Design AlgoExpert

Many systems design questions are intentionally left very vague and are literally given in the form of Design Foobar. It's your job to ask clarifying questions to better understand the system that you have to build.

We've laid out some of these questions below; their answers should give you some guidance on the direction of the problem. Before looking at them, we encourage you to take few minutes to think about what questions you'd ask in a real interview.

Clarifying Questions To Ask

Question 1

Q: Are we designing the entire AlgoExpert platform or just a specific part of it, like the coding workspace?

A: Since we only have about 45 minutes, you should just design the core user flow of the AlgoExpert platform. The core user flow includes users landing on the home page of the website, going to the questions list, marking questions as complete or in progress, and then writing and running code in various languages for each language. Don't worry about payments or authentication; you can just assume that you have these services working/registered (by the way, we mainly rely on third-party services here, like Stripe, PayPal, and OAuth2).

Question 2

Q: AlgoExpert doesn't seem like a system of utmost criticality (like a hospital system or airplane software); are we okay with 2 to 3 nines of availability for the system?

A: Yes, this seems fine--no need to focus too much on making the system highly available.

Question 3

Q: How many customers should we be building this for? Is AlgoExpert's audience global or limited to one country?

A: AlgoExpert's website receives hundreds of thousands of users every month, and tens of thousands of users may be on the website at any point in time. We want the website to feel very responsive to people everywhere in the world, and the U.S. and India are the platform's top 2 markets that we especially want to cater to.

Question 4

Q: Does AlgoExpert make changes to its content (questions list and question solutions) often?

A: Yes, every couple of days on average. And we like to have our changes reflected in production globally within the hour.

Question 5

Q: How much of the code-execution engine behind the coding workspace should we be designing? Do we have to worry about the security aspect of running random user code on our servers?

A: You can disregard the security aspects of the code-execution engine and just focus on its core functionality--the ability to run code in various languages at any given time with acceptable latency.

Question 6

Q: While we'll care about latency across the entire system, the code-execution engine seems like the place where we'll care about it most, since it's very interactive, and it also seems like the toughest part of our system to support low latencies; are we okay with anywhere between 1 and 3 seconds for the average run-code latency?

A: Yes--this seems reasonable and acceptable from a product point of view.

Design A Code-Deployment System

Design a global and fast code-deployment system.

Many systems design questions are intentionally left very vague and are literally given in the form of Design Foobar. It's your job to ask clarifying questions to better understand the system that you have to build.

We've laid out some of these questions below; their answers should give you some guidance on the problem. Before looking at them, we encourage you to take few minutes to think about what questions you'd ask in a real interview.

Clarifying Questions To Ask

Question 1

Q: What exactly do we mean by a code-deployment system? Are we talking about building, testing, and shipping code?

A: We want to design a system that takes code, builds it into a binary (an opaque blob of data--the compiled code), and deploys the result globally in an efficient and scalable way. We don't need to worry about testing code; let's assume that's already covered.

Question 2

Q: What part of the software-development lifecycle, so to speak, are we designing this for? Is this process of building and deploying code happening when code is being submitted for code review, when code is being merged into a codebase, or when code is being shipped?

A: Once code is merged to the trunk or master branch of a central code repository, engineers should be able to (through a UI, which we're not designing) trigger a build and deploy that build. At that point, the code has already been reviewed and is ready to ship. So to clarify, we're not designing the system that handles code being submitted for review or being merged into a master branch--just the system that takes merged code, builds it, and deploys it.

Question 3

Q: Are we essentially trying to ship code to production by sending it to, presumably, all of our application servers around the world?

A: Yes, exactly.

Question 4

Q: How many machines are we deploying to? Are they located all over the world?

A: We want this system to scale massively to hundreds of thousands of machines spread across 5-10 regions throughout the world.

Question 5

Q: This sounds like an internal system. Is there any sense of urgency in deploying this code? Can we afford failures in the deployment process? How fast do we want a single deployment to take?

A: This is an internal system, but we'll want to have decent availability, because many outages are resolved by rolling forward or rolling back buggy code, so this part of the infrastructure may be necessary to avoid certain terrible situations. In terms of failure tolerance, any build should eventually reach a SUCCESS or FAILURE state. Once a binary has been successfully built, it should be shippable anywhere within roughly 30 minutes.

Question 6

Q: So it sounds like we want our system to be available, but not necessarily highly available; we want a clear end-state for builds, and we want the entire process of building and deploying code to take roughly 30 minutes. Is that correct?

A: Yes, that's correct.

Question 7

Q: How often will we be building and deploying code, how long does it take to build code, and how big can the binaries that we'll be deploying get?

A: Engineering teams deploy hundreds of services or web applications, thousands of times per day, building code can take up to 15 minutes, and the final binaries can reach sizes of up to 10 GB. The fact that we might be dealing with hundreds of different applications shouldn't matter though; you're just designing the build pipeline and deployment system, which are agnostic to the types of applications that are getting deployed.

Question 8

Q: When building code, how do we have access to the actual code? Is there some sort of reference that we can use to grab code to build?

A: Yes, you can assume that you'll be building code from commits that have been merged into a master branch. These commits have SHA identifiers (effectively arbitrary strings) that you can use to download the code that needs to be built.

Design A Stockbroker

Design a stockbroker: a platform that acts as the intermediary between end customers and some central stock exchange.

Many systems design questions are intentionally left very vague and are literally given in the form of Design Foobar. It's your job to ask clarifying questions to better understand the system that you have to build.

We've laid out some of these questions below; their answers should give you some guidance on the problem. Before looking at them, we encourage you to take few minutes to think about what questions you'd ask in a real interview.

Clarifying Questions To Ask

Question 1

Q: What do we mean exactly by a stock broker? Is this something like Robinhood or Etrade?

A: Yes, exactly.

Question 2

Q: What is the platform supposed to support exactly? Are we just supporting the ability for customers to buy and sell stocks, or are we supporting more? For instance, are we allowing other types of securities, like options and futures to be traded on our platform? Are we supporting special types of orders like limit orders and stop losses?

A: We're only supporting the ability to buy and sell stocks. In simpler terms, a customer should be able to place an order to buy or sell a stock, we should try to execute the order as soon as possible regardless of the stock price. We also aren't designing any "margin" system, so the available balance is the source of truth for what can be bought.

Question 3

Q: Are we designing any of the auxiliary aspects of the stock brokerage, like depositing and withdrawing funds, downloading tax documents, etc.?

A: No -- we're just designing the core trading aspect of the platform.

Question 4

Q: Are we just designing the system to place trades? Do we want to support other trade-related operations like getting trade statuses? In other words, are we supporting the full CRUD operations on trades (i.e., supporting operations like getting the status of a trade, and potentially getting trades back in bulk)?

A: Great question. We'll also want to support operations like getting the status of a trade and getting trades back in bulk. For simplicity, let's support only two operations: placing a trade, and getting a trade status.

Question 5

Q: When a customer places a trade, do we assume that the customers have already deposited funds into the platform, and can just assume that we have a SQL table with the balance for each customer who wants to make a trade.

A: Yes, you can assume that this is the case.

Question 6

Q: How many customers are we building this for? Is our customer-base a global one?

A: Millions of customers, millions of trades a day. Let's assume that our customers are only located in the U.S.

Question 7

Q: What kind of availability are we looking for?

A: As high as possible, with this kind of service people can lose a lot of money if the system is down even for a few minutes.

Question 8

Q: Are we designing the UI for this platform? What kinds of clients can we assume we have to support?

A: You don't have to design the UI, but you should design the PlaceTrade API call that a UI would be making to your backend. Clients would be either a mobile app or a web app.

Question 9

Q: So we want to design the API for the actual brokerage, that itself interacts with some central stock exchange on behalf of customers. Does this exchange have an API? If so, do we know what it looks like?

A: Yes, the exchange has an API, and your platform's API (the PlaceTrade call) will have to interact with the exchange's API. As far as that exchange API goes, you can assume that the call to the exchange is made that takes completes at the exchange level (meaning, once the trade passes through the exchange, once the trade is placed successfully at the exchange level, we may have a SQL table with the balance for each customer who wants to make a trade.

Design Facebook News Feed

Many systems design questions are intentionally left very vague and are literally given in the form of Design Foobar. It's your job to ask clarifying questions to better understand the system that you have to build.

We've laid out some of these questions below; their answers should give you some guidance on the direction of the problem. Before looking at them, we encourage you to take few minutes to think about what questions you'd ask in a real interview.

Clarifying Questions To Ask

Question 1

Q: Facebook News Feed consists of multiple major features, like loading a user's news feed, interacting with it (i.e., posting status updates, liking posts, etc.), and updating it in real time (i.e., new status updates that are being posted to the top of the feed, in real time). What part of Facebook News Feed are we designing exactly?

A: We're designing the core functionality of the feed itself, which we'll define as follows: loading a user's news feed and updating it in real time, as well as posting status updates. But for posting status updates, we don't need to worry about the actual API or the type of information that a user can post; we just want to design what happens once an API call to post a status update has been made. Ultimately, we're designing the retrieval of a feed as well as the posting of a status update to a feed (only in terms of how this impacts the feed itself--again, not the actual API to post the status update itself).

Question 2

Q: To clarify, posts on Facebook can be pretty complicated, with pictures, videos, special types of status updates, etc... Are you saying that we're not concerned with the aspect of the system? For example, should we not focus on how we'll be storing this type of information?

A: That's correct. For the purpose of this question, we can treat posts as opaque entities that we'll certainly want to store, but without worrying about the details of the storage, the ramifications of storing and serving large files like videos, etc.

Question 3

Q: Are we designing the feed with regards to serving algorithms, you can assume that you have access to a ranking algorithm that you can simply feed a list of relevant posts to in order to generate an actual news feed to display.

Question 4

Q: Are we concerned with showing ads in a user's news feed at all? Ads seem like they would behave a little bit differently than posts, since they probably rely on a different ranking algorithm.

A: You can treat ads as a bonus part of the design. If you want to say to incorporate them in, great (and yes, you'd have some other ads-serving algorithm to determine what ads need to be shown to a user at any point in time). But don't focus on ads to start.

Question 5

Q: Are we serving a global audience, and how big is our audience?

A: Yes -- we're serving a global audience, and let's say that the news feed will be loaded in the order of 100 million times a day, by 100 million different users, with 1 million new status updates posted every day.

Question 6

Q: How many friends does a user have on average? This is important to know, since a user's status updates could theoretically have to show up on all of the user's friends' news feeds at once.

A: You can expect each user to have, on average, 500 friends on the social network. You can treat the number of friends per user as a bell-shaped distribution, with some users who have very few friends, and some users who have a lot more than 500 friends.

Question 7

Q: How quickly does a status update have to appear on a news feed once it's posted, and is it okay if this varies depending on user locations with respect to the location of the user submitting a post?

A: When a user posts something, you probably want it to show up on other news feeds fairly quickly. This speed can indeed vary depending on user locations. For instance, we'd probably want a local friend within the same region to see the new post within a few seconds, but we'd likely be okay with a user on the other side of the world seeing the same post within a minute.

Question 8

Q: What kind of availability are we aiming for?

A: Your design shouldn't be completely unavailable from a single machine failure, but this isn't a high availability requirement. However, posts shouldn't ever just disappear. Once the user's client gets confirmation that the post was created, you cannot lose it.

Design Google Drive

Many systems design questions are intentionally left very vague and are literally given in the form of Design Foobar. It's your job to ask clarifying questions to better understand the system that you have to build.

We've laid out some of these questions below; their answers should give you some guidance on the direction of the problem. Before looking at them, we encourage you to take few minutes to think about what questions you'd ask in a real interview.

Clarifying Questions To Ask

Question 1

Q: Are we just designing the storage aspect of Google Drive, or are we also designing some of the related products like Google Docs, Sheets, Slides, Drawings, etc.?

A: We're just designing the core Google Drive product, which is indeed the storage product. In other words, users can create folders and upload files, which effectively stores them in the cloud. Also, for simplicity, you can forget about these related products--Google Docs, Sheets, Slides, etc.

Question 2

Q: There are a lot of features on Google Drive, like shared company drive vs. personal drives, permissions on entities (ACLs), starred files, recently accessed files, etc... Are we designing all of these features or just some of them?

A: Let's keep things narrow and imagine that we're designing a personal Google Drive (so you can forget about shared company drives). In a personal Google Drive, users can store entities, and that's all that you should take care of. Ignore any feature that isn't core to the storage aspect of Google Drive; ignore things like starred files, recently-accessed files, etc... You can even ignore sharing entities for this design.

Question 3

Q: Since we're primarily concerned with storing entities, are we supporting all basic CRUD operations (create, reading, updating, and moving entities)?

A: Yes, but to clarify, creating a file is actually uploading a file, folders have to be created (they can't be uploaded), and we also want to support downloading files.

Question 4

Q: And we just designing the Google Drive web application, or are we also designing a desktop client for Google Drive?

A: We're just designing the web application.

Question 5

Q: Since we're not dealing with sharing entities, should we handle multiple users in a single folder at the same time, or can we assume that this will never happen?

A: While we're not designing the sharing feature, let's still handle what would happen if multiple clients were in a single folder at the same time (two tabs from the same browser, for example). In this case, we would want changes made in that folder to be reflected to all clients within 10 seconds. But for the purpose of this question, let's not worry about conflicts or anything like that (i.e., assume that two clients won't make changes to the same file or folder at the same time).

Question 6

Q: How many people are we building this system for?

A: This system should serve about 1 billion users, with roughly 15GB per user on average.

Question 7

Q: How important is data consistency and availability for this system? Is it okay if a customer doesn't find or see their data for some time, or if the data isn't stored with full integrity?

A: First and foremost, data loss isn't tolerated at all; we need to make sure that once a file is uploaded or a folder is created, it won't disappear until the user deletes it. As for availability, we need the system to be very highly available.

Design The Reddit API

Design an API for Reddit subreddits given the following information.

The API includes these 2 entities:

User (userId: string,
...)
Subreddit (subredditId: string,
...)

Your API should support the basic functionality of a subreddit on Reddit.

Many systems design questions are intentionally left very vague and are literally given in the form of Design Foobar. It's your job to ask clarifying questions to better understand the system that you have to build.

We've laid out some of these questions below; their answers should give you some guidance on the direction of the problem. Before looking at them, we encourage you to take few minutes to think about what questions you'd ask in a real interview.

Clarifying Questions To Ask

Question 1

Q: To make sure that we're on the same page: a subreddit is an online community where users can write posts, comment on posts, upvote / downvote posts, share posts, report posts, become moderators, etc... via this core set and a subreddit. Are we designing all of this functionality?

A: Yes, that's correct, but let's keep things simple and focus only on writing posts, writing comments, and upvoting / downvoting. You can forget about all of the other auxiliary features like reporting, sharing, moderators, etc.

Question 2

Q: So we're really focusing on the very narrow but core aspect of a subreddit: writing posts, commenting on them, and voting on them.

A: Yes.

Question 3

Q: I'm thinking of defining the schemas for the main entities that live within a subreddit and then defining their CRUD operations -- methods like CreatePost/EditDelete/List -- is this in line with what you're asking me to do?

A: Yes, and make sure to include method signatures -- what each method takes in and what each method returns. Also include the types of each argument.

Question 4

Q: The entities that I've identified are Posts, Comments, and Votes (upvotes and downvotes). Does this seem accurate?

A: Yes. These are the 3 core entities that you should be defining and whose APIs you're designing.

Question 5

Q: Is there any other functionality of a subreddit that we should design?

A: Yes, once you've designed a subreddit's core API, we want to make sure that you know how to correctly paginate and how to handle the feed aspect of a subreddit (i.e., how users might actually see posts and comments (one award per post / comment).