

RHDH Portal Customization & Copilot Agents Analysis

Awesome-Copilot Agents Selection + RHDH UX Customization Strategy
Three Horizons Implementation Accelerator

Date: February 28, 2026

Version: 1.0

Client: Microsoft LATAM — Platform Engineering

Team: Latam Software GBB

Author: paulasilva@microsoft.com

Microsoft Confidential

Table of Contents

[Right-click and select 'Update Field' to refresh Table of Contents]

Executive Summary

This document analyzes two interconnected work streams for the Three Horizons project: (1) which agents from the github/awesome-copilot repository should be added to our .github/agents/ directory to support portal UX/UI customization, and (2) the complete strategy for customizing Red Hat Developer Hub (RHDH) — including the login page, homepage, entity tabs, sidebar navigation, and custom pages. All analysis is based on the official RHDH 1.8 documentation, the project's existing new-features/ folder, and the awesome-copilot agents catalog.

Awesome-Copilot Agents — Recommended for Our Project

The github/awesome-copilot repository contains 50+ community-contributed agents. Based on our project needs — customizing the RHDH portal UI, designing login/home pages, and building React frontend components — the following agents are directly relevant and should be added to .github/agents/ in our repository.

Tier 1 — Must-Have (Directly Relevant)

Agent	File	Why We Need It
SE UX/UI Designer	se-ux-ui-designer.agent.md	Jobs-to-be-Done analysis, user journey mapping, and UX research artifacts for Figma and design workflows. Critical for defining the portal user experience, creating user stories, and mapping the developer journey through the RHDH tabs.
Expert React Frontend Engineer	expert-react-frontend-engineer.agent.md	React 19.2 specialist with hooks, Server Components, TypeScript, Fluent UI, Shadcn/ui, and design system architecture. Essential for building Backstage custom React components (SignInPage, HomePage, theme).
AEM Frontend Specialist	aem-frontend-specialist.agent.md	HTL, Tailwind CSS, and Figma-to-code workflows with design system integration. Useful for translating Figma designs into implemented components for the RHDH portal theme.

Tier 2 — Strongly Recommended

Agent	File	Why We Need It
Architectural Decision Record (ADR)	adr.agent.md	Creates comprehensive ADRs with Mermaid diagrams. Perfect for documenting portal customization decisions (theme choices, plugin selections, tab layouts).
SE UX Research	se-ux-researcher.agent.md	UX research specialist for user flow validation and browser testing. Validates our portal customization using automated UI testing.
Technical Writer	se-technical-writer.agent.md	Documentation specialist for developer docs and tutorials. Creates user guides for the customized RHDH portal.
SE Product Manager	se-product-manager.agent.md	Product management guidance for creating GitHub issues, aligning business value with user needs. Helps structure user stories for the customization work.
Responsible AI	se-responsible-ai.agent.md	Ensures accessibility compliance (WCAG), inclusive design. Critical for enterprise portal accessibility.

Tier 3 — Nice to Have

Agent	File	Why We Need It
Expert Vue.js Engineer	expert-vue-frontend-engineer.agent.md	Vue 3 Composition API specialist. Some Backstage community plugins use Vue.
Expert Next.js Developer	expert-nextjs-developer.agent.md	Next.js 16 specialist with App Router. Useful if we build a

		standalone landing page or documentation site.
SEO / AEO / GEO Specialist	se-seo-aeo-geo-specialist.agent.md	Search optimization expert. Useful for TechDocs discoverability within the portal.
Modernization Advisor	modernization-advisor.agent.md	Human-in-the-loop modernization assistant. Helpful for planning the migration from basic to customized portal.

Software Engineering Team Plugin (7-Agent Bundle)

The awesome-copilot repo also offers a pre-packaged plugin called 'software-engineering-team' that bundles 7 agents covering the full SDLC — from UX design to security. This can be installed in one command:

```
copilot plugin install software-engineering-team@awesome-copilot
```

This plugin includes: SE UX/UI Designer, SE Architect, SE Security Reviewer, SE DevOps, SE Product Manager, SE Responsible AI, and SE Technical Writer. It covers most of our Tier 1 and Tier 2 needs in a single installation.

How to Add These Agents

1. Clone or download the agent .md files from [github/awesome-copilot/agents/](https://github.com/awesome-copilot/agents/)
2. Place them in .github/agents/ in our repository
3. Each file uses YAML frontmatter with tools, description, and instructions
4. Agents become available in VS Code Copilot Chat, CCA (Copilot Coding Agent), and Copilot CLI
5. Configure handoffs between new agents and our existing 17 agents (e.g., @se-ux-ui-designer → @platform → @template-engineer)
6. Add the MCP server integration if the agent requires external tools

RHDH Customization — Complete Guide

RHDH 1.8 provides extensive customization capabilities, all driven by YAML configuration — no code changes required. Below is the complete breakdown of every customizable area, mapped to our Three Horizons project requirements.

3.1 — Login Page (SignInPage)

The SignInPage is the first thing developers see. In RHDH, there are two approaches to customize it:

Approach	How	Level of Customization
YAML Branding (recommended)	Configure app.branding in app-config-rhdh.yaml — set logo, colors, title, favicon	Colors, logo, title, favicon, font. The sign-in page inherits the configured theme automatically.
Custom SignInPage Component (advanced)	Create a dynamic plugin that exports a CustomSignInPage component. Reference it in dynamic-plugins.yaml via signInPage.importName	Full React component control — custom layout, animations, marketing copy, multiple auth provider buttons.

Our project already uses the YAML approach (Microsoft-branded login with GitHub OAuth). To add a fully custom sign-in page with marketing content, hero image, or animated background, we would need to create a custom dynamic plugin that exports a SignInPage React component. The RHDH docs (Chapter 8.8) describe the exact contract: the plugin must return a component that connects authentication provider factories.

3.2 — Homepage Customization

The RHDH homepage uses a 12-column grid system with configurable cards. RHDH 1.8 provides two modes:

Mode	Component	Description
Read-Only	DynamicHomePage	Admin-configured layout. Users see the cards but cannot rearrange them.
Interactive	DynamicCustomizableHomePage	Users can drag-and-drop, resize, add/remove widgets, and restore defaults. Settings

		persist via Backstage Storage API.
--	--	------------------------------------

Available Homepage Cards:

Card	importName	Description
Search Bar	SearchBar	Full-text search across catalog, templates, docs
Quick Access	QuickAccessCard	Links to key resources (configured via JSON)
Headline	Headline	Display announcements or status messages
Markdown	MarkdownCard	Render markdown content from URL or inline
Placeholder	Placeholder	Debug/layout testing card with borders
Catalog Starred Entities	CatalogStarredEntitiesCard	User's starred catalog items
Featured Docs	FeaturedDocsCard	Highlighted documentation links
Onboarding Section	OnboardingSection	Getting started guidance (default)
Entity Section	EntitySection	Recent/owned entities (default)
Template Section	TemplateSection	Software templates gallery (default)

Layout configuration example (grid-based):

Each card receives layouts for responsive breakpoints (xl, lg, md, sm, xs, xxs) with w (width in columns out of 12), h (height), and optional x (column offset). This allows 1-column, 2-column, or 3-column layouts depending on the screen size.

3.3 — Sidebar Navigation

The sidebar has two configurable parts: Main Menu Items (static core navigation) and Dynamic Plugin Menu Items (added by enabled plugins). Both are configured in `app-config.yaml`.

Configuration	Where	What It Controls
Menu item order & hierarchy	<code>dynamicPlugins.frontend.default.main-menu-items</code>	Order, priority, enable/disable, parent-child grouping of core items (Home, Catalog, APIs, etc.)
Plugin menu items	<code>dynamicPlugins.frontend.<package>.menuitems</code>	Sidebar entries added by dynamic plugins (Lightspeed, RBAC Admin, Topology, etc.)
Custom SidebarItem component	<code>dynamicRoutes[].menuItem.importName</code>	Use a custom React component for badge notifications, counters, etc.
Localization	<code>titleKey</code> property	Translate sidebar labels into user's preferred language
Logo in sidebar	<code>app.sidebar.logo: true</code> + <code>branding.fullLogo</code>	Display organization logo in collapsed/expanded sidebar

Our project currently configures 10 sidebar items (Home, My Group, Catalog, APIs, Learning Paths, Create, Docs, Notifications, Administration, Settings). New items can be added by enabling additional dynamic plugins — each plugin that registers a route can automatically appear in the sidebar.

3.4 — Entity Tabs (Catalog Entity Detail)

When you click on a catalog entity (Component, API, System, etc.), you see detail tabs. RHDH provides 14+ default tabs that can be renamed, reordered, removed, or extended.

Default Tab	Mount Point	Plugin
Overview	<code>entity.page.overview</code>	Core

Issues	entity.page.issues	GitHub Issues
Pull/Merge Requests	entity.page.pull-requests	GitHub PR
CI	entity.page.ci	GitHub Actions / Tekton
CD	entity.page.cd	ArgoCD
Kubernetes	entity.page.kubernetes	K8s plugin
Topology	entity.page.topology	Topology plugin
Tekton	entity.page.tekton	Tekton plugin
Image Registry	entity.page.imageregistry	Quay / JFrog
Monitoring	entity.page.monitoring	Monitoring plugin
API	entity.page.api	API Docs (kind: Service/Component)
Dependencies	entity.page.dependencies	Dependencies graph (kind: Component)
Docs	entity.page.docs	TechDocs
Definition	entity.page.definition	API Definition (kind: API)
Diagram	entity.page.diagram	System diagram (kind: System)

Tab customization uses the `entityTabs` section in dynamic plugin configuration. You can set title, path, priority (higher = appears first), and a negative priority removes the tab. Each tab's internal layout also uses a responsive 12-column grid for card placement.

3.5 — Theme & Visual Identity

Element	Configuration	Options
Color Palette	branding.theme.light/dark	primaryColor, headerColor1/2, backgroundColor, sidebarBackgroundColor, navigationIndicatorColor
Font	app-config.yaml branding	Any web font family via fontFamily property

Component Variant	options section per theme	Each component (buttons, cards, tabs, tables, dialogs, etc.) can use PatternFly or MUI variant
Page Themes	pageTheme section	Custom header backgrounds, fonts, and shapes per page type (home, catalog, apis, etc.)
Custom Theme Plugin	dynamic plugin with theme provider	Load an entirely custom theme via dynamic plugin for complete control
Ripple Effect	options.rippleEffect: on/off	Enable/disable Material ripple animation on interactive elements

3.6 — Quick Access Card

The QuickAccessCard on the homepage shows categorized links (Getting Started, Development, Operations, etc.). Content comes from two possible sources:

- Hosted JSON file (recommended): Publish a JSON file on GitHub/GitLab, reference it via proxy config in app-config.yaml. Easy to update without pod restart.
- Dedicated service: Build a REST API that returns the JSON data. Useful for dynamic content that changes based on user role or team.

3.7 — Tech Radar Page

The Tech Radar page (provided by backstage-community-plugin-tech-radar) visualizes technology adoption status. Configure via a hosted JSON file or a dedicated service. Data includes: quadrants (Languages, Frameworks, Tools, Platforms), rings (Adopt, Trial, Assess, Hold), and entries with movement indicators.

3.8 — Localization (i18n)

RHDH 1.8 supports full localization of the UI, including sidebar menu items, page titles, and plugin content. Configured via the localization framework with language persistence. Sidebar items support titleKey for translation key mapping. This is important for our LATAM deployment (Portuguese/Spanish).

Project new-features/ Folder Analysis

Based on the CLAUDE.md description, the new-features/ directory in our project contains RHDH-specific feature implementations organized into three sub-folders:

Folder	Contents	Purpose
new-features/configs/	Dynamic plugin configurations, RBAC policies, Helm values	All YAML files needed to enable/configure MCP plugins, Lightspeed AI, and role-based access control for AI features (lightspeed.chat, mcp.tools.execute permissions)
new-features/foundry/	Python-based AI agents for RHDH	Azure AI Foundry agents that power Developer Lightspeed: rhdh-agent.py (catalog/template queries) and lightspeed-monitor-agent.py (health/performance monitoring)
new-features/homepage/	Customized RHDH homepage with quick actions, links, status widgets	Homepage card configurations, QuickAccessCard JSON data, Three Horizons cards, Featured Templates, and potentially custom React components for homepage widgets

These files are the RHDH-specific counterpart to the Backstage source code customization. Since RHDH uses YAML-only configuration, these files are deployed via ConfigMaps and Helm values — no container rebuild required.

Relationship to DEPLOY_CUSTOMIZATIONS_GUIDE.md

The project's DEPLOY_CUSTOMIZATIONS_GUIDE.md already documents the deployment process for both Backstage (code-based) and RHDH (YAML-based). The new-features/ folder extends this with AI-specific features that go beyond basic branding.

User Stories for Portal Customization

The following user stories define the customization work needed to create a fully branded, intuitive Three Horizons portal experience.

Epic 1: Login Experience

ID	User Story	Acceptance Criteria	Agent
US-01	As a developer, I want to see a professional Microsoft-branded login page so I trust the portal	Microsoft 4-color logo, Segoe UI font, white background, 'Sign in with GitHub' button, responsive on mobile	@se-ux-ui-designer
US-02	As an admin, I want to customize the login page message and branding without rebuilding the container	Change title, subtitle, and logo via app-config-rhdh.yaml ConfigMap only. Pod restart applies changes.	@platform
US-03	As a developer, I want single-click GitHub OAuth login	Click 'Sign in with GitHub' → GitHub OAuth flow → redirect back to RHDH homepage with user identity loaded	@security

Epic 2: Homepage Experience

ID	User Story	Acceptance Criteria	Agent
US-04	As a developer, I want to see my most-used resources immediately on the homepage	QuickAccessCard shows: Getting Started, Development Tools, AI & Automation, Operations — categorized links with icons	@se-ux-ui-designer
US-05	As a developer, I want to discover and launch Golden Path templates from the homepage	TemplateSection shows 6 featured templates (Node.js, Python, React, .NET, Spring Boot, Go) with 'Create' action	@template-engineer
US-06	As an admin, I want to communicate the Three Horizons maturity model on the homepage	Three Horizons cards (H1: Foundation/Blue, H2: Intelligence/Green, H3: Autonomy/Yellow) with descriptions and links	@platform

US-07	As a developer, I want to personalize my homepage layout	DynamicCustomizableHomePage enabled — drag-and-drop cards, resize, add/remove widgets, restore defaults	@expert-react-frontend-engineer
US-08	As a developer, I want to see my starred entities and recent work on the homepage	CatalogStarredEntitiesCard and EntitySection visible by default. OnboardingSection for new users.	@platform

Epic 3: Navigation & Tabs

ID	User Story	Acceptance Criteria	Agent
US-09	As a developer, I want intuitive sidebar navigation with clear section hierarchy	10 sidebar items organized: Home, My Group, Catalog, APIs, Learning, Create, Docs, Notifications, Admin, Settings	@se-ux-ui-designer
US-10	As a developer, I want entity detail tabs to show relevant information for my component type	Component entities show: Overview, Issues, PRs, CI, CD, K8s, Topology, Docs, Dependencies. API entities show: Overview, Definition, Docs.	@platform
US-11	As an admin, I want to add a Lightspeed AI tab for entity-level AI assistance	New tab 'AI Assistant' on entity pages that opens Developer Lightspeed in context of the selected entity	@lightspeed-deployer
US-12	As an admin, I want to reorder entity tabs based on developer usage patterns	Most-used tabs (Overview, CI, K8s) have highest priority; less-used tabs (Image Registry, Monitoring) have lower priority	@platform

Epic 4: Custom Pages

ID	User Story	Acceptance Criteria	Agent
US-13	As a developer, I want a Learning Paths page with curated training content	Custom /learning route with categorized learning resources: Azure, Kubernetes, GitHub, RHDH, AI. Links to docs and tutorials.	@se-technical-writer
US-14	As a developer, I want a Tech Radar page showing our technology adoption status	Tech Radar plugin enabled with JSON data: 4 quadrants (Languages, Frameworks, Tools, Platforms), 4 rings (Adopt, Trial, Assess, Hold)	@architect
US-15	As an admin, I want a Scorecards page to visualize project health	Scorecard plugin enabled showing compliance, quality, and maturity scores per component	@sre

Custom Tabs Flow — How It Works in RHDH

Understanding how custom tabs work in RHDH is essential for planning the UX. Here is the end-to-end flow:

Flow 1: Adding a New Entity Tab

1. Enable the dynamic plugin that provides the tab content (e.g., backstage-community-plugin-topology) in dynamic-plugins.yaml with disabled: false
2. The plugin registers its mount point (e.g., entity.page.topology) and default tab configuration
3. RHDH auto-discovers the mount point and adds the tab to entity detail pages
4. Optionally customize the tab title, path, and priority in entityTabs configuration
5. Optionally customize the tab's internal layout using the 12-column grid (xs, sm, md, lg, xl breakpoints)

6. Apply ConfigMap changes and restart the pod — the new tab appears for all users

Flow 2: Reordering / Removing Tabs

1. In app-config.yaml, under dynamicPlugins.frontend.<plugin>.entityTabs, set priority values
2. Higher priority = tab appears further left (first position). Default priority is 0.
3. To remove a tab, set priority to a negative value (e.g., -1)
4. Tabs are displayed only if the associated plugin contributes content to the mount point
5. Entity-kind filtering: some tabs only appear for specific entity kinds (e.g., 'Definition' only for kind: API)

Flow 3: Custom SignInPage

1. Create a custom dynamic plugin (npm package) that exports a React component named CustomSignInPage
2. The component must accept props from Backstage's auth system and return a configured SignInPage
3. Package the plugin using npx @red-hat-developer-hub/cli@latest plugin package
4. Push the OCI image to your container registry (ACR, Quay.io, GHCR)
5. Reference in dynamic-plugins.yaml: package: oci://registry/plugin:tag!plugin-name
6. Configure signInPage.importName: CustomSignInPage in the frontend section
7. Restart RHDH — the custom login page is now active

Implementation Roadmap

Phase 1: Agent Setup (Week 1)

- Download 5 Tier 1+2 agents from awesome-copilot to .github/agents/
- Configure handoffs: @se-ux-ui-designer → @platform → @template-engineer → @deploy
- Install software-engineering-team plugin via Copilot CLI
- Validate agents work in VS Code Copilot Chat

Phase 2: Login & Branding (Week 1-2)

- Finalize Microsoft branding in app-config-rhdh.yaml (logo, colors, font, favicon)

- Configure Global Header with CompanyLogo component and sidebar logo
- Test login flow: GitHub OAuth → RHDH homepage redirect
- If needed, design custom SignInPage as dynamic plugin (Phase 3)

Phase 3: Homepage Customization (Week 2-3)

- Configure DynamicCustomizableHomePage for interactive layout
- Build QuickAccessCard JSON with 4 categories (Getting Started, Dev, AI, Ops)
- Add Three Horizons cards with color-coded design (Blue/Green/Yellow)
- Add TemplateSection with 6 featured Golden Path templates
- Add SearchBar, CatalogStarredEntitiesCard, and MarkdownCard for announcements

Phase 4: Tabs & Navigation (Week 3-4)

- Configure sidebar with 10+ items, proper priority ordering, and icon assignment
- Enable entity tabs: Overview, Issues, PRs, CI, CD, K8s, Topology, Docs, Dependencies
- Set tab priorities based on developer workflows
- Add Tech Radar and Scorecards as new sidebar pages
- Configure tab layouts using 12-column grid for responsive design

Phase 5: Advanced Customization (Week 4+)

- Build custom SignInPage dynamic plugin (if YAML branding is insufficient)
- Implement localization for Portuguese (Brazil) and Spanish
- Enable and configure Developer Lightspeed AI chat
- Add custom homepage cards via additional dynamic plugins
- Create custom entity tab for AI Assistant (Lightspeed in entity context)

Sources

Source: RHDH 1.8 Customizing Appearance (Official)

(https://docs.redhat.com/en/documentation/red_hat_developer_hub/1.8/)

Source: GitHub Awesome-Copilot Agents (<https://github.com/github/awesome-copilot/blob/main/docs/README.agents.md>)

Source: SE UX/UI Designer Agent (<https://github.com/github/awesome-copilot/blob/main/agents/se-ux-ui-designer.agent.md>)

Source: Expert React Frontend Engineer Agent (<https://github.com/github/awesome-copilot/blob/main/agents/expert-react-frontend-engineer.agent.md>)

Source: AEM Frontend Specialist Agent (<https://github.com/github/awesome-copilot/blob/main/agents/aem-frontend-specialist.agent.md>)

Source: DeveloperHub Custom Landing Page (<https://docs.developerhub.io/support-center/custom-landing-page>)

Source: DeveloperHub Custom HTML (<https://docs.developerhub.io/support-center/custom-html>)

Source: *Software Engineering Team Plugin* (<https://github.com/github/awesome-copilot/tree/main/plugins/software-engineering-team>)

Source: *RHDH Custom SignInPage*
(https://docs.redhat.com/en/documentation/red_hat_developer_hub/1.8/html/installing_and_viewing_plugins/)

Source: *RHDH Homepage Customization*
(https://docs.redhat.com/en/documentation/red_hat_developer_hub/1.8/html/customizing_red_hat_developer_hub/)

Source: *RHDH Entity Tabs Configuration*
(https://docs.redhat.com/en/documentation/red_hat_developer_hub/1.8/html/installing_and_viewing_plugins/)

Source: *Three Horizons RHDH Deploy Guide (Project)* ([rhdh_deploy_customization_guide_v1_0.md](#))

Source: *DEPLOY_CUSTOMIZATIONS_GUIDE (Project)* ([DEPLOY_CUSTOMIZATIONS_GUIDE.md](#))