lightspeed

email: info@lightspeedwp.agency
website: lightspeedwp.agency

46 Devon Street, Woodstock
Cape Town, 7925

Figma Design Process

LightSpeed Internal Design Workflow Process

Using Bugherd for Design Feedback: A Step-by-Step Guide

Overview

This document outlines the revised guidelines and tools now in use, emphasising efficiency, collaboration, and quality in our design process.

As part of our commitment to streamline and enhance our design and development workflow, we've made significant updates to our internal design process. This document outlines the revised guidelines and tools now in use, with a focus on integrating a Git-based workflow for our design systems.

This document reflects our current practices and tools, emphasising efficiency, collaboration, and quality in our design process. Please ensure you are familiar with these guidelines and utilise the outlined tools and processes in your work.

Design Tool Transition

LightSpeed consolidated designs tools and streamlined our design workflow:

Previous Design Toolset:

- Sketch App: Used for UI & UX web design.
- Abstract App: Used for design branching & version control of Sketch design files.
- Invision App: Used for gathering feedback from stakeholders on design outputs.

Previous Design Toolset:

- Figma.com: used our sole design tool for all web designs and for creating / publishing our design system libraries using Figma branches & a Figma plugin for changelogs.
    - NOTE: Figma has branching capabilities, read more about "Branching best practices", as well as a "Guide to branching" within the Figma help portal.
- Bugherd.com: Used for design feedback via integration with Figma and the team and clients provide feedback on websites or bug reports during design, development, testing and ongoing post launch.

Image File Naming and Exporting

Files must adhere to specific naming conventions for organization and efficiency. This includes lowercase naming, use of hyphens, and inclusion of image dimensions where applicable. Different file formats (.jpeg, .png, .svg) will be used based on the nature of the image and its use case, with a strong emphasis on optimising file sizes for web use.

Summary:

- **Naming Conventions:**
    - Use lowercase naming, hyphen separation, and include image dimensions.
- **File Formats:**
    - .JPEG for photographs and complex images.
    - .PNG for images requiring transparency.
    - .SVG for vector-based graphics.
- **Optimization:**
    - Aim to keep file sizes low without sacrificing quality.

Feedback Collection

Internal Projects

- Figma design system and prototype comments will serve as the initial feedback and discussion tool between Designer, Product Owner and Developers.
- During the design phase, Developers should always discuss designs with the Designer using Figma during the design process.
- Developers should discuss designs with the designer in Figma before development starts.
- Once development has started, then the Designer & Product Owner will use Bugherd for providing feedback to the Developers via contextual comments logged via the dev site being used and the Bugherd feedback toolbar.
- Bugherd streamlines the code feedback process, making it easier for both LightSpeed & Clients to communicate on the designs via a tool and prioritising tasks via a Kanban board.

Client Projects

- Bugherd should be used for gathering client feedback during the design and development phase, because it supports a Figma integration.
- In rare cases, clients will use Figma comments to provide feedback on client projects, this is likely when there are Figma prototypes involved.

Feedback and Approval

- **Using Bugherd for Feedback:** Clients will provide feedback directly on Figma designs via Bugherd. This approach ensures clear, actionable feedback and simplifies the revision process.
- **Approval Process:** Design approvals will be managed through Figma, with final sign-offs occurring within the platform to ensure clarity and accountability.

Design Process Guidelines

Meetings

- **Meetings:** Google Meet is our chosen platform for conducting all design-related meetings. This includes internal reviews, client presentations, and feedback sessions.

Task Management & Roadmap Planning

- **Harvest Time tracking:**
    - Use existing internal Harvest projects for open source projects, rough time logs are acceptable for internal projects as this is more for tracking the company's investment in open source projects.
    - For client projects, set up a new Harvest project and define its budget, it is essential client project time contains accurate time logs for billing purposes.
- **Asana Tasks:**
    - Asana continues to be our tool for internal task management for day to day workflow. It helps us keep track of design tasks, deadlines, and project progress.
    - A parent design task will be created in the Asana project for the duration of the design phase. This task will serve as the central hub for status updates, time logging, and overall project management.

    - Asana subtasks should include detailed requirements and time estimates, which are converted to development tasks post-design phase.
    - Sub-tasks for each design screen, including time estimates, will be created and converted to development tasks upon completion of the design phase.
- **Github projects for internal open source products:**
    - Planning for new releases and planning of plugin and theme roadmaps should start in Github projects
    - Asana tasks simply reference the github issue link, the Github Issue will contain the full requirements for said feature.

Git-Based Design Workflow

Reference links

- **Figma:**
    - Branching in Figma
    - Guide to branching – Figma Learn - Help Center
    - How to use Figma branching properly - LogRocket Blog
        - NOTE: Read about "***Resolving conflicts***"
- **Github:**
    - Git Workflow | Atlassian Git Tutorial
    - Gitflow Workflow | Atlassian Git Tutorial
    - GitHub flow | GitHub Tutorial
        - Creating and deleting branches within your repository - GitHub Docs
        - Creating a pull request - GitHub Docs
        - Reviewing changes in pull requests - GitHub Docs
        - Merging a pull request - GitHub Docs
        - Addressing merge conflicts - GitHub Docs

New Feature Planning

- **Create GitHub Issue:**
    - Use a "GitHub template for new features".
    - Plan features in GitHub Issues & Projects.
- **Create Figma Branch:**
    - Create a Figma branch for the respective design system corresponding to the GitHub issue.
- **Complete Design Updates:**

    - Work on design updates within the Figma branch.
- **Design Review:**
    - Conduct design reviews similar to pull request reviews.
- **Merge Figma Branch:**
    - Merge the Figma branch into the main design system upon approval.

Developer Handoff

Upon design approval, detailed handover notes will be provided to the development team through Asana. This includes all relevant Figma links, image assets, and any specific development considerations.

Reference links:

- Guide to developer handoff in Figma | Figma.com Best Practices
- Tips on developer handoff in Figma | Figma.com Best Practices
- Designing A Better Design Handoff File In Figma — Smashing Magazine

Process description:

- **Handover notes:**
    - Create detailed handover notes that include all relevant Figma links, image assets, and specific development considerations.
- **Export and Upload Designs:**
    - Export updated design screens from Figma.
    - Attach them to the relevant GitHub issue.
- **Create Corresponding GitHub Branch:**
    - Use consistent naming conventions between Figma and GitHub branches.
- **Commit Code and Create Pull Request:**
    - Develop the new feature and commit to the GitHub branch.
    - Create a pull request for review.
- **Review and Merge Code:**
    - Conduct a review of the pull request.
    - Merge the branch upon approval.
- **Close Issue and Update Board:**
    - Close the GitHub issue and update the projects board.

Changelog and Versioning

- **Maintain a Changelog:**

    - Keep a detailed changelog for both design systems.
    - Align versioning with plugins and themes.
    - Reference: **Keep a Changelog**
- **Versioning Alignment:**
    - Start the LSX Tour Operator Design System at version 2.0.0 to align with the 2.0.0 plugin release.
- **See example below:**

Changelog

July 24, 2024

LightSpeed on July 24, 2024

Added
- Free Consultation full page layout

July 5, 2024

LightSpeed on July 5, 2024

Added
- Prototype components:
  - Button prototype component with default, hover, pressed and disabled states.
  - Navigation drop-down prototype with a closed and open state.
  - Single Product drop-down prototype with a closed and open state.
  - Radio button prototype with default, hover, selected and selected-hover states
  - Checkbox prototype with default, hover, selected and selected-hover states
  - Text inputs for Search, Text, Message and Name with default and active states
  - Quantity input with toggles and a linked quantity indicator
  - Header
    - Logo - links to homepage when pressed
    - Navigation - links to shop page, as well as a collection drop down nav item which lists collections
    - My Account - default and hover state
    - Mini Cart - opens the mini cart overlay from the right
    - Product Search - expands when search button is pressed
- Prototypes patterns:
  - Mini-cart prototype
  - Cart prototype page
  - Product card example prototypes
  - 4x Normal Single products
  - 4x On Sale Single products prototypes
  - 11x simple product cards per Single product
  - Order confirmation prototype
  - Homepage prototype
  - Shop page carousel with 3 pages
  - Category pages for men's wear, women's wear, sale, men's shoes, women's shoes, sunglasses.
- Prototype variables:
  - Variable Collection: Prototypes - Global = Global variables used to control state and display of components in the header, cart, checkout, and order confirmation.
    - Cart & Checkout Group:
      - Cart Total Items = number variable tracker for the number of items added to the cart
      - Cart Total Price = number variable tracker for the total value of all the items in the cart
      - Items in Cart = Boolean variable dictating if the contents of the cart displays when there are items in it.
      - Empty Cart Message = Boolean variable dictating if the empty cart message displays in the mini cart and cart
      - Checkout Total = number variable tracker for the total value of all the items in the cart, as well as any additions like delivery.
      - Checkout Button State = variable controlling the state of the checkout button during the checkout process
      - Checkout Shipping Fee = number variable dictating the shipping fee
      - Complete Purchase Button = variable controlling the state of the Complete Purchase button
      - Payment Method = string variable dictating the payment method for the checkout process

    - Shipping Options Group:
      - Free Shipping = controller variable dictating the state of the Free Shipping option radio button in Checkout
      - Local Pickup = controller variable dictating the state of the Local Pickup option radio button in Checkout
      - Flat Rate = controller variable dictating the state of the Flat Rate option radio button in Checkout
      - Shipping Options Checked = boolean variable indicating if a shipping option has been selected, this activates the Checkout Button if a payment option is selected.

    - Payment Options Group:
      - Direct Transfer = controller variable dictating the state of the Direct Transfer option radio button in Checkout
      - Card Payment = controller variable dictating the state of the Card Payment option radio button in Checkout
      - Cash On Delivery = controller variable dictating the state of the Cash On Delivery option radio button in Checkout
      - Direct Transfer Dropdown = controller variable dictating the state of the Direct Transfer option drop down in Checkout
      - Card Payment Dropdown = controller variable dictating the state of the Card Payment option drop down in Checkout
      - Cash On Delivery Dropdown = controller variable dictating the state of the Cash On Delivery option drop down in Checkout
      - Payment Options Checked = boolean variable indicating if a payment option has been selected, this activates the Checkout button if a Shipping Option is selected

Using Bugherd for Design Feedback

- **Setting Up Your Project in Bugherd:**
  - Invite participants and install the browser extension.
- **Adding Comments on Designs using Bugherd:**
  - Use Bugherd to comment directly on design elements.
  - To get feedback from guests, share a single asset or an entire group.
  - Collect and manage feedback on Figma designs, images and PDFs.
  - Our team and your clients & stakeholders now have one place to review and get feedback on all the work you create.
  - Bugherd project guests and members can go to the Websites & Files folder for each project and leave feedback on any files uploaded.
- **Communicating Changes or Suggestions:**
  - Use clear, concise language and prioritise feedback.
- **Organising and Managing Feedback:**
  - Categorise tasks and assign them to specific team members.
- **Reviewing and Revising Designs:**
  - Update task statuses and communicate with the team for ongoing discussions.
- **Finalising Design Changes:**
  - Review resolved tasks and sign off on completed designs.

Figma Code Connect (Future Objective)

Figma Code Connect is a future Objective, we need to start with research to be able to implement the connection between Figma and the codebase.

Initial Steps for Figma Connect Integration with GitHub

- **Understand the Basics:**
  - Figma Code Connect: Begin by understanding what Figma Code Connect offers. Review the Figma Code Connect GitHub Repository for detailed documentation and examples.
  - GitHub API: Familiarise yourself with GitHub's API to understand how to interact with GitHub repositories programmatically.
- **Set Up Your Environment:**
  - GitHub Account: Ensure you have a GitHub account with the necessary permissions for the repositories you will be working on.

    - Figma Account: Ensure your Figma account has the required access to the design files you intend to integrate.
- **Research and Documentation:**
    - Figma Documentation: Review the Figma API Documentation to understand how to fetch, update, and manage design files programmatically.
    - GitHub Documentation: Review the GitHub API Documentation to learn how to create issues, pull requests, and manage repositories through API calls.
- **Tool and Resource Collection:**
    - Code Editor: Set up a code editor like VSCode or any other IDE you are comfortable with for writing integration scripts.
    - Postman: Use Postman to test API calls for both Figma and GitHub.
- **Define Integration Goals:**
    - Scope: Clearly define what you aim to achieve with the integration. Examples might include:
    - Automatically creating GitHub issues from Figma comments or tasks.
    - Syncing Figma design changes with corresponding GitHub branches.
    - Generating design tokens in Figma and updating corresponding styles in the codebase.
- **API Authentication:**
    - Figma Authentication: Set up OAuth or Personal Access Tokens for authenticating API requests with Figma.
    - GitHub Authentication: Set up OAuth or Personal Access Tokens for authenticating API requests with GitHub.
- **Prototype a Simple Integration:**
    - Create GitHub Issue from Figma: Start by creating a script that fetches comments or tasks from a Figma file and creates corresponding issues in a GitHub repository. This will help you get hands-on experience with both APIs.
    - Push Design Changes to GitHub: Create a simple workflow where changes in Figma trigger an update in a GitHub repository, such as committing JSON files of design tokens.
- **Testing and Iteration:**
    - Test the Integration: Rigorously test your prototype to ensure it handles different scenarios and edge cases.
    - Feedback Loop: Collect feedback from designers and developers to refine the integration.
- **Documentation and Training:**

    - Document the Integration Process: Create a detailed guide on how to use the integration for your team.
    - Training Session: Conduct training sessions to ensure all team members understand how to use the new integration.

Resources Needed

- **Documentation and Guides:**
    - Figma API Documentation
    - GitHub API Documentation
    - Figma Code Connect Repository
- **Tools:**
    - Code Editor (VSCode, Sublime, etc.)
    - Postman for API testing
    - GitHub Account with Repository Access
- **Figma Account with File Access:**
    - Sample Code and Libraries:
    - Example scripts and libraries for interacting with Figma and GitHub APIs (available on GitHub).
- **Team Collaboration Tools:**
    - Communication channels (Slack, Microsoft Teams) for coordinating between designers and developers.
    - Project management tools (Asana, Github) to track integration progress and tasks.

Next Steps

- **Kick-off Meeting:** Organise a meeting with key stakeholders to outline goals and assign responsibilities.
- **Set Up Environment:** Ensure all team members have the necessary accounts, permissions, and tools.
- **Start Prototyping:** Begin with a simple integration and iterate based on feedback.
- **Regular Check-ins:** Schedule regular check-ins to monitor progress and address any challenges.

Addendum

Best Practices for Naming and Exporting Image Files for Web Use

The following guidelines are designed to ensure efficient organization, optimal loading times, and the highest quality for images used on the web. Adherence to these practices is essential for maintaining a streamlined workflow and delivering professional results in our projects.

Naming Conventions

1. **Lowercase Only:** All file names should be in lowercase to avoid confusion and ensure consistency across different operating systems.
2. **Hyphen Separation:** Use hyphens (-) to separate words in file names. This improves readability and helps with SEO practices.
3. **Include Dimensions:** For images where size is crucial, such as banners, include the image dimensions at the end of the file name (e.g., `banner-homepage-1920x1080.jpeg`).

File Formats and Usage

- **.JPEG:** Ideal for photographs and complex images with gradients. JPEGs offer a balance between quality and file size, making them suitable for web content. When exporting JPEGs, adjust the quality slider to find the best balance between image quality and file size, aiming for images below 500kb. Use tools like Photoshop's "Save for Web" to optimise further.
- **.PNG:** Use PNG for images requiring transparency, such as logos and icons. PNGs maintain the quality of sharp lines and detailed graphics without loss. While PNGs generally have larger file sizes than JPEGs, they are preferable for graphics where clarity is paramount. Export directly from design tools at 2x size for quality, then scale down as needed.
- **.SVG:** For vector-based graphics such as logos and icons, SVG is the preferred format. SVGs scale without losing clarity, ensuring crisp visuals at any size. Before exporting SVGs, clean up any unnecessary layers to minimise file size and complexity.

Exporting for Web

- **Optimization:** Aim to keep the file size as low as possible without sacrificing visual quality. Use tools like TinyPNG or Adobe Photoshop for advanced compression options.
- **Testing:** Preview your images in the context they will be used. Compare the compressed image to the original to ensure there is no noticeable loss in quality.

- **Accessibility:** When naming files, consider descriptive names that convey the content of the image. This practice supports SEO and accessibility by providing context for screen readers.

By following these guidelines, we ensure our web projects are optimised for speed, aesthetics, and functionality. Proper naming and exporting practices contribute significantly to the overall performance and user experience of our websites and applications.

Using Bugherd for Design Feedback: A Step-by-Step Guide

Bugherd is a powerful tool for streamlining feedback during the design process. It allows team members and clients to provide precise, contextual feedback directly on the design itself. By following the steps below, we can leverage Bugherd to provide actionable, organised feedback that enhances the design process, ensuring efficient collaboration and high-quality outcomes.

Here's how to effectively use Bugherd for design feedback:

1. Setting Up Your Project in Bugherd

- **Invite Participants:** Begin by inviting your team and clients to the project. They'll receive an email with instructions on how to join and start providing feedback.
- **Install the Browser Extension:** For ease of use, participants should install the Bugherd browser extension. This allows for seamless integration with your design previews.

2. Adding Comments Directly on Designs

- **Navigate to the Design:** Open the design you wish to comment on through the link provided by Bugherd. This can be a live website or a design mock-up.
- **Activate the Bugherd Sidebar:** Click the Bugherd extension icon to activate the sidebar. This is where you'll see existing tasks and comments.
- **Create a New Task for Feedback:** Click on the webpage or design element where you want to leave feedback. A new task creation window will pop up.
- **Describe Your Feedback:** Clearly articulate your feedback in the task window. Be specific about what changes are needed or what elements you're referring to.

3. Effectively Communicating Changes or Suggestions

- **Be Specific:** Use clear, concise language to describe what needs to be changed or improved. Avoid vague terms.
- **Reference Design Elements:** Use the annotation and screenshot features to highlight specific elements related to your feedback.

- **Prioritise Feedback:** If possible, indicate the priority of your feedback (e.g., High, Medium, Low). This helps the design team manage their workflow effectively.

4. Organising and Managing Feedback

- **Categorise Tasks: Utilise** categories and tags to organise feedback into groups (e.g., UI, UX, Typography). This makes it easier to address feedback systematically.
- **Assign Tasks:** Assign feedback tasks to specific team members responsible for addressing them. This ensures accountability and streamlines the revision process.
- **Track Progress:** Use the Bugherd dashboard to track the status of feedback tasks. You can see which tasks are in progress, completed, or need further attention.

5. Reviewing and Revising Designs

- **Update Tasks:** Once feedback has been addressed, update the task status in Bugherd. You can mark it as resolved or add a note if further review is needed.
- **Communicate with the Team:** Utilise the comment feature within tasks for ongoing discussions or clarifications. This keeps all communication centralised and accessible.

6. Finalising Design Changes

- **Review Resolved Tasks:** Periodically review resolved tasks to ensure all feedback has been adequately addressed.
- **Sign Off on Designs:** Use Bugherd's task statuses to formally sign off on completed designs, indicating that they meet the project's requirements and standards.