🚀 Starting Your Build Journey

Define Clear Business or Personal Goals

- Begin with your **basic goal, idea, and desired features** — avoid technical jargon.
- Use **plain language** to describe what you want to build.

Craft Effective Initial Prompts

When working with Emergent Agents, structure your initial prompt as:

- **Business Goal:** [Clear, simple description]
- **Core Features:** [List 3–5 main features]
- **Target Users:** [Who will use this]
- **Success Criteria:** [How you'll measure success]

Best Practices:

- Be explicit and specific.
- Provide context and motivation (explain *why* features matter).
- Use positive framing (focus on what to do, not what not to do).
- Add quality modifiers:
  - Instead of: *"Create a dashboard"*
  - Use: *"Create a comprehensive analytics dashboard with interactive features and modern design principles."*

🧭 Strategic Planning Session

- The Agent will plan tasks and ask questions regarding priorities.
- Guide the Agent according to your use case.

- For brainstorming: *“Chat with me and don’t build right now”* followed by your questions.
- Discuss potential challenges and solutions upfront.
- Once aligned: *“Start building based on what we discussed.”*

🔧 Incremental Development Strategy

Feature Implementation Rules

- Implement **1–2 features per cycle**.
- Complete each feature fully before starting another.
- Test thoroughly before moving forward.

Optimal Prompting for Development

- Use **detailed modifiers**: *“Include hover states, transitions, and micro-interactions.”*
- Request **comprehensive implementations**: *“Go beyond basics and build a full implementation.”*
- Specify **quality expectations**: *“Apply modern design principles: hierarchy, contrast, balance, and movement.”*

⚡ Managing Development Flow

- **Never mix errors and features:** Separate bug reports from new features.
- **Clear context switching:** Finish bug fixes before requesting new features.
- **Avoid interruptions:** Let the Agent finish its current task.
- If needed, say: *“Please finish your current task, then we’ll discuss [new topic].”*

🪲 Effective Error Reporting

Test Before Reporting

- Test in your browser first.
- Document errors with steps, console logs, and screenshots if possible.

Error Communication Protocol

- Test thoroughly yourself.
- Document all errors.
- Provide reproduction steps.
- Share exact error messages.

If the Agent Can’t Resolve Errors

- Ask the Agent to test the frontend itself.
- Provide console logs for hidden issues.

✅ Quality Assurance

- Request targeted testing: *“Test all interactive elements and validate responsive design.”*
- Ask for code reviews: *“Review the code for potential issues and suggest improvements.”*

💾 Strategic Checkpoint Management

GitHub Discipline

- Commit after each major feature.

Context Management

Before clearing context, ask the Agent for a **comprehensive summary** including:

- Implemented features.
- Current status.
- Remaining tasks.
- Known issues/technical debt.
- Next recommended steps.

Rollback Strategy

- Roll back to a working state if needed instead of debugging broken code.
- Use checkpoints to avoid technical debt.

Session Continuity

- Start new sessions by pulling the latest code.
- Provide the last **implementation summary**.
- Clearly state: *"Continue from where we left off based on this summary."*

🔄 Scaling & Iteration

Managing Scope

- Avoid major scope changes mid-development.
- Small changes are fine, big pivots = start fresh.
- Complete current version before rebuilding.

Advanced Features

- Request performance improvements: *"Optimize for loading speed and smooth interactions."*
- Follow modern best practices.
- Ensure accessibility compliance.

🎯 Best Practices Summary

Communication Excellence

- Be explicit and detailed.
- Provide context (the *why*).
- Use positive instructions.
- Add quality modifiers.

Technical Excellence

- Build incrementally.
- Test thoroughly before moving on.
- Separate bug fixes from new features.
- Commit frequently to GitHub.

⚠️ Common Pitfalls

- Starting without clear business requirements.
- Overloading with unnecessary technical details.
- Mixing bug reports with feature requests.
- Interrupting mid-process.
- Changing scope drastically.
- Reporting bugs without testing.
- Forgetting GitHub commits.
- Not summarizing before clearing context.

📋 Master Checklist

✅ Define clear business/personal goals in plain language.

✅ Structure prompts with: **Business Goal, Core Features, Target Users, Success Criteria**.

✅ Be specific, contextual, and positive in instructions.

✅ Guide Agent during planning sessions, align before building.

✅ Limit to 1–2 features per cycle, test thoroughly.

✅ Use quality modifiers in development prompts.

✅ Separate bug fixes from new features.

✅ Document errors with steps, logs, and screenshots.

✅ Request testing and code reviews from the Agent.

✅ Commit code to GitHub after major features.

✅ Before clearing context, get a full implementation summary.

✅ Roll back to working checkpoints if issues persist.

✅ Maintain session continuity with summaries and clear directions.

✅ Avoid major scope changes mid-build.

✅ Optimize performance, follow best practices, ensure accessibility.
