面试常见
Behaviour Question解析

...

小红书 @momo职场陪跑

Agenda/目录(上半部分)

- 什么因素导致behaviour Question Round在北美越来越严格
- HM在看什么
- 讲话啰嗦:你以为的STAR型不是正确的STAR模型
  - 模版例子
  - 案例分析:
    - 修改前
    - 修改后
- 面试前要准备几个例子？什么类型的例子？
- 什么是亚马逊“14条军规”，哪些leadership principal要着重准备
  - 深入分析3个LP
    - 常见问题
    - 人设
    - 建议答案/模版
    - 面试官眼中的redflag

Agenda/目录(合订版)

1. 北美行为面试趋势 & HM关注重点
2. 正确使用 STAR 模型：真实案例前后对比
3. 如何准备高质量 BQ 示例（冲刺策略 + 模板）
4. 深度拆解10个亚马逊 Leadership Principle
    - 常见问题
    - 人设
    - 建议答案/模版
    - 面试官眼中的redflag
5. HM call解析
    a. 常见分类
    b. 面试技巧
6. 不同时间段（1个月/1周）冲刺计划建议
7. Staff 级别行为面试怎么准备

BQ为什么原来越卷

- 原来就比较卷的公司
  - Meta, Amazon, 字节
- 经济下行所有公司的performance考核都变得更加严格
  - 更频繁的PIP
  - 固定的淘汰比例5%
  - 频繁layoff
  - 更严格的绩效评估
  - 提升bar
  - Culture变差, 以完成产出为唯一目标
- AI 出现，对coding的要求从会完成代码-》会进行设计和review代码

Hire Manager关注什么

所以某些行为就变得极为重要，PIP的原因是perf吗？不是！其实是行为不符合要求

HM关注的行为：

- 按时完成，能够合作完成
- 在不能完成时的处理方式
- 多个紧急任务并行
- 能够超出招聘岗位更多的能力(mentor，带人，带队伍)
- 快速调整，应对不确定性的能力

使用正确的结构来讲故事

你以为的STAR型不是正确的STAR模型（5-8分钟讲完）

- 举个例子，我老公他的situation通常说的都很长，有时候还从头到尾讲一个项目，居然能花10分钟。
- 我不认为situation应该描述背景，situation应该描述你面临的问题和困难本身，而且不要超过三句话
- 我认为的STAR
  - Situation：我在XX项目中，碰上一个XX困难（技术难点/人事难点/deadline难点）
  - Task：我的目标是在困难的条件下，做成XXX
  - Action：我做了ABCD
  - Result：结果我成功的XXXX

模版例子（仅构架，无细节）

S：我在一个XX项目中担任XX migration的工作，但是在这个migration过程中，我发现XX团队提供的migration tooling有很多错误，这些错误导致我多花30%的时间进行矫正/改正/其他。

T：我希望能为我自己和其他小组节省30%的修改和矫正时间，当时有100+个组在使用这个tool，如果这个工具能够完善，将为这100多个组节省30%-50%migration的时间

A：

1. 我将migration过程中发现的问题收集起来，并主动的带领tooling小组make progress
2. 我自学tooling小组的Guice dependency injection技术，在学会之后，30%左右的tooling issue是由我完善的，我帮助修改收集到的tooling问题，用1-3天的时间修复了30%简单的bug
3. 其他70%较为复杂但是严重影响productivity的issue，我整理成文档，和tooling团队一起讨论修复

R：最后90%左右report的bug被成功修复了，我自己的项目从中节省了30%的时间，同时其他组的项目使用这个migration tool的，成功节省了最多50%的时间

Situation:
I was working on a project to create an automated workflow for generating test data based on production data. Since this involved sensitive information, it required Privacy Review approvals.

不清楚的challenge

Task:
Initially, I planned to complete the workflow within the given timeline. However, after engaging with the privacy team, I learned that the approval process would take several additional weeks, causing a delay.

虚假的task

错误的例子

Action:
I immediately notified the stakeholders and my manager, providing detailed context on the delay and its impact. Together, we reevaluated the project priorities. To mitigate the risk, I proposed a [ ] approach—sanitizing unnecessary attributes upfront to reduce the scope of the privacy review. This significantly shortened the approval time.

太简单的action

Result:
By implementing the [ ] solution, I was able to proceed with generating a minimal but usable dataset within the original timeline. The full privacy review was then completed in a second phase, ensuring long-term compliance without blocking progress.

平淡的结果

S:I was working on a project to create an automated workflow for generating test data based on production data. However, because the workflow involved sensitive production information, I engaged with the privacy team and learned that the approval process would take several additional weeks. This posed a major risk, as it would delay the overall project timeline. (直接描述问题和难点)

Task (T): My goal was to deliver the workflow on time while ensuring that the solution remained fully compliant with privacy regulations.

Action (A) - (**多层次的action，层层深入**)

1. Immediate Communication: I quickly notified stakeholders and my manager, providing detailed context on the delay and its potential impact. Together, we reevaluated project priorities.

2. Exploring Alternatives: To mitigate the risk, I proactively proposed and explored multiple solutions:

- A(whitelist) approach – By sanitizing unnecessary attributes upfront, I reduced the scope of the privacy review, significantly shortening approval time.
- Full synthetic data generation – Restarting the data creation process with fully synthetic data, eliminating privacy concerns.
- Manual backfilling – As an immediate workaround, manually filling in existing data to keep the project moving forward.

3. Implementing a Phased Approach: I convinced stakeholders to proceed with the A solution first, which allowed us to generate a minimal but usable dataset within the original timeline. The full privacy review continued in a second phase, ensuring long-term compliance without blocking progress.（短期解决方案和 长期计划都有了）

Result : By implementing the A solution, I successfully kept the project on track and delivered a usable dataset within the original timeline. The full privacy review was later completed in a second phase, ensuring long-term compliance. This approach balanced business needs and privacy constraints, allowing development to continue without disruption.（我造成的好的 结果）

面试之前，必须要准备几个例子？

- 亚马逊14条军规的例子：准备10个不同的故事（重复的故事会减分）
- 必须要准备的例子（你面试之前要背熟的）
  - Mistake/what you could do better/different
  - Conflict例子*2
  - Project you proud of/most challenge/most complicated case
  - 你的弱点
  - delay/miss deadline

亚麻十四条军规

https://www.amazon.jobs/content/en/our-workplace/leadership-principles

Customer Obsession ✅
Ownership ✅
Have Backbone; Disagree and Commit ✅
Think Big ✅
Learn and Be Curious
Earn Trust
Deliver Results ✅
Are Right, A Lot
Dive Deep ✅
Invent and Simplify
Bias for Action

Customer Obsession 常见的考察问题

- Give me an example of a challenging customer (internal or external) interaction. What did you do to work through it? What went well and what would you have done differently?
- Tell me about a time when what a customer wanted conflicted with company processes. What did you do to balance these two priorities?
- Tell me about a time when you worked to change a process or product based on customer feedback.
- Tell me about a time you made a decision that ultimately impacted the customer experience in a negative way. How did you find out about the impact? What did you do to improve the experience for that or future customers?

理论就是你合作的组，你接触的人，内部外部都算你的客户

这道题目的人设相对来说就比较复杂，如果说得简单点，应该就是平衡吧

面试官想听到什么

建议答案：

- 你面对的是一个复杂的棘手的客户需求
- 你积极主动**深度理解**用户的诉求，能拆解用户的诉求，找到最小核心
- 你理解现有资源，公司的商业模式，成本和收入来源
- 你的思考模式+解决方案（精心设计）能平衡用户诉求和公司利益，用最小的成本，满足用户需求

Redflag：

- 你不理解客户想要什么，以及背后的原因，只知道表层的（不合理）要求
- 不理解公司cost，在满足用户需求上，花费了过多精力
- ROI不佳，决策方式有问题

S: 我之前负责一个B端用户平台，当时有一些大客户反馈，导出的报表格式不符合他们财务系统的要求，每次都要手动调整字段顺序，非常费时，甚至有客户说要考虑换平台。
客户希望我们支持自定义导出格式，但如果完全支持自由配置，工程成本非常高，而且 这批客户的营收占比并不大，PM也不太想优先处理。

T:想办法在不增加太多资源投入的情况下，解决客户的核心问题，避免客户流失。

A:我做了三件事：

1. 主动访谈客户，把诉求拆解清楚，发现他们真正需要的是字段顺序和模板固定，并不是真的要完全自定义；
2. 评估技术成本，和后端确认可以通过配置方式预设三种模板，工程量很小；
3. 设计方案并推动落地，我提了“预设模板+使用说明”的轻量方案，说服PM排进下个版本，同时提前向客户沟通预期。

R:

- 功能上线后，95%的客户反馈问题得到解决；
- 满意度评分从3.8提升到4.6；
- 开发只用了1.5周资源，对主线没有影响；
- 这个方案后来还被复用到其他模块。

Ownership

- Tell me about the last project you led or in which you were involved as a team member. Would you have done anything differently? If so, what and why?
- What steps do you take to ensure that your work stays on task? Tell me about a time that you changed your approach to keep yourself on task.
- Can you share an example of a time when you took responsibility for a negative outcome?
- Tell me about a time that you took initiative to solve a problem or make an improvement at work.
- Describe a time when you were given a project but felt like you didn't have the information you needed to move forward. Why did you think this, and what did you do?

你是主动站出来，推进任务，带领大家，承担责任和错误，并不断总结经验，反思和改进的人

面试官想听到什么

建议答案：

- 在任务进度停滞的时候，你主动发现这个负面影响，你积极主动**站出来**
- 你作为主要责任人，花费时间精力去带领大家，解决问题
- 你克服了多个困难（最好2-3个，技术难点，人事难点，资源难点。。。）
- 你有系统性的反思，知道下次怎么做更好

Redflag：

- 你没有主动沟通，或者主动承担责任
- 主动承担的事情缺少商业价值（比如ROI低，对客户，对老板，对公司都不是正面影响的 task）
- 问题太小，冲突太弱
- 没有结果或结果不佳

举例子

- Proud of/challenge
- Junior没做好的时候
- 项目出现困难
- 也可以是小问题，大ROI
- 你的出现为结果带来实质性影响

举例子

Situation:
I was building a **scalable, config-based Person Generator** capable of generating **high-volume, high-fidelity data** with complex **inter-relationships and cross-references**. A key challenge was handling **circular and hierarchical dependencies** in the dataset while also **reducing latency**.

Task:
I needed to design a solution that could efficiently manage **dependencies and relationships** across data while ensuring **scalability and flexibility** for different dataset structures.

Action:
I explored two potential approaches:

1. **Leverage TDP's built-in features** (joins, transformations) to handle the hierarchical relationships and use **batch processing** for generation.
2. **Implement a multi-phase generator**, separating **hierarchy creation from batch processing** to allow more flexibility and performance optimizations.

After evaluating trade-offs, I realized that **TDP's workflow lacked reusability** for datasets with different structures. Instead, I designed a **multi-phase approach**:

- **Phase 1:** Generate the hierarchical structure first with **minimal metadata**, ensuring all inter-relationships are captured based on the user-provided config file.
- **Phase 2:** Use **Guice dependency injection** and **FlumeJava** to **parallelize backfilling** of remaining attributes, optimizing latency and scalability.

Result:

- My solution **scaled efficiently** to handle large data volumes while significantly **reducing processing time**.
- The generator became **configurable**, allowing users to **define custom structures** via a config file, increasing its flexibility across different datasets.
- This experience strengthened my ability to **navigate platform limitations, optimize large-scale data processing, and design reusable, scalable solutions**.

Have Backbone; Disagree and Commit - 典型的conflict问题

- Tell me about a time where your team was about to make a final business decision,but you felt that additional insights or perspectives were needed to make the best decision.
- Tell me about a time you and a coworker had a different set of expectations about how work needed to be executed.
- Have conflict with someone
- Difficult person to work with

人设:你敢于直面问题，了解问题，有解决复杂问题的手段，且总是造成积极影响(结果)

面试官想听到什么

标准答案：

1. 你自己的想法的原因表述清楚
2. 对方想法背后的逻辑表述清楚（不仅仅是对方的意见是什么，而是了解对方想法的背后逻辑）
3. 同时你通过合作让双方各退一步（这里有很多分化，比如听你的了，或者听他的了）
4. 最后结果一定是成功的：达成一致，到达中间点

Redflag：

1. 你说不清楚对方这个意见背后的原因（老板没有budget啦，团队有别的priority啦）
2. 最后结果negative或没结果 - 你造成的影响是负面的或没结果
3. 合作过程含糊或复杂度低 - 你说的哪一点打动了对方？

举例

向下冲突

- Mentor：和intern的例子，指导intern，有来有回，最后帮助对方成功学习到某个tech skill或者soft skill

向上冲突

- 和manager有不同意见，是怎么说服对方的？
- 和更高级别工程师有不同意见，是怎么说服对方的
- Principal 和 senior manager 冲突的例子

同级别冲突

- 向上汇报

Invent and Simplify - 负面的例子

Context
I built an data generator tool with high flexibility. Users could define structures through a custom configuration file. However, this design made the tool complicated for many users. They had to input all parameters manually every time, which took time and easily caused mistakes. There were also many unnecessary fields not relevant to most users.

Problem
The tool was too complex for general users, even though it was flexible.

没有task

Action

- Talked with users to better understand their needs.
- Added default values and a simple UI, so users could quickly generate data without filling in every field.

改动太小太简单，也不清楚价值是什么

Result

- Saved users time and reduced manual input errors.

Invent and Simplify

Junior level 的例子:

一个入职不久的初级工程师发现，新人加入团队时，设置本地开发环境通常需要 2~3 小时，步骤繁琐，容易出错（比如环境变量配置、依赖安装顺序、数据库迁移命令等）。

这个工程师自己经历完一次痛苦的setup 后，写了一个简单的 Bash 脚本 + README 指南，将以下步骤自动化：

- 安装依赖
- 设置环境变量
- 自动运行数据库迁移和 mock 数据注入
- 打开本地服务并验证端口

新人只需运行 ./setup.sh 一条命令，5 分钟内完成本地开发环境配置

所有人开始用这个脚本，新人 onboarding 体验大幅提升

减少了资深工程师重复指导的时间

Senior Level例子

公司内有多个microservice团队（A、B、C）各自维护自己的数据 mock 工具，用于本地开发和 CI 测试。工具各异，格式不统一，维护成本高，且容易造成数据不一致导致测试 flaky。

这位资深工程师主导调研了各团队使用的 mock 方式，发现其核心逻辑有80%是重复的，于是提出设计一个统一的 **可扩展的 mock service 平台**，核心包括：

- 定义标准化的 mock schema
- 提供 mock 生成器支持多语言（Python/Node）
- 可插拔的数据模板（支持随机性 +业务规则）
- 内建对 CI/CD 的接口支持

新服务只需几行配置即可启用mock，统一维护逻辑

减少重复构建的开销（平均每个团队减少 1～2 人月的维护工作/年）

测试数据一致性显著提升，CI flaky rate 降低 60%

最终被平台团队正式接管并推广到公司级别标准工具

Learn and Be Curious

- 初级和mid level常见的考察点
- Senior level考察这个问题的比较少，因为假设你应该在其他例子中有体现强学习能力了
- 建议在这个市场上，多加一些AI/ML相关的内容，结合自己的工作经验，出一个例子

结合当下流行的AI/ML关键词的一个例子

Situation:

I was developing a **synthetic test data generator** to produce high-fidelity datasets. Given the complexity of maintaining **realistic relationships and variability**, I wonder if **Machine Learning (ML) or AI** could provide a more **adaptive and scalable** solution.

Task:

My goal was to deliver the project using a **traditional rule-based approach** to ensure timely completion, and simultaneously **explore ML/AI techniques** in my own time to provide a proposal and share with related teams to streamline the procedures for synthetic data generation.

Action:

🔍 **Self-Learning & Research:**

- I **dedicated personal time** to learning about ML-driven synthetic data generation.
- I studied **generative models** like **GANs (Generative Adversarial Networks)** and **VAEs (Variational Autoencoders)** through online courses and technical papers.
- I also researched **real-world applications** of ML-based synthetic data generation in **privacy-preserving AI** and test automation.

📝 **Proposal for Future Integration:**

- Once I gained a basic understanding of ML in this domain, I **evaluated the feasibility** of incorporating ML into future versions of the generator.
- I drafted a proposal outlining how **ML could improve data realism and adaptability**, as well as the need for **collaborating with data scientists and ML engineers** to ensure its effectiveness.
- I shared this proposal with my manager, emphasizing how an ML-driven approach could enhance test data quality while ensuring compliance with data constraints.

Result:

✅ Delivered the **initial version of the generator on time** using a traditional rule-based approach.
✅ Gained foundational knowledge in ML/AI for synthetic data generation through **self-learning and research**.
✅ Proposed an ML-driven approach for **future iterations**, leading to discussions on **collaborating with ML experts** to explore its feasibility.
✅ Sparked interest in leveraging ML for **long-term improvements**, ensuring the generator could evolve to produce **more adaptive and high-fidelity test data**.

Bias for Action - speed matters, ok to take risk

- Tell me about a time you had to make an urgent decision without data
- Tell me about a time when you launched a feature with known risks
- Tell me about a time when you found an opportunity that no one else saw.
- 几件都很紧急的事情发生，你是如何决定优先级的
- Describe a situation where you made an important business decision without consulting your manager. What was the situation and how did it turn out?
- Tell me about a time when you had to analyze facts quickly, define key issues, and respond immediately to a situation. What was the outcome

线上崩溃 vs 发布流程卡住

情境：
你在准备一个重要的 feature 发布，发布流程刚卡在 staging，CI/CD pipeline 异常。此时你收到警报：线上老模块突然出现大面积 crash，影响 5 万用户。

你的 Bias for Action：
你立刻中止发布流程，把注意力转移到线上 crash 上，先快速分析 crash 日志，rollback 最近的依赖版本，减少影响范围，同时 ping oncall 组接手发布进度。

你的判断逻辑：

> “Release 只是延后影响，而 crash 是即时且不可逆的用户损害。我优先处理能立刻止血的。”

多项目冲突，如何分配资源

情境：
你负责的两个项目突然都需要 infra 支持，但 infra 组本周只能给出半天资源时间。一个项目是战略性功能，另一个是用户频繁抱怨的低稳定问题。

你的 Bias for Action：
你立刻收集 crash 数据，收集用户反馈量，快速写了两页决策简报，发给 TL 和 PM。最后决定让 infra 先修稳定性问题，因为有 clear impact data。你自己晚间加班推进另一个项目的替代方案。

你的判断逻辑：

> “我不是简单地选用户多的那个做，而是综合考虑了可落地性、风险暴露时间窗口，以及业务影响。我选择了‘短期能止痛，长期不误事’的方案。”

A/B 实验失控，数据异常

情境：
你在周五晚上收到A/B 实验结果的数据报警。原本计划周一早上分析，但你看到有一组用户异常流失。

你的 Bias for Action：
你放下手头任务，晚上紧急登录分析平台，快速定位问题是实验分组配置错误。你直接 disable 实验并通知数据科学家，避免了更多用户被影响。

你的判断逻辑：

> “等待不是选项。哪怕可能误报，我也要先止损，然后再分析错因。”

两组同时请求支援，你只能选一个

情境：
你是 senior 工程师，iOS 和 Android 同时遇到严重 UI 渲染崩溃问题。你只精通 iOS，但 Android 的问题似乎影响用户更多。

你的 Bias for Action：
你判断 iOS 的问题有 rollback path，而 Android 团队没有相关经验。你决定协助 Android 团队 debug 并发 patch，同时将 iOS 问题降级处理，并安排 junior 工程师协助。

你的判断逻辑：

> “我能立刻创造最大影响力的地方才是最优解，不是我熟悉的地方。”

面试官想听到什么

- 有理有据当然最好(或依据以往经验做出判断)
- 没有数据, 你的个人哲学是什么？
- 讲清楚自己的判断逻辑, 做到有道理可依, 讲清楚逻辑就可以了

Redflag:

- 没处理过任何紧急的情况
- 明显的错误决策(这边crash爆炸了, 还在慢悠悠做regular项目)
- 考虑局限(仅短期思考, 无长期策略), 不全面,
- 无风险意识, 忽略明显的高风险
- 可能缺少事后沟通

Earn Trust

- A co-worker constantly arrives late to a recurring meeting. What would you do?
- Tell me a piece of difficult feedback you received and how you handled it
- Give an example of a time where you were not able to meet a commitment to a team member. What was the commitment and what prevented you from meeting it? What was the outcome and what did you learn from it?
- Tell me about a piece of direct feedback you recently gave to a colleague. How did he or she respond?
- How do you like to receive feedback from coworkers or managers?
- Tell me about a time when you had to tell someone a harsh truth.
- Tell me about a time when someone (peer, teammate, supervisor) criticized you about a piece of work/analysis that you delivered. How did you react? What was the outcome?
- 这个问题大部分时候和feedback还有接受feedback有关
- 也有一些问题是关于如何融入小组，获得大家信任的

面试官想听到什么

- 你是否有花时间跟小组关键人物沟通
- 你对于feedback是否接受，不接受和接受的理由是什么
- 当别人做的有问题的时候，你是否指出来，怎么做的

Redflag：

- 在进入新环境或者和新人合作的时候，没有明显的正面结果（越来越相信）
- 不接受feedback的理由不常规
- 指出的方式太生硬，破坏同事关系，关系紧张

真实的负面例子

拒绝feedback

我曾面试一位候选人，她提到自己收到 经理反馈：需要提升对业务的理解。她声称接受了 这个反馈并采取了行动，但当我 进一步追问她具体做了什么，以及 这些行动带来的业务影响时，她无法清晰回答。她 说不出项目在业务上的贡献，也缺乏有 说服力的结果。最终我们判断她并未真正吸收并落 实反馈，反映出对成长的意愿和执行力不足，因此未能通 过面试。

缺乏理解和长短期决策

一位候选人，他分享了一个和上 级（或是 senior developer）有分歧的经历。当时他在项目中提出了一个关于 scalability 和 extensibility 的长期解决方案，认为这比老板提出的短期方案更好。但在 团队沟通中，老板更 倾向于落地短期方案，最 终他选择听从上级，完成了那个短期实现，并且交付了 结果。

听起来这是个中立的合作，但在我 进一步追问后我发现，他有几个问题

正面例子

无法按时交付，主动沟通风险

我曾负责一个 API 重构任务，初期预估两周内交付，但中途发现依赖系统更新滞后，严重影响进度。我第一时间主动跟 PM 沟通，明确指出延期风险，并提出短期替代方案，保障关键路径不被阻塞。虽然延迟了两天，但因为我的透明沟通，团队反而更信任我，后续还让我带另一个复杂项目。

给别人提意见的例子

在一次 code review 中，我发现一位新加入的 junior 同事在多个 MR 中重复使用了不必要的嵌套判断，代码虽然能跑，但可读性和扩展性都很差。我没有在评论里直接否定，而是用注释提出建议："这段逻辑可以通过 early return 精简，代码会更清晰。"并附上了一个精简后的对比写法。他在 review 后私下问我更多细节，我们还一起看了几段类似优化后的代码。之后他的 PR 可读性提升了不少，还在组内分享了一个"常见代码清理技巧"的小讲解。

Are Right, A Lot

- Tell me about a time when you had to work with incomplete data or information.
- Tell me about a time when you were wrong.
- Tell me about a time when you had to use your judgment to solve a problem.
- Tell me about a time when you incorporated a diverse set of perspectives into solving a problem.

是什么让你总能做出正确的选择，如果做错了，你是如何反思和改进你的决策思路的

面试官想听到什么

你在各种情况下的的决策思路：

- 有多个任务并行，什么决策思路选择优先级
- 缺少数据的情况下，什么决策思路做出对用户尽量好的选择
- 当充满不确定时，你是如何收集信息的？你找了谁？和谁聊天
- 做错事情时，有没有反思，反思后的结果是否能够帮助你未来更有经验？
- 多方信息有冲突有噪音，你是如何选择的？
- 风险控制，风险最低，计划b

Redflag：

- 我没做错过事情
- 我没学到什么对未来有帮助的能力（没有反思）

I was working on a **test migration project**. After successfully migrating all tests, I started noticing **flaky test failures** that were intermittent and difficult to reproduce. This was causing delays and frustration for the team.

Task:

My goal was to **stabilize the tests** by eliminating flakiness while maintaining test coverage and reliability. Since this was a critical migration, I needed to ensure that the new framework provided the same level of confidence in test results as the old one.

Action:

To quickly fix the flakiness, I decided to **mock one of the key dependent components**, thinking that this would make tests more deterministic and prevent failures caused by external dependencies. Initially, this seemed to work—flaky failures were reduced, and tests became more stable.

However, after a few weeks, a teammate pointed out that some **critical test scenarios were no longer covered**. I realized that by mocking out that component, I had inadvertently **reduced the effectiveness of test coverage**, making the tests pass too easily without catching real

Once I recognized this mistake, I immediately:

- **Analyzed the impact** by reviewing test coverage reports and identifying gaps introduced by the mock.
- **Rolled back the excessive mocking** and instead used a **controlled test environment** to simulate the dependency more realistically.
- **Worked with the team** to implement a **hybrid approach**, where we only mocked unstable external dependencies while preserving realistic interactions for critical scenarios.
- **Documented my learnings** and shared them in a post-mortem with the team to prevent similar mistakes in future migrations.

Lesson Learned:

This experience taught me that **stability should not come at the cost of test effectiveness**. Instead of quickly patching flaky tests with mocks, I learned to **investigate the root cause of flakiness** and apply **more thoughtful solutions** like dependency isolation, proper synchronization, and controlled test environments.

I was working on a test migration project. After successfully migrating all tests, I noticed that the CI/CD pipeline became unstable post-migration. The tests were flaky, preventing other engineers from deploying changes effectively. To mitigate the flakiness and improve the stability of the CI/CD pipeline, I discussed the issue with the tech lead and decided to mock the ABC component since we were using test accounts to run these tests. The goal was to eliminate the non-deterministic output returned from the ABC component.

However, after a few weeks, a teammate pointed out that some test results seemed to be false positives. Upon investigating the test logs, I found three tests that had passed when they should have failed. The root cause turned out to be the misconfiguration of the mock ABC component leading to incorrectly granting access to some test accounts resulting in false pass results.

Realizing this issue, I drafted a hybrid proposal and scheduled a meeting with the tech lead. In my proposal, I suggested applying the mock ABC component only in tests that were not sensitive to authentication. For tests related to authentication, I rerouted the test to use the real ABC components to ensure correctness.

Additionally, I documented my findings, created clear criteria for categorizing tests, and shared these insights with the team during a Lunch and Learn session. This experience reinforced my understanding that mocking certain modules carries risks. Therefore, before implementing mocks, it is crucial to conduct thorough research, gather as much information as possible, and fully understand the risks before making informed decisions.

🔹 Situation (S)

"I was on-call and noticed that some devices had consistent failures, leading to frequent reassignments and increased test latency. Our availability metrics showed that these devices were falsely marked as ‘available,’ but once assigned to a test, they would immediately fail with persistent errors. This was impacting test reliability and overall system performance."

🔹 Task (T)

"My goal was to identify the root cause of these false availability markings and implement a long-term, automated solution to improve device reliability and reduce unnecessary retries."

🔹 Action (A)

**"I took a structured approach to solve this problem:*
*1️⃣**Root Cause Analysis**: I created a cron job that auto-generated daily accuracy reports, aggregating failure counts by device. This helped me identify common failure patterns across the most unreliable devices.*
*2️⃣**Findings & Insights**: By analyzing the data, I discovered that devices often became ‘unready’ due to issues such as low battery, unstable WiFi, outdated ADB versions, and un-removed user accounts*
*3️⃣**Solution Implementation**: Based on these insights, I proposed and implemented a **health monitoring system** that:*

- *Periodically checks critical device metrics like battery level, WiFi stability, ADB version, and user account presence.*
- ***Automatically resolves issues** (e.g., reboots devices, reconfigures WiFi, restores user accounts).*
- *Prevents devices from being falsely marked as available.*

*4️⃣**Collaboration & Deployment**: I worked with the team to roll out the health monitor system and fine-tune its thresholds based on real-world test results."**

🔹 Result ®

Device reliability increased from 92% to 97%. **Availability metric accuracy improved**, leading to fewer false positives and better resource allocation. Reduced test latency and failure reassignments

- Situation (S)

"I was on-call and noticed that some devices had consistent failures, leading to frequent reassignments and increased test latency. Our availability metrics showed that these devices were falsely marked as 'available,' but once assigned to a test, they would immediately fail with persistent errors. This was impacting test reliability and overall system performance."

- Task (T)

"My goal was to identify the root cause of these false availability markings and implement a long-term, automated solution to improve device reliability and reduce unnecessary retries."

- Action (A)

To systematically diagnose and resolve the issue, I took a structured, data-driven approach, diving deeper into each layer of the problem:

1. ***Automated Data Collection*** *– I developed a* ***cron job*** *to automatically generate daily accuracy reports, aggregating failure counts by device. This provided clear visibility into which devices consistently exhibited issues.*
2. ***Targeted Investigation*** *– Using the report data, I* ***prioritized the top 20 problematic devices****, manually analyzing logs and system states to pinpoint root causes.*
3. ***Root Cause Analysis*** *– Through my investigation, I identified multiple factors contributing to devices being falsely marked as available, including* ***outdated ADB versions, unremoved user accounts, and low battery levels****, among others.*
4. ***Proposed a Scalable Solution*** *– Leveraging these insights, I designed a* ***health monitoring system*** *to periodically check device metrics and proactively* ***auto-remediate issues*** *where possible.*
5. ***Team Collaboration & Implementation*** *– I presented my findings and proposal, using* ***accuracy report data to support my recommendations****. The team immediately agreed with the approach, and together we discussed the* ***optimal cadence*** *for running the health monitoring system to minimize disruptions.*

Device reliability increased from 92% to 97%. **Availability metric accuracy improved**, leading to better resource allocation. Reduced test latency and failure reassignments

Deliver Results

Tell me about a time when you missed the deadline for something you were tasked to deliver. How did you communicate about the delay? What did you learn from that experience and implement for future deadlines?

Describe a time when you were given a complex problem with a tight deadline. What

steps did you take to ensure that you achieved a quality deliverable in the time you were given?

Tell me about a time where you had to make a trade-off to complete a specific task or meet a deadline. What was the trade-offs and how did it improve the final product?

Tell me about a time when you felt you could have delivered a better quality project or end result. What would you have done differently?

面试官想听到什么

- 在项目有可能延期的时候你是否及时和相关人员沟通？
- 项目延迟的时候你是怎么做的，灵活度如何
- 优先级有冲突的时候，你是如何排序的（类似bias for action），取舍的原则是什么
- 质量和速度的平衡，当两者冲突，你是如何处理的
- 是否有反思，有经验的积累，以后不会在发生？

Redflag：

- 没有miss过deadline，说明你的任务太简单，缺乏挑战性，或者目标太低，之前公司太佛系
- 缺乏反思，不知道造成delay的真正原因是什么

Situation:
I was working on an automated workflow for **XX data generation** using B Tool. A key requirement was **1-1 joins**, but B Tool only supported **1-many joins**, which blocked my project. Given the **tight deadline**, waiting for the B Tool team to implement this feature wasn't an option.

Task:
I needed to find a way to **deliver the solution on time** while ensuring **data accuracy and scalability**.

Action:
I quickly explored two approaches:

1. **Post-processing the output** to filter excess records.
2. **Implementing 1-1 joins myself**, making it reusable for similar cases.

Situation 和 Task 还可以
但是action 描述的非常模
糊，看不出来 难点和复 杂
性

Since post-processing wasn't scalable, I chose to **implement 1-1 joins myself** while consulting with the TDP team for best practices. I optimized the implementation to **minimize processing overhead**, ensuring it fit within our performance constraints.

Result:

- I delivered the project **on time** without depending on external team timelines.
- The implementation was later **adopted by other teams** facing similar issues.
- This experience reinforced my ability to **quickly assess trade-offs, make technical decisions under pressure, and deliver results within a tight deadline**.

A:
My manager initially suggested a quick post-processing workaround—filtering data after generation—but I had concerns about long-term maintainability and data accuracy. I proposed a more robust solution: implementing native 1-to-1 join support within our system.

I didn't jump in blindly. First, I:

- **Validated technical feasibility** by checking system constraints and data scale (millions of records)
- **Benchmarked memory usage** to ensure performance wouldn't degrade under large joins.
- **Engaged the TDP team early** to ensure they'd review and later own the code.

During development, I hit performance challenges—our naive implementation caused memory spikes. I iterated on the algorithm using hash-based deduplication and stream processing to reduce memory footprint. I also added a fallback for edge cases (e.g., null values, unbalanced datasets) and unit-tested the correctness across thousands of samples.

R:
I delivered the feature within 5 working days—still within our buffer. It not only unblocked my project but was soon reused by over ten other teams. The TDP team officially adopted and maintained it going forward. This avoided duplicated efforts and helped standardize 1-to-1 joins across the org.

Reflection:
Looking back, this experience taught me that delivering results doesn't mean just finishing a task quickly—it's about making the right trade-offs. Instead of applying a short-term fix, I delivered a scalable solution under pressure, while balancing stakeholder alignment, technical quality, and long-term ownership.

HM“推销型”面试？其实是在评估你什么？

- 推销型面试，先不要着急说话
- 多问问题，找准对方对这个岗位的定位
- 当 Hiring Manager 介绍团队结构和技术栈时，你能否立刻判断哪些与你经验高度相关？
- 面试官在讲产品和协作流程时，你有没有主动回应，展示你过往相关经验？
- 你是否能在技术不完全匹配时，展示你的适应力和学习力？
- 你能不能听懂对方在寻找什么样的人？并用你的经历回应这些潜在需求？
- 最后，你有没有表现出你真的想加入这个团队、这个公司？

Hiring Manager 行为面试，你准备好了吗？

- 短时间内，面试官需要听到足够多positive信号
- 无效内容越少越好
- 如果面试官不停追问某个技术细节，你知道这是好信号吗？
- 当被问到行为问题却不确定重点时，你会怎么处理？
- 你是否知道哪些例子能真正体现你对目标岗位的契合度？
- 面试中，你能展示出“成长曲线”而不仅是简单经历吗？
- 面试中有哪些小技巧提高HM通过率？

HM面试例子的准备和调整

1. **了解面试官的需求：**
    - **外部用户PM：**需要市场分析、收入数据和用户增长经验。
    - **内部工具PM：**需要理解技术细节，协调解决问题。
2. **根据岗位需求调整例子：**
    - **外部用户PM：**强调市场分析和用户增长。
    - **内部工具PM：**展示如何协调解决技术问题，推进技术项目。
3. **询问面试官需求：**
    - 问面试官：“你在找什么样的PM？”根据回答准备合适的例子。

Dive Deep

- Tell me about a time you identified and solved a recurring technical issue.
- Describe a situation where you had to debug a complex failure in a production system.
- Tell me about a time you automated a process to improve system reliability.
- Give me an example of a time when a project or system wasn’t performing well. How did you investigate and what did you find?
- Tell me about a time you challenged an assumption or a common belief based on your own findings.
- Describe a time when your intuition was wrong. How did digging into the details change your perspective?
- Have you ever found a root cause that was completely different from the initial assumption? How did you uncover it?
- Tell me about a time when you needed to understand a process you were not familiar with. How did you learn about it and what did you discover?

面试官想听到什么

- 想听到不是一次探究就找到答案了，而是层层深入，抽丝剥茧
- 你是否能够主动发现问题，你监测指标是什么
- 你是怎么一步一步挖掘问题的？
- 你是否挑战了错误的假设？有没有最终找出真正的 root cause？

Redflag：

- 过程简单直接，缺乏挑战
- 缺乏深度思考，发现的是表面问题

❌ Dive Deep 负面案例（结构完整但深度不足）

在一次 on-call 巡检中，我注意到多个设备频繁掉线，导致测试被反复重启。虽然这些设备在系统中显示为“available”，但一旦使用就会报错并失败。
目标是找出设备错误标记为可用的根本原因，并提出一个能长期提升设备可靠性的方案。

🔹 Action

我做了如下几个动作（这是此例子的问题点所在）：

1. 创建了一个 cron job，生成每日设备失败报告
2. 分析数据后发现失败设备通常有电量低、WiFi 不稳定、ADB 版本旧、未清理用户账户等问题。
3. 基于这些问题，我开发了一个设备健康监测系统，会定期检查电量、网络、ADB 等，并自动重启或修复设备。
4. 联合团队部署并重启系统剪小错误率。

🔹 Result

设备可靠性从 92% 提升到 97%，错误标记减少，测试延迟也显著降低。

🔹 Situation (S)

我注意到系统中大量设备频繁变为 unavailable，直接影响了数百万级用户的测试任务，也拖慢了多个交付节奏。。初期团队猜测问题源于设备注册流程不稳定（假设A），但始终找不到有效证据。

🔹 Task (T)

目标是在短时间内深入调查设备可用性问题，找到真正的 root cause，并设计一个高性价比、可长期维护的解决方案，同时避免消耗团队大量资源。

🔹 Action (A)

- 初期大家的猜测是 WiFi 或者 Battery 的问题，但我不满足于这个猜测。我写了一个 tool 自动收集和聚合失败原因的日志数据，结果发现真正主要的原因是B、C、D 三类，这直接推翻了团队最开始的假设 A。
- 分析数据后，我发现问题主要集中在三个新 root cause 上：低电量、WiFi 不稳定、ADB 工具版本过旧，完全不同于初期假设。
- 发现了 BCD 之后，最直接的办法当然是写三个 patch，一个个解决。但我意识到这样太浪费资源了，于是我开始思考有没有统一的解决方案。
- 我拉了组里的 principal engineer 和几个 senior engineer，一起复盘数据和讨论方案。我们通过对多个 bug 数据的横向分析，最终发现其实这几个问题背后有一个共性 root cause。于是我们设计了一个更系统性的解决方案，一次性修复多个路径，大幅减少了维护成本。

面试前一个月的准备BQ

Step 1：翻工作记录
去翻你的 Calendar、Docs、Slides、Performance Review，先把能找到的工作记录都整理出来。

Step 2：找出亮点
从中提炼出 10~20 个“可以拿出来讲”的场景，比如和 PM、QA 开的会、写过的文档、解决过的问题等等。记关键词就行，方便你回忆。

Step 3：对应 Leadership Principles
把这些亮点分别归类到常见的 Leadership Principles 里，比如 Ownership、Customer Obsession、Conflict 等，每类至少准备一个例子。常考的准备两个

Step 4：用 STAR 模型写故事 + 练习
把每个例子补充成 STAR 格式，练到能 5~8 分钟内讲清楚，不能超过 8 分钟。

⚠️注意：例子必须对口！
不要讲和岗位无关的经历，比如软件面试讲太多硬件项目，会让面试官觉得你不 match

面试前3天到一周

✅ **第一步：反复 练习**
把准备好的 10–20 个例子多练几遍，熟练到任何问题都能自然举例。

✅ **第二步：联系 Recruiter**
提前问清楚行为面试会问什么类型的问题，合理聚焦准备重点。

✅ **第三步：研究公司核心价 值观**
了解公司最看重的 3–5 个价值观，准备贴合这些点的故事。

✅ **第四步：准 备 “Why us” 回答**
查清楚公司背景、岗位信息，提前准备好“为什么你想加入这家公司”这种常见问题。

Weakness

根据公司风格来回答：

- 快节奏很卷的公司
  - Impatient
  - 你喜欢完成事情，所有有时候会忽视WLB，会burnout
- 比较传统的公司 - 我倾向于做比较完备的方案，喜欢按部就班推进，不喜欢打乱计划
- 成长型公司
  - 我在某些技能上还可以提高（public speaking，team leadership，provide constructive feedback，presentation skill）
  - 我曾经第一次做XX模块的时候没做好
  - 比较内向，再尝试公共场合多说话（上课）

面试官关心什么

标准答案：

1. 真实的weakness
2. 如果是重要的weakness，跟未来工作相关的要说明你改进的流程，而且要尽量详细
3. 结果要有一定进步

Redflag：

1. 不要说你没有weakness，每个人都有weakness
2. 不真实，虚假的弱点
3. Senior不要说太junior的弱点，比如不会跟别人合作
4. 你的弱点无法改变且对潜在工作造成负面影响
5. 你没任何可量化的进度

End

📌 课程回顾

- 覆盖了10大 Leadership Principles，拆解每个高频 Behaviour Question 的考察点，分析大量真实案例 + 面试红线提醒，教你如何整理个人故事+ STAR 模型高效表达

🌱 适合人群

- 跳槽面试包括bq轮的职场人
- 在职转岗、初入职场、无经验求职者
- 想要提前建立职场行为意识的人

🙏 致谢

- 感谢大家一路以来的支持和信任，我不是全职做咨询的，这门课是我过去15年工作与面试经验的浓缩总结
- 希望课程能成为你反复查阅的「行为面试工具书」文件夹会不定期更新干货，欢迎随时回来看看！

1:1 咨询 Momo's Chat Room: https://momochatroom.square.site/

Why us - 别小看这个问题，**回答好可以反 败为胜**

常见误区

只说“对公司产品感兴趣”是不够的，尤其在 hiring manager 面前。

正确答法：重点说这三点 ✅

1️⃣ **你是深度用 户**
用过产品、真的懂用 户痛点，进入公司后更有 ownership、更有动力优化产品。

2️⃣ **硬实力匹配公司需求**
不是“我很强”，而是“公司在 XX 技术做得好，我正好有相关 经验，可以贡献价值”。

3️⃣ **价值观匹配**（适合 peer 面试官）
你喜欢什么样的文化：bar 高、成长快、互相帮助？

面试技巧 - stick to the point

你想要说什么point先想清楚

- 先把**想表达的人 设**想清楚，才能保 证不偏题

你讲的故事，是为了证明你的观点

- 面试官不关心：你做了什么 项目，你在哪家公司
- 而是关心：**你是一个什么 样的人？**

面试官想知道的是：

你是一个——
✅ 关心客户、能处理复杂关系的人
✅ 在快节奏环境下能高效完成任务的人
✅ 有带人能力的人
✅ 能带大项目、拆解复杂任务的人
✅ 技术能力超群的人

❗太啰嗦只会让人忘记你的重点

BQ面试通常只有**短短45分钟**
废话不能帮你得分
清晰、有重点地展示你的人 设，才是关 键！

Staff/Senior+ BQ 回答关键点

- Staff 级别的 BQ 回答必须兼具**技术深度和组织影响力**。
- 首先，**你要影响一群人**，推动跨 org 的复杂项目，影响 Director、VP 或技术方向，而不是只和个别工程师合作完成任务。
- 其次，**体现你的技术领导力**：你提出并推动落地过关键架构升级、系统重构、性能瓶颈优化、AI/ML 应用、跨平台组件统一等方案。你解决的是别人搞不定的核心技术问题，具备系统性思维、架构判断力和前瞻性技术视角。
- 最后，**Delegation 是关键**：你制定方向，指导工程师实现方案，自己聚焦于突破性技术难题与协作瓶颈，而不是下场做每个细节。
- **Staff 是技术与领导力的双重体现，不是代码行数的叠加。**

如何准备 Staff/Senior+ 级别 BQ 面试

- 一场面试至少准备 3–5 个 solid 的 staff 级别例子。每个例子都要能体现出足够的影响力，比如推动多个团队合作、影响 几十位工程师、最终方案成功落地。
- 不管被问到什么问题，比如 ownership、customer obsession、conflict，都要能自然切入这些高影响力的项目。
- **你的目标不是“回答问题”，而是把影响力直接甩到面 试官脸上。**
- 不要藏着掖着，也别让对方去猜你有多强。
- 如果面试快结束了还没讲到你影响力最大的项目，就要主动切入：“Not sure if this fits exactly, but let me share…”
- Staff 面试，拼的就是技术深度 + 影响广度。

面试后被养鱼了怎么办？

用thank you letter去催结果！

可以准备一个thank you letter 模版

对于很想去的公司，如果被养鱼，最好写一封thank you letter去 “催一下”

发给谁：当然最好的就是每一个面试的人都发，HM，其他同事，recruiter，没时间的话，优先发给hire manager

Thank you letter 怎么写

内容：

1. 仔细的描述面试当中某些亮点，和聊的愉快的地方
   a. “面试当中您谈到了XXX这个地方，我觉得非常有趣，自己回去看了下，谢谢您指出来！”
   b. “您谈到这个岗位的XXX，产品XXX，我都有使用过，是深度用户，因此面试之后觉得更感兴趣了”
2. 表示对公司氛围的满意
   a. 我非常enjoy这次interview，我觉得和其他面试官同事聊的很愉快，公司氛 围很好，我们还聊了周末的球赛！
3. 表示自己可以为公司带来的价值
   a. I’m excited about the opportunity to contribute my experience in XXX
4. 表示对于加入公司的期待：looking forward to join XX/looking forward to hear back from you
5. 面试当中没来得及答完美的问题
6. 每个面试官最好都写不同的，如果没有时间，HM最重要，职位越高越重要