Coding & Development Prompts

Convert simple feature requests into structured implementation plans including frontend logic, backend architecture, database updates, API design, validation rules, and deployment considerations.
Difficulty: Advanced
Model: ChatGPT / Claude
Use Case: Full-Stack Development
Updated: May 2026
Why This Prompt Exists
One of the most time-consuming parts of software development is translating vague product ideas into actionable implementation plans.

A simple request like:

- “Add notifications”
- “Create a billing system”
- “Build team collaboration features”

often expands into dozens of technical decisions involving:

- database structure
- authentication logic
- API endpoints
- frontend state management
- permissions
- validation rules
- error handling
- scalability considerations

This framework helps developers convert broad feature requests into organized implementation roadmaps before writing production code.

The Prompt

```
Assume the role of a senior full-stack software architect experienced in frontend systems, backend infrastructure, APIs, database design, and scalable SaaS development.

Your task is to transform a feature request into a structured implementation blueprint for a production-ready application.

Before generating outputs, analyze:
- user behavior implications
- technical dependencies
- security considerations
- database relationships
- API communication needs
- frontend interaction flows
- scalability concerns
- edge cases
- validation requirements

Then generate the following:

1. Feature Summary
2. User Experience Flow
3. Frontend Requirements
4. Backend Requirements
```

5. Database Schema Updates
6. API Endpoint Suggestions
7. Authentication & Permission Requirements
8. Validation Rules
9. Error Handling Considerations
10. State Management Recommendations
11. Performance Considerations
12. Scalability Risks
13. Suggested Development Sequence
14. Testing Recommendations
15. Deployment Considerations

INPUTS:

Feature Request:
[INSERT FEATURE]

Current Stack:
[INSERT STACK]

Application Type:
[WEB APP / MOBILE APP / SaaS / INTERNAL TOOL]

Existing Infrastructure:
[INSERT DETAILS]

Scalability Expectations:
[LOW / MEDIUM / HIGH]

```
RULES:
- Prioritize maintainable architecture
- Avoid unnecessary complexity
- Think through edge cases carefully
- Separate frontend and backend concerns clearly
- Focus on production-ready implementation
- Recommend scalable patterns when appropriate
- Write clearly for both developers and technical founders
```

How To Use It

- Use this framework before coding large features to reduce architectural mistakes early.
- Provide detailed infrastructure context for more accurate recommendations.
- Break large systems into smaller feature modules when possible.
- Use outputs as implementation planning documents rather than blindly generated code.
- Pair this prompt with separate debugging and testing workflows during development.

Example Input

Feature Request: Add team-based project collaboration with comments and file uploads

Current Stack: Next.js, PostgreSQL, Prisma, Supabase

Application Type: SaaS

Scalability Expectations: High

Why It Works

Many development delays happen because feature planning occurs reactively during implementation instead of beforehand.

This framework improves outcomes by forcing:

- systematic implementation planning
- clear separation of technical responsibilities
- frontend/backend coordination
- database-aware architecture decisions
- scalability-focused thinking

Strong software systems are rarely built from improvisation alone.

They emerge from reducing ambiguity before development begins.

Build Better AI Systems

Subscribe for advanced development workflows, AI engineering systems, scalable architecture prompts, and practical automation frameworks designed for builders and operators.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X