most positive, lasting impact on improving how designers and developers work together at your company?"

Wrap-up (2-3 minutes)

- "This has been incredibly insightful, thank you so much for sharing your experiences and perspectives."
- "Is there anything else you'd like to add that we haven't covered, or any final thoughts on this topic?"
- Explain next steps in the research.
- Thank them again.

Interview Guide



Conducted the activity with 6 designers and 3 developers

AN01

AN01 - Designer

Experience: **~11-12 years**
Roles: **Head of Design**
Industry/Org Type: **Tech Startups, Data-driven Tech Company**
Environment Constraints: **Talent pipeline disparity (Eng vs Design), Pre-existing biases about design, Resistance to new collaborative tools.**

An experienced design leader (~11-12 yrs) focused on strategic collaboration and design maturity in tech companies. Critiques the 'task-taker' engineering mindset and misconceptions about design's role/complexity. Advocates for inclusive, structured critique, shared roadmaps, and emphasizing soft skills to bridge disciplinary gaps and manage ego effectively.

"A lot of engineering education globally trains people to be task completers... That approach misses the bigger picture and makes true collaboration difficult."

"Ultimately, soft skills are the tools. Negotiation, empathy, clear communication – that's how you bridge the gaps between disciplines."

"A structured churning of ideas. It needs facilitation to be productive, otherwise, it just devolves into ego battles."

Stacks/Platforms

Clear, as clear effective for real-time collaboration

EMAIL

Problem for adoption for original collaboration, faces resistance

FIGJAM

Standard Design Platforms

Project Management / Roadmapping Platforms

Core Motivations & Goals	Foster genuine, productive cross-functional collaboration (Design, Eng, Product).	Elevate the understanding, role, and ownership of design within the organization.	Create structured, productive forums for ideation and critique, minimizing ego.	Build mutual understanding, empathy, and respect between disciplines.	Improve cross-functional communication through education, shared processes, and soft skills.	Raise the overall quality of design execution and holistic problem-solving.	Ensure design pushes boundaries responsibly while remaining feasible.	Counteract misinformation and promote best practices.				
Key Pain Points & Frustrations	Significant disparity in talent pipelines (Eng vs. Design).	Prevalent misconceptions about design (easy, purely visual, lacks depth).	Lack of deep 'design sensibility' in engineering teams.	Ego clashes hindering productive cross-functional discussions.	Inconsistencies between user stories (Jira/ Confluence) and Figma designs.	Difficulty changing pre-existing biases against design's value.	Communication barriers (differing perspectives, articulation issues).	Engineers sometimes operating in 'task-taker' mode vs. collaborative problem-solving.	Resistance/unfamiliarity with modern collaborative tools (e.g., FigJam).	Identifying and addressing skill gaps ('bad talent') in problem-solving/ communication.	Misinformation online creating harmful biases.	Underestimation of design complexity and iterative effort.
Attitudes & Beliefs	Engineering education can foster a 'task-taker' mentality hindering collaboration.	Many non-designers perceive design as easy or superficial (visuals only).	Design's role is to improve user experiences and push boundaries responsibly.	Shared roadmaps between design and development are critical for alignment.	Structured critique is necessary for productive, ego-managed collaboration.	Designers need ownership; engineers need interest in the design domain.	Soft skills (communication, empathy, negotiation) are the most critical tools.	QA is crucial for understanding user experience and business logic, not just functional testing.	Good design ideas can come from anyone.	Designer ego is a barrier; process tools help remove personal attachment.	AI is a recycler, not a solver of novel design problems requiring deep empathy.	
Key Behavioral Patterns	Actively initiates educational sessions for developers on collaborating with design.	Advocates for and implements structured collaborative sessions	Engages team members personally to build rapport.	Strategically addresses mindset issues and cultural barriers.	Emphasizes objective rationale and respect in critiques.	Promotes adoption of collaborative tools like FigJam despite resistance.						
Key Tools & Artifacts	Shared Roadmaps	Sessions / Structured Critiques	Educational Sessions for Devs	Stakeholder Understanding Sessions	Design Artifacts (Concepts/Mockups)	Critique Documentation	Design Principles / Standards / Heuristics	Critique Methods (e.g., Dot Voting)	Soft Skills (Communication, Empathy, Negotiation)			
Collaboration Style & Key Interactions	Focuses on improving cross-functional collaboration between Design, Engineering, Product, QA.	Actively works to integrate non-designers into the design critique process.	Stresses the importance of shared roadmaps and stakeholder alignment sessions.	Believes in building personal connections, especially on long projects.	Leads by defining processes and fostering a culture of respect and objectivity.							

One-pager of interview

Keywords

Strategic Thinker

Process Leader

Culture-Focused

Inclusive Advocate

Manages Ego Proactively

Values Soft Skills

Design Educator

Realistic (Challenges)

Fosters Structured Critique

Cross-Functional Bridge Builder

Aware of Industry Issues

Promotes Shared Understanding

SP01

SP01 - Designer

Experience: **~9 years UX**
Role Focus: **Email Marketing Design (Current)**
Industry/Org Type: **Product Company**
Environment Constraints: **Highly constrained (basic HTML/CSS), requires standardization.**

He is an experienced designer navigating a highly constrained technical environment (email). He emphasizes standardization and clear documentation to ensure consistency and quality. He proactively collaborates with developers, valuing their technical input early in the process and acknowledging reciprocal learning to bridge the design-tech gap effectively.

"Documentation toh ek tarah se key hoti hai... Agar aapko developers ke saath achhe se kaam karna hai toh aapko har cheez documented karni hi padengi."

"Yeah, actually, in this case, developers helped me."

"...bahut se hote hain...aap kitni hi achhi documentation banake de do woh sirf aake aapka color code dekhenge... Woh baaki nahi dekhenge..."

Stacks/Platforms

Core Motivations & Goals	Establish/maintain brand consistency.	Create functional, reliable emails despite technical limitations.	Improve workflow efficiency via standards/ documentation/reusability.	Foster better collaboration with Dev/QA.	Deliver high-quality, error-free experiences.
Key Pain Points & Frustrations	Severe technical constraints limiting creativity.	Dealing with legacy inconsistency.	Rendering challenges (fonts, dark mode, localization).	Remote work hindering informal collaboration.	Developers skimming/ ignoring documentation details.
Attitudes & Beliefs	Values documentation immensely ("key") but frustrated when not used thoroughly.	Believes in proactive communication and early feasibility checks.	Appreciates reciprocal learning (learned tech limits from devs).	Values experienced developers who understand rationale and ask questions.	Believes standardization and design systems are crucial.
Key Behavioral Patterns	Proactive communicator (consults devs early).	Detail-oriented (drove standardization).	Pragmatic problem-solver (switched fonts, dark mode strategies).	Values process and standards.	Prefers analog notes for speed/accessibility.
Key Tools & Artifacts	Design Guidelines (in Confluence/Word) (Core Artifact - Highly valued but sometimes skimmed)	QA Process Document (Core Process Artifact - Established by him)	Component Library (in Adobe XD) (Core Reusability Artifact - Supports standardization)	Wireframes (Client Provided) (Input Artifact) (Sometimes)	Email Templates (Core Output Artifact)
Collaboration Style & Key Interactions	Collaborates closely with Developers, Testers.	Consults developers early for feasibility.	Direct client communication via Jira.	Interaction with Design Lead/Manager for feedback.	Resolves QA issues by providing clearer artifacts (mobile views).

Basic HTML/CSS (Implementation Constraint)

Keywords

Detail-Oriented

Process-Driven

Standardization Advocate

Values Documentation

Proactive Communicator

Collaborative (Seeks Early Input)

Pragmatic Problem-Solver

Values Expertise (Reciprocal Learning)

Frustrated by Skimmed Docs

Constrained Environment Navigator

Prefers Analog Notes

Quality-Focused

One-pager of interview

MV01

MV01 - Designer

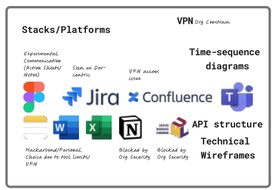
Experience: **3+ years UX**
Roles: **UX Designer**
Industry/Org Type: **EdTech Startup (Previous), Enterprise Cybersecurity Platform (Current, Large Org, ~20 products)**
Environment Constraints: **Security restrictions (VPN, tool limits like Notion), Siloed teams.**

He is an experienced designer navigating a highly constrained technical environment (email). He emphasizes standardization and clear documentation to ensure consistency and quality. He proactively collaborates with developers, valuing their technical input early in the process and acknowledging reciprocal learning to bridge the design-tech gap effectively.

"...most of the problem occurs... I am able to give that end-to-end flow... but at each screen there are a lot of edge cases... How do I give it to them?"

"...if I don't specifically give them that you have to make these two cards responsive, so they just give it fixed width..."

"...sometimes very basic questions that comes and which frustrates us... serious lack of ownership... mindset is that you have to give us this and our job is only to develop this."



Core Motivations & Goals	Achieve accurate, high-fidelity implementation.	Clearer, efficient handoff/ communication process.	Improve collaboration/ shared understanding with developers.	Effective, accessible design documentation.	Understand technical limitations early to reduce rework.	Find tools that genuinely support design workflow.
Key Pain Points & Frustrations	Difficulty conveying complex specs (edge cases, interactions, responsiveness).	Inefficient communication methods and information loss in siloed teams.	PM tools (Jira/Confluence) seen as dev-centric and have access issues (VPN).	Security restrictions limiting tool choices (Notion).	Perceived lack of developer ownership/proactivity.	Remote work hindering quick chats.
Attitudes & Beliefs	Handoff processes differ significantly by culture.	Static prototypes are limited for complex designs.	PM tools don't cater well to design needs.	Values early (senior) developer input for feasibility.	Face-to-face interaction is most efficient for quick clarifications.	Lack of ownership hinders collaboration.
Key Behavioral Patterns	Actively experiments with communication methods (Figma action sheets/ notes).	Proactively seeks face-to-face interaction.	Uses personal tools as workarounds.	Reflective on process differences (startup vs enterprise).	Frustrated by inefficiencies.	
Key Tools & Artifacts	Interactive Prototypes (Figma) (Core Handoff Artifact)	Figma Action Sheets/Notes (Experimental Communication Artifact/ Workaround)	Jira Tickets (Formal Task/ Communication Artifact)	Confluence Pages (Documentation Artifact)	Teams Messages/Calls (Communication Method)	Personal Notes (Mac Notes, Word, Excel) (Personal Tracking Artifact/ Hackaround)
Collaboration Style & Key Interactions	Frequent 1-on-1 with Developers (clarification/ feasibility), prefers face-to-face.	Direct with Designers, Scrum Masters, Team Lead.	Engages senior Devs/ Managers for early strategic input	Interacts with QA team.	Indirect with Stakeholders/ Clients	Relies on direct comms (Teams/in-person) despite formal channels.

Keywords

Adaptable (Contexts)

Experimental Communicator

Seeks Clarity

Values Early Feasibility Input

Proactive (Seeks F2F)

Uses Workarounds (Tools)

Frustrated by Inefficiency

Frustrated by Lack of Ownership

Needs Better Design Tools

Struggles with Complexity Handoff

Aware of Process Differences

One-pager of interview

AS01

AS01 - Designer

Experience: **~1 year**
Roles: **UI/UX Designer**
Industry/Org Type: **Tech Company**
Environment Constraints: **Immature Design System, Relatively late developer involvement in process.**

A UI/UX designer (~1 yr exp) using Figma primarily. Receives briefs (sometimes vague) from Mgr/Lead. Finds devs involved late, leading to feasibility issues post-design. Struggles conveying complex interactions with static screens and managing fragmented feedback (Slack, comments). Desires earlier technical input and better dev understanding of design rationale.

"Developers usually join later... jab design kaafi had tak final ho jaata hai."

"It would be helpful agar thoda technical input pehle mil jaaye... toh design hi waisa banega jo easily implement ho sake."

"Collaboration theek hai, but sometimes communication gap feel hota hai... design ke peeche ka 'why' shayad clear nahi hota devs ko."

Stacks/Platforms

Core Motivations & Goals	Receive clear, well-defined project requirements.	Create user-friendly, aesthetic, and technically feasible designs.	Collaborate effectively with developers and receive timely, constructive feedback.	Have a smoother handoff process with clear communication.	To minimize rework caused by late changes or unforeseen technical issues.	To ensure developers understand the reasoning behind design choices.	To work efficiently using tools like Figma.	To contribute to a consistent user experience across org's products.
Key Pain Points & Frustrations	Initial project briefs sometimes lack sufficient detail.	Designing without full awareness of technical constraints leads to rework.	Developers typically involved late in the design process.	Difficulty communicating dynamic design aspects (interactions, animations) via static artifacts.	Inconsistencies between user stories (Jira/ Confluence) and Figma designs.	Significant requirement changes after design completion/handoff.	Fragmented feedback across multiple channels (Slack, Figma comments, meetings).	Perceived communication gap regarding design rationale with developers.
Attitudes & Beliefs	Early technical input would help create more feasible designs.	Static screens have limitations for conveying complex interactions.	Late requirement changes are disruptive and frustrating.	Consolidating feedback from multiple sources can be tricky.	A communication gap sometimes exists regarding the 'why' behind designs.	A mature design system would improve efficiency and consistency.		
Key Behavioral Patterns	Relies heavily on Figma for the entire design process.	Follows existing style guides but adapts due to immature system.	Iterates based on feedback primarily from Design Lead/ Manager.	Uses Figma comments, Slack, and Jira for post-handoff clarification.	Expresses desire for process improvements (earlier tech input, better rationale sharing).			
Key Tools & Artifacts	Project Briefs (Input Artifact - Sometimes vague)	Wireframes, Mockups, Final UI Designs (in Figma) (Core Design Artifacts)	Figma Specs & Assets (Handoff Artifact)	Figma Comments (Communication Method/ Artifact - Limited effectiveness)	Prototypes / Videos (Workaround Artifact - For complex interactions)	Style Guides (Reference Artifact)	Official Mail (Communication Channel)	Direct Calls / In-person Chats (Communication Methods)
Collaboration Style & Key Interactions	Primary feedback loop during design is with Design Lead or Manager.	Developers involved later, primarily post-handoff via Figma links, Slack, Jira comments.	Direct collaborators include Managers.	Indirect collaborators: Content Writers, others via Mail/Whatsapp, Developer, CTO, Team Lead	Connection quality with direct collaborators (Managers) rated positively (Very approachable/ supportive)			

One-pager of interview

Keywords

Relies on Figma

Seeks Clarification

Values Lead/Manager Feedback

Desires Earlier Tech Input

Experiences Late Dev Friction

Struggles with Dynamic Spec Comms

Frustrated by Late Changes

Needs Feedback Consolidation

Aware of Rationale Gap

Adapting to Process

Notes Immature DS Impact

RV01

RV01 - Designer

Experience: **Not specified, but leading a team implies significant experience.**
Roles: **Design Lead**
Industry/Org Type: **Digital Solutions Company (Mobile apps, Web apps, Websites)**
Environment Constraints: **Team growth making current manual processes inefficient/risky.**

A Design Lead managing a growing team (5 designers/15 engineers) building various digital products. His primary challenge is an inefficient and error-prone Figma handoff process involving manual copy-pasting between 'playground' and 'handoff' files. This breaks change tracking, lacks a single source of truth, and hinders scaling. He's exploring solutions like Figma branching or Zeplin.

"Isse problem yeh hai ki Figma ka 'Compare Changes' kaam nahi karta... history track nahi hoti."

"And we lose the single source of truth, right? Design do jagah rehta hai, aur sync mein rakhna mushkil ho jaata hai."

"Yeh process tab theek tha jab team choti thi. But now, as we grow, the problems are getting more serious, it's becoming a bottleneck."

Stacks/Platforms

Handoff versioning in this step

Where development tools related to designs reside

Figma

Alternative to Figma for managing handoff and changes

Slack

Email

Jira

Core Motivations & Goals	To establish a robust and reliable single source of truth for all design work, from exploration to handoff.	To implement effective version control that allows clear tracking and comparison of changes, visible to both designers and engineers.	To streamline the process for communicating and implementing design updates after initial handoff.	To improve clarity and reduce ambiguity for engineers regarding what has changed in design specifications.	To find a scalable handoff and change management workflow suitable for a growing team.	To reduce the overhead and potential for errors associated with the current manual copy-paste process.	To make an informed decision on adopting appropriate tools or Figma features (branching, Zeplin) to solve their specific challenges.		
Key Pain Points & Frustrations	Inability to reliably track design changes in Figma due to copy-paste workflow breaking history.	Lack of a single source of truth (playground vs. handoff files).	Difficulty for engineers to efficiently identify specific updates within revised handoff files.	Current workflow for managing post-handoff changes is manual, inefficient, and error-prone.	Perceived limitations and reliability issues with Figma's native features like "Ready for Dev" and "Compare Changes".	Existing handoff process does not scale effectively with team growth.	Uncertainty about the best path forward (Figma Org/ branching or external tools).	Increased risk of miscommunication and implementation errors.	
Attitudes & Beliefs	The current copy-paste handoff workflow is problematic and risky.	Maintaining a single source of truth is crucial.	Clear change tracking is essential for efficient collaboration.	Native Figma features for handoff/versioning have limitations for their specific needs.	The process needs to adapt as the team scales.	Dedicated handoff tools (like Zeplin) might offer advantages.	Seeking best practices and guidance is important for process improvement.		
Key Behavioral Patterns	Manages a team workflow involving separate design ("playground") and handoff files in Figma.	Oversees a manual process for updating handoff files.	Has actively tested and evaluated native Figma features for handoff.	Is currently researching and considering major workflow changes (Figma Org/branching or Zeplin).	Actively seeks external best practices and guidance.	Identifies process bottlenecks related to team scaling.			
Key Tools & Artifacts	Figma "Playground" File (Internal Design Artifact - Initial creation space)	Figma "Handoff" File (Core Handoff Artifact - Problematic due to copy-paste)	Figma "Ready for Dev" status (Tested Feature - Found limiting)	Figma "Compare Changes" feature (Tested Feature - Broken by current workflow)	Figma Version History (Core Feature - Unreliable in current workflow)	Figma Branching (Potential Feature/Solution - Key reason for considering Org plan)	Best Practices Articles/ Guidance (Desired Input Artifact - Seeking process solutions)	Manual Copy-Paste Process (Current Workflow Step - Inefficient/Error-prone)	
Collaboration Style & Key Interactions	Leads a design team interacting with a larger engineering team.	Focus is currently internal to the design team's process and its interface with engineering (handoff).	Likely interacts with engineering leads/ managers regarding process changes.	Manages the flow of design information from designers to developers via the handoff files.					

Keywords

Process-Focused Leader

Identifies Bottlenecks

Actively Seeking Solutions

Aware of Scaling Issues

Seeks Best Practices

Concerned with Efficiency/ Risk

Values Single Source of Truth

Needs Better Version Control

Evaluating Tools

Manages Team Workflow

One-pager of interview

AK01

AK01 - Designer

Experience: **Experienced Designer**
Roles: **Senior Product Designer**
Industry/Org Type: **Fast-growing FinTech Company (FinSecure)**
Environment Constraints: **Cross-functional input requiring structured management, Designer ego potential barrier.**

An experienced Senior Product Designer in FinTech who champions an inclusive design process. Believes in bringing non-designers (Devs, POs) and their ideas into formal, rigorous critique sessions ("design jams"). Emphasizes objective, rationale-based evaluation using standard methods, collaborative validation decisions, and managing ego to leverage diverse perspectives effectively.

"Treat their suggestions just like you would in a critique with another designer. Ask for the rationale... Kyun socha aisa?"

"I often invite devs and POs to our design jams... They're almost always surprised by how rigorous it is."

"Too many designers get wrapped up in their ego... That's why so many process tools exist... to remove that personal attachment."

Stacks/Platforms

FigJam

Confluence

Core Motivations & Goals	To create an inclusive design culture where all team members feel empowered and heard.	To ensure all design ideas, regardless of source, are evaluated rigorously and objectively based on principles and potential user value.	To increase transparency and understanding of the design process and its inherent rigor among non-designers.	To make design decisions based on collective intelligence, validated assumptions, and objective rationale, rather than ego.	To leverage diverse perspectives and potentially valuable ideas from developers, POs, and others.	To foster mutual respect and collaboration between designers and other disciplines.	To improve the overall quality of design solutions by subjecting them to thorough, cross-functional critique.			
Key Pain Points & Frustrations	Non-designers feeling their ideas are ignored.	Designers reacting defensively/with ego to external input.	Lack of a structured, fair process for evaluating all suggestions.	Design decisions appearing subjective or based on personal preference.	Non-designers misunderstanding the depth of design validation.	Good ideas being lost due to poor communication or lack of forum.	Ego conflicts hindering open discussion and objective evaluation.			
Attitudes & Beliefs	Actively bringing non-designers into the formal design process is beneficial.	All ideas deserve initial exploration and consideration.	Making team members feel heard is crucial for morale and collaboration.	Design suggestions from non-designers should be critiqued with the same rigor as those from designers.	Critique must focus on rationale and identifying assumptions for validation.	Decisions on pursuing ideas should be collaborative.	Inviting non-designers to critiques increases their understanding of design rigor.	Feedback must be respectful, rationale-based, and not personal.	Good design ideas can originate from anyone.	Designer ego is a barrier; structured processes help remove personal attachment.
Key Behavioral Patterns	Actively invites developers and POs to internal "design jams" (critique sessions).	Treats all design suggestions equally within the critique process, demanding rationale.	Uses standard critique methods (standards, heuristics, comparative analysis).	Facilitates collaborative decision-making on which ideas warrant validation effort.	Emphasizes respect and objectivity throughout the critique process.	Utilizes (implicitly) structured design process tools to manage ego.				
Key Tools & Artifacts	"Design Jams" / Critique Sessions (Core Meeting Format/Process - Inclusive strategy for evaluation)	Design Rationale (Key input/output of critique - Central to his approach)	Design Standards / Heuristics (Reference Artifacts - Basis for objective critique)	Comparative Design Analysis (Critique Method/ Artifact - Used for evaluating trade-offs)	Validation Methods (User Testing, Data Analysis) (Process Step - Decided collaboratively)	Dot Voting / Silent Critique Methods (Implied Process Tools - For objectivity and managing ego)	Design Concepts/Mockups (Input Artifact for critique)			
Collaboration Style & Key Interactions	Highly collaborative and inclusive, actively drawing in Developers and Product Owners into design discussions.	Facilitates structured critique sessions ("design jams") as a core interaction mechanism.	Focuses on rationale-based discussion and objective evaluation to foster mutual respect.	Works to break down disciplinary silos and manage ego dynamics within the team.	Interacts regularly with a cross-functional team (Design, Dev, PO).					

Keywords

- Inclusive Collaborator
- Rationale-Focused
- Structured Critique Advocate
- Manages Ego Proactively
- Values Cross-Functional Input
- Objective Evaluator
- Process-Driven Facilitator
- Values Design Rigor
- Believes in Collective Intelligence
- Empathetic Facilitator
- Seeks Validation

One-pager of interview

C - Developer

Experience: **3+ years**
Roles: **Data Engineer, Python Developer, Full-Stack Developer**
Industry Size/Type: **Corporate environments**
Current Status: **Pursuing Masters (GPU focus)**
Environment Constraints: **Standard corporate workflows often lead to sequential handoffs.**

Bringing 3+ years of corporate backend/full-stack experience, C identifies a key friction: designers lacking technical awareness (constraints, effort) in standard sequential workflows using tools like Figma. This gap creates a post-handoff feasibility burden on developers. He believes earlier tech insight during design is crucial for efficiency.

"So the gap between designer and frontend developer, I feel is with respect to the technology, technology aspects."

"So if designers were aware that like this technology have this kind of responsibility or... functionalities... it would have been like very easier for for this workflow to like be efficient."

"...when developers get this... they have to map... there is a trade-off between what can be achieved and what is impossible... depending on the technology..."

Stacks/Platforms

Don't often don't mind understand it well — Workflow Core — Communication Artifact — Personal Choice/ Team Org

Google Workspace — Personal Choice/ Team Org

VR prototype

Core Motivations & Goals	Achieve a streamlined and efficient software development process.	Improve collaboration and understanding between design and development regarding technical feasibility.	Build products where the final implementation closely matches the intended design vision without major technical compromises discovered late.	Effectively balance Front-End and Back-End development needs and user experience.			
Key Pain Points & Frustrations	Designers often lack awareness of technical constraints, possibilities, or implementation effort during the design phase.	Developers bear the burden of assessing technical feasibility and mapping designs to code only after handoff.	The sequential nature of handoffs and the tech awareness gap lead to inefficient workflows and rework.	Potential for suboptimal implementation due to compromises made during post-handoff feasibility adjustments.	Ambiguity and missing details	Communication hindered when developers don't use or have access to the same design tools	
Attitudes & Beliefs	The primary "gap" between design and development is rooted in knowledge of technology aspects.	Early technical awareness integrated into the design process is crucial for efficiency.	The developer's role inherently involves making critical "trade-offs" after receiving designs.	Views the friction as a systemic workflow/ communication issue, not necessarily individual fault.	Acknowledges that designers cannot be expected to know all technology aspects innately.		
Key Behavioral Patterns	Thinks systematically about workflows and identifies bottlenecks.	Articulates processes and issues clearly.	Maintains a calm and thoughtful demeanor when discussing challenges.	Follows standard corporate processes but is analytical about their shortcomings.			
Key Artifacts	Requirements Documents / Briefs (Input Artifact)	PPT, Email (Communication Artifacts)	UML diagrams (Technical Design Artifact)	Time-sequence diagrams (Technical Design Artifact)	API structure documents (Core Technical Artifact)	Wireframes with technical aspects (Input Design/ Technical Artifact)	
Collaboration Style & Key Interactions	Interacts directly with Designers, Scrum Masters, Team Leads	Indirectly interacts with Stakeholders, Clients	Receives initial requirements often via Managers/PMs/POs	Communicates via a mix of pings, emails, formal meetings (e.g., specific feasibility calls)	Collaboration sentiment tends to degrade during design/handoff phases		

Keywords

- Backend-Focused
- Full-Stack Aware
- Analytical Workflow
- Identifies Bottlenecks
- Articulate
- Calm Demeanor
- Needs Designer Tech Awareness
- Experiences Handoff Friction Pragmatic
- Systemic Thinker
- Process Follower (Critical)
- Values Clear Communication
- Seeks Efficiency

One-pager of interview

R - Developer

Experience: **~3 years Frontend (ReactJS), also Backend (C++)**
Roles: **Full Stack Developer**
Industry/Org Type: **Working in product company**
Environment Constraints: **Fast-paced startup environment, Varying WFH schedules impacting sync communication, Emphasis on using existing libraries/components.**

A full-stack developer (~3 yrs ReactJS exp) primarily from startup/product backgrounds. He values developer input starting from Concept Exploration for feasibility checks. Uses Figma/Jira/Confluence and participates in formal handoffs. Finds collaboration "mixed" due to pace/WFH. Highlights specific handoff pain points: missing edge cases, disorganized Figma, pixel demands.

"Sometimes the error cases or loading cases, those kind of design will be missed."

"...it gets keeps on piling up right. And at the we have seen it it is never resolved properly and what will happen they will come up okay we are going to launch a second version..."

"I think second phase onwards... we can involve the developers."

Stacks/Platforms

Pain points: disorganized, empty, missing details

HLD/LLD
TBack

Used for defining API contracts

Core Motivations & Goals	Build features according to specifications efficiently.	Ensure technical feasibility of designs early in the process.	Receive clear, complete, and actionable design handoffs.	Build scalable and performant applications.	Use development time efficiently, minimizing rework and technical debt.	Have clear API contracts agreed upon for parallel FE/BE development.				
Key Pain Points & Frustrations	Missing design details in handoffs (edge cases, loading states, errors).	Designs lack consideration for scalability or responsiveness.	Disorganized/non-developer-friendly Figma files (scattered elements, canvas vs. DOM structure).	Pixel perfection requirements causing implementation difficulties.	Inconsistencies between user stories (Jira/Confluence) and Figma designs.	Information required for development scattered across multiple sources.	Frustration from repeated back-and-forth for missing design info.	Accumulation of technical debt due to time pressure ("fix in v2").	Difficulty connecting synchronously with collaborators (fast pace, WFH).	Unit discrepancies between design (px) and development (rem).
Attitudes & Beliefs	Developer input is less critical in early Inception but valuable from Concept Exploration onwards for feasibility.	Formal handoff meetings with multiple stakeholders are standard.	Clear API contracts enable efficient parallel development (FE/BE).	Technical debt is a common issue in fast-paced environments, often deferred.	Figma's canvas nature doesn't always translate well to structured HTML/CSS.	Collaboration quality can be mixed and affected by work environment dynamics.				
Key Behavioral Patterns	Collaborates with designers on feasibility, referencing Figma files or external examples.	Participates in formal design walkthrough meetings and uses Jira for tracking issues found.	Works in parallel on FE/BE based on agreed API contracts.	Interacts with QA and Designers during testing phases.	Experiences frustration but adopts pragmatic approaches (e.g., accepting tech debt).					
Key Tools & Artifacts	Figma files (Design Input Artifact)	References (Prototypes/Websites) (Design Input Artifact)	User Stories (in JIRA/Confluence) (Requirement Artifact)	API Contracts (Swagger format implied) (Technical Agreement Artifact - Crucial for parallel dev)	HLD / LLD Documents (Technical Design Artifacts)	TBack Document (Knowledge Transfer/Documentation Artifact)	Licenses Documentation	Informal daily chats, Scrums, Design Walkthroughs, Feature Delivery meetings (Communication Methods)		
Collaboration Style & Key Interactions	Direct collaborators include Designers, Product Teams.	Indirect interaction with QA, Customer Success (CX), DevOps, Legal, 3rd party software contacts	Communication via Informal chats (startups), Formal meetings (Scrums, Design Walkthroughs, Post-meeting reviews), Jira tickets.	Collaboration quality described as 'mixed', impacted by fast pace and remote work schedules.						

Keywords

Full-Stack Perspective

Pragmatic (Pace/Debt)

Values Clear Process (API Contracts)

Identifies Spec Gaps

Provides Feasibility Input

Collaborative (Within Process)

Affected by Remote Work

Needs Comprehensive Specs

Aware of Tech Debt

Notes Tool Discrepancies

One-pager of interview

PS01

PS01 - Developer

Experience: **7 years total (significant Angular/ TypeScript)**
Roles: **Frontend Developer**
Industry/Org Type: **Mid-sized Product Company (HealthTech Dynamics)**
Environment Constraints: **Heavy reliance on specific component library (PrimeNG), Agile sprint structure.**

A senior frontend dev (7 yrs exp) working with Angular and PrimeNG in a HealthTech product company. Emphasizes maintainability via library components, pushing back on custom designs incompatible with PrimeNG due to effort/ constraints. Finds handoffs (Figma) often lack detail (states, edge cases). Frustrated by mid-sprint changes and values understanding design 'why'.

"Woh Figma mein custom design bana rahe the, lekin humara standard PrimeNG hai. Implement karna mushkil tha, time consuming bhi."

"Mid-sprint changes are the worst... kaam already start ho gaya hota hai, phir change aata hai toh sab rework karna padta hai."

"Sometimes you need to understand the 'why' behind a design choice, not just 'build this'. It helps us suggest better technical solutions."

Stacks/Platforms

Heavily influences implementation; source of friction vs. custom design.

HTML CSS TS PRIMENG Jira Confluence

Core Motivations & Goals	To build features efficiently and maintainably using established tools and libraries (Angular, PrimeNG).	To receive clear, complete, and unambiguous design specifications for implementation.	To understand the user context and design rationale behind features to make better technical decisions and contribute effectively.	To have smooth, timely, and effective communication with designers for clarifications and feedback.	To minimize rework and disruptions caused by late-stage design changes.	To ensure the final product is consistent, accessible, and high quality.	To foster a collaborative environment where technical constraints and possibilities are considered early.	
Key Pain Points & Frustrations	Designs proposed are sometimes incompatible or difficult to implement using the established PrimeNG component library, leading to rework or difficult negotiations.	Lack of designer awareness regarding the technical effort required for custom components versus leveraging the existing library.	Incomplete specifications in design handoffs, particularly around interaction details, edge cases, and error states.	Communication inefficiencies when using asynchronous channels like Figma comments for complex clarifications; risk of missed information.	Disruptive and inefficient mid-sprint design changes requiring rework after development has commenced.	Insufficient communication of design rationale ('the why') behind UI/UX decisions, hindering developers' ability to make informed technical choices or suggest alternatives.	Inconsistent levels of detail and context provided by different designers.	Accessibility considerations sometimes being addressed late in the design process.	Difficulty in maintaining UI/ UX consistency across different product features without strong guideline enforcement or communication.
Attitudes & Beliefs	Using established component libraries (PrimeNG) is better for maintainability and efficiency.	Designers sometimes don't fully grasp technical limitations or implementation effort.	Figma comments are okay for small clarifications but poor for complex discussions.	Understanding the 'why' behind design helps developers suggest better technical solutions.	Early, brief developer involvement before design finalization is highly beneficial.	Mid-sprint changes are highly disruptive.	Accessibility should be considered early.	Consistency requires clear guidelines and communication.	
Key Behavioral Patterns	Actively pushes back on designs conflicting with the component library, providing rationale.	Uses Figma comments but prefers sync meetings or Slack for urgent/complex issues.	Advocates for understanding design rationale.	Works within a two-week Agile sprint structure.	Highlights the need for explicit responsive design specs.				
Key Tools & Artifacts	Jira Tickets (Task/ Requirement Artifact)	Figma Links/Specs (Core Design Input Artifact - Often incomplete)	Figma Comments (Communication Method/ Artifact - Limited effectiveness)	Design Rationale Documentation (Desired Artifact - Aids implementation)	Responsive Design Specs (Desired Artifact)	Accessibility Guidelines (Reference Artifact - Needed Earlier)	Centralized Design Guidelines (Reference Artifact - Needed for Consistency)	Agreed API Process/ Contracts	Sprint Ceremonies (Planning, Review)
Collaboration Style & Key Interactions	Interacts with Designers via Figma links, comments, Slack, sync meetings.	Interacts with team via scheduled sprint ceremonies (planning, review).	Engages in negotiation/ pushback when designs conflict with technical constraints/library.	Advocates for early involvement to provide feasibility input.	Level of collaboration/detail received varies by designer.				

Keywords

Technically Grounded (Libs)

Pragmatic (Advocates Library Use)

Strong Communicator (Rationale)

Values Design Rationale

Process-Aware (Sprints)

Identifies Spec Gaps

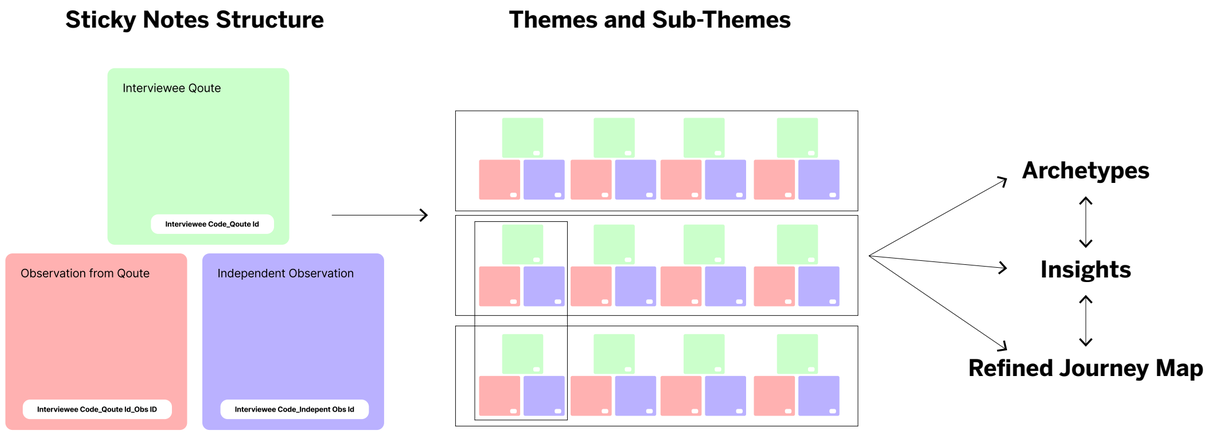
Frustrated by Mid-Sprint Changes

Needs Consistent Detail

Advocates Early Involvement

Maintainability Focus

One-pager of interview



Sensemaking Process

Handoff & Specification Breakdown

Misaligned Collaboration & Feasibility Blindspots

Lack of Shared Context & Rationale

Inefficient Communication Channels & Workflow

Cultural & Mindset Disconnect

5 Major Themes

Incomplete/Missing Specs: Edge Cases & States

Designs frequently lack details for non-happy paths, error states, empty states, loading states, and complex interaction nuances.

1.1 (Theme 1)

Ambiguous Responsive Design Specs

Lack of clear specifications for how designs should adapt across different screen sizes, leading to implementation inconsistencies or defaults.

1.2 (Theme 1)

Difficulty Conveying Dynamic Interactions

Static design artifacts (e.g., Figma screens) are often insufficient for communicating complex animations, transitions, or nuanced user flow behaviors.

1.3 (Theme 1)

Inconsistent Handoff Artifacts/Deliverables

Variation in what constitutes a 'complete' handoff package (e.g., Figma files vs. interactive prototypes) across teams/companies, causing confusion.

1.4 (Theme 1 and Theme 4)

Technical Specification Mismatches

Designs created without adhering to established technical constraints (e.g., email HTML/CSS limits, specific component libraries like PrimeNG, defined units like px vs. rem).

1.5 (Theme 1 and Theme 2)

Versioning & Change Tracking Failures in Handoff Files

Manual copy-pasting or current tool limitations break reliable version history and make it hard for developers to identify specific changes in updated handoff files.

1.6 (Theme 1 and Theme 4)

Lack of/Poorly Utilized Design Systems & Guidelines

Absence of a mature design system, or inconsistent adherence to existing style guides/component libraries, leading to bespoke design efforts and specification inconsistencies.

1.7 (Theme 1 and Theme 4)

Handoff Artifacts Not Optimized for Developer Consumption

Figma files or other design outputs not structured in a way that aligns with development needs (e.g., DOM structure, logical grouping of assets/layers).

1.8 (Theme 1)

Sub Themes : Handoff & Specification Breakdown (Theme 1)

Late Developer Involvement in Design Process

Developers are often brought into the design process too late, typically at or near handoff, preventing early feedback on technical feasibility and potential implementation challenges.

2.1 (Theme 2)

Designers' Lack of Awareness of Technical Constraints/ Effort

Designers may not fully understand or consider the technical limitations, implementation effort, or constraints of the specific technology stack or component libraries being used, leading to designs that are difficult or time-consuming to build.

2.2 (Theme 2 and Theme 1)

Post-Handoff Discovery of Feasibility Issues & Rework

Technical feasibility problems or significant implementation challenges are often only discovered by developers *after* the design handoff, necessitating rework, compromises, or difficult negotiations.

2.3 (Theme 2, Theme 1 and Theme 3)

Value & Benefit of Early Developer Consultation

Positive experiences and stated desires highlight the significant benefits of involving developers earlier in the design process for feasibility checks, understanding technical blockers, and reducing downstream iterations.

2.4 (Theme 2)

Reciprocal Learning & Cross-Skilling (Designer ↔ Developer)

Instances where designers learn technical constraints from developers, or developers gain design understanding, leading to better mutual awareness. This also includes formal training exchanges.

2.5 (Theme 2 and theme 5)

Misalignment on Effort Estimation & Timelines

Discrepancies in understanding the time and effort required for implementing certain designs, sometimes due to lack of early collaboration on scope and complex...

2.6 (Theme 2 and Theme 4)

Considering Non-Functional Requirements (e.g., Accessibility, Maintainability) Late

Critical non-functional requirements like accessibility or long-term maintainability (e.g., use of component libraries) are sometimes considered too late in the design process, leading to rework or suboptimal solutions.

2.7 (Theme 2 and Theme 1)

Sub Themes : Misaligned Collaboration & Feasibility Blindspots (Theme 2)

Insufficient Communication of Design Rationale by Designers

Designers sometimes do not adequately articulate or document the reasoning, user needs, or strategic goals behind their design choices, providing mainly the 'what' (UI) without the 'why'.

3.1 (Theme 3)

Developers Not Engaging With or Understanding Provided Rationale

Even when rationale is provided (e.g., in documentation), developers may skim, overlook, or not fully internalize it, leading to a focus on superficial implementation details rather than intent.

3.2 (Theme 3 and Theme 5)

Importance of 'The Why' for Developer Problem-Solving & Contribution

Developers express that understanding the design rationale empowers them to make better technical decisions, suggest more suitable implementation approaches, and contribute more effectively to achievin...

3.3 (Theme 3)

Loss of Context in Large or Rotating Teams

In larger organizations or with rotating development teams, the original context and rationale behind design decisions can be lost during handovers or as new individuals join, requiring repetitive explanations.

3.4 (Theme 3)

Need for Explicit Rationale in Documentation & Communication

A clear need is expressed for design documentation and communication to explicitly include not just the visual specifications but also the underlying design rationale and user context.

3.5 (Theme 3 and Theme 1)

Differing Perspectives Hindering Shared Context (Holistic vs. Specific)

Designers often approach problems holistically (user journey, overall experience), while engineering might focus on specific technical symptoms or components, creating a gap in shared understanding if these perspectives aren't bridged.

3.6 (Theme 3 and Theme 5)

Impact of Unclear Rationale on QA and Testing

When the rationale behind specific design choices (e.g., why an element behaves differently on mobile) is not clear to QA, it can lead to unnecessary bug reports or misunderstandings of intended functionality.

3.7 (Theme 3 and Theme 1)

Sub Themes : Lack of Shared Context & Rationale (Theme 3)

Fragmented Communication Across Multiple Channels

Design discussions, feedback, and clarifications are scattered across various tools (Slack, Jira, Figma comments, email, meetings), making it difficult to track decisions, consolidate information, and maintain a single source of truth.

4.1 (Theme 4)

Tool Limitations & Unsuitability for Cross-Functional Needs

Specific tools used in the workflow present limitations or are not well-suited for all roles; e.g., Jira being perceived as dev-centric and not design-friendly, Figma's limitations in versioning/change tracking, or access issues with Confluence (VPN).

4.2 (Theme 4 and Theme 1)

Inefficient Manual Processes & Lack of Single Source of Truth

Reliance on manual processes for updates (e.g., copy-pasting designs between files), leading to a lack of a single source of truth, broken version histories, and increased risk of errors or outdated information.

4.3 (Theme 4 and Theme 1)

Disruptions from Mid-Sprint Changes & Unclear Change Management

Design changes occurring after development has started (mid-sprint) cause significant disruption, rework, and frustration, often due to a lack of clear processes for managing such changes.

4.4 (Theme 4)

Challenges of Remote/Asynchronous Collaboration

Remote work environments and asynchronous communication make spontaneous, quick collaboration checks and informal problem-solving more difficult compared to co-located setups.

4.5 (Theme 4)

Inconsistent or Undefined Handoff/Workflow Frameworks

Lack of standardized, clearly defined workflows or frameworks for handoff and collaboration across different teams or company cultures, leading to confusion and inefficiency.

4.6 (Theme 4 and Theme 1)

Workarounds for Inefficient Official Tools/Processes

Team members resort to personal tools or informal workarounds (e.g., Macbook Notes, pen & paper, direct in-person visits) due to perceived inefficiencies, access issues, or unsuitability of official company tools and processes.

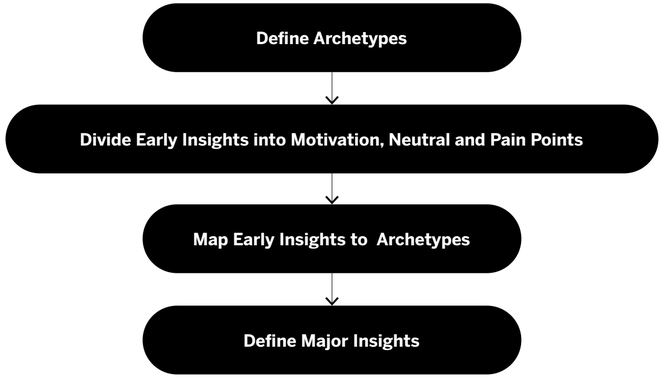
4.7 (Theme 4)

Sub Themes : Inefficient Communication Channels & Workflow (Theme 4)

'Task-Taker' vs. 'Collaborative Problem-Solver' Mindset	Misconceptions & Undervaluing of Design Complexity/ Role	Ego, Defensiveness, & Lack of Openness to Critique	Differing Role Expectations & Perceived Ownership Gaps
A prevalent mindset where some (often developers, as per perception) operate as 'task completers' focused on literal execution, rather than engaging as collaborative problem-solvers who seek to understand context and contribute to the overall solution.	Non-designers (engineers, business stakeholders) often perceive design as easy, purely visual, or superficial, underestimating its complexity, iterative nature, and strategic value.	Non-designers (engineers, business stakeholders) often perceive design as easy, purely visual, or superficial, underestimating its complexity, iterative nature, and strategic value.	Confusion or disagreement about who is responsible for certain details (e.g., responsiveness specifications, deep documentation engagement) and a perceived lack of proactive ownership from some team members.
5.1 (Theme 5)	**5.2 (Theme 5)**	**5.3 (Theme 5)**	**5.4 (Theme 5 , Theme 1 and Theme 3)**

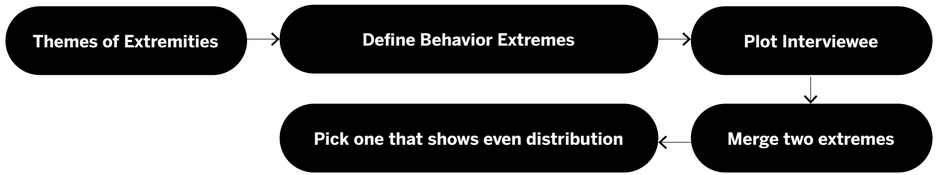
Impact of Pre-existing Biases & Difficulty Changing Mindsets	Importance of Empathy, Soft Skills, & Human Connection	Creating an Inclusive & Respectful Critique Culture	Disconnect from User/Market Realities
Strongly held pre-existing negative opinions or biases about design (or other disciplines) are difficult to change and can significantly impede collaboration and adoption of new practices.	Recognition that soft skills (empathy, communication, negotiation) and building personal rapport are crucial tools for bridging disciplinary gaps, fostering understanding, and improving collaboration.	The value of, and strategies for, fostering an inclusive culture where ideas from all team members are heard, respectfully critiqued based on objective rationale, and design is not seen as an exclusive domain.	Instances where team members (designers or developers) may lack familiarity with common user behaviors, platforms, or market contexts, potentially impacting design decisions or empathy.
5.5 (Theme 5)	**5.6 (Theme 5, Theme 3 and Theme 4)**	**5.7 (Theme 5)**	**5.8 (Theme 5 and Theme 3)**

Sub Themes : Cultural & Mindset Disconnect (Theme 5)



Process for writing Insights

Archetypes Making Process



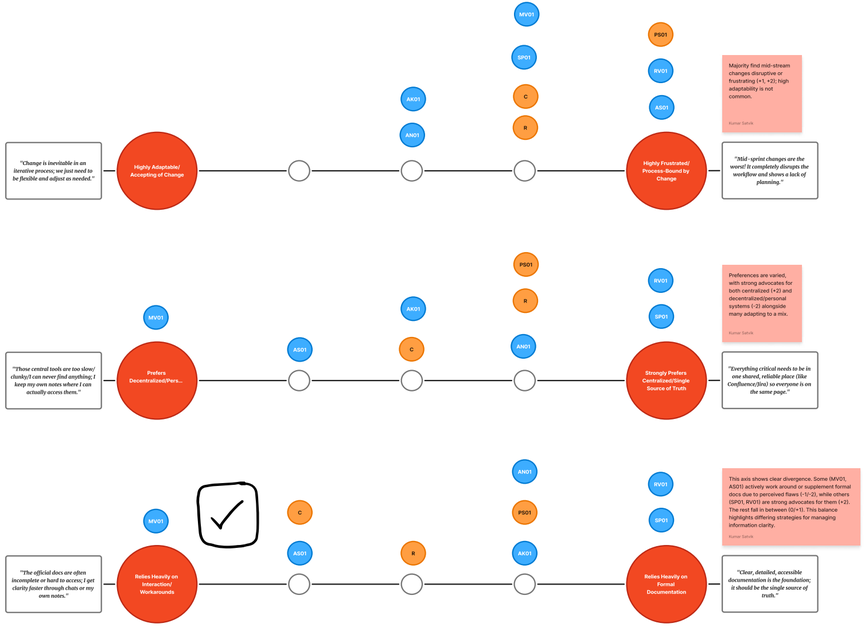
Process Adherence & Documentation

Collaboration & Communication Style

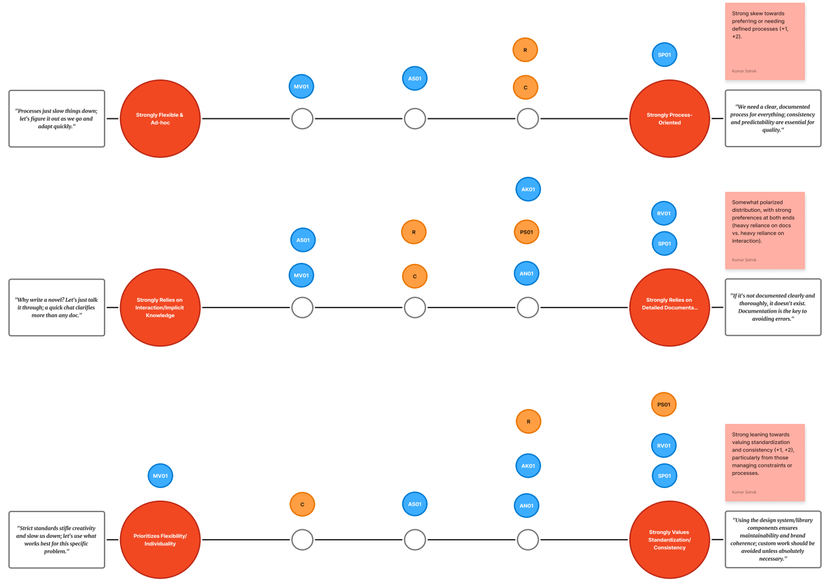
Technical Awareness & Integration

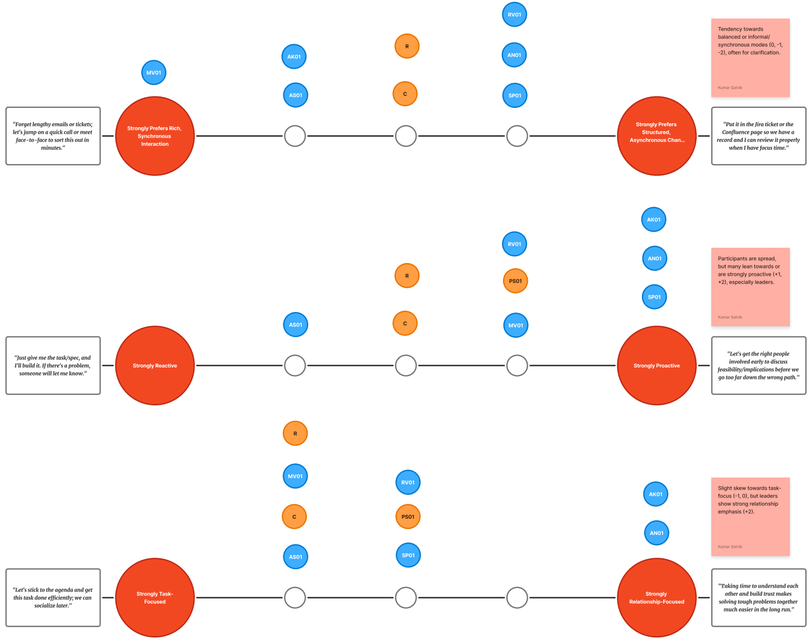
Problem Solving & Attitude

Themes of Extremities

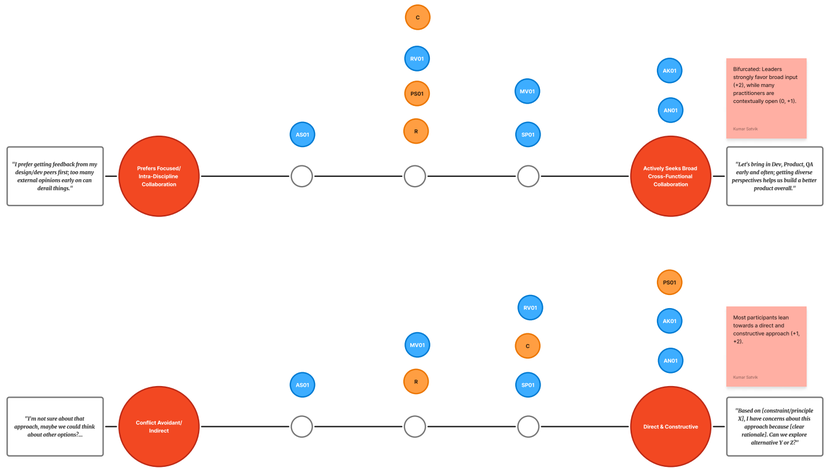


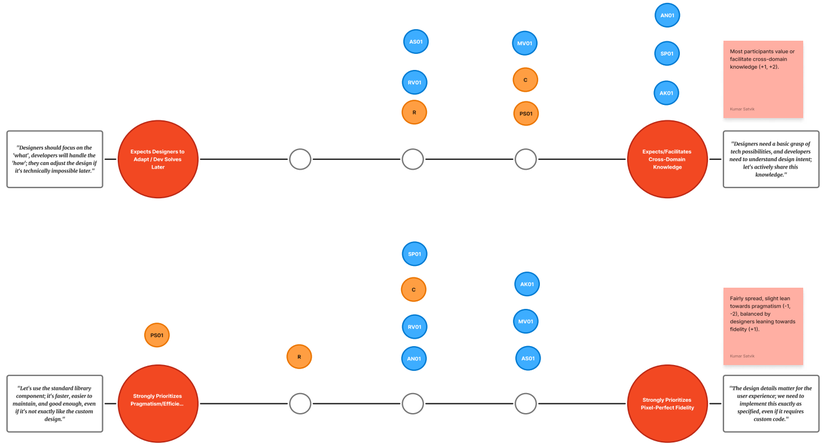
Process Adherence & Documentation

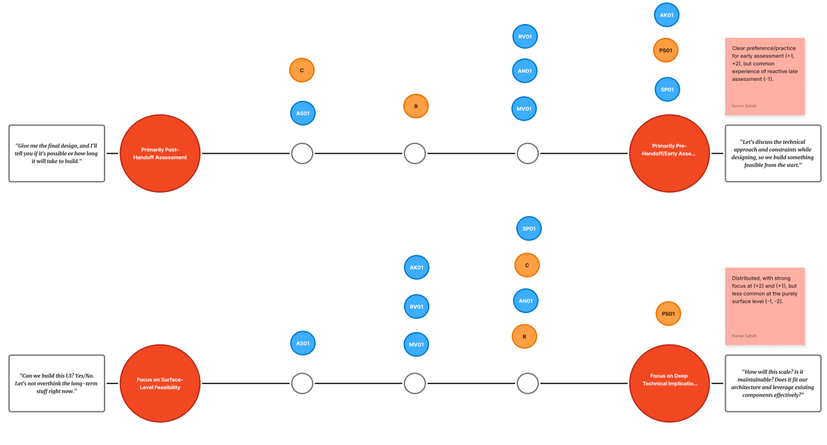




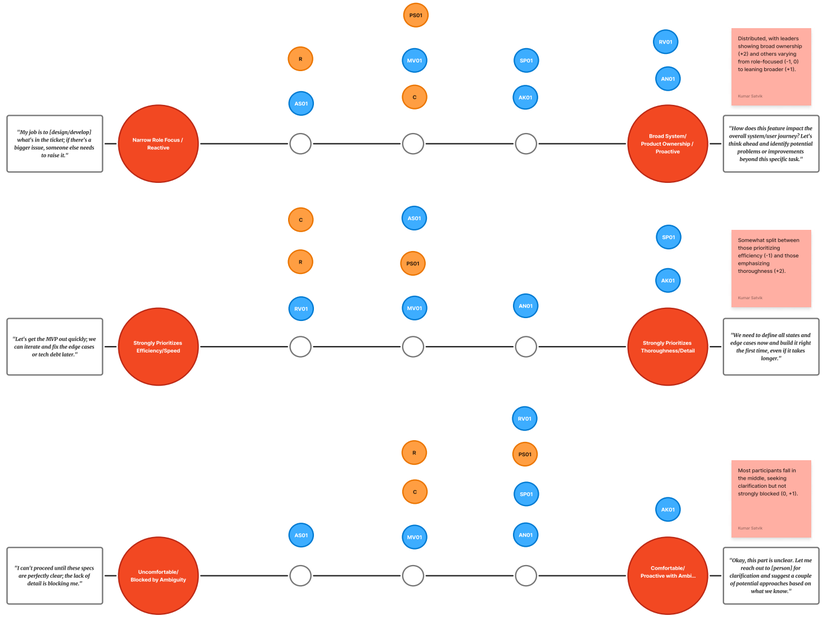
Collaboration & Communication Style

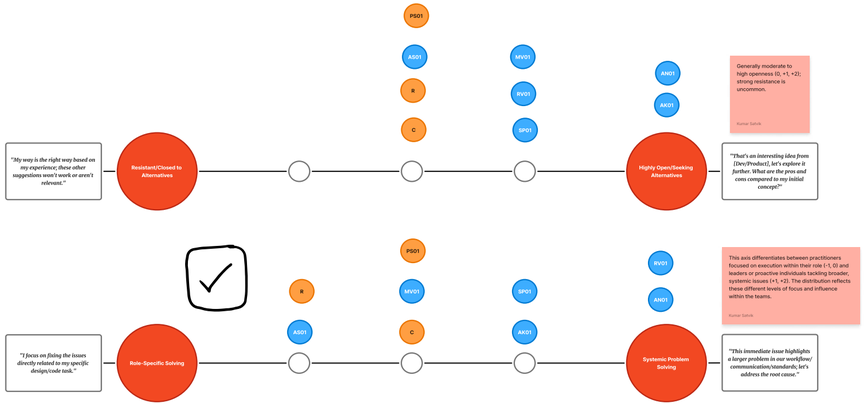




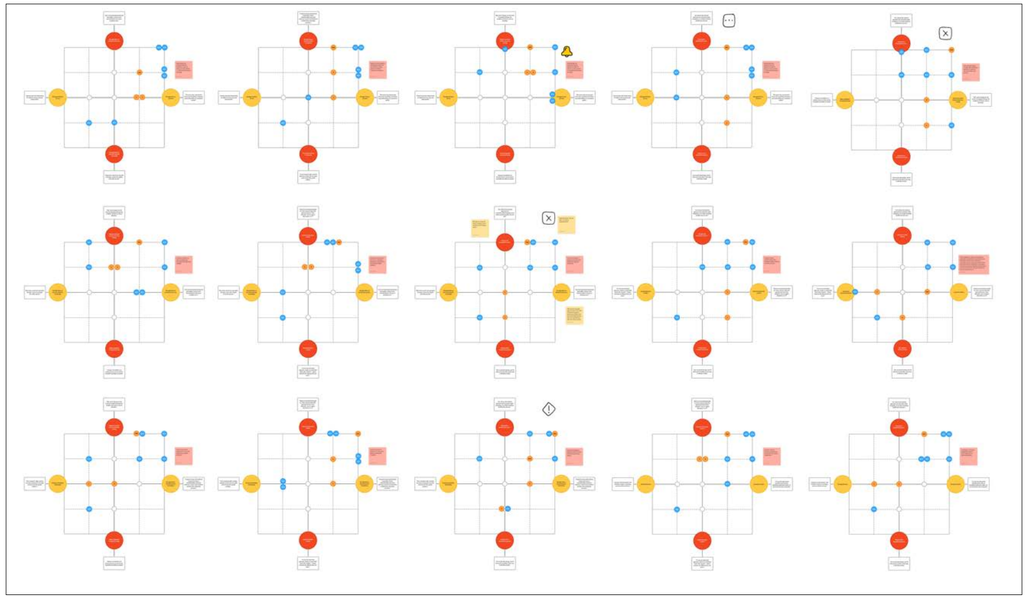


Technical Awareness & Integration

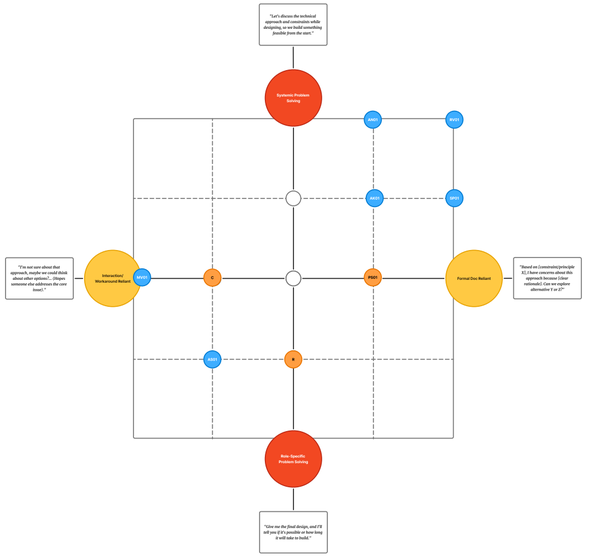

PS01
R
MV01
SP01
RV01
AS01
C
AK01
AN01
"My job is to {design/develop} what's in the ticket; if there's a bigger issue, someone else needs to raise it."
Narrow Role Focus / Reactive
Broad System/ Product Ownership / Proactive
"How does this feature impact the overall system/user journey? Let's think ahead and identify potential problems or improvements beyond this specific task."
Distributed, with leaders showing broad ownership (+2) and others varying from role-focused (-1, 0) to leaning broader (+1).
Kumar Satvik
C
AS01
SP01
R
PS01
AK01
RV01
MV01
AN01
"Let's get the MVP out quickly; we can iterate and fix the edge cases or tech debt later."
Strongly Prioritizes Efficiency/Speed
Strongly Prioritizes Thoroughness/Detail
"We need to define all states and edge cases now and build it right the first time, even if it takes longer."
Somewhat split between those prioritizing efficiency (-1) and those emphasizing thoroughness (+2).
Kumar Satvik
RV01
PS01
R
SP01
C
AK01
AS01
MV01
AN01
"I can't proceed until these specs are perfectly clear; the lack of detail is blocking me."
Uncomfortable/ Blocked by Ambiguity
Comfortable/ Proactive with Ambi...
"Okay, this part is unclear. Let me reach out to [person] for clarification and suggest a couple of potential approaches based on what we know."
Most participants fall in the middle, seeking clarification but not strongly blocked (0, +1).
Kumar Satvik

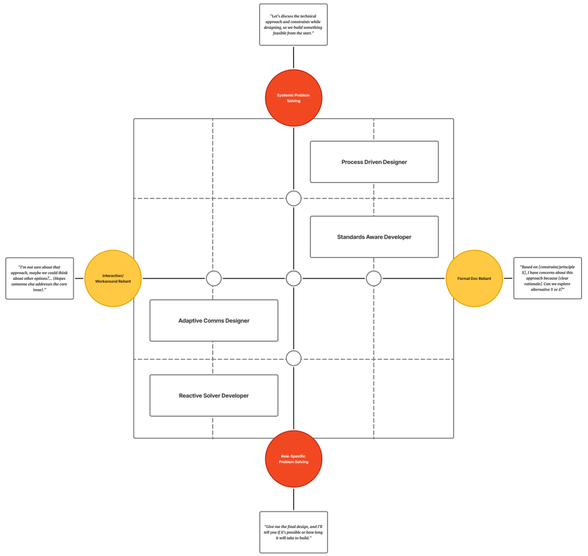

PS01
AS01
MV01
AN01
R
RV01
AK01
C
SP01
"My way is the right way based on my experience; these other suggestions won't work or aren't relevant."
Resistant/Closed to Alternatives
Highly Open/Seeking Alternatives
"That's an interesting idea from [Dev/Product], let's explore it further. What are the pros and cons compared to my initial concept?"
Generally moderate to high openness (0, +1, +2); strong resistance is uncommon.
Kumar Satvik
PS01
R
MV01
SP01
RV01
AS01
C
AK01
AN01
"I focus on fixing the issues directly related to my specific design/code task."
Role-Specific Solving
Systemic Problem Solving
"This immediate issue highlights a larger problem in our workflow/ communication/standards; let's address the root cause."
This axis differentiates between practitioners focused on execution within their role (-1, 0) and leaders or proactive individuals tackling broader, systemic issues (+1, +2). The distribution reflects these different levels of focus and influence within the teams.
Kumar Satvik

Exploring different axises



Finalized Graph





Archetype 1

Process Driven Designer

Experienced designer/lead creating clarity and quality through structured processes, standards, and proactive cross-functional collaboration.

Attitudes & Beliefs

- Clear processes and documentation are essential for quality and collaboration.
- Standardization (Design Systems, guidelines) is crucial for efficiency and consistency.
- Early cross-functional input, especially technical feasibility, is vital.
- Structured critique based on rationale improves design quality and manages ego.
- Soft skills and relationship building are important tools for collaboration.
- Good ideas can come from anywhere, but need rigorous evaluation.
- A single source of truth for information is highly valuable.

Pain Points & Frustrations

- Inconsistency stemming from lack of standards or process adherence.
- Inefficiencies caused by poor handoffs, late changes, or unclear communication.
- Collaborators ignoring or skimming crucial documentation/ guidelines.
- Ego clashes or lack of objectivity hindering productive critique/ collaboration.
- Difficulty scaling manual or inefficient processes (like handoffs).
- Misconceptions about design's complexity or value.
- Resistance to adopting structured processes or new collaborative tools.

Functional Needs

- Reliable tools for documentation, version control, and collaboration (e.g., Confluence, Figma Org features).
- Accessible and enforced design standards/heuristics.
- Shared roadmaps for alignment.
- Clear change management processes.
- Mechanisms for early technical input and validation.
- Structured meeting/ critique frameworks.

Emotional Needs

- Team buy-in and respect for established processes.
- Recognition of design expertise and the value of rigor.
- A culture supporting objective, constructive feedback (low ego).
- Empowerment to identify and improve systemic issues.
- Mutual understanding and respect across disciplines.

How likely are they to initiate collaboration or address issues preemptively?

Reactive — Proactive

How do they prefer to exchange information for clarity and efficiency?

Informal — Recorded

When do they prefer/tend to integrate technical feasibility into the workflow?

Late — Early

Archetype 2

Standards Aware Developer

Experienced developer focused on maintainability and efficiency through established technical frameworks and clear rationale.

Attitudes & Beliefs

- Using established libraries/standards is crucial for efficiency and long-term maintainability.
- Understanding the 'why' behind a design enables better technical solutions.
- Pushing back on technically problematic designs with clear rationale is necessary.
- Mid-sprint changes reflect poor planning and are highly inefficient.
- Early technical input from developers is beneficial for the design process.
- Accessibility and consistency are important technical considerations.

Pain Points & Frustrations

- Receiving designs incompatible with established component libraries or technical constraints.
- Designers lacking awareness of the effort/ implications of custom components vs. library use.
- Incomplete specifications lacking technical details (states, edge cases, accessibility).
- Disruptive mid-sprint design changes leading to rework.
- Insufficient design rationale provided with specs.
- Inconsistent levels of detail across different designs or designers.
- Maintaining consistency when standards aren't followed.

Functional Needs

- Clear guidelines on library usage vs. custom components.
- Comprehensive specs including states, edge cases, accessibility, and rationale.
- Stable requirements during sprints.
- Efficient channels for technical clarification (Slack/calls for urgent issues).
- Access to centralized design guidelines.
- Processes for early technical involvement.

Emotional Needs

- Designers receptive to technical feedback/ constraints.
- Respect for technical expertise and maintainability concerns.
- Predictable workflow with minimal disruption from late changes.
- Shared understanding of the impact of deviating from standards.

How likely are they to initiate collaboration or address issues preemptively?

Reactive — Proactive

How do they prefer to exchange information for clarity and efficiency?

Informal — Recorded

When do they prefer/tend to integrate technical feasibility into the workflow?

Late — Early

Archetype 3

Adaptive Comms Designer

Adaptable designer navigating process flaws by relying heavily on informal communication and workarounds for clarity.

Attitudes & Beliefs

- Static prototypes/specs are often insufficient for conveying complex enterprise designs.
- Direct, often informal, communication is necessary to bridge gaps.
- Formal PM tools don't always cater well to design documentation needs.
- Personal workarounds are sometimes required to manage workflow effectively.
- Early technical input is valuable but not always easy to get in the current process.
- Face-to-face interaction is highly efficient for resolving minor issues.

Pain Points & Frustrations

- Difficulty conveying nuances (interactions, edge cases, responsiveness) via static specs/ prototypes.
- Inefficiency and confusion from fragmented communication across multiple channels.
- Formal tools (Jira/ Confluence) seen as dev-centric, inadequate, or having access barriers (VPN).
- Receiving late technical feedback leading to rework.
- Late requirement changes disrupting workflow.
- Perceived lack of developer ownership or understanding of design rationale.
- Information loss in siloed teams (enterprise).
- Immature design systems requiring more bespoke work.

Functional Needs

- Better methods/tools to communicate dynamic/ complex specs contextually (beyond static Figma).
- Efficient and consolidated feedback channels.
- Accessible and design-friendly documentation/ tracking tools (or better personal workarounds).
- Earlier, reliable integration points with developers for feasibility checks.
- Clearer process for managing late requirement changes.

Emotional Needs

- Feeling understood by developers regarding design intent.
- Reduced frustration from tool/process limitations and inefficiencies.
- Easier, faster ways to get quick clarifications without lengthy meetings.
- A sense of shared ownership and reduced communication friction.

How likely are they to initiate collaboration or address issues preemptively?

Reactive — Proactive

How do they prefer to exchange information for clarity and efficiency?

Informal — Recorded

When do they prefer/tend to integrate technical feasibility into the workflow?

Late — Early

Archetype 4

Reactive Solver Developer

Pragmatic developer who often assesses technical feasibility post-handoff, making trade-offs to implement designs effectively.

Attitudes & Beliefs

- The gap between design and dev is often technical knowledge held by designers.
- Post-handoff, the developer's role involves crucial technical trade-offs.
- Clear API contracts are key for efficient parallel development.
- Workflow inefficiencies are common and impact delivery.
- Pragmatism is needed to navigate constraints and pressures (e.g., tech debt).

Pain Points & Frustrations

- Receiving designs with insufficient technical awareness or detail (edge cases, loading states).
- Bearing the burden of feasibility assessment and trade-offs post-handoff.
- Inefficient rework cycles caused by late discovery of issues.
- Ambiguity in specifications leading to repeated clarification.
- Scattered information across different tools/ documents.
- Accumulating technical debt due to time pressure or quick fixes.
- Communication friction if designers use tools inaccessible to developers.
- Collaboration challenges due to remote work or fast pace impacting communication.

Functional Needs

- Designs informed by technical awareness earlier in the process.
- More comprehensive specs including edge/ error/loading states.
- Clear API contracts for efficient development.
- Reliable and efficient channels for seeking clarification from designers.
- Consistent and centralized information sources.

Emotional Needs

- Recognition of the post-handoff technical complexity and effort involved.
- Clearer, more technically informed communication from design side.
- A more streamlined workflow reducing ambiguity and rework.
- Less pressure leading to technical debt compromises.

How likely are they to initiate collaboration or address issues preemptively?

Reactive — Proactive

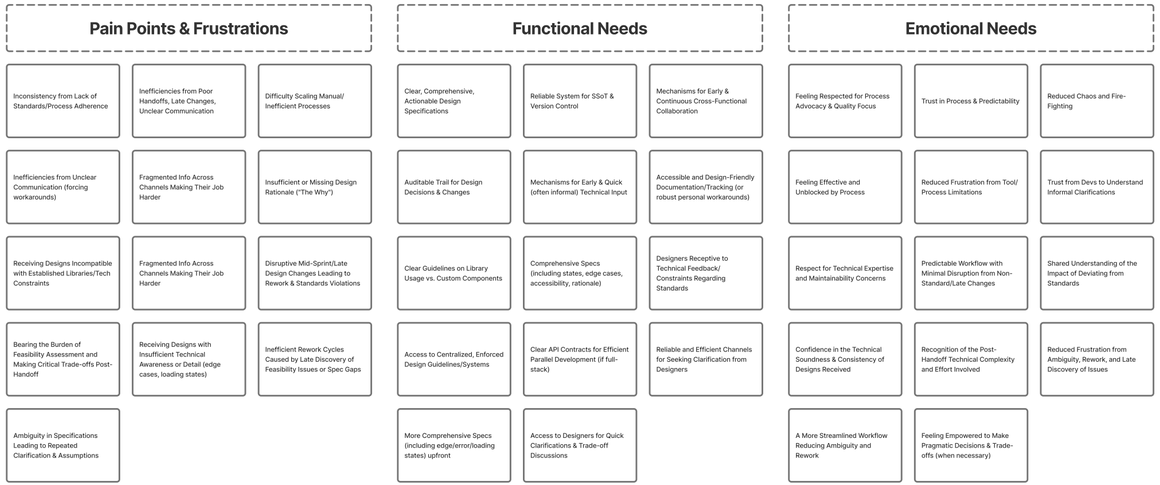
How do they prefer to exchange information for clarity and efficiency?

Informal — Recorded

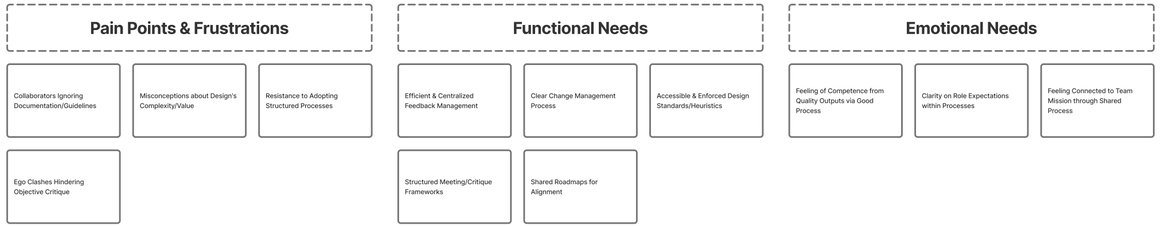
When do they prefer/tend to integrate technical feasibility into the workflow?

Late — Early

Critical

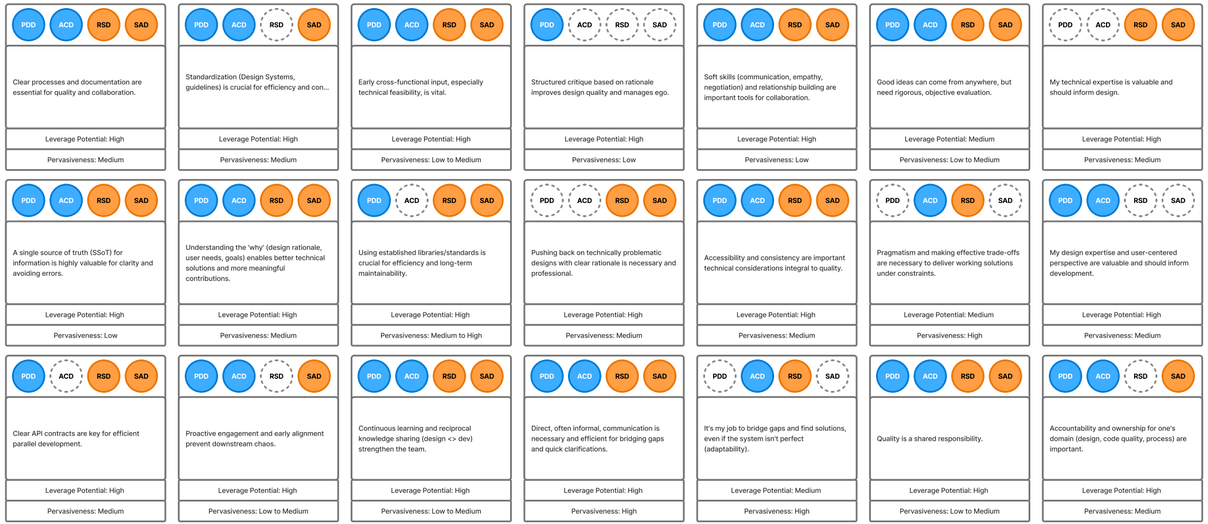


Important

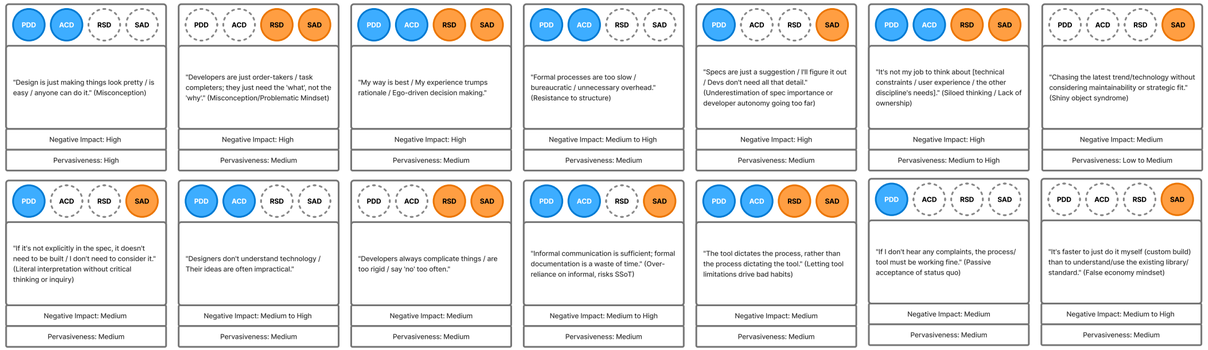


Good to have

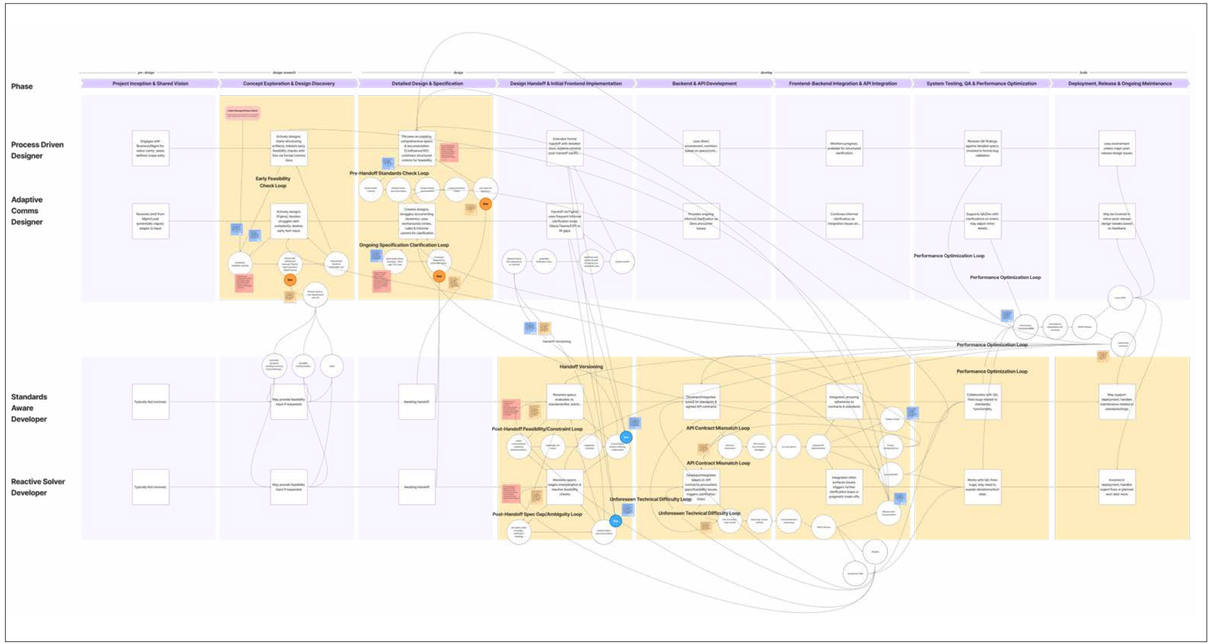
Classification of Needs and Pain Points based on Survey



Shared Productive Attitudes & Beliefs

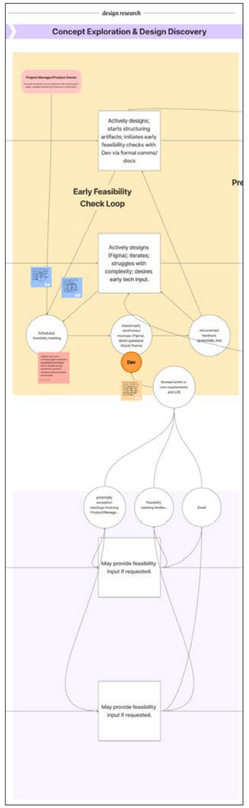


Shared Problematic Attitudes & Beliefs



Journey mapping of Archetypes

Link to journey map



CRITICAL Pain Points

- Receiving Designs with Insufficient Technical Awareness or Detail +3
- Inefficient Rework Cycles Caused by Late Discovery of Feasibility Issues +3
- Bearing the Burden of Feasibility Assessment Post-Handoff +3

CRITICAL Functional Needs

- Mechanisms for Early & Continuous Cross-Functional Collaboration +2
- Mechanisms for Early & Quick (often informal) Technical Input +2
- Designers Receptive to Technical Feedback (facilitated by this loop) +2

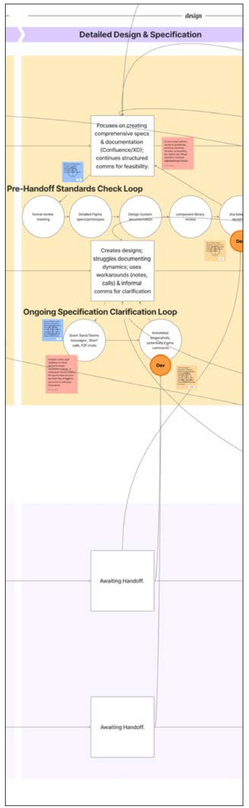
CRITICAL Emotional Needs

- Trust in Process & Predictability +1
- Reduced Frustration from Rework and Late Discovery +1
- Feeling Respected for Expertise (both designer showing work early, dev giving input) +1

Frequency - 3 Points total

Total Point = 21

Early Feasibility Check



CRITICAL Pain Points

- Inconsistency from Lack of Standards/Process Adherence +3
- Receiving Designs Incompatible with Established Libraries/Tech Constraints +3

CRITICAL Functional Needs

- Clear, Comprehensive, Actionable Design Specifications (re: standards) +2
- Clear Guidelines on Library Usage vs. Custom Components +2
- Access to Centralized, Enforced Design Guidelines/Systems (loop ensures adherence) +2

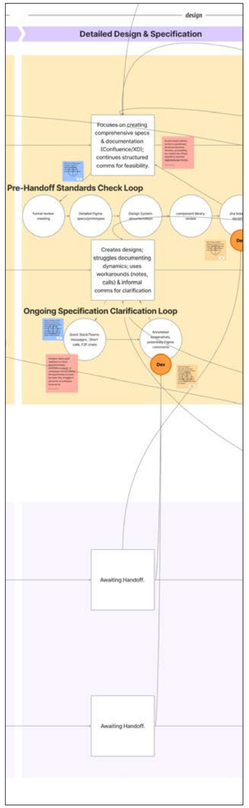
CRITICAL Emotional Needs

- Confidence in the Technical Soundness & Consistency of Designs +1
- Trust in Process & Predictability +1

Frequency - 3 Points total

Total Point = 17

Pre-Handoff Standards Check



CRITICAL Pain Points - 3 Points each

- Inconsistency from Lack of Standards/Process Adherence +3
- Receiving Designs Incompatible with Established Libraries/Tech Constraints +1

CRITICAL Functional Needs - 2 Points each

- Reliable and Efficient Channels for Seeking Clarification from Designers +2
- Clear, Comprehensive, Actionable Design Specifications (its absence triggers this) +2

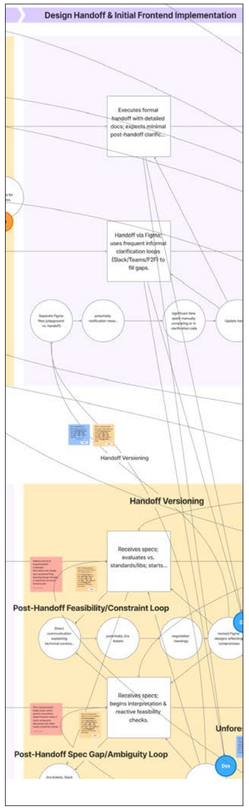
CRITICAL Emotional Needs - 2 Points each

- Reduced Frustration from...Ambiguity, Rework +1

Frequency - 5 Points total

Total Point = 14

Ongoing Specification Clarification



CRITICAL Pain Points

- Ambiguity in Specifications Leading to Repeated Clarification & Assumptions +3
- Inefficiencies from Poor Handoffs, Late Changes, Unclear Communication +3
- Receiving Designs with Insufficient Technical Awareness or Detail +3

CRITICAL Functional Needs

- Clear, Comprehensive, Actionable Design Specifications (its absence is the cause) +2
- Reliable and Efficient Channels for Seeking Clarification from Designers +2

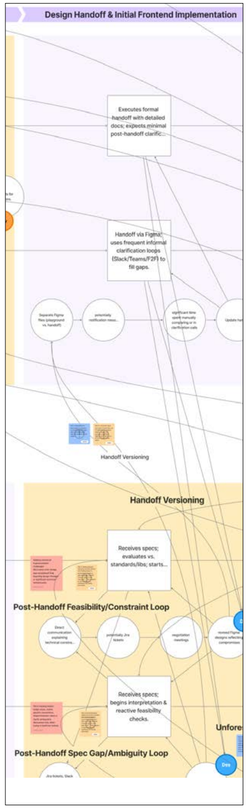
CRITICAL Emotional Needs

- Reduced Frustration from Ambiguity, Rework, and Late Discovery +1
- Trust in Process & Predictability (eroded by this loop) +1

Frequency - 5 Points total

Total Point = 20

Post-Handoff Spec Gap



CRITICAL Pain Points

- Receiving Designs Incompatible with Established Libraries/Tech Constraints +3
- Bearing the Burden of Feasibility Assessment and Making Critical Trade-offs Post-Handoff +3
- Inefficient Rework Cycles Caused by Late Discovery of Feasibility Issues +3
- Disruptive Mid-Sprint/Late Design Changes Leading to Rework (often a result) +3

CRITICAL Functional Needs

- Mechanisms for Early & Continuous Cross-Functional Collaboration (its absence leads here) +2
- Designers Receptive to Technical Feedback (essential for resolution) +2

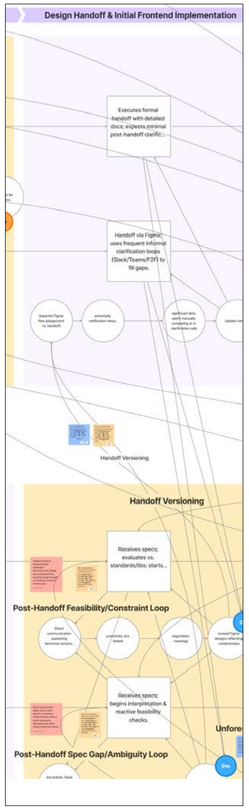
CRITICAL Emotional Needs

- Reduced Frustration from Ambiguity, Rework, and Late Discovery +1
- Trust in Process & Predictability (severely impacted) +1
- Feeling Respected for Expertise (for devs raising major issues) +1

Frequency - 3 Points total

Total Point = 22

Post-Handoff Feasibility



CRITICAL Pain Points

- Inefficiencies from Poor Handoffs, Late Changes, Unclear Communication +3
- Difficulty Scaling Manual/ Inefficient Processes (this loop is often manual/inefficient) +3
- Disruptive Mid-Sprint/Late Design Changes Leading to Rework (if change comms are poor) +3

CRITICAL Functional Needs

- Reliable System for SSoT & Version Control +2
- Auditable Trail for Design Decisions & Changes +2

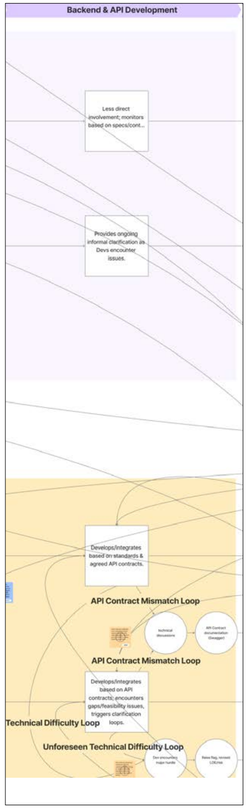
CRITICAL Emotional Needs

- Trust in Process & Predictability +1
- Reduced Chaos and Fire-Fighting +1
- Reduced Frustration from Tool/ Process Limitations +1

Frequency - 5 Points total

Total Point = 21

Handoff Versioning



CRITICAL Pain Points

- Inefficiencies from Unclear Communication (between FE/ BE if contract is issue) +1
- Disruptive Mid-Sprint/Late Design Changes (if design must change due to API) +3

CRITICAL Functional Needs

- Clear API Contracts for Efficient Parallel Development +2

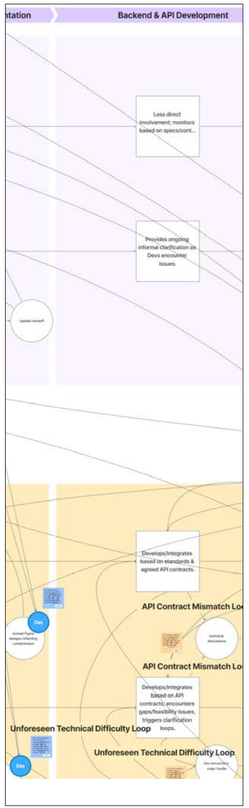
CRITICAL Emotional Needs

- Trust in Process & Predictability (between FE/BE teams) +1
- Reduced Frustration... +1

Frequency - 3 Points total

Total Point = 11

API Contract Mismatch



CRITICAL Pain Points

Disruptive Mid-Sprint/Late Design Changes (if design must adapt) +3

Inefficient Rework Cycles... (if difficulty causes rework) +1

CRITICAL Functional Needs

CRITICAL Emotional Needs

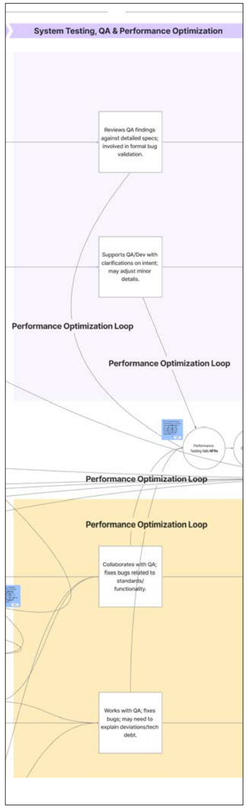
Reduced Frustration... +1

Feeling Empowered to Make Pragmatic Decisions (for dev solving it) +1

Frequency - 1 Points total

Total Point = 7

Unforeseen Technical Difficulty



CRITICAL Pain Points

Disruptive Mid-Sprint/Late Design Changes (if design is root cause of poor performance) +3

CRITICAL Functional Needs

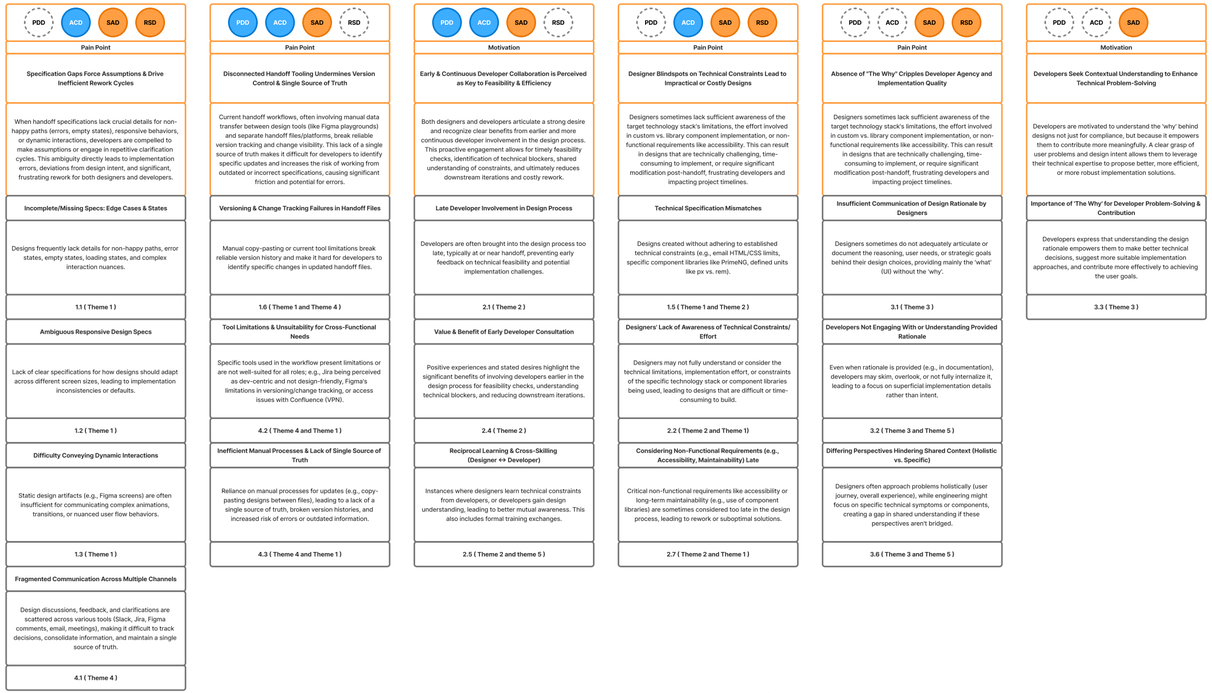
CRITICAL Emotional Needs

Reduced Frustration... +1

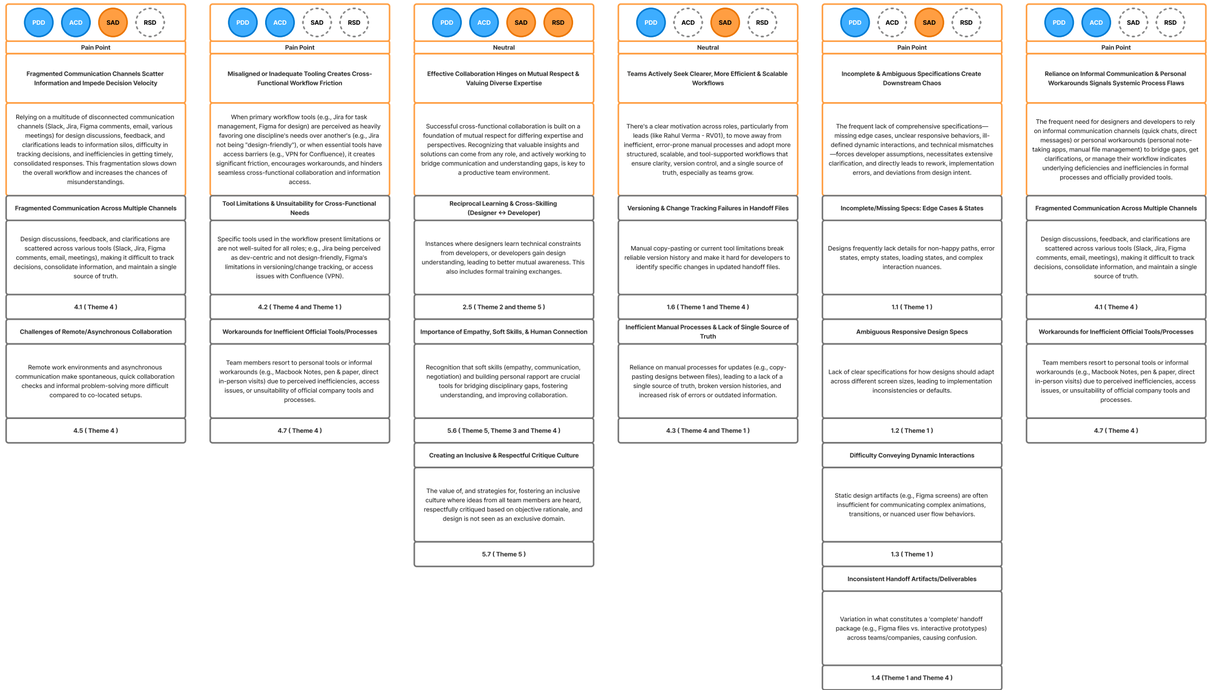
Frequency - 1 Points total

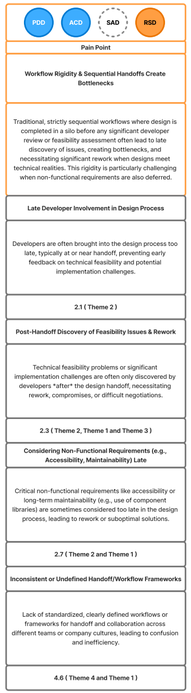
Total Point = 5

Performance Optimization



Early Insights with Archetypes





Early Insights with Archetypes

Checking tech ideas late causes big delays, early talks are often missed but vital.

Missing design details (states, logic) causes confusion, rework, and bad user experiences

No single official design version causes confusion, errors, and wasted time for everyone.

Devs build better when they know the 'Why' behind designs; this is often unclear.

Major Insights

1. How might we...

...make it easy for designers and developers to check tech ideas together, right from the start?

B2 "Constraint Cards" or "Technical Principles" Document for Designers

Developers (SAD/Tech Lead) collaboratively create and maintain a simple, living document or set of "constraint cards" (physical or digital) that outline key, overarching technical principles, known limitations of the current stack, or "no-go" areas for design to consult before even starting detailed concepts.

Process & Workflow

C2 Figma Plugin for "Request Tech Review" / Flagging Uncertainty

A Figma plugin where a designer can flag a specific frame or component in their early-stage design and add a comment like "Need tech feasibility check on this interaction" which then pings relevant developers or creates a small task in a linked system (like a lightweight Jira board for these checks).

Tools & Artifacts

C6 "Feasibility Scorecard" Template

A simple, standardized scorecard template (digital or even a mental model) that developers (SAD/RSD) can use to quickly evaluate early design concepts against key technical criteria (e.g., "Uses Standard Components: Y/N," "Estimated Effort: Low/Med/High," "Known Performance Risks: Y/N," "Accessibility Concerns: Y/N"). Designers get consistent feedback points.

Tools & Artifacts

A6 "Design Buddy" System with Devs

Pair each designer (or a small design pod) with a specific developer "buddy" (ideally an SAD or experienced RSD interested in early collaboration) for the duration of a project or major feature. The dev buddy is the first point of contact for informal technical questions and participates in more frequent, brief design check-ins.

People & Collaboration

C19 "Sandboxed Dev Environments" for Designers

Provide designers with extremely simplified, sandboxed development environments or low-code or one storybook platforms where they can play with actual front-end components or basic API interactions to get a tactile feel for technical constraints and possibilities without needing to code proficiently.

Tools & Artifacts

B4 "Living" Technical Constraint Document Linked in Design System

For every major component or pattern in the Design System, include a "Technical Considerations" section maintained by developers (SADs primarily). This section would highlight known performance caveats, common implementation challenges, or best-practice usage from a technical perspective, directly accessible to designers using the system.

Process & Workflow

C10 "What if...?" Scenario Building Tool

A simple template or digital tool where designers can outline a core user interaction and then, alongside developers, quickly list out potential "What if...?" technical scenarios (e.g., "What if API is slow?", "What if user has no network?", "What if this component fails to load?"). This surfaces edge cases and constraints early.

Tools & Artifacts

C7 AI-Assisted "Constraint Checker" for Early Designs

An AI tool (potentially a Figma plugin) trained on the organization's tech stack, design system, and past feasibility issues. Designers could run early concepts through it to get an initial, automated flag for potential conflicts with known technical constraints or overly complex patterns. This would not replace dev consultation but could act as a first pass.

Tools & Artifacts

A10 "Reverse Shadowing" - Designers Shadowing Devs

Designers spend a short amount of time (e.g., a few hours per quarter) "shadowing" developers to understand their workflow, the tools they use for implementation, common technical hurdles they face, and how design decisions translate into code.

People & Collaboration

C20 "Automated Documentation of Feasibility Decisions"

When a feasibility discussion happens (e.g., in a specific Slack channel C1, or a Figma plugin C2), have a bot or integration that prompts participants to summarize the key constraint discussed and the decision made, then automatically logs this to a central "Feasibility Decision Register"

Tools & Artifacts

C24 Gamified "Tech Constraint Quiz" for Designers

A lighthearted, optional, gamified quiz or learning module for designers (ACD/PDD) that periodically tests their knowledge of common or recently updated technical constraints or design system capabilities relevant to their projects. Not for performance review, but for fun, self-paced learning.

Tools & Artifacts

C21 "Shared Glossary of Design ↔ Technical Terms with Visual Examples"

A collaboratively maintained glossary that defines common design terms for developers and common technical terms/constraints for designers, ideally with simple visual examples or analogies for each to bridge the vocabulary gap.

Tools & Artifacts

C22 "Feasibility Consideration Score" within Design Tool

An AI tool (potentially a Figma plugin) trained on the organization's tech stack, design system, and past feasibility issues. Designers could run early concepts through it to get an initial, automated flag for potential conflicts with known technical constraints or overly complex patterns. This would not replace dev consultation but could act as a first pass.

Tools & Artifacts

A23 "Joint KPI Setting" for Design & Development on Feature Success

Design and Development leads collaboratively define not just product success KPIs, but also internal process KPIs related to successful early collaboration (e.g., "number of major feasibility issues caught pre-handoff," "percentage of stories requiring no post-handoff spec changes due to tech feasibility").

People & Collaboration

A28 "Feasibility Facilitator" Role

Introduce a dedicated (or rotational, senior team member) "Feasibility Facilitator" role. This person is not necessarily the designer or the dev but is skilled in bridging design intent with technical possibilities. They would proactively ensure the right conversations happen at the right time and help translate between disciplines during early stages.

People & Collaboration

2. How might we...

...help designers easily detail all design parts (states, logic, responsive) consistently and clearly?

C27 "State & Interaction Libraries" in Figma

Beyond basic components, create pre-defined common interaction patterns (e.g., standard modal flows, data table interactions with all states, form validation patterns) as library elements that designers can pull from, ensuring completeness and consistency.

Tools & Artifacts

C27 "State & Interaction Libraries" in Figma

Beyond basic components, create pre-defined common interaction patterns (e.g., standard modal flows, data table interactions with all states, form validation patterns) as library elements that designers can pull from, ensuring completeness and consistency.

Tools & Artifacts

C29 Automated Spec Annotation Tools (AI-assisted)

Explore AI tools that can analyze a design in Figma and automatically suggest or generate initial annotations for states, spacing, typography, or even identify potentially missing edge cases based on common patterns. Designer then reviews and refines.

Tools & Artifacts

C30 "Specification Dashboard" for Project Overview

A simple dashboard (e.g., in Confluence or a dedicated tool) that tracks the completion status of key specification elements (states, responsive, dynamic interactions) for each major feature or screen in a project. Provides visibility to PMs, Designers, and Dev Leads.

Tools & Artifacts

C32 "Micro-Interaction Prototyping Snippets" Repository

For common but complex micro-interactions (e.g., a specific type of animated reveal, a multi-step input validation flow), create a library of very small, isolated, interactive prototypes (in Framer, ProtoPie, or even advanced Figma prototypes) that designers can reference or link to in their main specs to show exact intended dynamic behavior.

Tools & Artifacts

C34 Visual "State Matrix" Generator

A tool or Figma plugin that helps designers quickly generate a visual matrix of all possible states for complex components (e.g., a button with primary/secondary, default/hover/active/disabled/loading variations). This ensures no state is overlooked.

Tools & Artifacts

C33 "Dynamic Spec Comments" in Figma

A Figma plugin or convention where designers can attach short screen recordings (e.g., Loom, or native if Figma develops it) directly to components or frames to demonstrate a dynamic interaction or state change, rather than just relying on text annotations.

Tools & Artifacts

C38 "Spec Completeness Score" AI Feature

An AI tool (extending C29/C34) that not only flags potential issues but also gives a "completeness score" for a design against the defined spec framework (e.g., "85% of required states defined," "Responsive notes missing for 3 components"). Helps designers identify gaps before handoff.

Tools & Artifacts

C40 "Visual Diffing Tool" for Specifications

Explore or advocate for a tool that can visually "diff" two versions of a detailed specification document (could be exported Figma frames or a Confluence page structure) and clearly highlight textual and visual changes relevant to developers, going beyond what native Figma compare might offer for non-Figma outputs.

Tools & Artifacts

C41 Centralized "Spec Issues & Resolutions" Knowledge Base

A searchable knowledge base where common specification gaps or ambiguities previously encountered (from Loop 4 incidents) and their resolutions/clarifications are documented. Designers can consult this to proactively avoid repeating common spec mistakes.

Tools & Artifacts

B22 "Progressive Disclosure" Specification Framework

- Level 1: High-level user flow, key screens, core happy path (for very early dev consults or MVS).
- Level 2: Adds major states, responsive considerations, key interactions.
- Level 3: All edge cases, detailed animation specs, accessibility notes.

Designers progressively complete these layers, and handoffs can occur at different levels of detail depending on project needs

Process & Workflow

B24 Automated Reminders/Nudges for Spec Framework Adherence

A Figma plugin or convention where designers can attach short screen recordings (e.g., Loom, or native if Figma develops it) directly to components or frames to demonstrate a dynamic interaction or state change, rather than just relying on text annotations.

Process & Workflow

C46 "Printable Spec Framework 'Cheat Sheet' Poster"

A visually appealing, concise A3 poster that summarizes the key elements of the specification framework. Designers (and interested devs) can have it on their wall as a quick physical reminder of what needs to be covered.

Tools & Artifacts

A48 "Spec Framework 'Escape Room' Challenge"

A team-building and learning activity where designers (and maybe devs) have to solve a "puzzle" (e.g., figure out how to design a complex feature to launch successfully) by correctly using elements of the specification framework to find clues or unlock the next steps. Makes learning the framework engaging.

Tools & Artifacts

C48 Voice-Annotated Specifications

Designers use tools to add short voice notes directly onto Figma frames or spec document sections, verbally explaining a complex interaction, a tricky edge case, or the rationale for a particular state. This supplements written text for a richer, more personal explanation.

Tools & Artifacts

3. How might we...

...make sure everyone always knows which design is the latest official one and what changed?

A56 Regular "SSoT Health Check" Audits & Retrospectives

Periodically (e.g., monthly or post-project), the SSoT Czars or a small working group audit how well the SSoT and versioning system is being used, identify pain points or areas of non-compliance, and hold a retrospective to improve the process/tool usage.

People & Collaboration

B30 "One-Button Publish to SSoT" from Design Tools

Strive for a workflow where approved designs from Figma (or other primary design tools) can be "published" to the designated SSoT (e.g., a specific folder in cloud storage, a Zeplin project, a versioned Confluence space) with a single, controlled action, automatically creating a new version and triggering notifications. Discourages manual file copying.

Process & Workflow

B33 Automated "Stale Link" or "Outdated Version" Checker

A script or tool that periodically scans Jira tickets or Confluence pages for links to design files and flags any that point to known outdated versions or non-SSoT locations.

Process & Workflow

C57 Dedicated Design Handoff & Versioning Tools

Evaluate and implement a specialized tool designed for design handoff, version control, and dev-designer communication, which acts as the clear SSoT and provides robust change tracking.

Tools & Artifacts

C58 "SSoT Dashboard" for Project Design Status

A simple dashboard that shows, for each active project/feature, the link to the current SSoT design file, its version number, last update date, and a link to the latest change summary.

Tools & Artifacts

C58 Visual "Diffing" Tools for Design Specifications

Implement or integrate tools that can visually compare two versions of a Figma file (or exported specs) and highlight exactly what has changed (pixels, layers, properties, text), making it much easier for developers to see updates.

Tools & Artifacts

B36 Bi-Directional Linking between SSoT Design & Jira Tasks

Ensure that not only Jira tasks link to the SSoT design, but the SSoT design artifact (e.g., specific Figma frame, Zeplin screen) also contains a link back to the relevant Jira task(s) it fulfills. This creates a closed loop for reference.

Process & Workflow

C62 "Figma SSoT Health Dashboard" Plugin

A Figma plugin that provides a dashboard for design leads showing:

- How many "Dev Ready" files haven't been updated/re-confirmed in X days.
- If any unmerged branches contain critical feature work.
- Links to external Jira tasks that might point to outdated Figma file versions (if an API existed for this).

Tools & Artifacts

C63 "Human-Readable Changelog Generator" from Figma Version History

A tool or script that parses Figma's detailed version history (or commit messages from branching) and generates a more concise, human-readable changelog summarizing key updates, specifically formatted for developer consumption (e.g., "Changes to: Login Screen (error states), Dashboard Widget X (responsive tablet)").

Tools & Artifacts

C65 Using Blockchain/Distributed Ledger for Design Version Provenance

For environments requiring absolute, irrefutable proof of design versions and changes (e.g., highly regulated industries, IP disputes), explore using distributed ledger technology to timestamp and secure design iterations. This is very out there for typical product design but explores ultimate SSoT integrity.

Tools & Artifacts

A61 "SSoT Scavenger Hunt" Onboarding for New Devs/Designers

As part of onboarding, new team members participate in a "scavenger hunt" where they have to find specific pieces of information or past design versions using only the official SSoT and version control tools. This playfully forces them to learn and use the system.

People & Collaboration

C67 Browser Extension to Flag Non-SSoT Design Links

A custom browser extension for team members that detects when they are viewing a design file (e.g., a Figma link) and cross-references it with a list of official SSoT project links. If it's not an official SSoT link (e.g., someone's personal draft), it displays a subtle warning banner.

Tools & Artifacts

C69 "Subscription Model" for Design Updates by Developers

Developers can "subscribe" to specific feature designs or component libraries within the SSoT platform. They automatically receive tailored notifications only for the changes relevant to the items they are subscribed to, reducing notification fatigue.

Tools & Artifacts

C70 "AI-Powered SSoT Query Bot"

A chatbot (in Slack/Teams) that developers can ask questions like, "What's the current SSoT link for the Profile Page?" or "When was the SSoT for Feature X last updated, and what changed?" The bot pulls info from the SSoT system.

Tools & Artifacts

C71 "Design SSoT Embeds" Directly in Developer IDEs

Imagine a plugin for a developer's IDE (like VS Code) that allows them to view relevant parts of the Figma SSoT (e.g., component specs, spacing tokens) directly within their coding environment, reducing context switching to check the SSoT.

Tools & Artifacts

4. How might we...

...make it simple for everyone to know and find the reasons behind our design choices?

B53 Linking Jira Epics/Stories/Tickets Directly to Research Findings & Goal Docs

Ensure every Jira Epic and key User Story contains direct links to the underlying user research reports, persona documents, or strategic briefs that define the "Why" for that piece of work.

Process & Workflow

C90 Figma Plugin/Annotations for "Why This Way?"

A Figma feature or plugin allowing designers to easily attach rich annotations or link to external rationale documents directly from specific UI elements or frames, explaining why it was designed that way. Developers can access this in dev mode.

Tools & Artifacts

C93 "Rationale Builder" Template/Tool

A template or simple tool that guides designers through articulating rationale by asking prompt questions: "What user problem does this solve? How does this align with Goal X? What alternatives were considered and why was this chosen?" AI could potentially summarize rationale from linked research.

Tools & Artifacts

A81 "Voice of the User" Regular Playbacks

Beyond initial research readouts, have short, regular sessions (e.g., bi-weekly) where relevant new user quotes, video clips from usability tests, or customer support feedback related to current or upcoming work are played back. Developers (SAD/RSD) are strongly encouraged to attend to keep the user's "Why" fresh.

People & Collaboration

B57 "Traceability Matrix" for Design Rationale

For complex projects, maintain a simple traceability matrix linking strategic goals → user needs/problems → key design principles → specific features/UI elements. This makes "The Why" traceable through layers.

Process & Workflow

C98 "Rationale Toggle" in Figma Dev Mode

In Figma's Dev Mode, provide a "Show/Hide Rationale" toggle. When enabled, it reveals specific annotations or linked documents containing the "Why" for selected components or frames, making it an opt-in layer of information for developers.

Tools & Artifacts

C100 "Team Value Proposition Canvas" Visible to All

Collaboratively create and keep highly visible (digitally or physically) a "Value Proposition Canvas" (or similar strategic framework) for the product/feature, clearly outlining customer jobs, pains, gains, and how the product delivers value. This is the overarching "Why."

Tools & Artifacts

Concept ID	Concept Title	p	AI?	Add new steps/ increase effort for Designers	Add new steps/ increase effort for Developers	Extends Core Tools	New Tool/Method Type	Visible Change in a Day of Designer	Visible Change in a Day of Developer	Does the concept propagate from users to org.
C20	"Automated Documentation of Feasibility Decisions"	0	0	0	0	1	1	1	1	
C27	"State & Interaction Libraries" in Figma	0	0	0	0	1	0	1	1	
C30	"Specification Dashboard" for Project Overview (Completeness Focus)	0	0	0	0	0	1	1	1	
C33	"Dynamic Spec Comments" in Figma (e.g., screen recordings)	0	0	0	0	1	1	1	1	
B30	"One-Button Publish to SSoT" from Design Tools	0	0	0	0	1	1	1	1	
B33	Automated "Stale Link" or "Outdated Version" Checker	0	1	0	0	1	1	1	1	
C67	Browser Extension to Flag Non-SSoT Design Links	0	0	0	0	1	1	1	1	
C69	"Subscription Model" for Design Updates by Developers	0	1	0	0	1	1	1	1	

Filtering out ideas

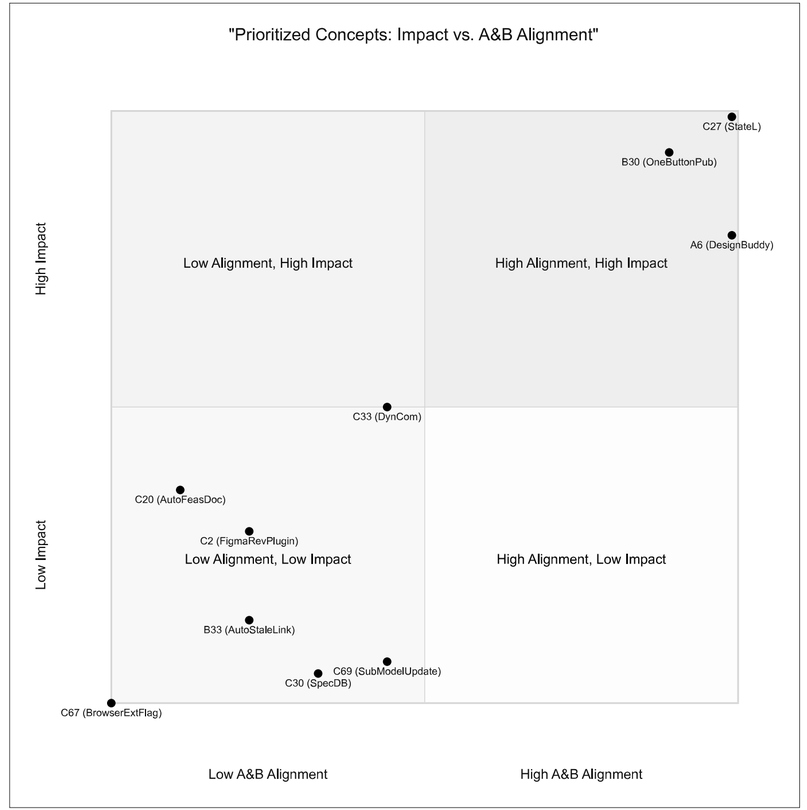
Should not add significant effort for designers and developers
Should create visible change in their workflows
Targeting Journey Map's loop >= 14
1=Yes , 0=No

Link to Sheet of Concept Evaluation, Impact Score + Alignment + Filtering

Impact Score: Sum of all pain points and needs targeted (Critical +3, Important +2, Good to have +1 each)
Alignment Score: Sum of all shared Attitude and Beliefs targeted(All 4 +4, Only 3 +3, Only 2 +2, Only 1 +1 each)

Normalized both Calculated values for 10 filtered down concepts and plotted on the graph

Prioritizing Ideas



Creating Solution

Selected Idea

C27: Ready-to-use design patterns in Figma for common interactions and all their states.

Concept Catalogue

What is it?

A Figma plugin duo: one helps designers quickly create all UI states (variants) using common patterns and check compliance; the other helps Design System managers review, integrate, and manage these submissions and new token requests.

What it does?

This suite accelerates how designers produce complete, consistent UI components. It streamlines checking designs against company standards and adding them to the official Design System, boosting quality and simplifying developer handoff.

How it works?

- Plugin 1 (Variant Generator - for PDD/ACD): Designers select common UI patterns to rapidly generate visual states. The plugin checks DS compliance and aids in requesting new tokens. A Designer Widget tracks requests and allows replies.
- Plugin 2 (Design System Integration - for DS Managers): DS Managers review submitted designs, focusing on flagged non-compliant items. They approve and integrate into the DS. A Manager Widget shows tasks and DS stats.

How it benefits?

- **Faster Variant Creation:** Pattern library speeds up defining states (benefits PDD, ACD).
- **Improved DS Compliance:** Early checks reduce non-standard designs (benefits PDD, SAD).
- **Streamlined Token Management:** Clear process for new token requests (benefits PDD, ACD, SAD).
- **Better DS Governance:** DS Managers have a dedicated review tool.
- **Clearer Handoffs:** Developers (SAD/RSD) get well-defined, DS-aligned components.
- **Reduced Rework:** Fewer compliance and spec gap issues caught late (benefits all).

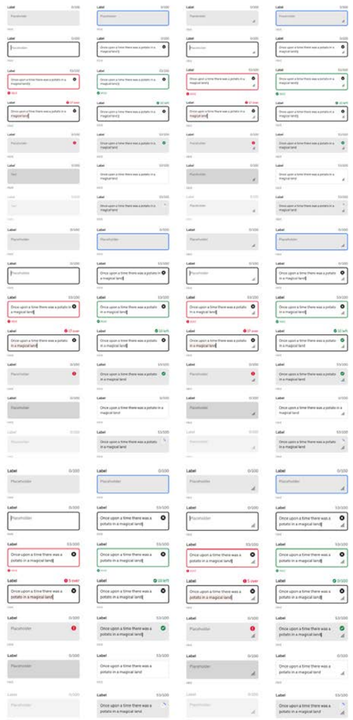
Roadblocks

- Significant plugin suite development effort.
- Requires curating a robust common UI patterns library.
- Needs well-defined, machine-readable Design System rules.
- Team workflow adoption and consistent usage.
- Ensuring widget performance and genuine utility.

Input →
Text area

1 Properties
State = 14 States

3 Properties
Size = 3
Platform = 2
State = 14 States



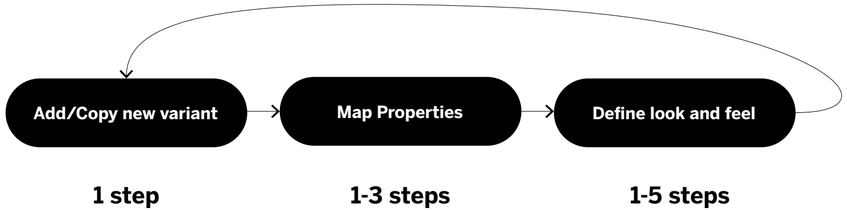
Input →
Text area

"So after creating the initial components of text area for mobile application and mapping all the states, the other designers might figure out requirements for another platform and 2 new different sizes for the same component"

70 new variants needed

Current task flow analysis - Extend DS

Missing any of the variant might cause red flag from developers, or even if using a custom values for colour, stoke width, etc...



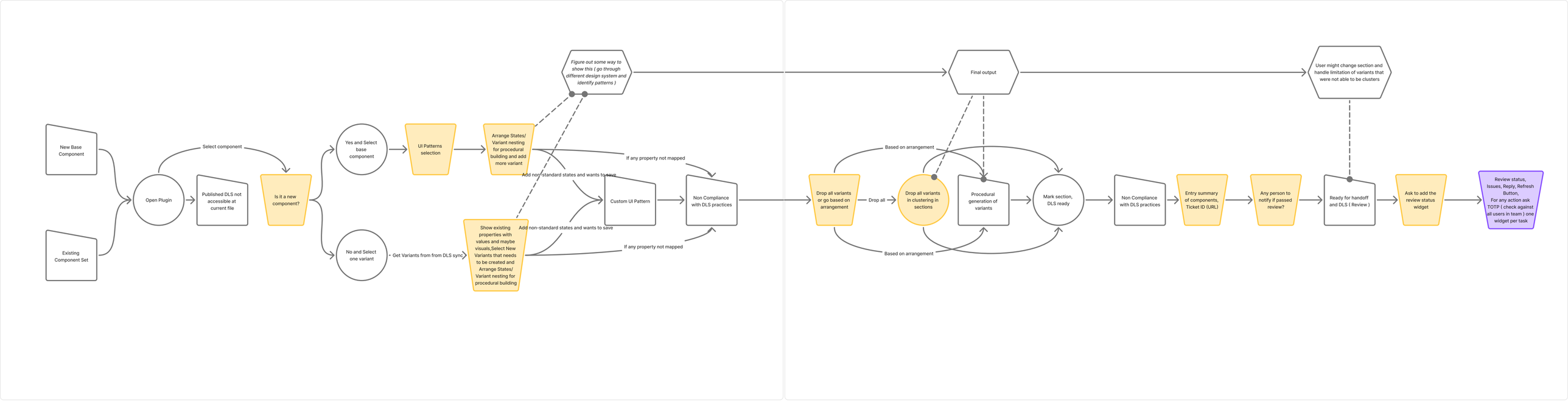
1 step

1-3 steps

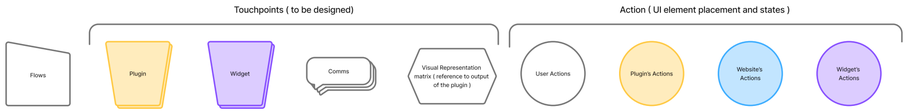
1-5 steps

Make sure to use design tokens, if any custom value is used need to add that to

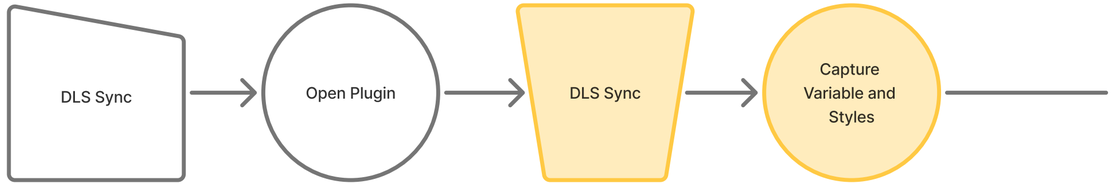
140 - 630 Steps needed

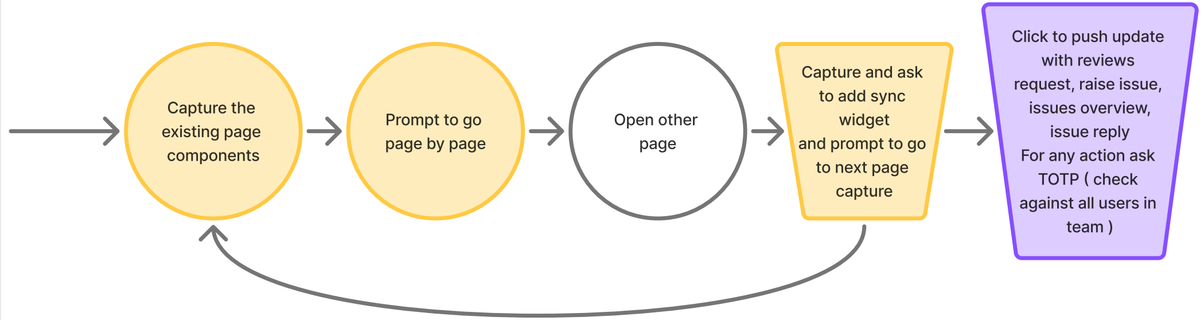


Task flow for Solution

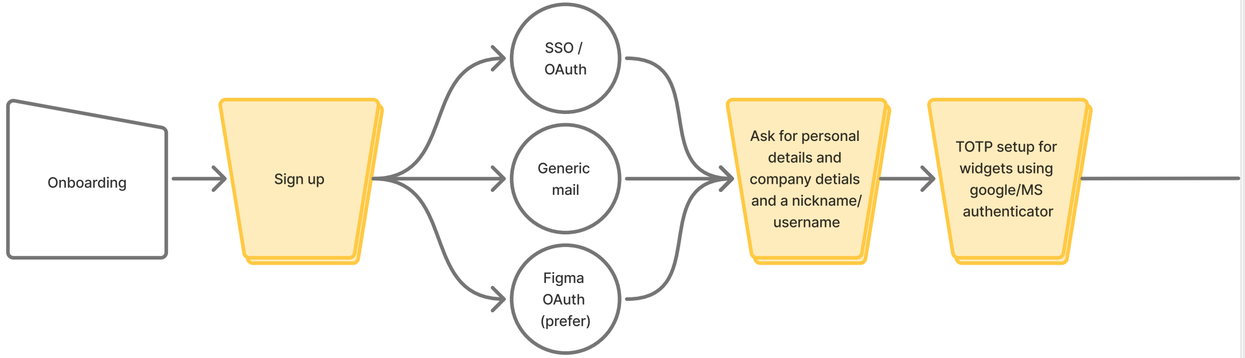


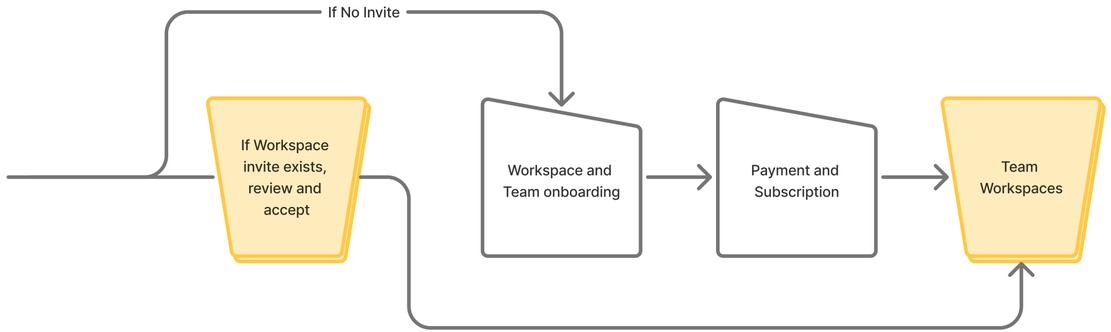
Create Component



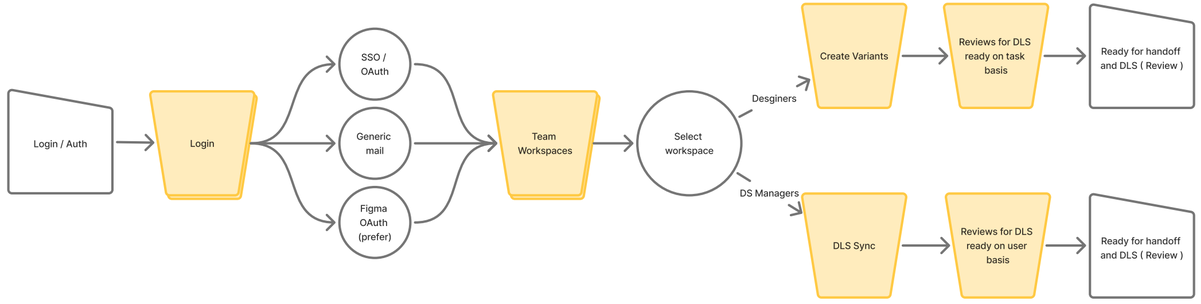


How the DLS will get Sync

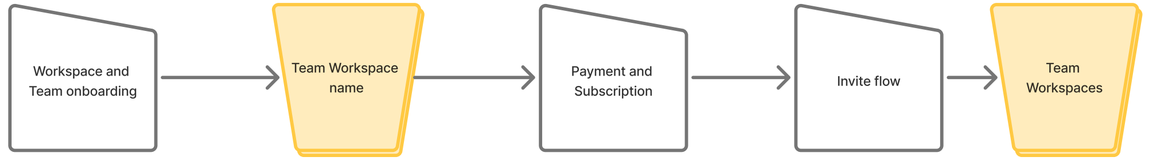




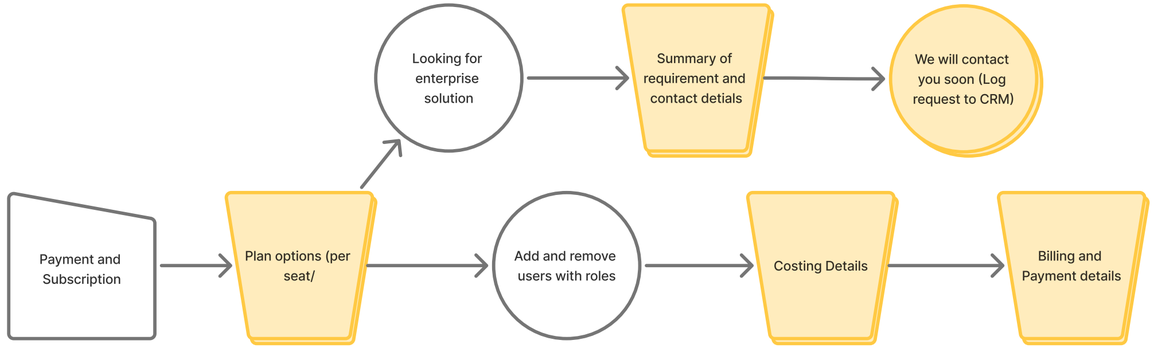
Onboarding Flow



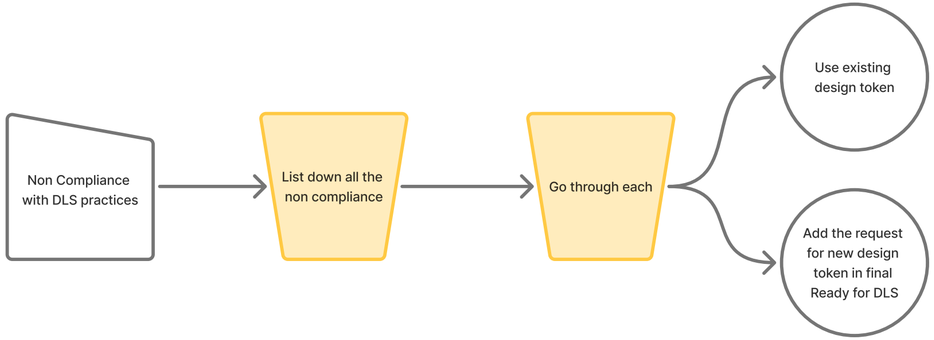
Login Flow



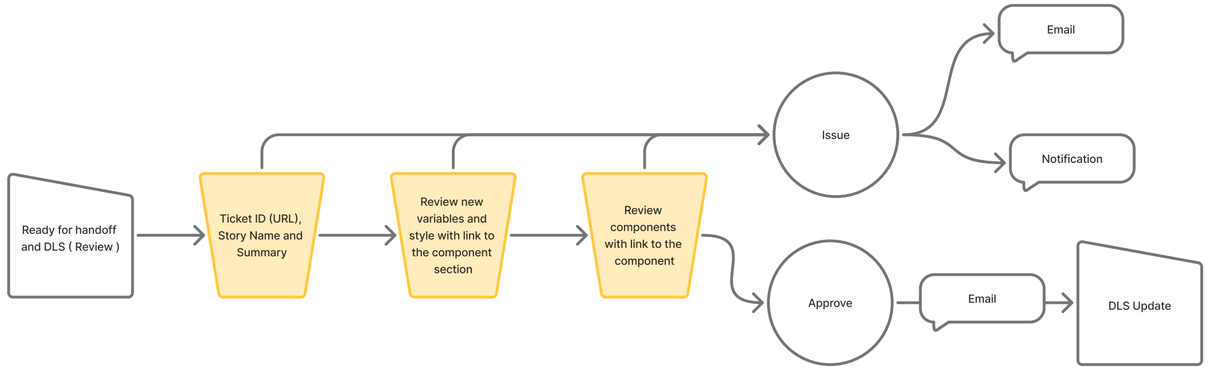
Workspace Creation Flow



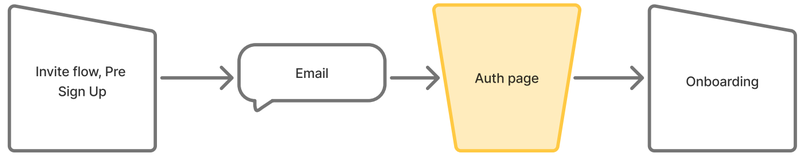
Payment Plan Setup



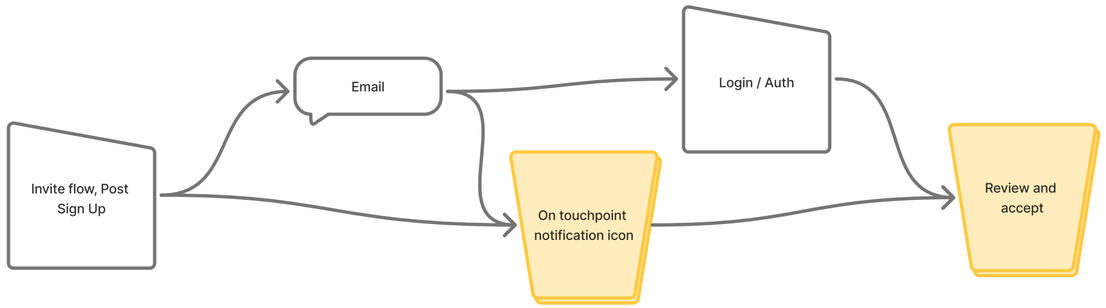
Design System Non Compliance



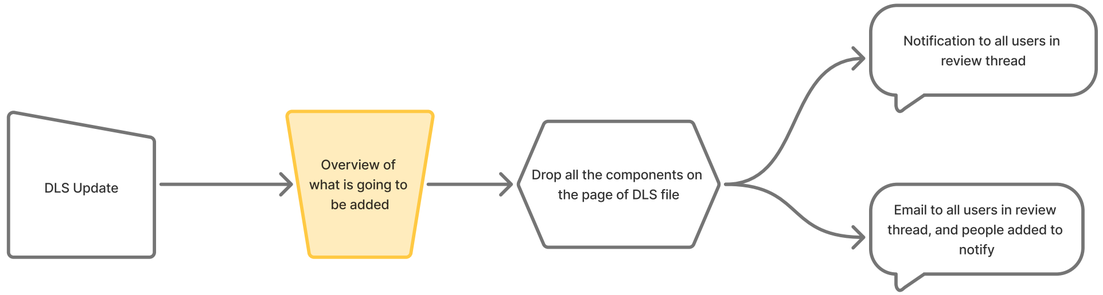
Ready for handoff and DLS (Review)



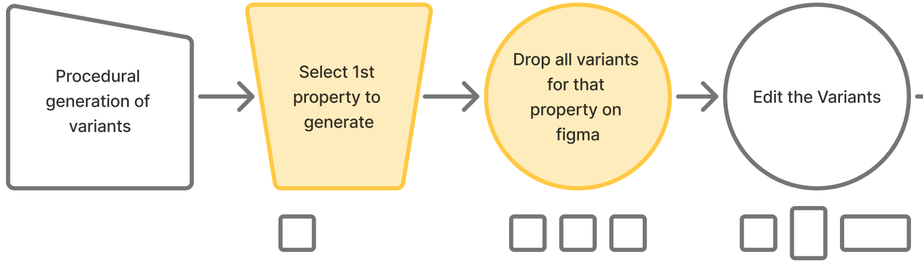
Invite Pre Sign-up



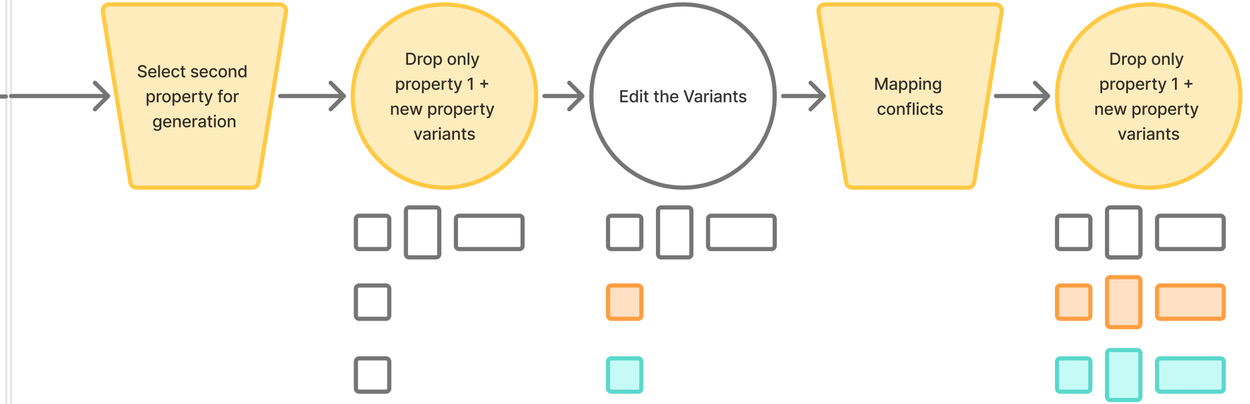
Invite Post Sign-up



Updating Variants in Design System

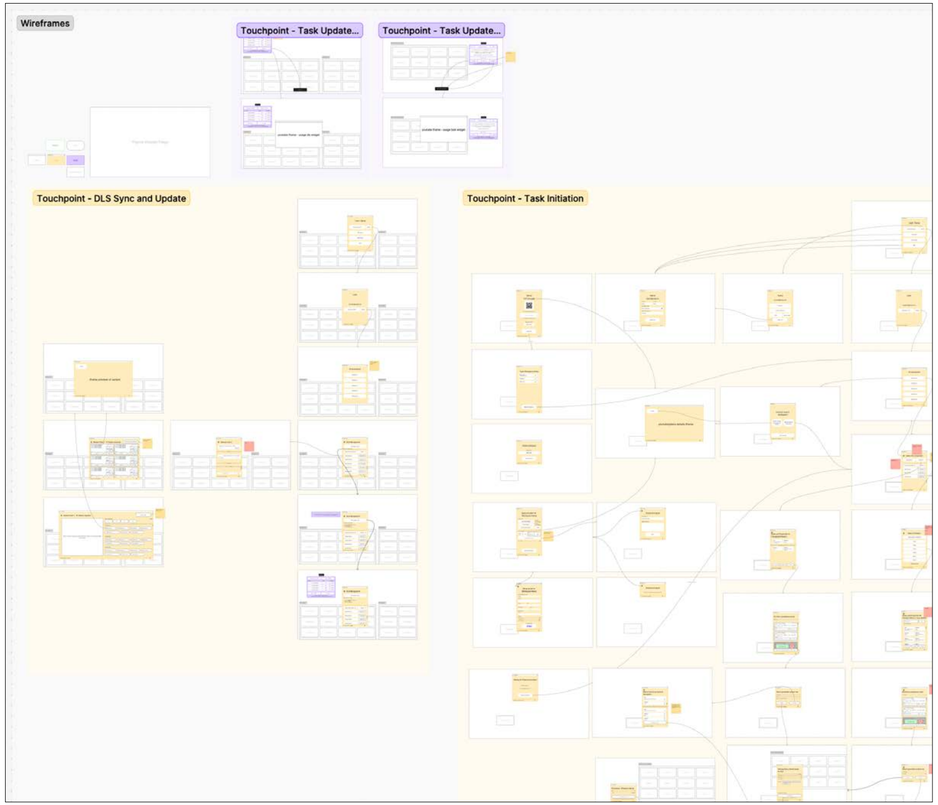


Variants Generation flow



Reduced to 14-32 steps

6 initial setup steps + 2-5, 1st property steps + 3 x (2-7 step for rest)
Exponential growth the more properties getting generated together and larger the existing component



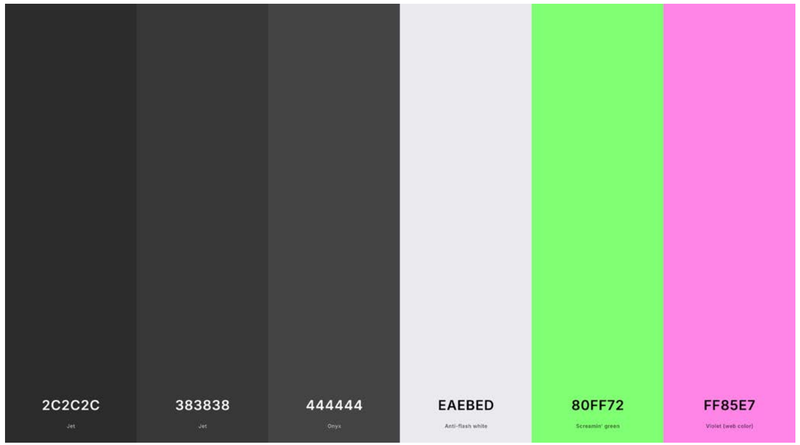
Wire Frames

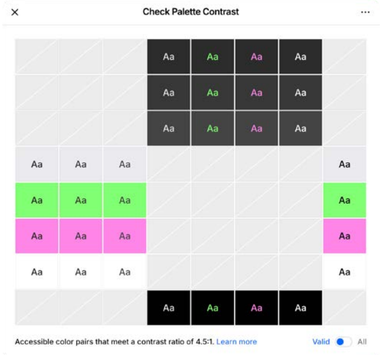
Feedback from testing with Visual Designers

Too much info on small window of plugin

DS Compliance flow and Mapping Conflict flow are not needed always

Brand Colour





Extending figma base colour to look more like same ecosystem

Variable Name	Value	Variable Name	Value	Variable Name	Value
Base		**Green**		**Pink**	
base-900	#2c2c2c	green-50	#f2fff1	pink-50	#fff3fd
base-800	#383838	green-100	#d8ffd3	pink-100	#ffd9f8
base-700	#444444	green-200	#c5ffbe	pink-200	#ffc7f4
base-300	#aeaeae	green-300	#aaffa1	pink-300	#ffadef
base-200	#d5d5d5	green-400	#99ff8e	pink-400	#ff9dec
base-100	#eaebed	green-500	#80ff72	pink-500	#ff85e7
base-600	#747474	green-600	#74e868	pink-600	#e879d2
		green-700	#5bb551	pink-700	#b55ea4
		green-800	#468c3f	pink-800	#8c497f
		green-900	#366b30	pink-900	#6b3861

Typography

Inter Medium
Inter Regular

Used Inter because the default font
available for the widgets in figma

Variant Companion

Variant Companion

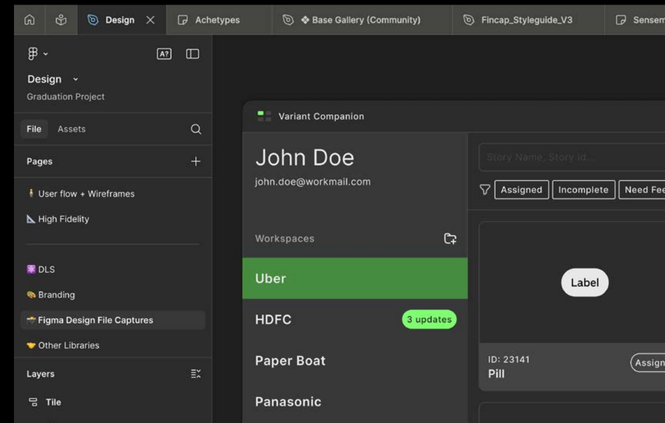
Variant Companion

System Companion

System Companion

System Companion

The grayed out boxes showcase the variants the will be generated from the plugin

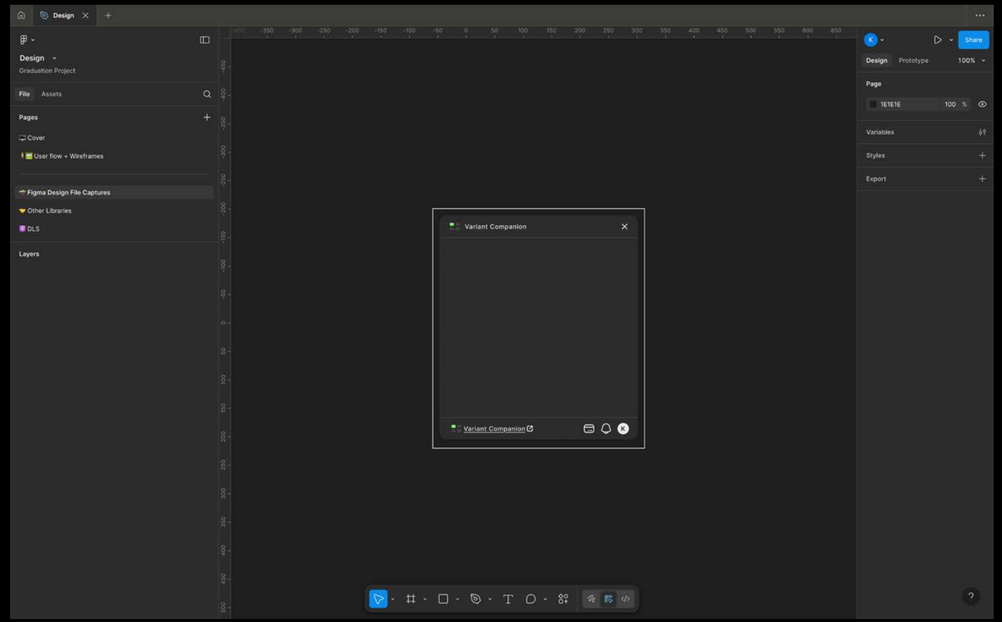


Logo + Name

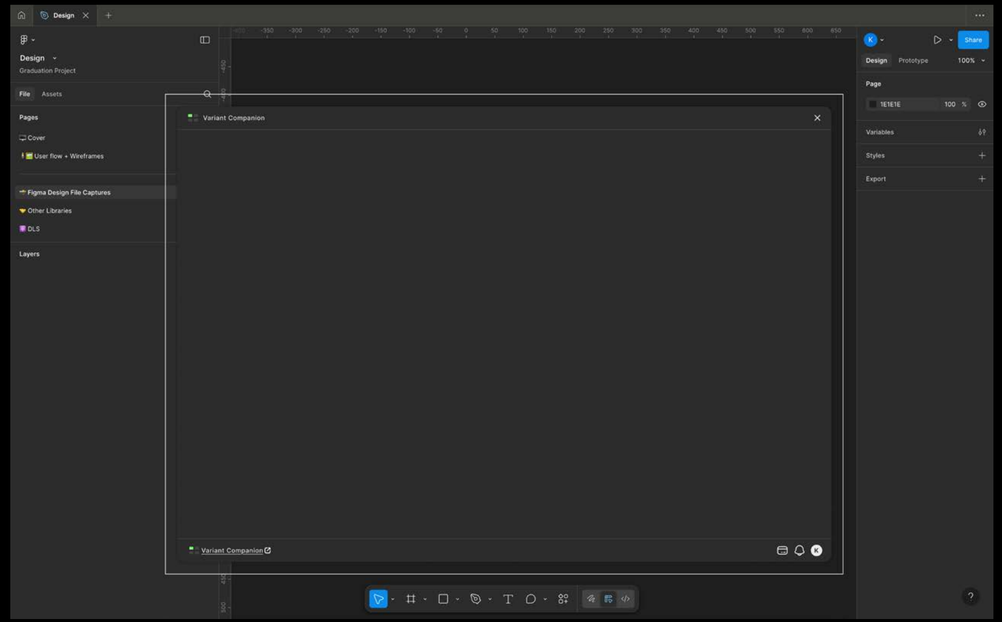
Design and Testing / Branding

Addressing “Too much info”

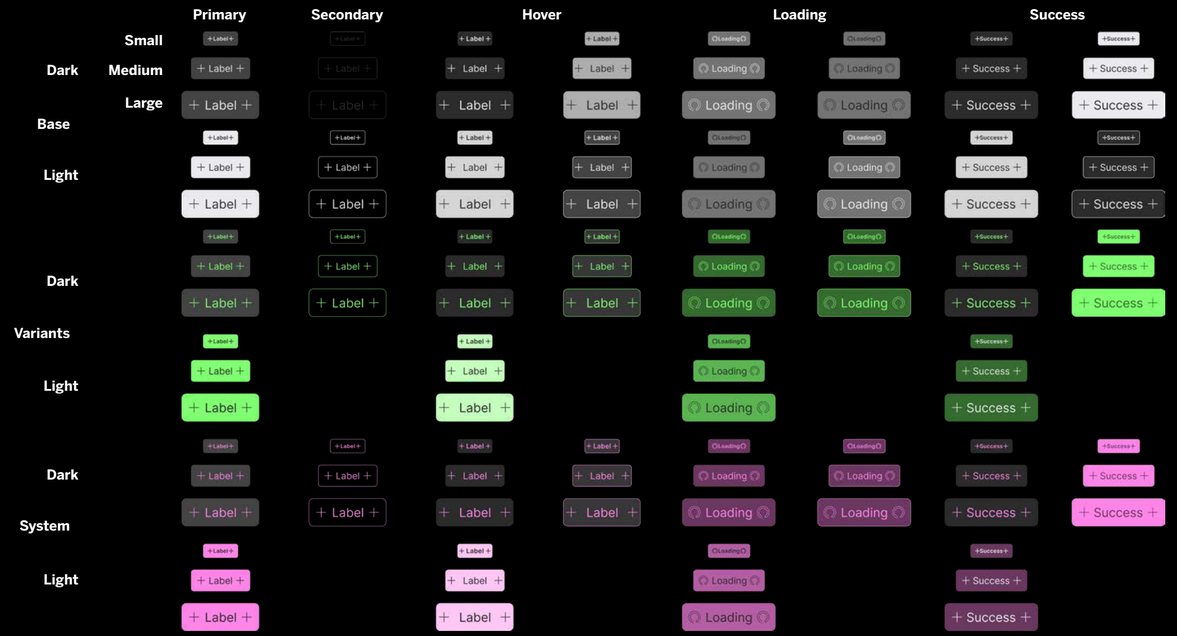
When action is needed in plugin/iframe
plugin size - 1150 x 800 - frame size, 1728 x 1117



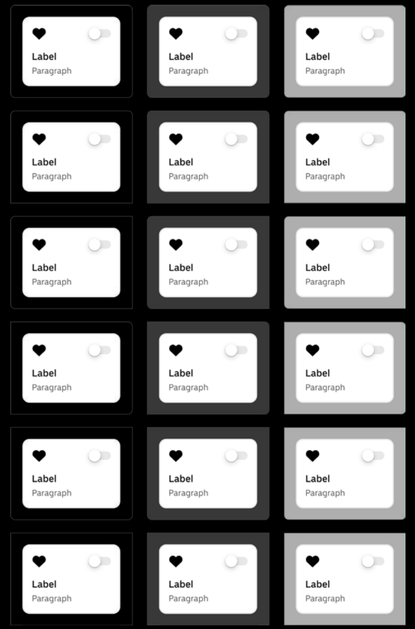
When action is needed at Figma file
plugin size - 373 x (400 to 800) - frame size, 1728 x 1117



Button Components



Component Renderer



Tags

Pills

Title

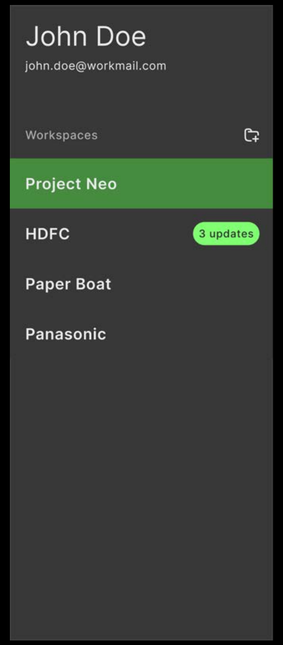
Label Label Label Label Label Label
Label Label Label Label Label

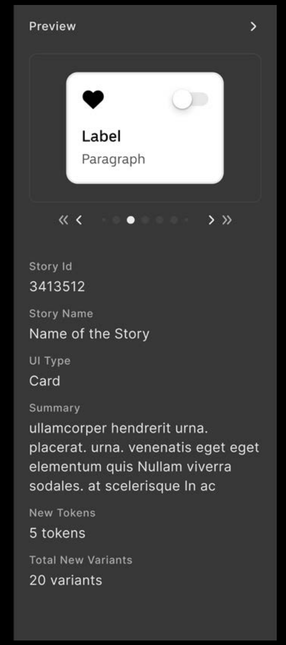
Title

Label Label Label Label Label Label
Label Label Label Label Label

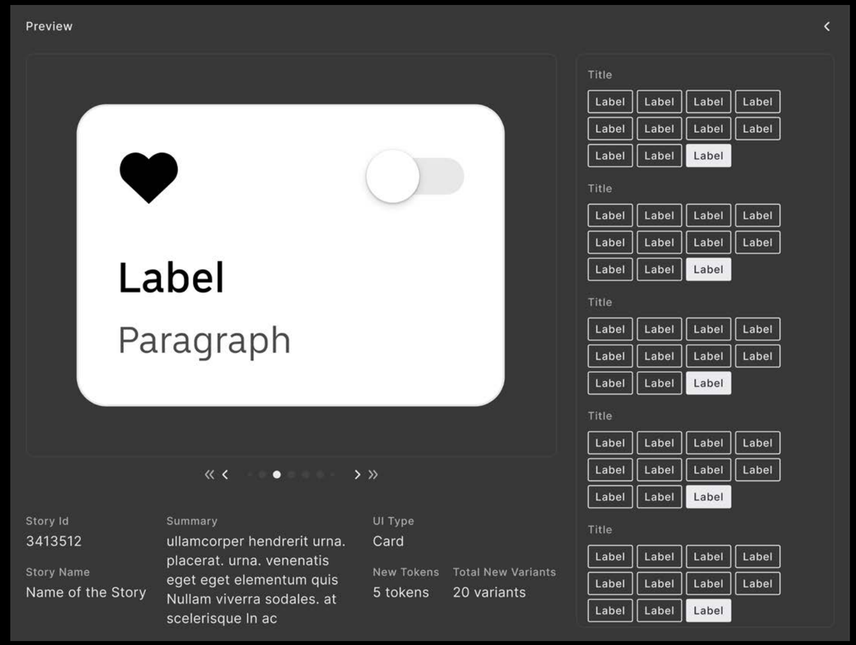
Title + Tags or Pills

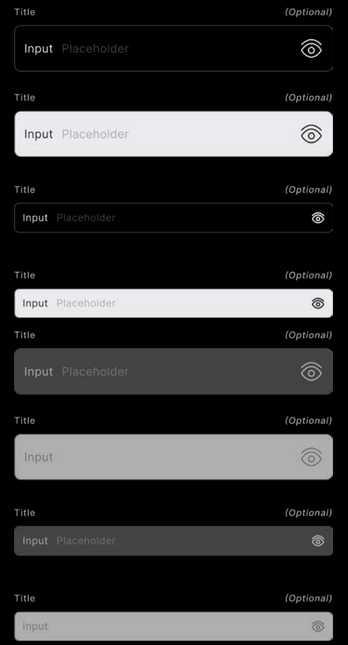
Preview Controller





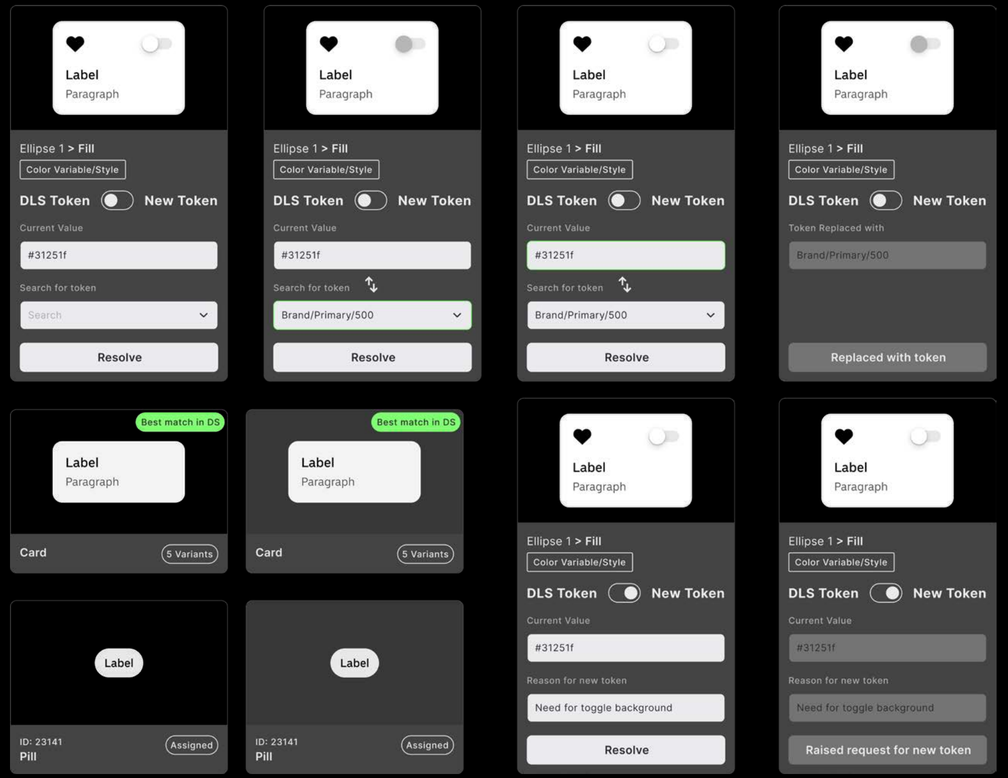
Side Nav



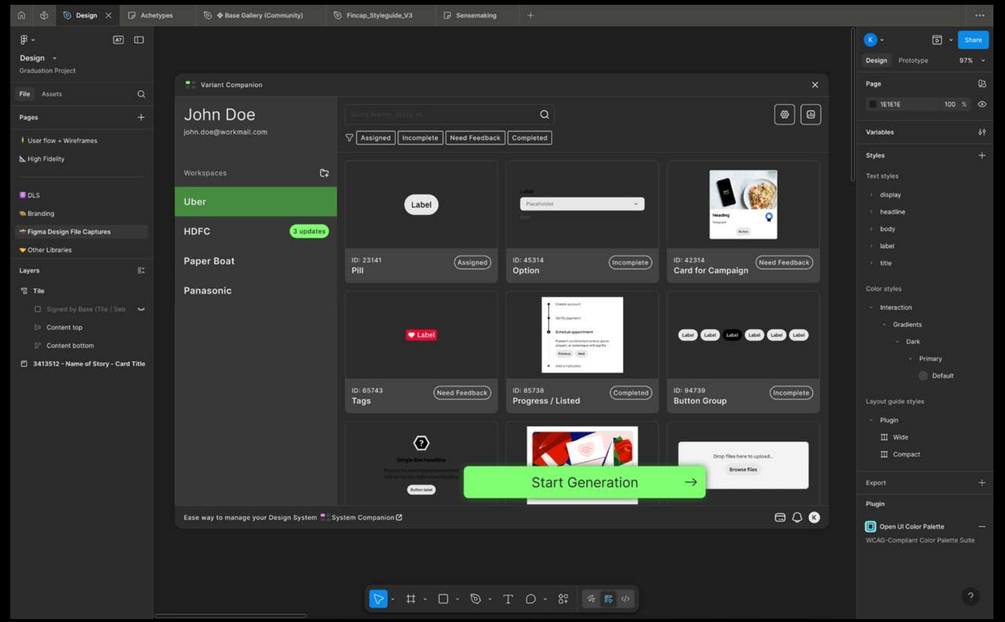


Input

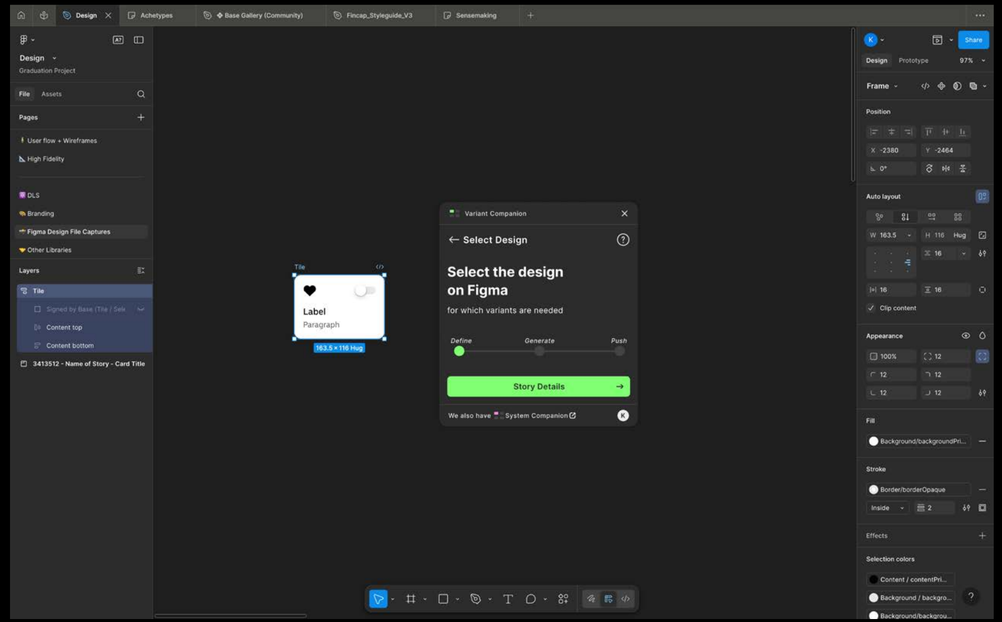
Cards



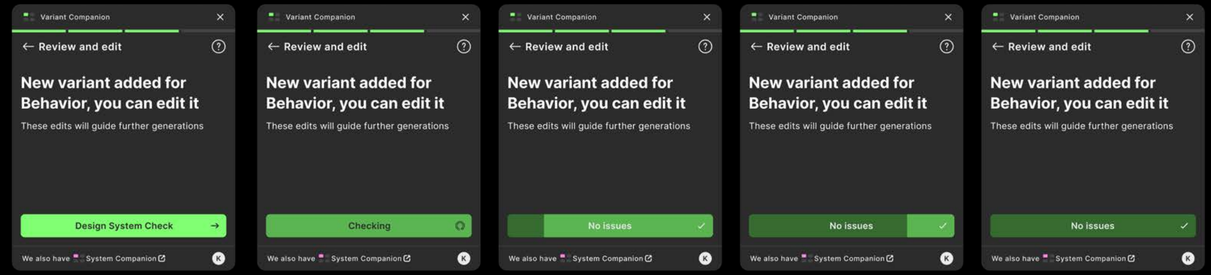
Workspace



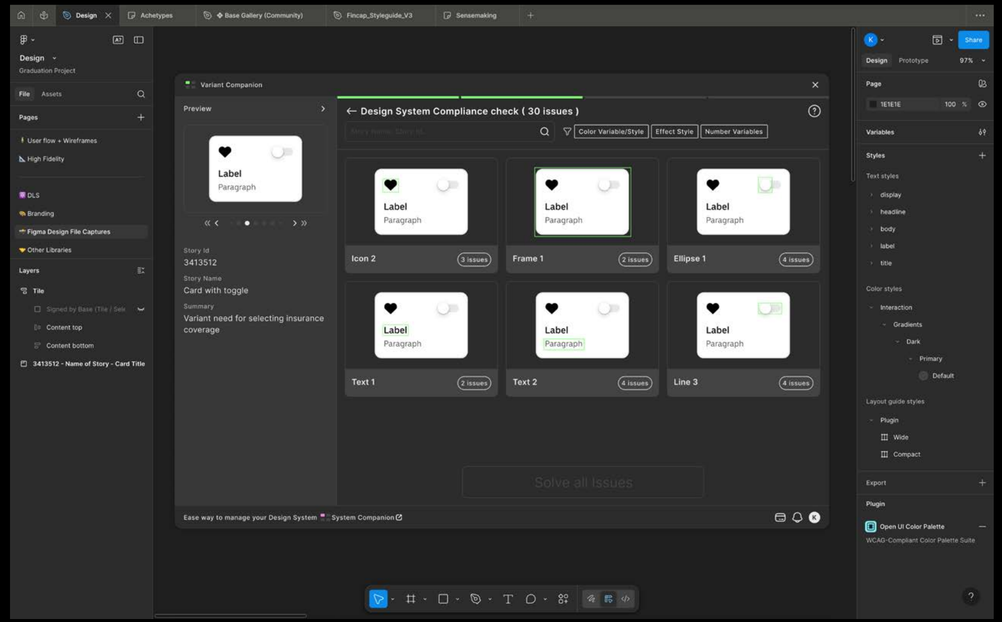
Select Component



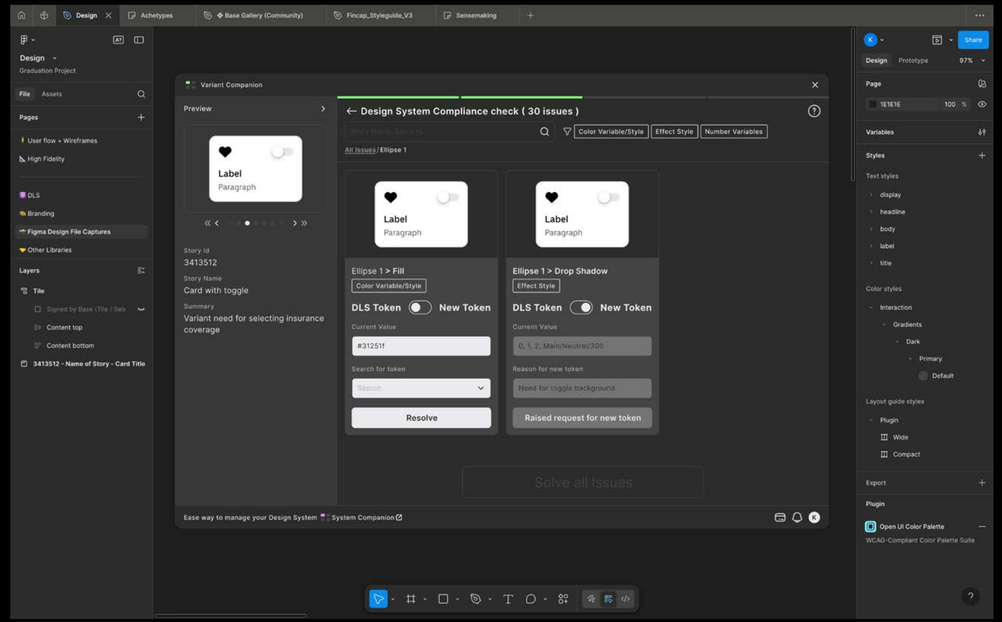
Skip compliance check and auto proceed, based on feedback



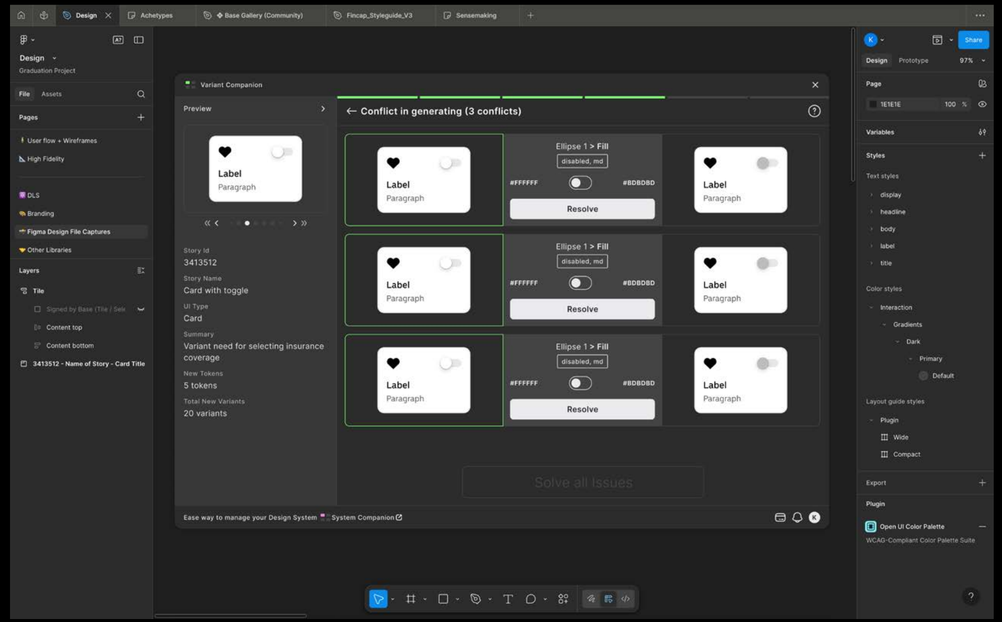
Design System Compliance (Nested)



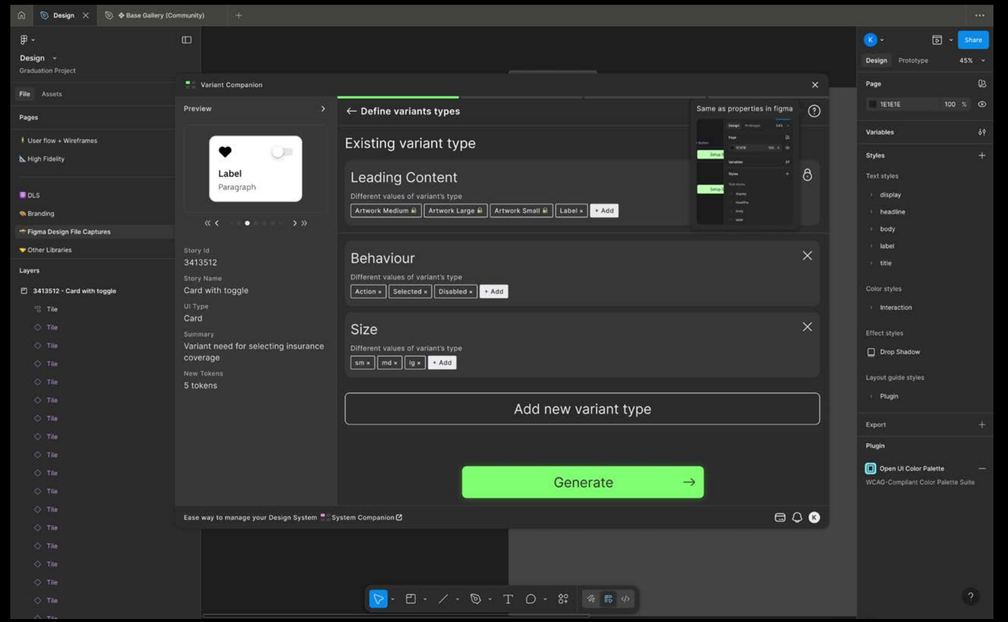
Design System Compliance



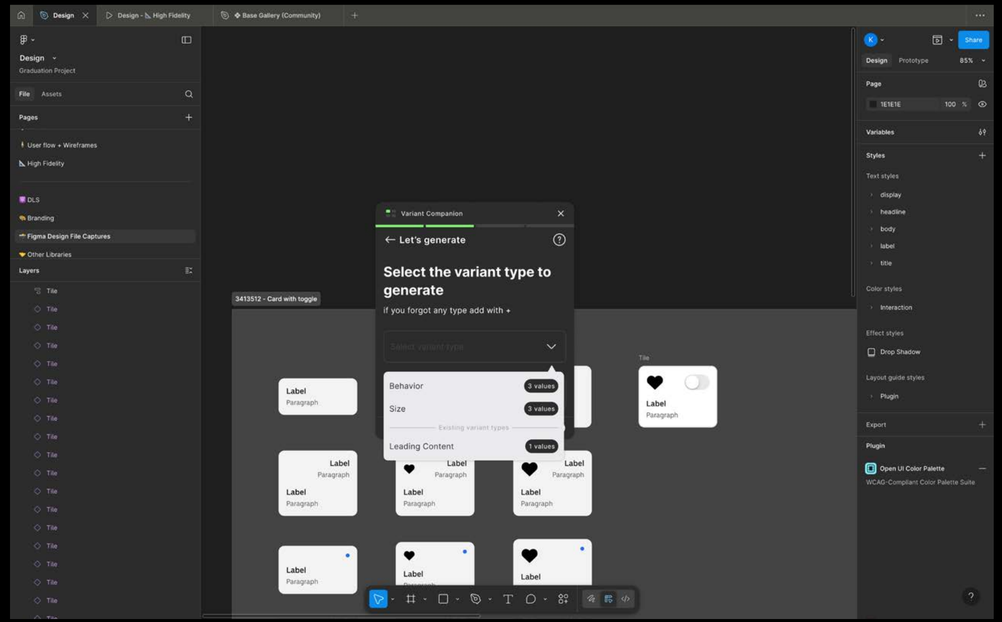
Managing conflicts in generation



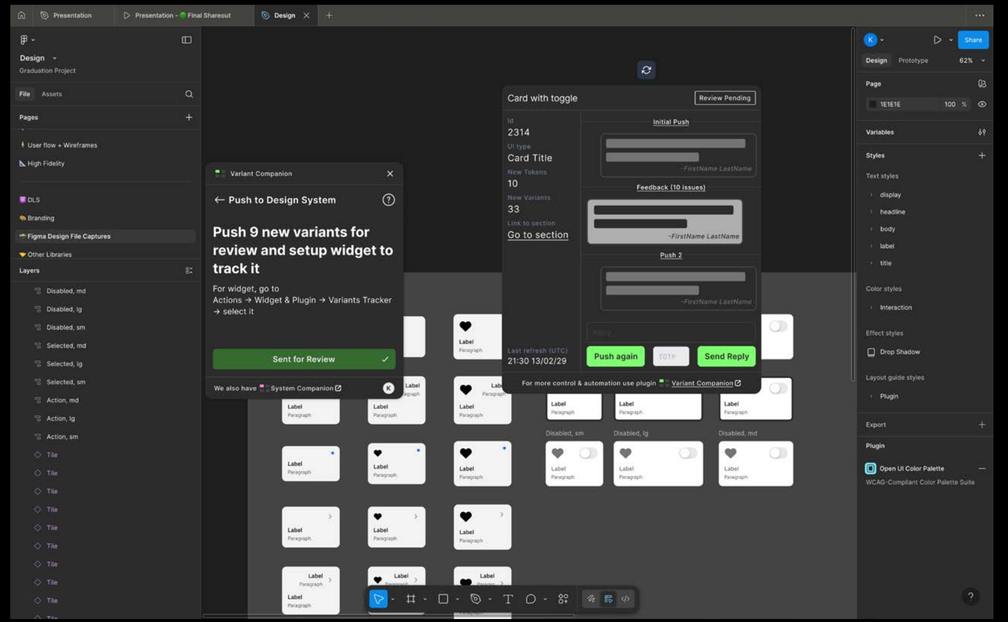
Selecting variants types + tooltips

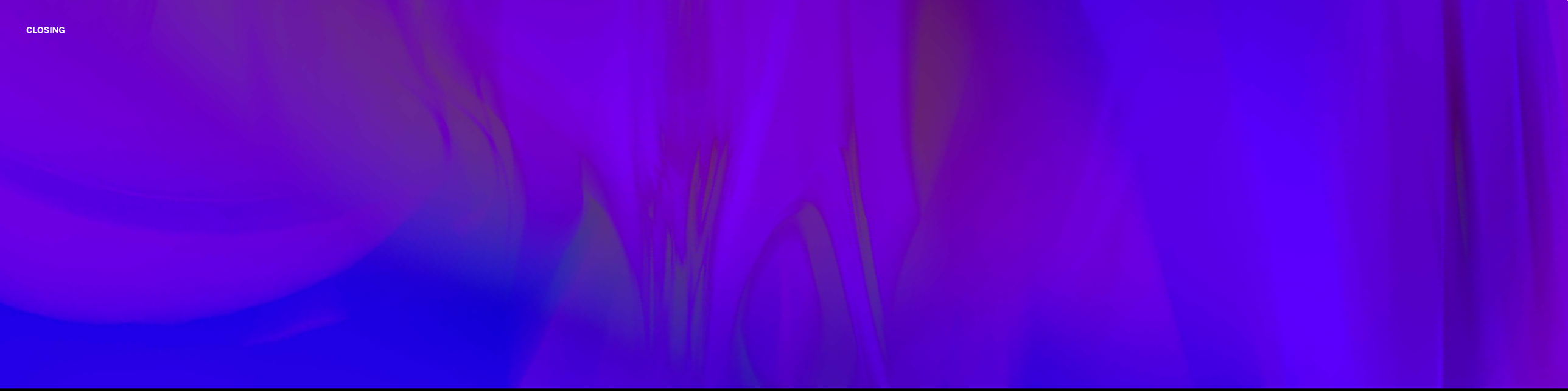


Selecting variant type for generation



Pushing for review and Widget Setup





Conclusion

Future Development & Enhancements

Automated Linkage to Developer Documentation (e.g., Storybook)

To bridge the gap between design specifications and live code, a significant enhancement would be to integrate the “Design System Companion Plugin Suite” (specifically Plugin 2 - DS Integration & Review) with developer documentation platforms like Storybook. Upon approval and integration of a component into the Design System via the plugin, an automated process could create or update its corresponding entry in Storybook, linking design specs, states, and tokens directly to the coded component, ensuring a true single source of truth that spans both design and development.

Expanding Plugin Functionality to Other Utility Flows

The principles and architecture of the “Design System Companion Plugin Suite” can be extended to streamline other critical but often repetitive utility design flows. This could include developing specialized modules or companion plugins for tasks such as generating comprehensive user flow diagrams with linked screen states, managing accessibility annotations systematically, or even facilitating the creation of consistent presentation slide decks based on brand guidelines, further enhancing designer efficiency and output quality across various deliverables.

Integrating Developer Feedback Loops within the Plugin Suite

While the current concept focuses heavily on empowering designers to create comprehensive specifications, a crucial next step involves building robust mechanisms within the “Design System Companion Plugin Suite” for developers to directly provide feedback, flag issues, or request clarifications on specific components or states. This would create a more seamless, bi-directional communication channel directly within the design environment, further reducing friction and ensuring that developer insights are captured and actioned efficiently throughout the iterative process.

Key Learnings

Proactive Mitigation is Key for Developer Collaboration

The most effective way to mitigate issues with developers is not just through better handoffs, but by fostering early, continuous, and empathetic communication, ensuring technical feasibility is considered from the outset, and that design rationale is clearly shared.

Understanding Interconnected Workflows is Crucial

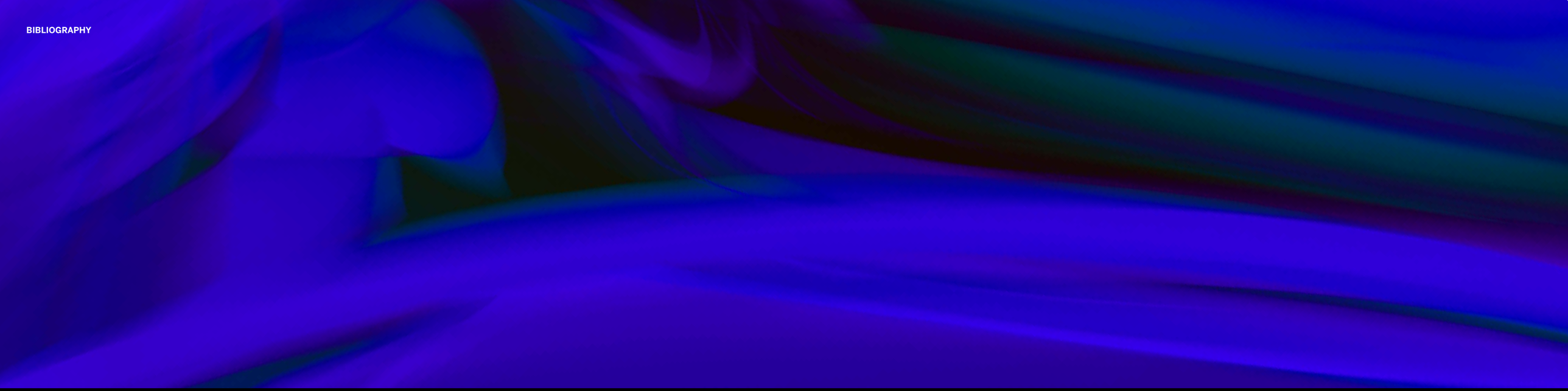
Designer and developer workflows are deeply intertwined. Optimizing one in isolation is insufficient; true efficiency comes from understanding and improving the interfaces, dependencies, and feedback loops between these distinct yet collaborative processes.

Standards are the Bedrock of Scalable Quality

The importance of well-defined design systems, clear specification frameworks, and consistent adherence to standards cannot be overstated. They are fundamental to reducing ambiguity, ensuring quality, and enabling teams to scale effectively.

Rigorous Design Research & Iterative Interaction Design Drive Effective Solutions

A human-centered approach, grounded in thorough design research to understand user (designer/developer) needs and pain points, followed by iterative interaction design and validation, is essential for creating tools and processes that genuinely solve problems and enhance user experience.



Bibliography

Bibliography

https://medium.com/@krsatvik/list/devexp-7d2867031f58
https://medium.com/@bradleybirch09/responsive-over-non-responsive-web-design-7a59e6a8b7e3
https://medium.com/@pavelparradomarin/how-material-3-is-improving-user-experience-on-android-eae570f048ec
https://medium.com/@mrijalulkahfi/figma-dev-mode-how-it-helps-flutter-developers-66f5308967af
https://medium.com/@rashi.karanpuria/from-developer-to-designer-day-01-5baad6ab31
https://medium.com/prototypr/agile-ux-what-i-learned-from-living-with-an-agile-developer-4173a60a8172
https://medium.com/appwrite-io/announcing-console-2-0-2e0e96891cb0
https://medium.com/dx-heroes/how-to-build-a-developer-portal-that-developers-will-adore-1a5bb51b36ee
https://medium.com/@tusharupadhyay691/advanced-react-component-design-with-typescript-part-2-2b76b3e0e52e
https://medium.com/holochain/improving-the-developer-experience-debug-logging-3e27a8e7be05
https://medium.com/holochain/developer-experience-the-details-matter-358f0d57d125
https://medium.com/@codefinger/meet-developers-where-they-are-8b667e2c79f5
https://medium.com/@riyajawandhiya/how-do-we-involve-developers-in-the-design-system-before-asking-them-to-write-code-3cb131ec6ff5
https://medium.com/@vitiya99/taming-the-beast-structuring-large-scale-react-applications-119ba172de1c
https://medium.com/vmwaredesign/introducing-transport-and-vmware-developer-experience-dx-98aaf5146ba6
https://medium.com/words-from-ultima/developer-experience-101-f7773d2055f6
https://medium.com/bitsrc/top-10-ui-ux-concepts-every-developer-should-know-e9a6b9ae53da
https://medium.com/user-experience-design-1/color-management-for-better-dx-developer-experience-640044a9e21a
https://medium.com/@broccolini/design-systems-at-github-c8e5378d2542
Verdecchia et al., 2021: The core focus of this paper is ATD. It develops a comprehensive theory of ATD grounded in empirical data.
Soliman et al., 2021: This paper studies ATD specifically in the context of architectural design decisions (ADDs).
Yli-Huumo et al., 2016: This paper mentions this as a general description of the term technical debt.
Verdecchia et al., 2021: Uses the financial debt analogy as part of the broader discussion of ATD.
Andrews et al., 2005: Use TD terms in general, and mention how important it is to find a balance between them.
Plösch et. Al, 2018: Use financial terms to derive a framework to manage debt.
https://kumarsatvik.notion.site/Secondary-Research-18b95233353a80c59d83f5d515117dd9