ConnectSoft UI/UX Design System Brief

Prepared for UI/UX designer estimation and planning

Note: tip "Shareable PDF"

A PDF version for external UI/UX designers is available here:
[connectsoft-ui-ux-designer-brief.pdf](../assets/briefs/connectsoft-ui-ux-designer-brief.pdf).

Purpose

We need help designing a unified UI/UX system for the ConnectSoft product ecosystem.

The designer will not need direct access to our repositories. We will prepare and provide a structured design input package containing screenshots, documentation summaries, product maps, existing visual references, Figma exports, component notes, and examples from the current codebase.

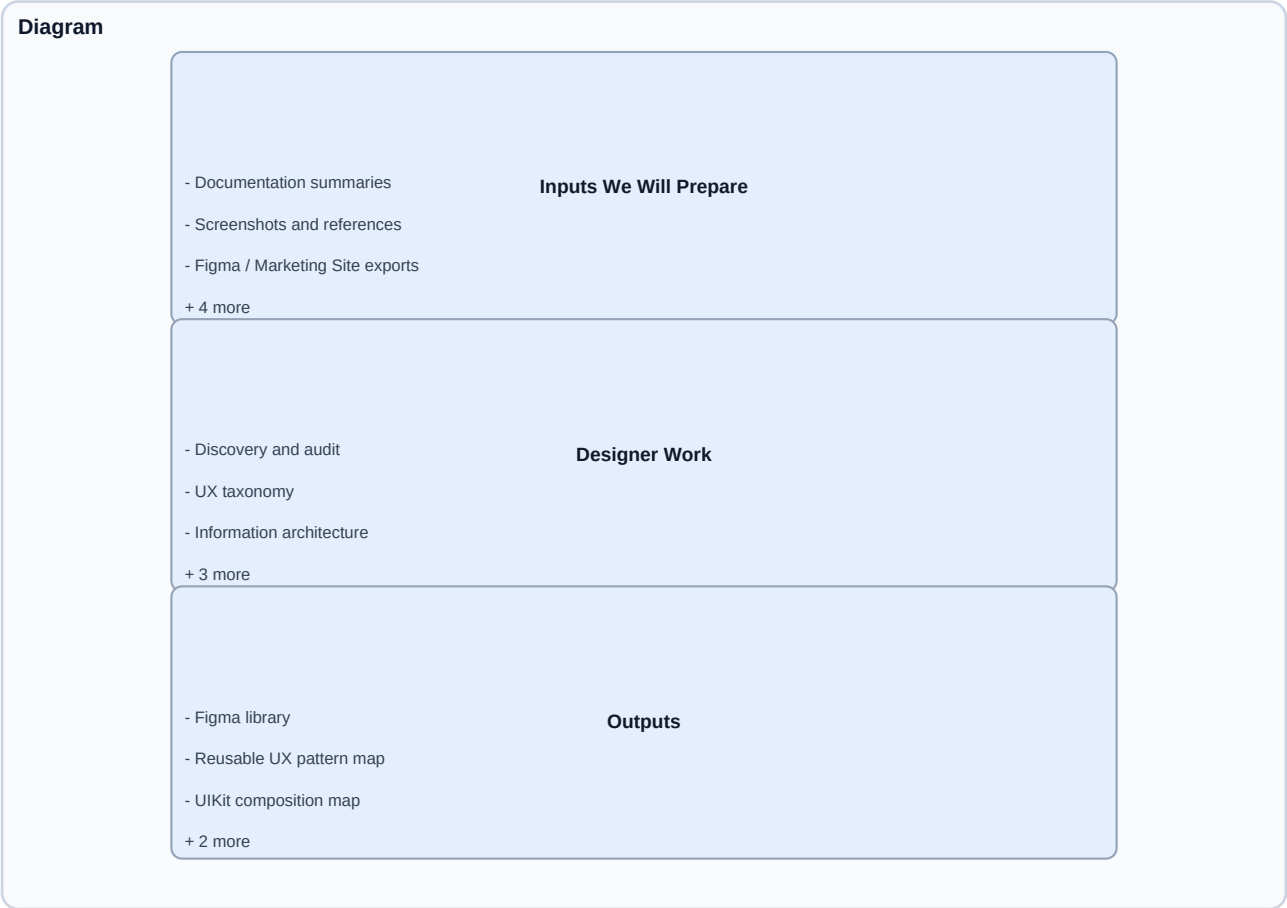
The goal is to create a practical design system and reference product experience that can later be implemented by our developers and AI coding agents.

This is not a request for isolated page mockups. We need a scalable UI/UX foundation that can support many products and modules over time.

Project Context

ConnectSoft is an ecosystem that includes:

- AI Software Factory Studio
- Enterprise SaaS platforms
- Admin portals
- Identity and authorization modules
- Configuration and feature flag modules
- Audit and compliance modules
- Billing, metering, and entitlements
- Shell and microfrontend applications
- Marketing website
- Documentation websites
- Reusable Blazor UIKit components
- Flowbite-based adapter layer
- Many planned SaaS modules described in documentation



The key product idea is the **ConnectSoft AI Software Factory**: a system that turns business ideas, blueprints, product definitions, and domain models into production-ready SaaS systems through AI agents, workflows, quality gates, deployment, observability, and feedback loops.

The design system must also support a broader SaaS ecosystem. Documentation already describes many planned, started, documented, and live modules, including identity, authorization, auditing, configuration, feature flags, billing, entitlements, metering, notifications, integrations, marketplaces, vertical SaaS products, and more.

There is also an existing **Marketing Site** that was designed in Figma and implemented in code. The content of that site may change, but its visual language, spacing, layout rhythm, and design-token direction should be treated as an important visual starter.

We also want to use the **Azure Cloud Portal** as an enterprise UX reference for product and admin surfaces. This does not mean copying Azure branding or visual identity. It means using Azure Portal as a benchmark for dense operational UX: persistent shell navigation, global search, command bars, resource/service menus, dashboards, monitoring, IAM-style access control, settings, auditability, and resource detail pages.

What We Will Provide

Before design starts, we will prepare a design input package for the designer.

The package will include the following materials.

1. Product Overview

A concise overview of the ConnectSoft ecosystem and how the major product families relate to each other.

This will explain:

- ConnectSoft as an ecosystem of SaaS platforms, templates, microservices, microfrontends, documentation, and AI-assisted delivery tools
- The relationship between public marketing surfaces, documentation surfaces, admin portals, SaaS modules, and AI Software Factory Studio
- The difference between live products, documented products, in-progress templates, planned modules, and future catalog items
- The intended product feel: enterprise-grade, technical, calm, operational, precise, reusable, and scalable

The designer should use this to understand the whole product landscape before designing individual screens.

2. AI Software Factory Overview

A product summary of the AI Software Factory and its core lifecycle:

```
Vision / Blueprint
→ Planning
→ Agent Orchestration
→ Generation
→ Quality Gates
→ Deployment
→ Observability
→ Feedback / Evolution
```

This will include short explanations of the main entities:

- **Blueprint** - structured product, domain, or feature input
- **Factory Run** - one execution of the factory lifecycle
- **Workflow** - deterministic sequence of phases, gates, retries, and handoffs
- **Agent** - specialized AI role such as planner, architect, programmer, tester, ops, security, docs, or designer
- **Agent Cluster** - group of agents working on a phase or product slice
- **Quality Gate** - validation checkpoint for tests, security, contracts, SLOs, policies, approvals, or deployment readiness
- **Trace** - audit trail connecting blueprint, agents, outputs, decisions, and artifacts
- **Artifact** - generated code, UI, docs, infra, tests, APIs, diagrams, or deployment assets
- **Approval** - human decision point when the workflow requires explicit review

The AI Software Factory Studio should be treated as the human command center for the factory. It is not a chatbot and not an IDE. Its job is to make autonomous work observable, steerable, reviewable, and governed for multiple tenants and projects.

The AI Factory documentation identifies these designer-facing Studio surfaces:

```
Project Command Center
Blueprint Designer
Agent Flow Explorer
Artifact Lineage Browser
Knowledge Graph Explorer
Human Review Center
QA Center
Security Center
DevOps Center
Runtime Center
Cost Center
Marketplace Install Center
Admin / Settings
```

The current Studio documentation also describes operational modules that map into those final-state surfaces:

```
Platform Status Dashboard
Agent Orchestration Manager
SaaS Factory Workflows Launcher
Knowledge Base & Document Management
Security & Access Control
Observability & Metrics
Versioning & History Tracking
Cost & Resource Optimization
Governance & Compliance Center
```

The designer should understand the core AI Factory workflow domains:

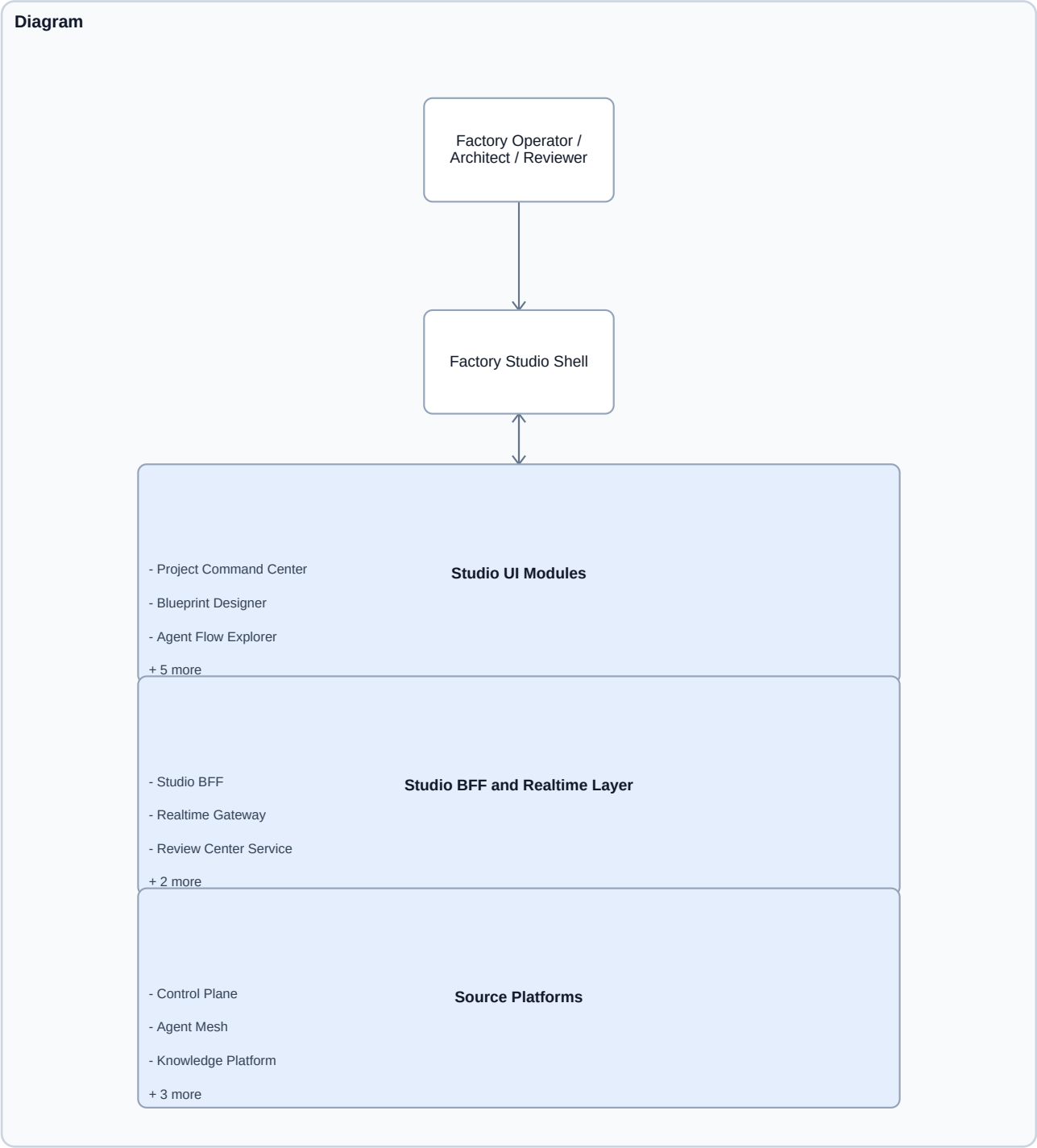
```
Vision and Product Planning
Research and UX/UI Design
Enterprise and System Architecture
Software Engineering
Testing and Quality Assurance
Security and Compliance
Deployment and Observability
Monitoring and Observability
Documentation and Knowledge Management
Customer Success
Growth and Marketing
Platform Evolution
Platform Cost Optimization
SaaS Template Selection
```

Important UI states for the factory experience:

```
draft
queued
running
parallel execution
waiting for agent
waiting for quality gate
waiting for human review
blocked
retrying
failed
passed
approved
rejected
overridden
deployed
observing
optimizing
archived
```

Important AI Factory artifacts and views:

```
factory run summary
phase timeline
agent task graph
event stream
quality gate result
review request
review decision
generated artifact preview
artifact diff
artifact lineage
knowledge graph neighbourhood
trace/log detail
cost and token usage
runtime signal
security/governance finding
version history
notification/escalation
```



The designer should use this to design the Factory Studio experience without falling into a generic "AI dashboard" pattern.

3. SaaS Ecosystem Overview

A summary of the SaaS ecosystem catalog, including the fact that ConnectSoft documentation describes roughly 1,500 candidate products, services, modules, and components across 30 categories.

Important clarification: this does **not** mean 1,500 unique screens or 1,500 microservices. Most items are reusable module types, service modules, UI modules, connectors, workflow templates, AI agents, or vertical solution packs.

The package will summarize major platform categories such as:

- SaaS Core Platform and Tenant Control Plane
- Identity, Auth, Access, and Security
- Configuration, Feature Flags, Entitlements, and Policy
- Billing, Subscriptions, Payments, and Revenue
- Notifications, Messaging, Email, SMS, and Webhooks
- Audit, Compliance, Privacy, and Governance
- Data Platform, Analytics, AI, and Knowledge
- APIs, Integrations, Connectors, and Developer Ecosystem
- Cloud Infrastructure, SRE, Observability, and Runtime
- Developer Experience, SDLC, QA, and Software Factory
- CRM, Customer Success, Support, and Service Ops
- Marketing, E-commerce, Finance, Healthcare, Education, Logistics, and other vertical solution families

The designer should use this to create reusable UX patterns, not one-off module designs.

4. Existing Marketing Site Screenshots

Screenshots from the existing ConnectSoft Marketing Site.

These will show:

- Public homepage structure
- Hero layout and first-screen visual rhythm
- Header and footer behavior
- Feature grid patterns
- CTA sections
- Pricing or plan-style layouts
- Testimonials and social proof patterns
- Comparison-table patterns
- Contact and lead-capture sections
- Responsive behavior where available

Important: current marketing content may change. The designer should treat these screenshots as visual and structural references, not as final content.

5. Existing Figma-Derived Visual References

Exports and screenshots from the existing Marketing Site Figma design.

These will be used to understand:

- Color direction
- Typography direction
- Spacing rhythm

- Section composition
- Button styles
- Card styles
- Visual density
- Public brand expression
- Responsive layout assumptions

The designer should use these as a starting point for visual continuity, while adapting the system for enterprise SaaS and operational product UI.

5a. Azure Cloud Portal UI/UX Reference

Reference notes and screenshots from Azure Portal and Microsoft Fluent design guidance.

These will be used to understand enterprise-grade cloud portal patterns:

- Unified web console for creating, managing, and monitoring resources
- Persistent global shell with portal menu, page header, global search, notifications, settings, support/help, and account controls
- Contextual command bars for the current resource or module
- Resource/service menu with grouped navigation, search, favorites, and remembered expanded/collapsed state
- Working pane for overview, properties, monitoring, activity, access control, settings, and related operations
- Dashboard model with customizable tile grids, private/shared dashboards, pinned resources, role-specific dashboards, and project/task-oriented workspaces
- Access Control / IAM pattern with scoped role assignments, role selection, members, conditions, review, and disabled actions when permissions are missing
- Monitoring pattern with scope picker, metric selection, time range, filters, chart interaction, alerts, sharing, and dashboard pinning
- High-density tables and lists that support filtering, sorting, search, column customization, status, and bulk actions
- Clear separation between global navigation, resource navigation, page actions, and page content

Apply this reference to ConnectSoft as follows:

```
Azure Portal shell pattern
→ ConnectSoft Shell / Microfrontend Host

Azure resource overview page
→ SaaS module detail page

Azure service menu
→ module navigation / tenant admin navigation / Studio module navigation

Azure dashboard tile grid
→ tenant dashboard / Factory Studio command center / observability dashboards

Azure Access Control (IAM)
→ identity, authorization, policy, role assignment, tenant access screens

Azure Monitor metrics explorer
→ runtime health, factory run metrics, cost, quality, and observability screens

Azure command bar
→ create, refresh, pin, export, approve, reject, retry, deploy, disable, delete actions
```

The designer should use Azure Portal as a UX maturity reference for enterprise operations, not as a visual skin. ConnectSoft should remain visually aligned with our Marketing Site/Figma direction and UIKit implementation path.

Useful external references:

```
Azure Portal overview:
https://learn.microsoft.com/en-us/azure/azure-portal/azure-portal-overview

Azure Portal dashboards:
https://learn.microsoft.com/en-us/azure/azure-portal/azure-portal-dashboards

Azure RBAC role assignments in portal:
https://learn.microsoft.com/en-us/azure/role-based-access-control/role-assignments-portal

Azure Monitor metrics explorer:
https://learn.microsoft.com/en-us/azure/azure-monitor/metrics/analyze-metrics

Microsoft Fluent 2:
https://fluent2.microsoft.design/
```

6. Existing UIKit / Component Inventory

A summarized inventory of existing reusable UI components from ConnectSoft.Blazor.UIKit and related templates.

Expected component areas include:

- Buttons and async buttons
- Badges and indicators
- Cards and layout cards
- Tables and data grids
- Tabs and accordions
- Breadcrumbs and navigation
- Navbar, dashboard navbar, footer, dashboard footer
- Drawer and modal/dialog patterns
- Toast notifications
- Forms, text fields, selects, checkboxes, radio groups, date pickers
- Progress bars, spinners, skeletons, busy overlays
- Avatars and user cards
- Pagination and utility components
- Application shell and dashboard shell patterns

The package will also identify whether each component is:

```
available
available but needs visual refinement
missing
should become a higher-level composition
product-specific and not part of UIKit
```

The designer should use this to avoid designing components that cannot map cleanly to implementation.

7. SaaS Module Taxonomy

A classification of SaaS modules and product areas into reusable UX types.

Example taxonomy:

```
Platform Product
Portal / UI Module
Microfrontend
Module-in-service
Workflow Template
Connector
AI Agent
Shared Library
Vertical Solution Pack
Marketplace Item
```

Example module groups:

- Tenant management
- Product catalog
- Product editions
- Feature catalog
- Entitlements
- Billing
- Metering
- Identity
- Authorization server
- OAuth client management
- Users, roles, permissions
- RBAC / ABAC policies
- Configuration server
- Tenant settings
- Feature flags
- Experiments and rollouts
- Audit logs
- Compliance reporting
- Notifications and webhooks
- Integrations and connectors
- Observability and health checks
- Developer portal
- Marketplace and module catalog

The designer should use this taxonomy to define repeatable screen patterns.

8. Shell / Microfrontend Architecture Summary

A non-technical summary of how ConnectSoft expects portals and modules to be composed.

This will explain:

- Global Shell as the host application

- Microfrontends as feature/product modules inside the Shell
- Shared navigation, top bar, side bar, breadcrumbs, command bar, and module switchers
- Tenant switcher and product/module switcher
- Shared loading, error, permission, and empty states
- Shared entity-detail and settings-page patterns
- How a module can be independently developed but still feel like part of one product

The designer should use this to design layouts that scale across many modules without each module inventing its own navigation.

9. Existing Design Tokens

A summarized token package from existing ConnectSoft design assets and code.

Token areas include:

- Primary blue scale
- Secondary cyan/teal scale
- Neutral gray scale
- Success, warning, error, and info colors
- Light and dark surfaces
- Text colors
- Spacing scale
- Border radius scale
- Shadows
- Typography scale
- Font stacks, including Hebrew/RTL considerations
- Dark mode support
- RTL/Hebrew support

Additional status tokens will be needed for product and Factory experiences:

```
running
waiting
blocked
failed
passed
deployed
observing
draft
archived
permission denied
waiting for approval
```

The designer should normalize these into a clean design-token system and identify gaps.

10. Reference Screenshots From Current Products / Templates

Screenshots or rendered examples from current product templates and module starters.

These may include:

- Marketing Site
- Marketing Site Template
- Blazor Shell Template
- Blazor Microfrontend Template
- Authorization Server Template
- Identity Backend Template
- SaaS Tenants Template
- SaaS Products Catalog Template
- SaaS Entitlements Template
- SaaS Metering Template
- SaaS Billing Template
- Documentation Site Template
- Admin-style Blazor microfrontends

The designer should use these to understand the current implementation baseline and where visual consistency is missing.

11. List Of Required Screens

A prioritized list of screens we need designed or standardized.

The proposed order starts with clearer SaaS/admin surfaces first, then moves into AI Software Factory-specific screens:

```
Shell / Microfrontend Host Layout
Tenant Admin Dashboard
Host Admin Dashboard
Super Admin Dashboard
Tenant Onboarding Wizard
Product Catalog Module
Product Editions Module
Feature Catalog Module
Entitlements Module
Billing / Subscriptions Module
Metering / Usage Module
Identity / Authorization Module
OAuth Clients Module
Users / Roles / Permissions Module
Policy Management Module
Configuration / Feature Flags Module
Tenant Settings Module
Experiment / Rollout Module
Notifications / Webhooks Module
Integrations / Connectors Module
Audit / Compliance Module
Observability / Health Checks Module
Developer Portal Module
Marketplace / Module Catalog Pattern
Marketing Landing Page Pattern
Documentation Landing Page Pattern
AI Factory Studio Overview
SaaS Factory Workflows Launcher
Project Command Center
Blueprint Designer
Platform Status Dashboard
Agent Orchestration Manager
Agent Flow Explorer
Artifact Lineage Browser
```

```
Knowledge Graph Explorer
Human Review Center
QA Center
Security Center
DevOps Center
Runtime Center
Cost Center
Governance / Compliance Center
Security / Access Control Center
Knowledge Base Management
Versioning / History Tracking
Factory Control Tower
Blueprint Intake
Vision-to-Production Timeline
Agent Execution Graph
Quality Gates Dashboard
Trace Viewer
Artifact Explorer
Human Approval Queue
```

The designer should help decide what belongs in MVP and what can be deferred.

12. UX Patterns We Need Standardized

Reusable UX patterns that should work across many modules:

```
Dashboard overview
Customizable dashboard tile grid
Pin to dashboard / save view / share view
List + filters + search + sorting
Bulk actions
Entity detail page
Resource overview page
Resource/service side navigation
Contextual command bar
Create/edit form
Settings page
Policy editor
IAM-style role assignment wizard
Permission-disabled action states
Feature flag rollout
Approval workflow
Audit log
Activity timeline
Integration setup wizard
Billing/subscription page
Tenant onboarding wizard
Workflow timeline
Execution graph
Quality gate status
Trace detail panel
Artifact diff
Artifact lineage browser
Knowledge graph explorer
Agent orchestration board
Workflow launcher
Human review queue
Review decision detail
Event stream / live updates
Run replay / version history
Cost and token usage dashboard
Provider/model policy panel
Security/governance findings panel
DevOps pipeline board
Runtime health board
Metric explorer with scope, time range, filters, and chart controls
Marketplace listing
Documentation landing page
Marketing conversion section
Empty/loading/error states
Permission-aware actions
```

The designer should define these as reusable patterns with examples, variants, and states.

13. Technical Implementation Constraints

The designer does not need to implement code, but the design should respect the implementation direction.

Implementation model:

```
Product pages
→ reusable UI compositions
→ reusable UI primitives
→ Flowbite-based adapters
→ runtime implementation
```

Important constraints:

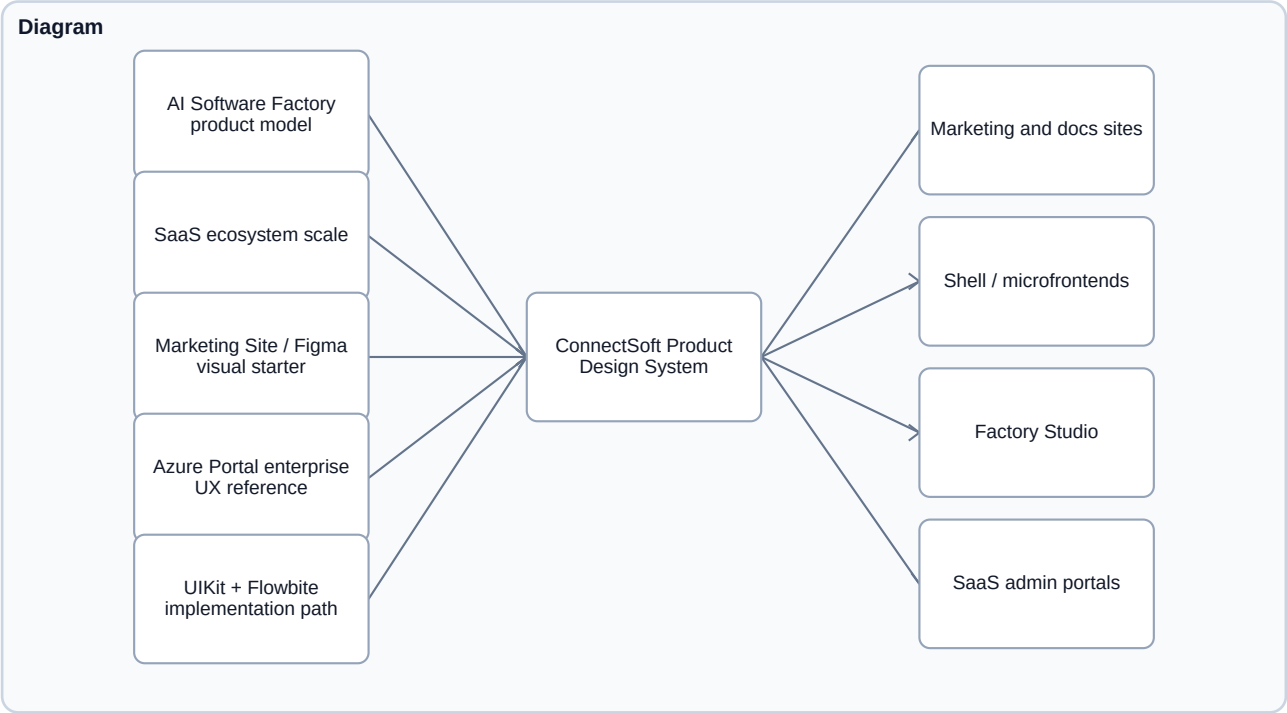
- Reusable design should map to ConnectSoft.Blazor.UIKit where possible
- Flowbite is a rendering/behavior foundation, not the public design API
- Product pages should not rely on large one-off custom blocks when a reusable composition is appropriate
- Shell and microfrontend layouts need shared navigation and shared states
- Dark mode and RTL/Hebrew should be considered early
- Accessibility should follow enterprise expectations
- Components should include states, variants, and responsive behavior
- Handoff should be detailed enough for developers and AI coding agents to implement consistently

The designer should estimate based on this package, not by inspecting the repositories directly.

Main Design Goal

Create a ConnectSoft Product Design System that defines:

- How ConnectSoft products should look
- How AI Software Factory Studio should work
- How SaaS admin modules should be structured
- How Shell and microfrontend apps should share layout and navigation
- How marketing, docs, and product UI should visually relate
- What reusable components and compositions are needed
- How developers should implement future screens consistently
- How AI coding agents should consume and follow the design system



Design Direction

The UI should feel:

```
enterprise-grade
technical
calm
precise
modern
trustworthy
AI-powered but not flashy
operational rather than decorative
scalable across many SaaS modules
consistent with existing Marketing Site/Figma
informed by Azure Cloud Portal enterprise UX patterns
```

Avoid:

```
generic AI dashboard look
random SaaS template look
too much decoration
one-off custom screens
unnecessary visual noise
marketing fluff inside product UI
copying Azure visual identity or Microsoft branding
```

Important Product Model

The AI Software Factory experience should be designed around this lifecycle:

```
Vision / Blueprint
→ Planning
→ Agent Orchestration
→ Generation
→ Quality Gates
→ Deployment
→ Observability
→ Feedback / Evolution
```

Important concepts:

```
Blueprint
Factory Run
Workflow
Agent
Agent Cluster
Task
Trace
Quality Gate
Artifact
Deployment
Project
Tenant
Provider
Channel
Approval
Feedback Loop
```

SaaS Areas To Support

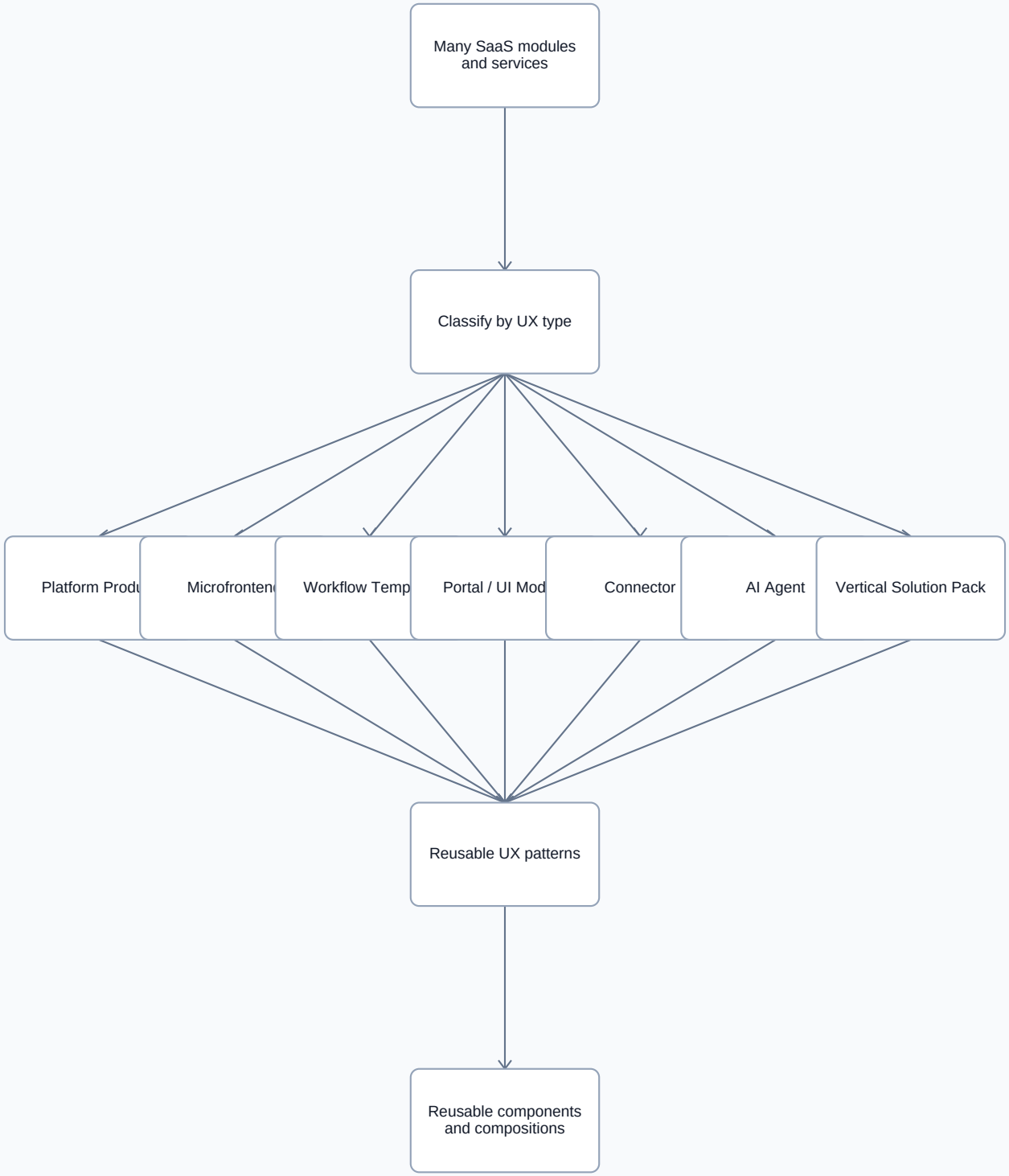
The design system should support product areas such as:

```
SaaS Control Plane
Host Admin Portal
Tenant Admin Portal
Super Admin Portal
Tenant Onboarding
Product Catalog
Product Editions
Feature Catalog
Entitlements
Billing
Subscriptions
Metering
Identity
Authorization Server
OAuth Clients
Users
Roles
Permissions
RBAC / ABAC Policies
Audit Logs
Compliance
Configuration Server
Tenant Settings
Feature Flags
Experiments
Canary Releases
```

```
Kill Switches
Notifications
Webhooks
Integrations
Developer Portal
Marketplace
Observability
Health Checks
```

The goal is not to design every module separately. The goal is to create reusable UX patterns that can support them.

Diagram



SaaS Ecosystem Scale

The broader ConnectSoft ecosystem catalog contains roughly **1,500 candidate products, services, modules, and components across 30 categories**.

This does **not** mean we need 1,500 unique UI designs.

Most of these should be classified into reusable UX patterns and reusable UI compositions, such as:

```
Platform Product
Portal / UI Module
Microfrontend
Module-in-service
Workflow Template
Connector
AI Agent
Shared Library
Vertical Solution Pack
Marketplace Item
```

The designer should help us reduce this complexity into reusable patterns.

Shell And Microfrontend Requirements

ConnectSoft uses or plans to use Blazor Shell and microfrontend patterns.

The design system should include shared patterns for:

```
Global application shell
Tenant switcher
Product/module switcher
Side navigation
Top navigation
Breadcrumbs
Command bar
Shared filters
Microfrontend loading state
Microfrontend error state
Entity detail layout
Settings layout
Audit/activity panel
Permission-aware actions
```

Existing Marketing Site / Figma Role

We already have a Marketing Site that was designed in Figma and implemented.

The content may change, but its visual direction should be used as a starter for:

```
brand expression
colors
typography
spacing
layout rhythm
hero sections
CTA sections
pricing sections
feature grids
testimonials
comparison tables
header/footer behavior
responsive behavior
```

Do not treat current marketing copy as final product truth. Content will change.

Azure Portal Reference Role

Azure Cloud Portal should be used as a practical reference for enterprise product UX.

Use it for:

```
global shell and portal navigation
resource/service menu organization
resource overview and detail page structure
contextual command bars
dashboard tile grid behavior
pinning, saving, sharing, and cloning dashboard views
IAM-style access control workflows
monitoring and metrics exploration
scope pickers and time range filters
permission-aware disabled actions
activity, audit, and notification patterns
```

Do not use Azure Portal as:

```
ConnectSoft visual identity
marketing style
exact component skin
exact navigation taxonomy
license to overcomplicate simple SaaS modules
```

The target is an Azure-like level of operational maturity with ConnectSoft's own brand, component system, and product model.

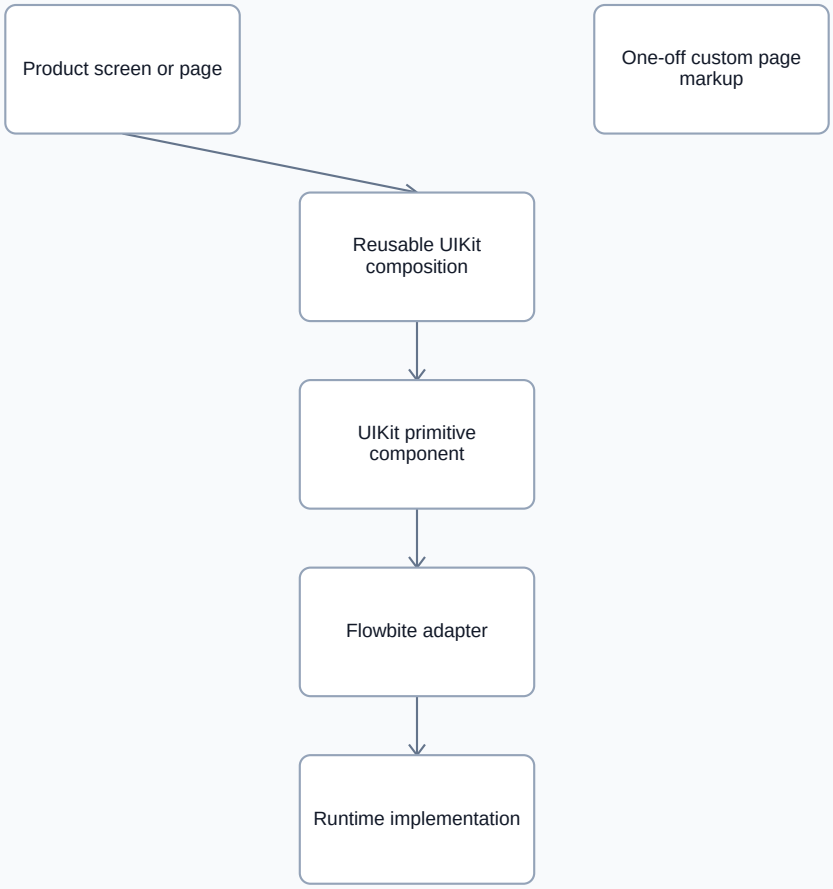
Implementation Constraint

Our implementation direction is:

```
Product pages
→ reusable UI compositions
→ reusable UI primitives
→ Flowbite-based adapters
→ runtime implementation
```

The designer does not need to implement this, but the design should be structured so it can be translated into reusable components.

Diagram


Product screen or page
One-off custom page markup
Reusable UIKit composition
UIKit primitive component
Flowbite adapter
Runtime implementation

Expected Designer Deliverables

Please estimate the work for the following deliverables.

1. Discovery Review

Review the design input package we provide.

Output:

```
observations
risks
missing information
recommended design direction
scope clarification
```

2. UX Taxonomy

Create a clear classification of product/module types.

Output:

```
product/module taxonomy
screen type taxonomy
reusable UX pattern map
recommendation for MVP scope
```

3. Information Architecture

Define navigation and hierarchy for:

```
AI Software Factory Studio
SaaS Admin Portal
Tenant Admin
Host Admin
Super Admin
Shell/microfrontend apps
Marketing/docs relationship
```

Output:

```
navigation model
app shell model
main user journeys
module hierarchy
role/access considerations
```

4. Design System Foundation

Define:

```
colors
typography
spacing
radius
shadows
icons
status colors
agent/workflow colors
dark mode approach
RTL/Hebrew approach
accessibility rules
responsive rules
```

Output:

```
design tokens
visual principles
usage rules
component style guidance
```

5. Component And Composition Design

Define reusable UI elements.

Primitives:

```
Button
Badge
Card
Table
Tabs
Accordion
Drawer
Modal
Toast
Form controls
Breadcrumbs
Pagination
Progress
Skeleton
Avatar
Navigation
```

Compositions:

```
FactoryRunCard
AgentExecutionGraph
QualityGatePanel
TraceViewerPanel
ArtifactExplorer
TenantSwitcher
ProductSwitcher
PolicyEditor
FeatureFlagRolloutPanel
AuditTimeline
BillingPlanCard
SettingsLayout
MarketingHeader
MarketingFooter
PricingCards
FeatureGrid
ConversionCta
```

Output:

```
Figma components
component documentation
states
variants
responsive behavior
```

6. Reference Screens

Prepare key high-fidelity screens.

The execution order should start with the clearer SaaS/admin product surfaces first, then move into the more specialized AI Software Factory experience.

Suggested first set:

```
Shell / Microfrontend Host Layout
Tenant Admin Dashboard
```

```
Host Admin Dashboard
Super Admin Dashboard
Tenant Onboarding Wizard
Product Catalog Module
Product Editions Module
Feature Catalog Module
Entitlements Module
Billing / Subscriptions Module
Metering / Usage Module
Identity / Authorization Module
OAuth Clients Module
Users / Roles / Permissions Module
Policy Management Module
Configuration / Feature Flags Module
Tenant Settings Module
Experiment / Rollout Module
Notifications / Webhooks Module
Integrations / Connectors Module
Audit / Compliance Module
Observability / Health Checks Module
Developer Portal Module
Marketplace / Module Catalog Pattern
Marketing Landing Page Pattern
Documentation Landing Page Pattern
AI Factory Studio Overview
SaaS Factory Workflows Launcher
Project Command Center
Blueprint Designer
Platform Status Dashboard
Agent Orchestration Manager
Agent Flow Explorer
Artifact Lineage Browser
Knowledge Graph Explorer
Human Review Center
QA Center
Security Center
DevOps Center
Runtime Center
Cost Center
Governance / Compliance Center
Security / Access Control Center
Knowledge Base Management
Versioning / History Tracking
Factory Control Tower
Blueprint Intake
Vision-to-Production Timeline
Agent Execution Graph
Quality Gates Dashboard
Trace Viewer
Artifact Explorer
Human Approval Queue
```

Output:

```
high-fidelity desktop screens
responsive/mobile variants where needed
key states
interaction notes
```

7. UX States

Define common states:

```
empty
loading
skeleton
error
blocked
permission denied
waiting for approval
running
completed
failed
deployed
observing
draft
```

```
archived
```

Output:

```
state library
status language
visual treatment rules
```

8. Developer Handoff

Prepare implementation-ready guidance.

Output:

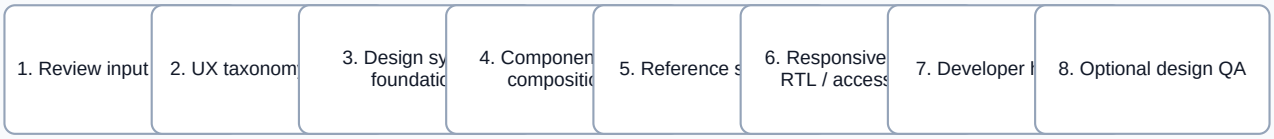
```
Figma handoff
component specs
token specs
responsive rules
accessibility notes
implementation notes
developer checklist
AI coding agent instructions
```

Suggested Work Phases

Please estimate by phase:

```
Phase 1 - Review input package and clarify scope
Phase 2 - UX taxonomy and information architecture
Phase 3 - Design system foundation
Phase 4 - Component and composition design
Phase 5 - Reference screens
Phase 6 - Responsive, dark mode, RTL, accessibility
Phase 7 - Developer handoff
Phase 8 - Optional design QA during implementation
```

Diagram



Estimate Requested

Please provide:

```
scope included
scope excluded
```

```
timeline
milestones
price
required inputs
risks
assumptions
optional add-ons
cost per additional screen
cost per additional component
cost for mobile/responsive variants
cost for dark mode
cost for RTL/Hebrew support
cost for design QA during implementation
```

Questions For The Designer

Please answer:

```
1. What minimum input package do you need from us before starting?
2. How much discovery time do you recommend?
3. Would you split this into MVP and full design system?
4. How many reference screens should be included in MVP?
5. How many components/compositions should be included in MVP?
6. What can be estimated fixed-price?
7. What should remain time-and-materials?
8. What budget range do you recommend?
9. What timeline is realistic for first usable design package?
10. What timeline is realistic for a full design system?
```

Recommended MVP Scope

If the full scope is too large, propose an MVP around:

```
input package review
core visual direction
design tokens
global shell layout
Tenant Admin Dashboard
Identity Module
Feature Flags Module
Audit Log Module
AI Factory Studio Overview
Project Command Center
SaaS Factory Workflows Launcher
Blueprint Designer
Agent Flow Explorer
QA Center / Quality Gates Dashboard
Human Review Center
Artifact Lineage Browser
Trace Viewer
Cost Center
Security / Governance Center
core component map
developer handoff
```

What We Need From The Designer

We need the designer to help us turn a complex ecosystem into a clear, reusable design system.

The designer should not try to design hundreds of modules one by one. Instead, the designer should help define the reusable patterns, components, and reference screens that make future modules easy to design and implement consistently.

Final Principle

The design should combine:

```
AI Software Factory product model
SaaS ecosystem scale
existing Marketing Site/Figma visual starter
reusable UIKit/component implementation path
```

The output should give us a practical foundation for future product design, development, and AI-assisted implementation.