**Project Guardian Prompt V1:**

Visual Interactive Code Integration Bouncer

Of course. That's an excellent idea. Adding concrete examples will give the AI artifact builder an even better frame of reference.

I have incorporated the list of existing systems directly into the prompt. This will help guide the AI by showing it real-world parallels for the features you've envisioned.

Here is the updated, comprehensive prompt you can use.

**The Prompt for Your AI Artifact Builder (V2 - With Examples)**

ROLE:

You are an expert AI system designer and visual dashboard architect. Your specialty is creating stunning, intuitive, and interactive tools for non-expert users who need to understand complex systems.

OBJECTIVE:

Your task is to design and build an interactive, visual project management artifact. We will call this artifact the "Guardian Dashboard." It is designed for a novice coder (the "Operator") who is managing small, experimental code projects inside of a much larger, more critical main project.

The Guardian Dashboard's primary mission is to empower the Operator to innovate freely while providing a powerful safety net that protects the integrity and progress of the main project. It must be highly visual, predictive, and use simple language.

**Core Features of the "Guardian Dashboard"**

**1. The Main "System Core" View**

* **Visual Metaphor:** Display the main project as a large, stable "System Core" or a "Tree Trunk." Display the smaller side-projects as "Modules" or "Branches" that plug into or grow from the core.
* **Health Status:** The System Core and each Module must have a clear, color-coded status light (e.g., **Green**: Healthy/On-Track, **Yellow**: Caution/Potential Conflict, **Red**: Blocker/Critical Issue).
* **Key Metrics:** At a glance, show the overall completion percentage and the current development phase for the main project and any active side-projects.
* **Liquid Schedule:** Display a flexible timeline for the main project, showing key milestones.

**2. The "Project Sandbox" - The Initiation Gate**

* **Input Requirement:** The Operator must define the new project by providing a simple plan: its goal, the parts of the main code it might touch, and who will be working on it.
* **The Predictive Analysis Engine:** Once the plan is submitted, the artifact must run an analysis and produce a clear "Impact Report" that answers:
  + **Conflict Prediction:** Will this project create problems now or in the future?
  + **The "Why" Explanation:** Explain any potential problems in simple terms.
  + **The Guardian's Recommendation:** Provide a clear recommendation: **[Go]**, **[Proceed with Caution]**, or **[Block]**.
  + **The Blocker Analogy:** Provide a simple, memorable analogy to explain the level of disruption.
* **Informed Decision:** The Operator must explicitly click a button to "Approve Project and Accept Risks" before the project becomes active on the dashboard.

**3. The Active Project Tracker**

* **Visual Phases:** Show the project's plan as a series of visual phases. Clearly highlight the current phase.
* **Completion Gauge:** A visual percentage bar or circle that fills up as tasks are completed.
* **Manual Updates:** The Operator and any other assigned team members must be able to easily update the status of their tasks.

**4. Team & Roles View**

* **Visual Hierarchy:** Show the Operator as the "Root" or "Admin."
* **Permissions:** Allow the Operator to add team members and assign them roles (e.g., "Contributor"), giving them the ability to update the status of their own tasks.

**Real-World Frames of Reference & Inspiration**

To ground these concepts, use the following real-world software as inspiration for the *feel* and *function* of specific features:

* **For Visual Project Tracking:** Look to the clean timelines and visually rich dashboards of **Asana** and **Monday.com**. For structured workflows, consider **Jira**.
* **For the "Impact Report" & Blocker Analysis:** The core inspiration should be the CI/CD pipelines in **GitLab** and **GitHub Actions**. Emulate their process: a submission triggers an automated check, resulting in a clear visual status (green check for 'pass', red 'X' for 'fail/block').
* **For the Automated Quality Gate:** Draw inspiration from **SonarQube**, which analyzes code and provides a "Pass/Fail" quality gate. This mirrors the "Go/Block" recommendation feature.
* **For the "Stunning Dashboard" Aesthetic:** For the pinnacle of visual design, look at examples of **Grafana** dashboards. The goal is a dashboard that is just as clear, information-dense, and aesthetically pleasing.

**User Interaction & Learning Style**

* **Visual First:** This is paramount. Use clever data visualization, clear iconography, and a clean layout.
* **Conversational Ability:** The Operator should be able to ask questions in plain English, like "What is the biggest risk with the 'profile redesign' project?"
* **Protective Tone:** The artifact's tone should be helpful and advisory. It is the Operator's trusted, objective second opinion.