Standard Operating Procedure (SOP)

Building New Projects from Scratch Using AI-Native Tools

Document Version: 1.0
Date Created: April 2, 2025
Created By: Siddharth Bhansali, based on insights from Prajwal Tomar (@PrajwalTomar_)

Purpose: To provide a structured, AI-native workflow for development teams to build new projects from scratch using Cursor Agent, ChatGPT, CodeGuide, and Vercel, ensuring high-quality, consistent, and efficient code generation.

---

1. Objective

This SOP outlines a step-by-step process for leveraging AI tools to plan, document, and code new projects. The goal is to minimize errors in AI-generated code, ensure alignment with project requirements, and maintain consistent coding standards across the team.

---

2. Scope

This SOP applies to all development team members involved in building new software projects from scratch using AI-native tools, specifically:

- **Cursor Agent** for AI-assisted coding.
- **ChatGPT Voice** for project planning.
- **CodeGuide** for generating project documentation.
- **Vercel AI SDK** for building AI-powered applications.
- **Starter Kits** to streamline project setup.

---

3. Responsibilities

- **Project Manager:** Oversees the planning phase, ensures documentation is complete, and tracks progress using the Implementation Plan.
- **Developers:** Follow the SOP to set up, document, and code the project using AI tools.
- **Tech Lead:** Reviews documentation, ensures adherence to coding standards, and updates Project Rules as needed.

---

4. Tools Required

- **Cursor Agent:** AI code editor for generating and updating code.
- **ChatGPT Voice:** For project planning and outlining.
- **CodeGuide:** For generating project documentation (PRDs, tech stack, file structure, etc.).
- **Vercel AI SDK:** For building AI-powered applications (if applicable).
- **Starter Kits:** Pre-configured project templates (e.g., from CodeGuide or custom kits).
- **Version Control System (e.g., Git):** For tracking changes and collaboration.

---

5. Procedure

Step 1: Project Planning with ChatGPT Voice

Objective: Define the project scope and structure before coding begins.
Responsible: Project Manager, Tech Lead
Steps:

1. Use ChatGPT Voice to brainstorm and plan the project:
    - Define the core idea of the project.
    - List essential features.
    - Outline the app flow (e.g., pages, navigation, user actions).
2. Ask ChatGPT to draft a structured project outline, including:
    - High-level goals.
    - Key features and user flows.
    - Potential risks and mitigation strategies.
3. Save the outline as a reference for the next steps.

Output: A detailed project outline document.

Step 2: Generate Project Documentation with CodeGuide

Objective: Create comprehensive documentation to provide context for Cursor Agent.
Responsible: Tech Lead, Developers
Steps:

1. Sign up for CodeGuide using a Google account for quick access.
2. Create a new project in CodeGuide:
    - Describe the core functionality and goals of the project.
    - Select Cursor as the AI coding tool for compatibility.
3. Complete CodeGuide's questionnaire to generate the following documents:
    - **Product Requirements Document (PRD):** Detailed project requirements.
    - **Tech Stack Overview:** Recommended technologies (e.g., React, Next.js, Firebase).
    - **File Structure:** Suggested directory structure.
    - **Frontend & Backend Guidelines:** Best practices for each.
    - **.cursorrules File:** Initial rules for Cursor (to be refined later).
    - **Implementation Plan:** A 50-step roadmap for coding the project.
4. Review and optimize the generated documentation using CodeGuide's AI suggestions.
5. Store all documents in a centralized location (e.g., Google Drive or Git repository).

Output: A set of project documentation files, including PRD, tech stack, file structure, guidelines, and an Implementation Plan.

Step 3: Set Up the Project with a Starter Kit

Objective: Establish a strong foundation for the codebase to avoid setup issues.
Responsible: Developers
Steps:

1. Select a pre-built Starter Kit (e.g., from CodeGuide or a custom kit) that matches your tech stack. CodeGuide offers kits for:
    - React + Supabase
    - React + Firebase
    - Next.js + Firebase
    - React Native + Expo + Firebase
2. Download and initialize the Starter Kit in your local environment:
    - Ensure pre-configured file structures and dependencies are in place.
    - Verify the presence of a documentation folder in the kit.
3. Commit the initial setup to your version control system (e.g., Git).

Output: A pre-configured project directory with a structured codebase.

Step 4: Prepare Cursor Agent with Project Context

Objective: Ensure Cursor understands the project scope before coding.
Responsible: Developers
Steps:

1. Create an `Instructions` folder in the root directory of your project.
2. Add all generated documentation from CodeGuide (PRD, tech stack, file structure, etc.) to the `Instructions` folder.
3. Open Cursor Agent and run the following prompt:
    - "Go through all files in the Instructions folder and summarize what you understand about my project."
4. Review Cursor's summary to ensure it aligns with your project goals. If discrepancies are found, revise the documentation and repeat this step.

Output: Cursor Agent has a clear understanding of the project scope.

Step 5: Define and Apply Project Rules (.mdc Files)

Objective: Set granular coding rules to ensure Cursor generates code that matches your style and standards.
Responsible: Tech Lead, Developers
Steps:

1. Create a `.cursor/rules/` directory in your project.
2. Define modular Project Rules (.mdc files) for different parts of the project:
    - **general.mdc:** Global rules for clean code and readability (e.g., naming conventions, formatting).
    - **frontend.mdc:** Rules for frontend files (e.g., `.tsx` for React components).
    - **backend.mdc:** Rules for backend logic (e.g., `api/**/*.ts` for API routes).
    - **database.mdc:** Rules for database queries (e.g., `*.sql`).
3. Example rules to include:
    - Use functional components for React.
    - Follow RESTful conventions for API routes.
    - Use camelCase for variable names.
4. Test the rules by generating a small piece of code with Cursor and verify compliance.

Best Practices:

- Keep rules modular and specific to file types or modules.
- Use precise scope targeting (e.g., `.tsx` for React, `api/**/*.ts` for backend).

- Regularly update rules as the project evolves.

Output: A set of `.mdc` files in `.cursor/rules/` that enforce coding standards.

Step 6: Code the Project Using Cursor Agent

Objective: Generate code step-by-step using Cursor, following the Implementation Plan.
Responsible: Developers
Steps:

1. In Cursor, run the following prompt to start coding:
    - "Follow the @.implementation-plan.md file and start coding from Step 1."
2. Monitor Cursor's progress as it generates code for each step.
3. After each step, ask Cursor to:
    - Update the `Implementation Plan` file with a "DONE" status for completed steps.
    - Summarize the changes made and confirm alignment with the project goals.
4. Review the generated code for accuracy and adherence to Project Rules.
5. If issues arise, refine the documentation or Project Rules and regenerate the code.

Output: A progressively built codebase, with each step tracked in the Implementation Plan.

Step 7: Integrate Vercel AI SDK (If Applicable)

Objective: Add AI-powered features to the project using Vercel AI SDK.
Responsible: Developers
Steps:

1. If your project requires AI-powered features (e.g., chatbots, recommendation systems), integrate the Vercel AI SDK:
    - Install the SDK using npm: `npm install @vercel/ai-sdk`.
    - Follow Vercel's templates for your framework (e.g., Next.js, React).
2. Use Cursor to generate AI-related code, ensuring it adheres to the Project Rules.
3. Test the AI features thoroughly to ensure they meet project requirements.

Output: AI-powered features integrated into the project.

Step 8: Review and Iterate

Objective: Ensure the codebase meets quality standards and project goals.

Responsible: Tech Lead, Developers
Steps:

1. Conduct a code review to verify:
    - Adherence to Project Rules.
    - Functionality of all features.
    - Alignment with the PRD and project outline.
2. Test the application thoroughly (unit tests, integration tests, etc.).
3. If issues are found, update the documentation, Project Rules, or Implementation Plan, and regenerate the affected code using Cursor.
4. Commit all changes to the version control system.

Output: A fully functional, high-quality codebase ready for deployment.

---

6. Best Practices

- **Planning is Key:** Spend 70% of your time planning and 30% on execution to ensure AI tools have the right context.
- **Modular Rules:** Keep Project Rules specific to modules or file types to avoid conflicts.
- **Regular Updates:** Update documentation and rules as the project evolves to maintain alignment.
- **Track Progress:** Use the Implementation Plan to monitor progress and ensure no steps are missed.
- **Starter Kits:** Always use a Starter Kit to avoid setup issues and provide Cursor with a strong foundation.

---

7. Expected Outcomes

- **Improved Code Quality:** AI-generated code aligns with your coding style and project requirements.
- **Faster Development:** Structured planning and modular rules reduce the need for repetitive corrections.
- **Consistency:** Project Rules ensure consistent coding standards across the team.
- **Efficiency:** Starter Kits and detailed documentation streamline the setup and coding process.

---

8. References

- **Original Thread:** Prajwal Tomar's X Thread
- **Cursor Website:** www.cursor.com
- **CodeGuide:** www.aisharenet.com
- **Vercel AI SDK:** sdk.vercel.ai

---

9. Approval

Approved By: Siddharth Bhansali
Date: April 2, 2025
