Coding & Development / Debugging

Explain a cryptic error message in plain English, with likely causes and specific fixes.
Difficulty: Beginner → Intermediate
Model: GPT-4 / Claude / Gemini
Use Case: Error Diagnosis, Debugging, Learning
Updated: May 2026
Why This Prompt Exists
Most error messages are cryptic — developers waste hours searching for answers.

You get:

- stack traces that don't point to the real problem
- error messages that are too generic ("Something went wrong")
- no explanation of why the error happened
- no specific fix — just vague suggestions
- hours of searching instead of solving

But an error message is not a dead end.

It is a clue with a solution.

- Translation: what the error actually means in plain English
- Likely causes: common reasons this error occurs
- Specific fixes: line-by-line or code-level solutions
- Prevention: how to avoid the error in the future

Without explanation, errors waste time.

This framework forces AI to translate cryptic errors into actionable fixes.

The Prompt

```
Assume the role of a debugging expert who translates cryptic errors
into plain English.

Your task is to explain an error message and provide fixes.

Generate:

1. PLAIN ENGLISH TRANSLATION
   - What the error actually means

2. LIKELY CAUSES (2-4)
   - Common scenarios that trigger this error

3. SPECIFIC FIXES (2-3)
   - Code-level solutions
   - Configuration changes

4. PREVENTION TIPS
   - How to avoid this error in the future

5. RELATED ERRORS
   - Similar errors that might appear next

INPUTS:

Error Message (paste the full error):
[PASTE ERROR HERE]
```

```
Language/Framework:
[PYTHON / JAVASCRIPT / REACT / NODE / DJANGO / OTHER]

Stack Trace (if available):
[PASTE OR "NONE"]

What You Were Trying to Do:
[DESCRIBE]

Environment:
[DEVELOPMENT / PRODUCTION / BOTH]

RULES:
- Translate technical jargon into plain English
- List specific causes (not "configuration issue")
- Provide copy-paste fixes where possible
- Include code examples in fixes
- Note if the error is security-related
- Distinguish between dev and prod environments
```

How To Use It

- Paste the exact error message (not just a description).
- Include the stack trace — it contains critical clues.
- Describe what you were trying to do — context matters.
- Try the specific fixes in order (most likely first).
- If the error persists, the “related errors” section points to next steps.

Example Input

**Error Message:** “TypeError: Cannot read property ‘map’ of undefined”

**Language/Framework:** JavaScript / React

**Stack Trace:** at UserList.render (UserList.js:15) at finishClassComponent (react-dom.development.js:12345)

**What You Were Trying to Do:** Render a list of users from an API response

**Environment:** DEVELOPMENT

Why It Works
Most error messages are cryptic.

This framework improves outcomes by forcing:

- plain English translation (understanding)
- specific causes (diagnosis)
- copy-paste fixes (action)
- prevention tips (learning)
- related errors (anticipation)

Great error explanations don’t just translate — they teach you how to fix and prevent.

## Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

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

See also  The Null/Undefined Error Fixer