When entering into a relationship with a client, there are a number of questions to be answered in order to ensure clarity on the scope of and direction of the project. Many clients are idea-smart, but not tech-smart, so are unaware of the amount of potential effort it will take to generate quality code, test it and release it. Making "quick" changes can lead down rabbit holes that can be lengthy, and of course expensive, especially if charged at an hourly rate.

With that in mind, then the goal of these questions becomes three fold:

1) minimize the potential costs for the client,
2) improve the accuracy of any time/cost estimates at the beginning of the project and
3) delivering not just high quality results, but delivering results that are what the client needs, which may not naturally match what was proposed when the client thought to ask about upon initial consultation.

The Questions/Considerations

A lot of these questions are exploratory. If they can't be clearly answered, then the consultant will have to do a lot more research once a code review begins.

IN NO PARTICULAR ORDER (and sometimes vaguely worded). "We" is the consultant and "you" is the the client:

1. Do you have someone(s) who will be doing detailed testing, or will you be relying on the consultants? [This adds time, complexity, and cost.]
2. Did your previous coders do TDD? If not, are you willing to take on the cost of adding TDD? [It's likely that for a mature code base TDD can't be added in retroactively, so adding it would only be for new/young code bases]
3. Does existing code follow best practices and accepted idioms for the language? Is it well organized or follow a model such as MVC or MVVC? [Without some sort of structure, modularity or standardization, the code will be harder to work with.]
4. What is the full technology stack?
5. How many other coders do you have committed to the project and when was the last time they made a modification to the code base?
6. Quality results come for quality communication; during the project, we will provide as much detail as possible and explain what we can in terms you can understand (i.e. for the non-technical). Will you be able/willing to respond back in kind? [This is a leadership question; some tasks can't be decided upon by the consultant]
7. There may be times when a coder enters a "rabbit-hole." The coder will try to recognize this as soon as possible and get feedback from you so that you can determine the value of continuing on. We will explain the potential work to be done; will you be able to respond? [This is a question of scope; if the client has chosen tasks that aren't well defined, then there is a higher degree of rabbit-holing, especially if the code base is messy or simply unknown (like at the beginning of a project)]
8. For each expected task, will you be will to provide step-by-step details of what you expect to happen (as far as the UI is concerned)? You can black-box or hand-wave parts that you aren't

sure of or don't know how would work (such as with non-UI elements and backend items). The details should be more business oriented rather than technically oriented. [The key takeaways with this question is to 1) see the degree of scope (and possibly find relations with other tasks and 2) see how much UI/UX design will also need to be engaged]

9. Do you have UI/UX mockup or a UX designer of your own we can communicate with? Are they empowered to make decisions concerning the UI?
10. Of any listed items can you rank them by importance on a scale of 1 to 500? Start with large increments, say with a spread of 20 to 50 for obvious items, and then use smaller numbers if you find that later items would fit in between you earlier items. Not two items can share the same importance. [This is needed for two reasons: 1) it properly prioritizes the task list and 2) it may shine light on possible patterns of intent that show a different route to take; it's possible that the client might get more bang for their buck if they do something related to, but yet different from their list, but this might only be revealed in seeing what the client considers to be important first.]
11. What is your preferred mode of communication?
12. Does your existing code base exist in version control?
13. How easy can your existing code base be deployed or at least setup in a development environment?
14. Quality code comes with quality time. Coding cannot be done 24/7, so the speed of delivery is dependent on several factors (existing code quality, specificity of tasks, relationship between tasks, testing), so will you be okay accepting that many things that cannot be seen (especially on the server side) will take time?