

ChatGPT for Coding Cheat Sheet

Practical prompt templates, debugging workflows, and refactoring strategies for software engineers.

1. CODE GENERATION & SCAFFOLDING

New Component / Module Boilerplate

GENERATION

Generate clean modular code with explicit modern standards and error handling.

"Act as a senior [Language/Framework] developer. Write a clean, self-contained module for [Functionality]. Follow modern best practices, include full TypeScript types, and handle edge cases (e.g., null values, network failures). Do not use external dependencies unless specified."

Database Schema & SQL Queries

DATABASE

Draft normalized SQL tables and efficient query joins.

"Design a normalized SQL schema for a database supporting [Use Case, e.g., e-commerce orders]. Provide table definitions with primary/foreign keys, proper indexes, and write a query that calculates [Metric, e.g., monthly revenue per customer]."

API Endpoint Implementation

BACKEND

Build robust HTTP route handlers with validation and status codes.

"Create a RESTful API route in [Node.js/Express/Python FastAPI] that accepts a POST request for [Action]. Include input payload validation, appropriate HTTP status responses (200, 400, 500), and structured error JSON payloads."

2. DEBUGGING & TROUBLESHOOTING

Error Log Diagnosis

DEBUGGING

Identify root causes and fixes from stack traces rapidly.

"I received the following runtime error in my [Framework] application:\n\n[Paste Stack Trace / Log]\n\nAnalyze the error, explain the most likely root cause in 2 sentences, and provide corrected code snippet options to resolve it."

2. DEBUGGING & TROUBLESHOOTING (CONT.)

Logical Bug Detection

CODE REVIEW

Find subtle edge-case issues or state bugs without explicit compiler errors.

"Review the code below for hidden logical flaws, race conditions, or memory leaks. Explain what could break under high load or edge-case inputs:\n\n[Paste Code Block]"

3. REFACTORING & CODE OPTIMIZATION

Performance & Algorithmic Complexity

OPTIMIZATION

Reduce time/space complexity and streamline execution paths.

"Optimize the following function to improve execution time complexity. State the current Big-O complexity and the optimized Big-O complexity after your changes:\n\n[Paste Code]"

Clean Code & Design Patterns

REFACTORING

Apply DRY principles and modern design patterns to legacy scripts.

"Refactor this legacy script into clean, readable code using SOLID principles. Extract inline magic values into named constants and split monster functions into smaller reusable units:\n\n[Paste Legacy Code]"

4. TESTING & DOCUMENTATION

Unit Test Suite Generation

TESTING

Generate comprehensive test coverage including happy paths and boundary cases.

"Write unit tests for the following function using [Jest/PyTest/JUnit]. Include test cases for normal operation, boundary inputs, and unexpected error throwing:\n\n[Paste Function]"

5. BEST PRACTICES FOR CODING PROMPTS

Maximize Output Quality

GUIDELINES

- **Provide Context:** Always specify language version, framework version, and environment (e.g., Node 20, Python 3.11).

