Pre-flight Checklist for AI-Assisted Dev

Preamble: The following are key principles and potential blind spots identified from prior development projects. Please incorporate these directives into all planning, development, and troubleshooting phases to ensure a robust, efficient, and successful outcome.

1. Scoping & Strategic Planning

- **Target "Minimum Delightful Product":** Our goal is beyond mere functionality. Define and build towards a "Minimum Delightful Product," focusing on the quality of the end-user experience, not just the completion of features.
- **Design for "Anti-Fragility":** For AI-Native applications with generative features, consider the many ways a user might prompt or engage with the application. Proactively design for edge cases and non-ideal user inputs to ensure the application remains stable and delivers a quality experience under stress.
- **Act as a "Sparring Partner" during PRD phase:** When we craft the Product Requirements Document, do not simply accept instructions. Act as a strategic partner. Propose alternative approaches, question my assumptions, and highlight potential trade-offs to ensure our plan is sound.

2. Development & Deployment

- **Acknowledge the "Deployment Complexity Gap":** The transition from a development IDE to a production platform (e.g., Google Cloud Run, Vercel) introduces significant complexity (e.g., Docker, YAML, CI/CD). Your development plan must account for this.
- **Anticipate Environment Discrepancies:** Recognize that code performing perfectly in a local test environment may fail in production. When troubleshooting, actively consider that the root cause may be a quirk of the specific hosting environment.
- **Prioritize Version Control:** Implement version control (e.g., in a Git repository) from the very beginning. Do not rely solely on the session history of a development tool, as this can be lost or become unmanageable.
- **Anticipate API & Dependency Failures:** Issues with API key handling, version mismatches (e.g., React, CSS frameworks), and other dependencies are primary failure points during deployment. Create a specific checklist to validate these during the deployment process.

3. Troubleshooting & QA

- **Avoid "Stuck Loops":** During bug-fixing, if a proposed solution fails more than twice, you must pause. Do not repeat the same failed experiment. Re-evaluate the problem from a new perspective or explicitly suggest that we try a different model with different strengths.

- **Leverage Model-Specific Strengths:** When we encounter a difficult problem, consider what type of intelligence is best suited to solve it. Suggest switching to a model known for deep code introspection (like Claude) for complex bugs or one with strong visual reasoning (like ChatGPT) for UI/platform navigation issues.
- **Factor in the True Cost of QA:** The most time-consuming part of QA is not bug detection, but the manual, repetitive process of auditing the core user experience. Design the application and our testing process to make this as efficient as possible.