Quality of a good easy project

1. ***SRS.***
2. Visual. Professional UI.
3. Solves, business problem.
4. With wow factor . Make it look Cool. Bootstrap, copy paste Bootstrap template.
5. interactivity
6. easy to understand
7. taught you something
8. 1) Solve a lot of coding problems and never stick to the easy and comfortable ones, slowly increase the difficulty (Leetcode, GeeksForGeeks, Codeforces) 2) Try to do a lot of projects with different technologies and frameworks and always "always" commit your code on GitHub it helps a lot! 3) Study System Design (even for SDE1). I was asked questions around System Design in almost every company I sat for. Best of Luck NDC.
9. NDC, ASP.NET,
10. Use Database.
11. CRUD.
12. Security. Authentication and Authorization. Alth0
13. Make programming related problem solver which may earn money.
14. Financial systems.
15. Task Scheduler / Management of web services and tasks.
16. Use Bug Tracker

Patience is virtue when learning to code. Everybody has to go through lots of trial and error when learning. Things that I personally recommend are: 1. Get a whiteboard. Whiteboards are nice because you can map out a general idea for your program and simply run your finger across to erase things that you want to change. Paper is fine too but I've always used whiteboards. 2. Learn pseudocode so you can map out your classes and algorithms on a whiteboard/paper and desk-check the logic. This can significantly chop off a lot of trial and error when you go to write it in the programming language of your choice. The book that I learned from is, "Simple Program Design: A Step-By-Step Approach 5th Edition", by Lesley Anne Robertson. You can learn before, during, or after learning the basics of coding. 3. Once you learn the general basics, learn about data structures, Big O, and algorithms for time complexity. The O'Reilly book, "Algorithms In A NutShell", is a good one but learning from videos might be easier to most. 4. Patience is important and so is taking breaks. Step away to decompress when you're stuck on a problem. Google helps a lot.

Biggest Self-Taught Programmer Mistakes

Summary: 1. Set a hard date as to *when* you'll start applying for jobs for your new career, then work backward to execute your "plan" 2. Have a clear game plan, you can start with a vague, "not perfect" game plan, but have a plan and iterate over it as you go 3. Track your progress against the goals you've set in your plan. Track Sleep/Study time(active vs Passive) / Exercise etc. Make adjustments as you go 4. Focus on task at hand (this is where your Plan comes in) Don't fret about things down the road. Knock down the problem at hand and the big picture will get clearer as you go.