Figma Custom Plug-in Development - LayerMatch

VDS Figma Plugin

**Last updated:** Nov 15, 2024

# Overview

# Inspired by [Similayer](https://www.figma.com/community/plugin/735733267883397781/similayer), a plugin that we love, but that Verizon Security has blocked.

# LayerMatch is a Figma plugin designed to enhance the layer selection AND manipulation capabilities within Figma. It extends Figma's native selection functionality, allowing users to select layers based on a wide range of properties with greater precision AND flexibility. This plugin aims to significantly improve workflow efficiency for designers working on complex projects with numerous layers.

# How this sets us up for 2025

As VDS moves into the tools space, these learnings will teach us how to develop and push plugins through the Verizon security ecosystem.

# Outputs

## Key Features

* Advanced Layer SelectionSelect layers based on multiple properties simultaneously
* Batch EditingApply changes to all selected layers at once
* Selection PresetsSave and reuse complex selection criteria
* Selection HistoryUndo/redo selection actions
* Layer AnalysisGenerate reports on layer usage and properties
* Smart SelectionSelect layers based on their similarity to a chosen layer
* Nested Layer SelectionSelect layers within groups or frames
* Selection InversionInvert the current selection to select all unselected layers
* Selection ExpansionExpand selection to include layers with similar properties
* Custom Selection RulesCreate and save custom selection rules using boolean logic

## User Stories

1. As a user, I want to select all button layers with a specific color and border radius, so I can quickly update them to match new brand guidelines.
2. As a user, I want to find all text layers using deprecated fonts, so I can ensure consistency across the design.
3. As a user, I want to save my commonly used selection criteria, so I can work more efficiently across different projects.
4. As a user, I want to share my selection presets with my colleagues, so we can maintain consistency in our workflow.
5. As a user, I want to select all layers similar to a chosen layer, so I can quickly apply changes to visually similar elements.
6. As a user, I want to invert my current selection, so I can work on all other layers not currently selected.
7. As a user, I want to select all layers with a specific effect applied, so I can standardize the use of effects across my design.
8. As a user, I want to create a custom selection rule that combines multiple properties, so I can perform complex selections with a single action.
9. As a user, I want to select all layers within a specific frame that match certain criteria, so I can organize my design components more efficiently.
10. As a user, I want to expand my current selection to include layers with similar properties, so I can ensure consistency across related elements.
11. As a user, I want to generate a report of all layer properties in my design, so I can audit my design system for inconsistencies.
12. As a user, I want to undo my last selection action, so I can quickly revert to a previous selection state if needed.
13. As a user, I want to select all layers with a specific naming convention using regular expressions, so I can efficiently manage layers with systematic naming.
14. As a user, I want to batch edit the properties of all selected layers, so I can make consistent changes across multiple elements quickly.
15. As a user, I want to adjust the similarity threshold when using smart selection, so I can fine-tune the precision of my selections based on visual similarity.

## Acceptance Criteria

**Feature: Advanced Layer Selection**

Scenario: Select layers by multiple properties

GIVEN the user has opened a Figma file

WHEN the user chooses to select layers by properties

AND the user specifies color "#FF0000" AND type "Rectangle"

Then all red rectangle layers should be selected

Scenario: Select layers using regular expressions

GIVEN the user has opened a Figma file

WHEN the user chooses to select layers by name

AND the user enters the regex pattern "btn-[a-z]+"

Then all layers with names matching the pattern should be selected

**Feature: Batch Editing**

Scenario: Apply color change to multiple layers

GIVEN the user has selected multiple layers

WHEN the user chooses to change the fill color to "#00FF00"

THEN all selected layers should have their fill color changed to green

**Feature: Selection Presets**

Scenario: Save AND apply a selection preset

GIVEN the user has defined a complex selection criteria

WHEN the user saves the criteria as a preset named "Header Elements"

AND the user applies the "Header Elements" preset in a new file

THEN the layers matching the preset criteria should be selected

**Feature: Smart Selection**

Scenario: Select similar layers with threshold adjustment

GIVEN the user has selected a layer

WHEN the user initiates smart selection with a similarity threshold of 80%

THEN all layers that are 80% or more similar to the selected layer should be selected

**Feature: Nested Layer Selection**

Scenario: Select nested layers within a frame

GIVEN the user has a frame with nested layers

WHEN the user chooses to select all text layers within the frame

THEN only the text layers inside the frame should be selected

**Feature: Selection Inversion**

Scenario: Invert current selection

GIVEN the user has selected some layers

WHEN the user chooses to invert the selection

THEN all previously unselected layers should become selected

AND all previously selected layers should become unselected

**Feature: Custom Selection Rules**

Scenario: Create AND apply a custom selection rule

GIVEN the user creates a rule to select all blue rectangles with a shadow effect

WHEN the user applies the custom rule

THEN all blue rectangle layers with a shadow effect should be selected

**Feature: Layer Analysis**

Scenario: Generate layer property report

GIVEN the user has a complex Figma file

WHEN the user requests a layer property report

THEN a report should be generated listing all layers AND their properties

AND the report should highlight any inconsistencies in the design system

# Non-Functional Requirements

1. **Platform Compatibility:** The plugin must function on macOS.
2. **Performance:** The plugin should process selections for up to 1,000 layers in under 3 seconds.
3. **Compatibility:** The plugin must work with the latest version of Figma AND be backwards compatible with the last major release.
4. **Localization:** Support for English language only in the initial release.
5. **Accessibility:** The plugin interface must be navigable using keyboard shortcuts.
6. **Scalability:** The plugin should handle files with up to 1,000 layers without significant performance degradation.
7. **Reliability:** The plugin should have an uptime of 99.9% during Figma's operational hours.
8. **Security:** All data processing should occur locally within the Figma environment, with no external data transmission.
9. **Usability:** The plugin should have an intuitive interface that requires no more than 5 minutes of onboarding for a new user.
10. **Maintainability:** The plugin's codebase should follow best practices for code organization AND documentation.
11. **Extensibility:** The plugin architecture should allow for easy addition of new selection criteria in future updates.
12. **Resource Usage:** The plugin should not consume more than 100MB of memory during operation.
13. **Error Handling:** The plugin should provide clear error messages for any issues encountered during use.
14. **Response Time:** User interactions with the plugin interface should have a response time of no more than 200ms.

# Constraints

1. The plugin must adhere to Figma's API limitations AND security guidelines.
2. Initial release will be limited to desktop versions of Figma.
3. The plugin must pass Verizon's IT security stANDards for third-party software.
4. Development must be completed using only approved libraries AND frameworks.
5. The plugin must not interfere with or modify Figma's core functionality.
6. All user data must be stored locally within the Figma environment.

# Risks

1. Value
2. Feasibility
3. Viability
   1. Brand Reputation
   2. Legal
      1. [Legal Contacts](https://docs.google.com/spreadsheets/d/1F7sBtKpE8Owi4D67H_68Ly0VE6_nJB_4LtR3f89Egj8/edit?gid=0#gid=0)
4. Usability

# Timeline

4 weeks for a TBD thin vertical slice

# Observing User Behavior

As developers, you know that building features is only half the battle. The other half is understanding how users interact with those features in the wild. For LayerMatch, our goal isn’t just to deliver functionality—it’s to deliver functionality that works seamlessly for our Verizon designers and adds measurable value to their workflows. To do that, we need to be intentional about how we **observe and analyze user behavior**.

## Event Tracking: The Heart of Usability Data

Within the Figma plugin environment, we should have control over tracking how users interact with LayerMatch. By embedding analytics directly into the plugin, we can capture key actions, such as:

1. How often the plugin is launched and used.
2. Which features (e.g., Advanced Layer Selection, Batch Editing, Smart Selection) are used and in what combinations.
3. The time spent completing tasks like generating a report or applying a selection rule.

This gives us a clear picture of what’s working and what’s being ignored. For example, if we see frequent use of Selection Presets but limited use of Custom Rules, we can investigate whether users find rules too complex or redundant.

## Localized Logging: Understanding User Workflows

We can add local logging within the plugin to track sequences of actions during a session. For example, if a designer uses Layer Analysis followed by Batch Editing, we can infer that they’re auditing layers for issues and then correcting them. This data stays local to the user’s machine, respecting privacy while giving us anonymized insights into real workflows. By analyzing patterns across these logs, we can prioritize improvements to high-friction workflows.

## Embedded Feedback Mechanism

A built-in feedback button allows users to report issues or suggest improvements directly from LayerMatch. This feature can guide us in identifying unanticipated pain points or unmet needs. Designers might tell us, for example, that Selection Expansion isn’t intuitive or that Nested Layer Selection is too rigid. This qualitative data will complement our analytics by providing the "why" behind user behavior.

## Usage Metrics: Proving Productivity Gains

We can track and benchmark productivity-related metrics. For instance, if a user applies a batch edit to 50 layers in five seconds, we can compare that to manual estimates for the same task. Over time, these benchmarks allow us to quantify LayerMatch’s impact on efficiency, which is critical for demonstrating its value to Verizon.

## Collaboration Metrics: Tracking Sharing Behavior

While we can’t see how teams work in Figma itself, we can track specific LayerMatch collaboration features. For example, if designers are saving and sharing Selection Presets, we can log how often these actions occur. If sharing is rare, we’ll know to investigate and refine the feature to make it more appealing and usable.

## Error Logging: Improving Reliability

By capturing and analyzing error logs, we can identify where users encounter issues. Whether it’s a failed batch edit, an invalid regular expression in a selection rule, or difficulty undoing a selection, these logs will show us where to focus bug fixes and improvements.

## User Surveys: A Low-Cost Insight Generator

Surveys don’t require integration into Figma, making them a scalable option for gathering user insights. We can send out short, targeted surveys to users after major updates or periodically to assess satisfaction and gather feedback. Questions could include:

1. “Which LayerMatch feature do you use most often, and why?”
2. “What task did you struggle with recently while using LayerMatch?”

This qualitative feedback is invaluable for shaping future development.

# What We Can Build to Observe

By focusing on event tracking, localized logging, embedded feedback, and surveys, we can gather rich behavioral insights entirely within the plugin environment. These methods ensure we’re aligned with Figma’s platform constraints and Verizon’s security standards while still understanding how users engage with LayerMatch. This approach ensures that every feature we refine or build is driven by real data and user needs. Let’s focus on building what’s realistic—and making it powerful.