Emphasis: Discovery & Design

LI.FI Widget – Product Development Blueprint

1. Discovery: Decide what is worth building

Goal
Find and validate widget iterations that grow adoption, unlock new use cases, or remove real friction for integrators and end-users.

Inputs
– Feedback from integrators (BD/DevRel calls, support tickets, recurring SDK/API workarounds)
– Product data (funnel metrics, drop-offs, error patterns, low-performing chains/wallets)
– Ecosystem signals (new chains, wallets, AA/intents, gas abstraction, DeFi/vault flows partners request)
– Relevant standards (EIP-712, ERC-2612, Permit2, EIP-7702, EIP-5792)
– Quick scan of 2–3 comparable widgets or embedded routing UIs

Process
(1) Define the problem – where the widget falls short and who is affected.
(2) Size the opportunity – partners impacted, flows involved, volume or adoption at stake.
(3) Check it belongs in the widget – ensure it should be widget-native rather than SDK/API-level.
(4) Validate demand & feasibility – 3-5 partner conversations, a feasibility check with the tech lead, and a quick data check that the problem shows up in usage.
(5) Prioritise – impact × confidence × effort, with extra weight on widely reusable flows and strategic partners.

Output
A one-page Problem & Opportunity Brief with: problem, who it affects, evidence, why now, and a proposed widget iteration with a success hypothesis.

2. Design: Turn the idea into something buildable

Goal
Translate the opportunity into a clear widget iteration with no ambiguity around scope, UX, or technical behavior.

PRD contents (2-3 pages)
– Objective & success criteria – the problem we're solving and how success is measured
– Personas & JTBD – which integrators and end-users are affected and what they need to accomplish
– Users & flows – core journeys in scope and where the widget intervenes
– Solution outline – where the feature lives in the widget, which LI.FI capabilities it relies on (routing, bridges, DEXs, gas, supported contract calls), plus simple UX diagrams
– Functional requirements – main flow, edge cases, error/fallback behavior; required WidgetConfig options and defaults; events/telemetry to emit
– Non-functional requirements – latency, reliability, compatibility, accessibility
– Non-goals – what will not be part of this iteration
– Dependencies & risks – infra work, wallet/chain quirks, partner or product constraints

Stakeholders & input
– Tech lead / engineers – feasibility, architecture, effort, phased rollout

Emphasis: Discovery & Design

– Design – flows, widget states, copy, empty/error/blocked states
– BD / DevRel – partner relevance and selection of beta testers
– Protocol / chain experts – confirmation of assumptions around standards or contract behavior

End result of Design
A short PRD, UX flows, and example WidgetConfig snippet(s); MVP scope defined, phase-2 ideas noted; acceptance criteria ready for ticket creation.

3. Development: Build and iterate with a small team

Goal
Ship an MVP that works reliably on real routes and wallets, on time and within scope.

Planning & structure
– Break work into 2–4 epics, for example:
(1) Core flow (wallet interactions, routing, on-chain execution)
(2) Widget UI & configuration
(3) Telemetry & documentation
– Estimate with the tech lead and agree on what fits into the first release.

Refinement & PM involvement
– Regular refinement sessions break epics into tickets tied to PRD requirements
– Tickets include clear acceptance criteria and expected failure behavior
– PM responsibilities: unblock quickly, keep scope tight, review early slices in staging with real wallets/chains, and keep Design & BD/DevRel aligned

Output
A release candidate in staging that meets acceptance criteria, works across key chains/wallets, and has draft docs and changelog ready.

4. Deployment: Release and iterate

Goal
Release safely, help partners integrate quickly, and confirm the iteration solves the original problem.

Release process
(1) Testing & QA across chains, wallets, frameworks, and failure paths
(2) Beta rollout behind a config flag to selected partners; track integration effort, completion rate, and errors; gather qualitative feedback
(3) General release with updated docs, a Playground preset, a clear changelog entry, and partner communication via BD/DevRel

Success criteria
– Adoption by targeted integrators within the release window
– Improvement in the primary metric (e.g., completion rate, fewer approval errors, reduced reliance on SDK/API workarounds)
– Lower support load for the original issue

Future iterations
Expand coverage (chains, wallets, supported DeFi/vault flows), refine UX/defaults based on real usage, or retire the feature if it does not provide lasting value.