

Vibe Coding 大厂面试完整指南（2025-2026）阿东玩AI

适用场景：阿里、字节、美团、滴滴、百度、大疆等大厂 AI 编程机考 / Vibe Coding 面试环节

目录

- 1. [什么是 Vibe Coding 面试](#)
- 2. [标准面试流程（7 步）](#)
- 3. [核心文档模板](#)
- 4. [模拟面试实录（真题）](#)
- 5. [高频追问话术](#)
- 6. [常见翻车场景与解法](#)
- 7. [面试高分行为清单](#)
- 8. [工具配置](#)
- 9. [开场白模板](#)
- 10. [准备建议](#)

一、什么是 Vibe Coding 面试 阿东玩AI

Vibe Coding 由 Andrej Karpathy（前特斯拉 AI 总监、OpenAI 联创）于 2025 年 2 月提出：开发者用自然语言描述需求，让 AI（Cursor、Claude Code、Copilot 等）生成代码，人负责**指导、迭代、审查、管理**，而非逐行手写。2025 年 11 月被柯林斯词典选为年度词汇。

面试形式变化

传统形式	2025-2026 新形式
手写 LeetCode 算法题	用 AI 实现完整业务功能
考手速和记忆	考 AI 驾驭能力和工程判断
使用 AI = 作弊	部分大厂明确要求使用 AI
看代码对不对	看流程规范、设计思维、沟通能力

面试官真正考察的

- 需求理解与拆解能力
- 方案设计与工程思维（文档先行）
- AI 沟通与管理能力（Prompt 质量）
- 代码审查与质量意识
- 全程表达清晰、与面试官实时同步

二、标准面试流程（7 步）

Step 1 需求澄清 + 编写 design.md
Step 2 人工确认①：与面试官对齐设计方案
Step 3 任务拆解 + 编写 task.md
Step 4 人工确认②：边界与优先级二次对齐
Step 5 AI 辅助分模块编码（TDD）
Step 6 代码审查 + 质量加固
Step 7 总结交付 + 补充未完成项设计思路

Step 1：需求澄清 + 编写 design.md

你要做的事：

1. 用一句话复述需求，确认理解方向
2. 识别核心功能模块（3-5 个）
3. 明确输入 / 输出 / 约束条件（性能、安全、异常处理）
4. 写出整体架构：模块划分、数据流向、核心接口
5. 说出技术选型及理由
6. 标注 1-2 个核心风险点及预案

时间控制：5 分钟内完成

说话方式示例：

"我先把需求整理成设计文档，确认一下我的理解——核心是做一个商品自动审核系统，输入是商家提交的商品信息，输出是上架 / 拒绝 / 转人工三种状态。我计划拆成违禁词检测、图片合规审核、价格校验三个模块，中间加一个置信度路由层。我把这个写成 design.md，您看一下有没有问题？"

写好 design.md 的关键：暴露你的设计判断，不要写废话。

错误示范：

"系统分为前端、后端、数据库三层，前端负责展示，后端负责逻辑。"

正确示范：

"核心难点是置信度路由策略——检测结果在 0.4-0.7 的灰区时既不能直接上架也不能直接拒绝，必须转人工，阈值需要可配置而非硬编码。"

Step 2：人工确认①（与面试官对齐设计方案） 阿东玩AI

你要做的事：

1. 把 design.md 呈现给面试官
2. 逐模块过一遍，主动说出设计决策和取舍
3. 等待面试官提问，不要自己一直讲（给 15-20 秒沉默时间）
4. 根据反馈当场更新文档
5. 明确问："我们对这个方案达成一致了吗？"

注意：确认完成前不开始写代码。

面试官沉默的处理：

- 他在思考 → 等待，不要打破沉默
- 他在等你推进 → 说"您看这个方向有没有问题，如果没有异议我们就按这个往下走？"

Step 3：任务拆解 + 编写 task.md 阿东玩AI

你要做的事：

1. 基于确认后的 design.md 拆解开发任务
2. 每个任务：2-5 分钟内可完成，有明确交付物
3. 标注优先级 P0 / P1 / P2 和依赖关系
4. 每个任务写清楚验收标准

粒度控制：

剩余时间	建议任务数
30 分钟	4-6 个
20 分钟	3-4 个

10 分钟	直接说"重点实现核心模块，其余给出设计思路"
-------	------------------------

说话方式示例：

"我把任务拆成了 6 个，P0 是核心逻辑必须跑通，P1 时间够再做。整个开发顺序是先做违禁词检测和路由层，主流程跑通了再做图片审核。您看这个拆解合理吗？有没有您特别想看我实现的部分？"

Step 4：人工确认②（边界与优先级二次对齐） 阿东玩AI

你要做的事：

- 1. 把 task.md 呈现给面试官
- 2. 重点确认：优先级划分是否合理、边界逻辑是否对齐
- 3. 询问："您希望我重点演示哪个模块？"
- 4. 明确时间分配

Step 5：AI 辅助分模块编码（TDD） 阿东玩AI

编码原则：

- 严格按 task.md 顺序，一个模块一个模块来
- 先写**测试用例**（TDD），再让 AI 生成实现代码
- 拿到 AI 生成代码后，先人工审查再运行
- 每个模块完成后做一句小结，再进下一个
- 全程边做边讲，让面试官随时能跟上

黄金 Prompt 五段式（每次调用 AI 前先写好）：

角色：资深 [语言] 工程师
任务：实现 [具体功能]
输入：[入参类型 / 格式 / 示例]
输出：[出参类型 / 格式 / 示例]
约束：

- 性能：[具体指标，如耗时 <50ms]
- 安全：[异常处理要求，空值/越界/特殊字符]
- 代码风格：[类型注解/行内注释/函数命名规范]

格式：完整可运行代码 + 行内注释 + 3 个测试用例（正常/异常/边界）+ 时间空间复杂度分析

示例 Prompt（违禁词检测模块）：

角色：资深 Python 后端工程师

任务：实现违禁词检测模块

输入：{text: str}（商品标题 + 描述拼接文本）

输出：{hit: bool, keywords: list[str], confidence: float}

约束：

- 词库从外部 JSON 文件加载，支持热更新
- 支持模糊匹配（编辑距离 ≤ 1 ）
- 耗时 $< 50\text{ms}$ ，空字符串/None 输入不报错
- 所有函数有类型注解和行内注释

格式：完整可运行代码 + 3 个测试用例（正常命中/未命中/空输入）+ 复杂度分析

编码过程中说给面试官听的话：阿东玩AI

"我现在要实现 T1，Prompt 里我加了模糊匹配的约束，因为违禁词变体是实际业务里最常见的绕过手段..."

"AI 生成的代码我先看一下——这里用的是 list 遍历，复杂度是 $O(n \times m)$ ，词库大了会慢，我让它改成 Trie 树..."

"这里 AI 没有处理 None 输入的情况，我来告诉它修一下..."

Step 6：代码审查 + 质量加固 阿东玩AI

人工检查三个维度：

5. 逻辑正确性

- 核心算法是否正确
- 边界 case 是否全部覆盖（空值、超长输入、特殊字符、Unicode）
- 数据结构选型是否合理

6. 代码质量

- 命名是否清晰（函数名、变量名见名知义）
- 注释是否有意义（不写废话注释）
- 结构是否合理（单一职责、函数不超过 30 行）

7. 安全性

- 异常处理是否完整

- 输入校验是否到位
- 有无注入风险

审查时间自己：

"如果有人想绕过这个检测，他会怎么做？" "这个函数如果输入是 None / 空列表 / 超长字符串，会发生什么？"

Step 7：总结交付

你要做的事：

1. 跑通主流程，全量演示
2. 说清楚：整体思路、模块关系、复杂度、可扩展性
3. 主动交代未完成项，给出设计思路（不要沉默）
4. 整理交付物：代码 + 注释 + 测试用例 + 说明

未完成项说法示例：

"T5 审核日志我没有实现，但设计上它是一个异步写入模块——审核结果通过消息队列异步落库，不阻塞主流程，接口已预留 log_result 占位函数，生产环境接入时替换实现即可。"

三、核心文档模板

design.md 完整模板

设计文档 - [系统名称]

需求理解

- 业务目标：
- 核心输入：
- 核心输出：
- 约束条件：

整体架构

[模块划分]

- 模块 A：
- 模块 B：
- 模块 C：

[数据流向]

输入 → 模块A → 模块B → 路由层 → 输出

核心接口

接口	入参	出参	说明
-----	-----	-----	-----

技术选型

选项	选/不选	理由
-----	-----	-----

风险点

风险	预案
-----	-----

task.md 完整模板

任务拆解 - [系统名称]

P0 核心任务（必须完成）

- [] T1: [任务名]
 - 输入:
 - 输出:
 - 验收标准:
- [] T2: [任务名]
 - 输入:
 - 输出:
 - 验收标准:

P1 重要任务（时间够则做）

- [] T3:
- [] T4:

P2 优化任务（有余力则做）

- [] T5:
- [] T6:

开发顺序与并行关系

[串行] T1 → T2 → T3

[可并行] T1 与 T4（无数据依赖）

四、模拟面试实录（真题）阿东玩AI

题目：商家提交商品后，系统自动判断能否上架。请全程使用 AI 工具完成设计与实现，并随时解释你的决策。

面试官：你怎么理解这道题？

这是一个内容审核系统，输入是商家提交的商品信息（标题、描述、图片、价格），输出是三种状态：直接上架、直接拒绝、转人工审核。核心难点不是某一个检测算法，而是多个检测模块的结果融合和路由策略——什么情况下由 AI 决策，什么情况下转人工，置信度阈值怎么设计。我先把这个理解写成 design.md，您确认一下方向对不对。

面试官：你为什么要把结果分三类，直接给上架 / 拒绝两类不行吗？

硬二分法在线上风险很高。两个典型 case：一是违禁词检测命中了“枪”这个字，但商品是水枪玩具，直接拒绝会大量误杀；二是图片审核模型置信度只有 0.6，这种模糊情况硬判断很容易出错。三态路由的好处是把“模型不确定”的 case 显式暴露出来交给人工，而不是把风险藏在系统里。这也是我在 design.md 风险点里提到的：置信度阈值可配置，审核日志可追溯。

面试官：你的任务拆解里 T1 和 T3 你说可以并行**，为什么？**

因为这两个模块的输入来源不同——T1 消费文本字段，T3 消费图片 URL，两者没有数据依赖关系，可以同时发起检测。结果都返回后再交给置信度路由层 T2 做汇总。并行能把总延迟从串行的 200ms 降到单个最慢模块的时间，大约 100ms 左右。

面试官：你写的 Prompt 里要求耗时 <50**ms**，AI 能保证这个吗？

AI 生成的代码不能直接保证性能，这是我在代码审查阶段要做的事。拿到违禁词检测代码后，我会看它用的是什麼数据结构——如果是 list 遍历就是 $O(n \times m)$ ，我会让它改成 Trie 树，把匹配复杂度降到 $O(m)$ 。这也是为什么代码审查是一个显式步骤，而不是 AI 生成完就直接用。

面试官：如果 AI 生成的代码有 Bug，你怎么处理？

我不会让 AI 自己猜怎么改，那会引入新的 Bug。三步处理：第一步，人工定位问题——看报错信息和测试用例，确认问题在哪一行；第二步，精确描述给 AI："第 23 行的 confidence 计算没有处理 keywords 为空列表的情况，会导致除零错误，请只修复这一处"；第三步，拿到修复代码后只看修改的那一处，不重新看整个文件。精准定位 + 精准描述 + 局部验证，不会越修越乱。

面试官：你整体用 AI 完成了多少，你自己做了多少？

分工很清楚——AI 负责代码实现，我负责所有判断和决策。具体说：design.md 和 task.md 是我写的；技术选型和架构决策是我的；每个模块的 Prompt 是我设计的，包括约束条件和边界 case；代码生成后的审查、发现问题、给修复指令是我做的；测试用例设计是我的。AI 是我手里的工具，加快了实现速度，但所有工程决策都是我在主导。

面试官：如果让你把这个做到生产级，还缺什么？

目前的 demo 缺三块：第一，模型热更新——违禁词库和审核模型要能在不重启服务的情况下更新，用配置中心 + 版本管理实现；第二，可解释性——拒绝商品时要告知商家具体原因，现在的输出结构有 keywords 字段但还不够完整；第三，人工审核回流——人工审核结果反哺模型，形成数据飞轮，这是审核质量持续提升的关键，对应 task.md 里的 T6。

面试官：你怎么评估审核质量？

核心三个指标：第一，精准率（Precision）——被拒绝的里面真正有问题的占多少，太低说明误杀严重；第二，召回率（Recall）——真正有问题的里面被拒绝的占多少，太低说明漏检；第三，人工审核占比——理想情况低于 5%，比例反映置信度阈值是否合理。上线后做 A/B 测试，对比不同阈值配置下这三个指标，找到业务可接受的最优点。

面试官：你的方案在高并发下能撑住吗？

当前 demo 是单机同步处理，生产环境需要三个改造：第一，异步化——审核请求进消息队列，消费者并行处理；第二，无状态化——检测模块不持有状态，可以水平扩展；第三，缓存层——相同商品内容（如同款商品二次上架）命中缓存，不重复跑模型。这些是 task.md 里 P2 级别的工作，核心逻辑先跑通，扩展性是生产化的下一步。

面试官：你的方案有什么缺点？

有两个明显局限：第一，置信度阈值是静态配置的，没有根据业务反馈自动调整，需要人工定期复盘和校准；第二，各模块是独立检测再汇总，没有考虑模块间的相关性——比如图片和标题同时可疑时，联合置信度应该比单独可疑更高，这个联合建模在 demo 里没有实现。

主动说缺点，反而体现了你对系统的全局认知。

五、高频追问话术 阿东玩AI

追问：你为什么选这个方案，有没有考虑过别的？

答法模板（MECE 对比法）：

"我考虑过两个方案。方案 A 是 [没选的]，优点是 [xxx]，但缺点是 [核心缺陷]；我选的方案 B 牺牲了 [xxx]，但解决了 [核心问题]。结合这道题的约束条件，我选了 B。"

追问：你让 AI 写的代码你真的看懂了吗？

"我审查代码的方式是三步：第一看数据结构选型对不对；第二看边界处理有没有遗漏（空值、超长输入、特殊字符）；第三看异常处理是否完整。如果哪行看不懂，我会让 AI 加注释或换一种写法。生产代码里不应该有我自己看不懂的代码块。"

追问：如果面试时 AI 工具崩了怎么办？

"有两个备选：一是切换工具——Cursor 崩了用 Claude.ai 网页版；二是如果所有 AI 工具都不可用，我会基于已确认的 design.md 和 task.md，手写关键函数的核心逻辑，把数据结构

和算法选型说清楚。设计文档已经做完了，编码只是把思路翻译成代码，工具崩了我还有文档托底。”

追问：你在整个过程中最担心的风险是什么？

“我最担心边界 case 遗漏——AI 生成的代码在 Happy Path 上通常没问题，但实际业务里的异常情况（Unicode 特殊字符、超长文本截断、违禁词变体拆字）在正常测试里不会暴露，只有专门设计对抗性测试用例才能发现。所以代码审查阶段我会问自己：如果有人想绕过这个检测，他会怎么做？”

追问：你整个流程最耗时的是哪一步？

“实际耗时最长的往往不是编码，而是 Prompt 的迭代——AI 第一次生成的代码如果约束描述不够精准，来回修改会很耗时。所以我把 Prompt 设计放在编码前，用五段式结构强迫自己想清楚输入输出和约束，一次出对比反复修要高效得多。”

六、常见翻车场景与解法

场景一：AI 反复生成错误代码

根本原因：Prompt 太模糊，AI 不知道真正的约束是什么。

解法：

1. 停下来，不要继续让 AI 猜
2. 主动说：“这里我重新整理一下需求再给它”
3. 回到 task.md，重写一个更精准的 Prompt
4. 还不对就说：“我用伪代码写出核心逻辑，让 AI 翻译成完整代码”

说给面试官听：

“AI 连续两次没出对，说明我的 Prompt 约束条件不够精准，我重新梳理一下再试。”

这反而体现了工程判断力，不要沉默。

场景二：面试官问了一个完全没想到的边界 case

正确做法：

"好问题，这个 case 我没有在 design.md 里覆盖到。我现在想一下——[停 5 秒思考]——这种情况应该走 [xxx] 路径，因为 [理由]。这个应该加到 task.md 的 P1 里，我来更新一下。"

当场更新文档，让面试官看到你的思维过程。

场景三：时间不够，P0 任务没全部完成

提前 5 分钟主动预警：

"时间还剩 5 分钟，T1 和 T2 已经完成并测试通过，T3 图片审核来不及实现了。快速说一下 T3 的设计思路：调用外部图片审核 API，拿到置信分后交给 T2 路由层统一处理，接口已经预留好了。T4 和 T5 同理，都是独立模块，不影响已完成的主流程。"

面试官评分的是工程判断力，不是代码行数。

场景四：面试官全程沉默不给反馈

这是压力测试，考察你能否在不确定环境下推进。

策略：

- 每完成一个关键节点，主动说："T1 完成，测试通过，接下来做 T2，您有问题吗？"
 - 对方还是沉默就继续推进，不要反复寻求认可（问超过两次显得不自信）
 - 结束时做完整总结，把未完成项也说清楚
-

场景五：设计阶段卡住，不知道怎么拆模块

不要沉默超过 10 秒，直接说：

"我先把我想到的列出来，可能还不完整，我们一起来看看有没有遗漏——目前我能想到的是 [xxx] 和 [xxx]，您觉得还有哪些核心模块需要考虑？"

主动暴露思考过程 + 邀请面试官参与，是正确的沟通方式，不是示弱。

七、面试高分行为清单

必须做到

- ☐ 开口第一句是复述需求，不是马上动手
- ☐ design.md 在 5 分钟内写完并主动呈现
- ☐ 两次显式确认，等面试官认可再继续
- ☐ 每写一个 Prompt 前先说思路
- ☐ AI 生成代码后先人工审查再运行
- ☐ 遇到 Bug 先定位再精准告诉 AI 怎么修
- ☐ 每完成一个任务做一句小结
- ☐ 主动交代未完成项的设计思路
- ☐ 主动说出方案的缺点和局限

绝对不做

- 不经过需求确认直接让 AI 开始写代码
- 不看 AI 生成的代码直接运行
- 面试官提问时继续打字不停下来回答
- 遇到问题沉默超过 10 秒不说话
- 说"这是 AI 生成的我也不知道为什么"
- 反复问"您觉得这样可以吗？"超过两次

八、工具配置

.cursorrules 模板（放项目根目录）

你是一个资深后端工程师。

编写代码时：

- 所有函数必须有类型注解
- 必须包含完整异常处理
- 复杂逻辑必须有行内注释（不写废话注释）
- 每个函数写完后附上 2-3 个测试用例（正常/异常/边界）
- 数据结构选型需说明时间和空间复杂度

Claude Code 使用技巧

- 用 Plan 模式（Shift+Tab 两次）先规划，不直接执行
- 复杂任务分多次给，不要一次给太长的需求
- 拿到代码后补充说："review this code, focus on edge cases and error handling"

面试前检查清单

- ☐ Cursor / Claude Code 登录状态正常
 - ☐ 有备用方案 (Claude.ai 网页版)
 - ☐ 本地有 Python / Node 运行环境
 - ☐ 熟悉基本 Git 操作 (init / add / commit)
 - ☐ 准备好 2-3 个常用 Prompt 模板, 熟记五段式结构
-

九、开场白模板 (直接背)

面试开始后主动说这段话, 建立专业印象:

"在开始之前我想说一下我的工作方式——我会先花几分钟写一份设计文档, 把需求理解、模块划分和核心接口梳理清楚, 跟您确认方向对了再开始; 然后拆解任务列表, 再用 AI 工具逐模块实现。全程我会把思路说出来, 您随时可以打断提问。这样您能看到的不只是最终代码, 也能看到我的工程判断过程。我们可以开始吗?"

这段话做了三件事:

1. 主动说明流程, 让面试官知道你有章法
 2. 预告"随时可以打断", 降低面试官心理门槛
 3. 强调"工程判断过程", 把评分重心从代码结果引向过程
-

十、准备建议

练习方法

1. 练习写 **design.md**: 找 3 道场景题, 每道题 5 分钟内写出 design.md, 反复练到不假思索。
2. 练习 **Prompt 结构**: 背住"角色 + 任务 + 输入输出 + 约束 + 格式"五段式, 任何需求套进去就能出高质量代码。
3. 练习代码审查: AI 生成代码后, 先找边界 case 和异常处理, 这是面试官最常插问的点。
4. 练习口头表达: 边做边说, 确保思路和操作同步输出, 家里对着镜子练。
5. 熟悉工具: Cursor 或 Claude Code 至少用一个到熟练, 知道怎么快速定位和修改特定代码块。

推荐模拟练习题

题目	核心考察点
用户登录系统（含 JWT + 异常处理）	安全性设计、Token 管理
日志收集 Agent（多源采集 + 过滤 + 存储）	数据管道、异步处理
简历自动筛选系统（JD 匹配 + 评分 + 路由）	NLP 应用、评分算法
智能客服 Agent（意图识别 + 工具调用 + 多轮记忆）	Agent 架构、Memory 设计
商品自动审核系统（违禁词 + 图片 + 价格）	多模块融合、置信度路由

整理时间：2026 年 4 月