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

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: 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 workingalready (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 globallywithin 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 . It's your job to ask clarifying questions to better understand the system that you have to build.`Design Foobar`

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 reposity, 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 to all machines globally within 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 . It's your job to ask clarifying questions to better understand the system that you have to build.`Design Foobar`

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 market orders on stocks in this design. A market order means that, given a placed 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, how comprehensive should the API that's going to support this platform be?

A: In essence, you're only designing a system around a PlaceTrade API call from the user, but you should define that API call (inputs, response, etc.).

Question 5

Q: Where does a customer's balance live? Is the platform pulling a customer's money directly from their bank account, or are we expecting that customers will have already deposited funds into the platform somehow? In other words, are we ever directly interacting with banks?

A: No, you won't be interacting with banks. You can assume that customers have already deposited funds into the platform, and you can further assume that you have a SQL table with the balance for each customer who wants to make a trade.

Question 6

Q: How many customers are we building this for? And 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 1 region – the U.S., for instance.

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 also 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 webapp.

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 yes, do we know what it looks like, and do we have any guarantees about it?

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's concerned, you can assume that the call to the exchange to make an actual trade will take in a callback (in addition to the info about the trade) that will get executed when that trade completes at the exchange level (meaning, when the trade either gets FILLED or REJECTED, this callback will be executed). You can also assume that the exchange's system is highly available–your callback will always get executed at least once.

Design Facebook News Feed

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

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: 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., adding 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 primarily want to design the feed generation/refreshing piece of the data pipeline (i.e, how/when does it get constructed, and how/when does it get updated with new posts).

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 this 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 relevant-post curation system (i.e., the system that decides what posts will show up on a user's news feed)?

A: No. We're not designing this system or any ranking 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 find a way 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 . It's your job to ask clarifying questions to better understand the system that you have to build.`Design Foobar`

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: 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, we can refer to folders and files as "entities".

Question 2

Q: There are a lot of features on Google Drive, like shared company drives 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 like creating, deleting, renaming, 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: Are 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 functionality of the Google Drive 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 a billion users and handle 15GB per user on average.

Question 7

Q: What kind of reliability or guarantees does this Google Drive service give to its users?

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 this system to be 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, ...

Both of these entities likely have other fields, but for the purpose of this question, those other fields aren't needed.

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 . It's your job to ask clarifying questions to better understand the system that you have to build.`Design Foobar`

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: 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...–is this correct, and 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 auxiliary features like sharing, reporting, moderating, 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 Create/Get/Edit/Delete/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, you should also allow people to award posts. Awards are a special currency that can be bought for real money and gifted to comments and posts. Users can buy some quantity of awards in exchange for real money, and they can give awards to posts and comments (one award per post / comment).

Design Netflix

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.

Q: From a high-level point of view, Netflix is a fairly straightforward service: users go on the platform, they're served movies and shows, and they watch them. Are we designing this high-level system entirely, or would you like me to focus on a particular subsystem, like the Netflix home page?

A: We're just designing the core Netflix product–so the overarching system / product that you described.

Question 2

Q: Should we worry about auxiliary services like authentication and payments?

A: You can ignore those auxiliary services; focus on the primary user flow. That being said, one thing to note is that, by nature of the product, we're going to have access to a lot of user-activity data that's going to need to be processed in order to enable Netflix's recommendation system. You'll need to come up with a way to aggregate and process user-activity data on the website.

Question 3

Q: For this recommendation system, should I think about the kind of algorithm that'll fuel it?

A: No, you don't need to worry about implementing any algorithm or formula for the recommendation engine. You just need to think about how user-activity data will be gathered and processed. It sounds like there are 2 main points of focus in this system: the video-serving service and the recommendation engine. Regarding the video-serving service, I'm assuming that we're looking for high availability and fast latencies globally; is this correct? Yes, but just to clarify, the video-streaming service is actually the only part of the system for which we care about fast latencies.

Question 4

Q: So is the recommendation engine a system that consumes the user-activity data you mentioned and operates asynchronously in the background?

A: Yes.

Question 5

Q: How many users do we expect to be building this for?

A: Netflix has about 100M to 200M users, so let's go with 200M.

Question 6

Q: Should we worry about designing this for various clients, like desktop clients, mobile clients, etc.?

A: Even though we're indeed designing Netflix to be used by all sorts of clients, let's focus purely on the distributed-system component–so no need to get into details about clients or to optimize for certain clients.

Design The Uber API

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: Uber has a lot of different services: there's the core ride-hailing Uber service, there's UberEats, there's UberPool–are we designing the API for all of these services, or just for one of them?

A: Let's just design the core rides API – not UberEats or UberPool.

Question 2

Q: At first thought, it seems like we're going to need both a passenger-facing API and a driver-facing API–does that make sense, and if yes, should we design both?

A: Yes, that totally makes sense. And yes, let's design both, starting with the passenger-facing API.

Question 3

Q: To make sure we're on the same page, this is the functionality that I'm envisioning this API will support: A user (a passenger) goes on their phone and hails a ride; they get matched with a driver; then they can track their ride as it's in progress, until they reach their destination, at which point the ride is complete. Throughout this process, there are a few more features to support, like being able to track where the passenger's driver is before the passenger gets picked up, maybe being able to cancel rides, etc... Does this capture most of what you had in mind?

A: Yes, this is precisely what I had in mind. And you can work out the details as you start designing the API.

Question 4

Q: Do we need to handle things like creating an Uber account, setting up payment preferences, contacting Uber, etc...? What about things like rating a driver, tipping a driver, etc.?

A: For now, let's skip those and really focus on the core taxiing service.

Question 5

Q: Just to confirm, you want me to write out function signatures for various API endpoints, including parameters, their types, return values, etc., right?

A: Yup, exactly.