Software Engineer Interview Guidelines - 2020Q3

We are hiring strongly qualified candidates for Software Engineer at levels 25-27. The job descriptions are listed in the Appendix. The candidates we seek will help us build the science infrastructure behind a new CPC advertising infrastructure and our other components. We want to hire candidates which are listed as “exceeds expectations” below and should raise the overall average within the team. We have some notes on collecting the feedback below.

The hiring process follows the eBay standard practice [[1](https://careers.ebayinc.com/how-we-hire/%E2%80%A9)]. Candidates will pass through 2 phone interviews and an onsite. The responsibilities for each interview are as follows:

**Phone Screen 1**. Covers Basic Coding: Coding question related to ML

**Phone Screen 2**. System Design

**Onsite**. Mix of ML Implementation, Big Data Coding, Design , and breadth, coding, and behavioral questions [[2](https://hub.corp.ebay.com/site/people/page/manage-interview-question-library/us/manager)]. In the days of shelter-at-home, all interviews are virtual, but we expect the onsite to occur all within one day. Behavioral questions include Customer Focus, Cultivates Innovation, Manages Complexity.

For Software Engineers, we generally seek candidates who have a strong background in computer science and code and math or machine learning. Collaborating closely with scientists on complex algorithmic solutions, they need to understand the math and hold high standards on coding and pipelines. We do not want engineers who want to produce hacky proof of concept systems. Why? Because we have scientists for that. Instead, we want engineers to drive high standards so that our systems are more reliable, more robust, and more flexible than other traditional services.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MTS1 - T25** | **MTS2 - T26** | **Sr. MTS - T27** |
| **Domain experience (e.g., Ads)** | No | Some familiarity | Yes |
| **Experience in web / IT** | Nice to have | Yes | Yes |
| **ML Implementation** | Basic understanding of learning problems and implementation | Understand pros and cons of implementation strategies | Experience implementing large-scale modeling architectures |
| **Basic Coding** | Can write correct and logical code easily. Understands key data structures. | Writes clean and well-structured code. Can derive new existing or new data structures. | Writes clean and well-structured code. Can derive or optimize data structures |
| **Coding standards** | Writes clean code and follows best practices | Follows practices consistently, implements maintainable code | Follows practices consistently, implements maintainable code |
| **Design** | Model basic design architecture problems | Design for scalable and maintainable solutions, failover strategies | Design robust and logical architectures no more complex than needed |
| **Big data coding** | Can write basic data processing code in pseudocode, such as map reduce, spark, or sql | Writes modular data processing code that can be put into data pipelines | Writes modular data processing code that can be put into data pipelines |

# Software Engineer Competencies

Assessing each component, though varying by level, roughly falls into four major areas. This is in addition to behavioral questions [[2](https://hub.corp.ebay.com/site/people/page/manage-interview-question-library/us/manager)]. Each competency is rated individually according to our expectations, relative to the average of existing members at that level. Comparing against the average person, rather than any particular person, drives increasing performance standards. The calibration is as follows:

**Below Expectations**. Below the existing expectations of an average member at the role and level. Had this person been hired, they would lower the average for the competency.

**Meet Expectations**. Close to what we expect from an average member at the role and level. Had this person been hired, they would perform at the average but not increase it.

**Exceed Expectations**. Better than the average member at the role and level. Had this person been hired, they would raise the overall average of the team.

The technical competencies and ratings are detailed are as follows:

### Machine Learning Implementation

In assessing ML implementation, we test the candidate’s flexibility in applying multiple techniques to a hypothetical problem statement. We do not expect engineers to take on research and science work, but they are involved in the implementation and should have some understanding of the basics. Ideally, candidates should have understanding of someone who has taken a course in machine learning--understanding basic modeling approaches, the general training cycle, and can understand a learning problem that is posed. We expect engineers to focus on the implementation aspect of ML: updating models, scheduling, monitoring, efficiency, etc. For engineers, we expect to level them below the level of a T24 Applied Researcher.

**Below expectations**. Candidate knows the terms, but introduces keywords or buzzwords without clearly explaining them or the details. They may say “use decision tree” but cannot explain how a decision works or why it would be preferred as opposed to something else.

**Meet expectations**. The candidate can explain the algorithms at a high level and can focus on the implementation. When presented with a simple problem the candidate can speak about what might be involved in implementing the solution.

**Exceed expectations**. Candidate knows the algorithms in detail, they can write out the formulas, explain challenges in implementing model-based solutions in production environments.

### 

### Basic Coding

We expect our Software Engineers to be able to write code which is algorithmically complex and embedded into different teams. Our engineers need to be able to understand code from other teams, see where to insert self-contained logic. Engineers build components that scientists can use to introduce algorithmic logic. Therefore we focus our coding interviews on math and ML coding questions. Some examples include: Distributed implementation of sparse matrix or vector products, implementing specific ML models, and information retrieval.

**Below expectations**. Candidates write reasonably good code but need significant help in understanding algorithms.

**Meet expectations**. Candidates write good code and once walked through the algorithm can understand what is needed. They are able to think through possible edge cases and highlight optimizations.

**Exceed expectations**. Candidates may have seen multiple algorithms before and are quickly able to see hotspots and discuss strategies around them.

### Coding Standards

Our engineers need to be able to understand code from other teams, see where to insert self-contained logic. They think carefully about how code is maintained and how it is structured. Engineers build components that scientists can use to introduce algorithmic logic. Thus, they are maintaining code that they do not write themselves. We focus our coding standards interviews on thinking through libraries, package design, and testing.

**Below expectations**. Candidates write reasonably good code but need significant help in understanding algorithms.

**Meet expectations**. Candidates write good code and once walked through the algorithm can understand what is needed. They are able to think through possible edge cases and highlight optimizations.

**Exceed expectations**. Candidates may have seen multiple algorithms before and are quickly able to see hotspots and discuss strategies around them.

### Design

Our engineers need to make precise designs that are both robust yet flexible. This is hard to do and we look for engineers who can think through design problems and make appropriate tradeoffs. We focus our design interviews on realistic search use cases, such as designing an experimentation framework, making a component experimentable, dealing with ML model versioning.

**Below expectations**. Candidates focus on a design and have a hard time changing it for future evolution.

**Meet expectations**. Candidates build a reasonable solution but may need hints to think through edge cases.

**Exceed expectations**. Candidates can think through a design, highlight problem areas, and can articulate trade offs while making a recommendation.

### Big data coding

We expect our Software Engineers to be able to build data infrastructure to be used by scientists and to maintain data pipelines in production. Production data pipelines run reliably on a frequent basis are typically written by our researchers as the prerequisite for running experiments. Engineers build components that scientists can use to access data, validation components for testing, and data pipeline systems so that code once built by scientists is jointly owned and maintained by the team. The coding interview should focus on fundamentals of data processing, joins, grouping, aggregation, lookup tables, etc. We may not expect syntactically correct code, but they should be able to parse a dataset and solve data problems. Some examples include: Distributed implementation of sparse matrix or vector products, aggregations by key, joins, group statistics, skewed joins, and structuring data pipelines. These questions will likely take the form of pseudo code with core map-reduce concepts, SQL statements, or spark code snippets.

**Below expectations**. Candidate believes that data processing is someone else’s problem. Resort to single-process code, or cannot express basic concepts, struggles with parsing simple text formats or forming tuples.

**Meet expectations**. Candidate can write a basic script in SQL or map reduce, but they struggle with optimizations or analyzing performance. They may not recognize when skewed joins happen or use primitives that violate distributed processing rules, such as assuming all data is available equally on all machines without storing it.

**Exceed expectations**. Candidate knows the solution and can suggest improvements directly. Code is clean and well structured. If simplifying assumptions are made, they are called out and can be deep-dived if needed.

### Coding Standards

We expect our Software Engineers to be able to build data infrastructure to be used by scientists and to maintain data pipelines in production. Production data pipelines run reliably on a frequent basis are typically written by our researchers as the prerequisite for running experiments. Engineers build components that scientists can use to access data, validation components for testing, and data pipeline systems so that code once built by scientists is jointly owned and maintained by the team. The coding interview should focus on fundamentals of data processing, joins, grouping, aggregation, lookup tables, etc. We may not expect syntactically correct code, but they should be able to parse a dataset and solve data problems. Some examples include: Distributed implementation of sparse matrix or vector products, aggregations by key, joins, group statistics, skewed joins, and structuring data pipelines. These questions will likely take the form of pseudo code with core map-reduce concepts, SQL statements, or spark code snippets.

**Below expectations**. Candidate believes that data processing is someone else’s problem. Resort to single-process code, or cannot express basic concepts, struggles with parsing simple text formats or forming tuples.

**Meet expectations**. Candidate can write a basic script in SQL or map reduce, but they struggle with optimizations or analyzing performance. They may not recognize when skewed joins happen or use primitives that violate distributed processing rules, such as assuming all data is available equally on all machines without storing it.

**Exceed expectations**. Candidate knows the solution and can suggest improvements directly. Code is clean and well structured. If simplifying assumptions are made, they are called out and can be deep-dived if needed.

## Writing interview feedback

When writing interview feedback, it would be helpful to give the committee some insights into what was asked of the candidate and what specific aspects of the interview went well and did not go well. Here are some guidelines.

### Select a rating

Use the standard that we should hire candidates the improve the overall average of the team. By team we refer to the logical team of all people at this level and role. Using the following guidelines to select the best rating.

1. **Strong no hire**. Candidate is far outside the average and we should not consider this candidate ever again for this role (rare).
2. **No hire**. Candidate lowers the average of the team by performing below expectations on one or more competencies--beyond standard mentoring.
3. **Maybe not**. Candidate does not raise the average of the team and did not perform above expectations on any particular competency. The candidate could be considered for a lower level.
4. **Hire**. Candidate raises the average of the team on all competencies and did not perform below expectations on any.
5. **Strong Hire**. Candidate raises the average of the team significantly, on multiple competencies, and may be considered for a higher level.

### Overall Summary

Provide an overall summary of in support of the evaluation. Summarize the positive and negative points. Why do you believe the candidate is right for the role. Is the candidate at the right level as stated.

### Detailed feedback

For each competency area, describe the question that was asked and provide details of the candidate’s solution. Write the feedback in a sort of Question-Answer format. For example, consider following feedback for ML Depth:

*“I asked the candidate to design a search ranking model. They described a basic inverted index correctly. Next I asked to formulate the ranking with only text match features. They were able to write the formula for BM25 and TF IDF, but did not understand why TF made sense only that this was a common ranking function. I asked how the candidate would apply machine learning to this. They described using a click-based logistic regression and were able to derive the maximum likelihood formulation. They correctly describe the gradient update rule, but I had to provide several hints. I asked why regularization is needed and although they could describe L1 and L2, could not write the formula or explain why they accomplish regularization--or why it is necessary”*

### Notes

It is helpful when writing feedback to just write a point-by-point Questions-Answer in a set of notes appended to the feedback. That way the interviewer can recall this during the debrief and everyone else can see these notes.

## FAQs about the process

### Why are we comparing candidates to existing members?

We are not really comparing candidates to actual people, just to a hypothetical average person. This helps us drive increasing standards, so as we continue to higher, the average quality of our members improves.

### Are these research engineers or regular engineers?

We are NOT hiring for the common role of research engineer or machine learning engineer. The dominant view in the industry for these roles is that candidates will do some engineering and some science or analysis, but neither particularly well or they focus on implementing highly effect model inference engines only. This isn’t what we want. We want highly competent engineers who can design a very flexible architecture so that scientists can write their own code safely.

# Appendix

## Further Reading

[[1](https://careers.ebayinc.com/how-we-hire/%E2%80%A9)] Official guidelines: https://careers.ebayinc.com/how-we-hire/

[[2](https://hub.corp.ebay.com/site/people/page/manage-interview-question-library/us/manager)] Behavioral interview questions:

## 

## Leveling Guide

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MTS 1, Software Engineer / 25** | **MTS 2, Software Engineer / 26** | **Sr. MTS, Software Engineer / 27** |
| Job Scope | ● Owns and is accountable for the design and development of a product feature or sub-system (example: owns search autosuggest, or a substantial part of the classification autotagger, or owns Button Factory, or a substantial part of Account Setup).  ● Spends most of the time developing code, and efficiently produces elegant, clean code with no unnecessary complication or abstraction.  ● Code is well-commented, easy to maintain, and can be reused across a sub-system or feature. Code may persist for the lifetime of a software version  ● Code is thoroughly tested with very few bugs, and is supported by unit tests.  ● Regularly leads feature or sub-system design reviews and code reviews and is fully recognized as the go-to developer for that component.  ● Participates in architecture discussions, regularly proposes and discusses solutions to system and product changes that are directly related to their area of focus. | ● Owns and is accountable for the design and development of a product or major product sub-system (example: owns the Shopping Cart or search backend aggregator, or owns Invoicing or Account Management services).  ● Spends most of the time developing code, and efficiently produces exemplar and thoroughly tested code for the most challenging features. Code typically persists for the lifetime of a software version  ● Produces clear, well-communicated, complete designs for product features or sub-systems.  ● Recognized as the go-to developer for a product or major sub-system and is seen as a leader in their specialized field.  ● Leads product design and code reviews; can competently review any aspect of their product or major sub-system.  ● Drives architecture discussions, and proposes solutions to system and product changes. | ● Owns and is accountable for the design and development of a product or major product sub-system (example: owns the Shopping Cart or the search backend aggregator, or owns Invoicing or Account Management services).  ● Spends most of the time developing code, and efficiently produces exemplar and thoroughly tested code for the most challenging features. Code typically persists for the lifetime of a software version  ● Produces clear, well-communicated, complete designs for product features or sub-systems.  ● Recognized as the go-to developer for that product or major sub-system and is seen as a leader in their specialized field.  ● Leads product design and code reviews; can competently review any aspect of their product or major sub-system.  ● Drives architecture discussions, and proposes solutions to system and product changes. |
| Interaction | ● Frequently interfaces with other functional teams on all aspects of a feature or sub-system.  ● Influences senior management on product direction, particularly related to their area of focus and expertise.  ● Frequently visible across major engineering organizations, or through external forums such as conferences, workshops, or open source initiatives.  ● Provides leadership to others, particularly junior engineers who work on the same team or related features or product sub-systems. | ● Frequently interfaces with other functional teams' leadership  ● Influences Vice Presidents on product direction.  ● Highly visible across major engineering organizations, and regularly visible through external forums such as conferences, workshops, or open source initiatives.  ● Provides leadership to others, particularly junior engineers who work on the same team or related features or product sub-systems. | ● Key interface with other functional teams' leadership.  ● Influences Vice Presidents on product direction  ● Highly visible across major engineering organizations, and regularly visible through external forums such as conferences, workshops, or open source initiatives.  ● Provides leadership to both junior and senior engineers. |
| Architecture and Design | ● Designs may require refinement by more senior engineers, but are generally correct and consider the broad implications of the design upon larger systems. | ● Designs are typically exemplars.  ● Complex architectural designs typically require refinement and iteration with more senior engineers. | ● Designs are typically exemplars.  ● Complex architectural designs typically require only minor refinement. |
| Skills | ● Can be relied on to deliver features and sub-systems on time and to requirements, without quality issues.  ● Generally correctly estimates software schedules.  ● Basic mastery of feature or subsystem reusability, modularity, or scaling.  ● Works well within a team, and contributes effectively to the success of those that they interact with regularly.  ● Understands and is able to reason about the business, as it relates to their area of expertise.  ● Can triage and resolve site issues with supervision.  ● Able to evangelize innovations, through prototyping or other means. | ● Can be relied on to deliver products on time and to requirements.  ● Correctly estimates software schedules, and delivers on time without quality issues.  ● Works well within a team, and contributes effectively to the success of the team and related teams.  ● Understands the business and is able to contribute to technology direction that contributes to measurable business improvements.  ● Can triage and resolve site issues without supervision.  ● Able to evangelize innovations, through prototyping or other means.  ● Identifies opportunities for engineering productivity improvements or directions, and evangelizes these successfully. | ● Can be relied on to deliver products on time and to requirements.  ● Correctly estimates software schedules, and delivers on time without quality issues.  ● Works well within a team, and contributes effectively to the success of the team and related teams.  ● In depth understanding of the business and regularly contributes to technology direction that delivers measurable business improvements.  ● Identifies opportunities for engineering productivity improvements or directions, and evangelizes these successfully. |
| Supervision/  Management  (Direction) | ● Receives direction in terms of the desired outcome or strategic objectives.  ● Can be given new feature or subsystem assignments with minimal supervision; has established working relationships that enable self-direction in gathering requirements and scoping work.  ● Frequently coordinates activities of junior engineers assigned to the same feature or subsystem. | ● Provides leadership within their team, and is frequently consulted by directors throughout their vice-president’s organization.  ● Can be given new product assignments with minimal supervision.  ● May be relied on by an engineering director to provide technical direction for an engineering team of 5 to 10 engineers. | ● Provides leadership within their team, and is frequently consulted by directors throughout their vice-president’s organization.  ● Can be given new product assignments with minimal supervision.  ● May be relied on by an engineering director to provide technical direction for an engineering team of 5 to 10 engineers. |

## Sample Questions

**ML Implementation**. Explain a linear model. Ask how would you implement the sparse inner product calculation? Then, discuss how should server obtain the model, hard coded, pulling from a file, feature server? How to handle version control? How to handle A/B tests.

**ML Implementation**. Explain a single decision tree. Ask how would you implement the scoring logic for a single decision tree. Could you automatically generate compiled code? How to handle model updates? How would you test the model?

Coding questions related a component of ML or Science or Math

* Entropy of partitioned data
* Assigning clusters from means
* Reservoir Sampling
* Sparse inner product
* Matrix multiplication
* Evaluate a provided decision tree
* Find the longest substring matching a dictionary (how would you store the dictionary)
* Given a list of items with two numeric fields, write an algorithm to count the number of inversions in order between the two orderings

System Design:

* LRU Cache
* Model deployment
* Real-time Feedback loop
* Design a key-value store
* Inverted index
* Top-frequent query
* Autocomplete
* Design an ecommerce: shopping cart

## 

## New Hire Tool Feedback Template

Enter the following information in the New Hire Tool [feedback form](https://hire.corp.ebay.com/admin/caterpie/scheduled/feedback/).

*Overall: [2: Strong No Hire, 4: No Hire, 6: Maybe Not, 8: Hire, 10: Strong Hire]*

*[1-2 paragraphs summarizing your feedback and justifying your recommendation]*

*<Competency 1>: [Below Expectation, Meets Expectations, Exceeds Expectations]*

*[2-3 paragraphs explaining the question you asked, summary of the answer, and justification for the rating]*

*<Competency 2 if present>: [Below Expectation, Meets Expectations, Exceeds Expectations]*

*[2-3 paragraphs explaining the question you asked, summary of the answer, and justification for the rating]*

*Q&A / Notes:*

*Any detailed notes you may be taking during the interview. Mostly for you to review*

For phone screens, use the score in the rating above.

For onsites, recommend Hire for scores greater than or equal to 8.

## Interview Assignments

The assignment protocol for interviews is as follows. Try to check for time off when assigning.

**Phone Screens 1**. Select one person randomly from the pool of interviewers at the same level. Covers ML Breadth.

**Phone Screen 2**. Select one person randomly from the pool of interviewers at the same level or above. Covers ML Depth and Coding.

**Onsite**. Select three persons randomly from the pool of interviewers at the same level or above who were not in previous phone screens. Select the HM and one other manager or tech lead at least 1 level above.

## Interview Description

Request an interview and assign a person and competency in the Hiring Tool. This triggers recruiters to begin scheduling.

Role: Software engineer

Level: <Level>

Please find your assigned competency here:

<https://searchhire.corp.ebay.com/admin/caterpie/scheduled/feedback/>

Please follow the template for feedback and guidelines in the doc here:

<https://docs.google.com/document/d/10d-figforoU_gqZkoxMBQQsrM_JnIAeM_T7WrHgbQ7k/edit#bookmark=kix.ga0fv98omtkg>