How To Structure A Programming Project…

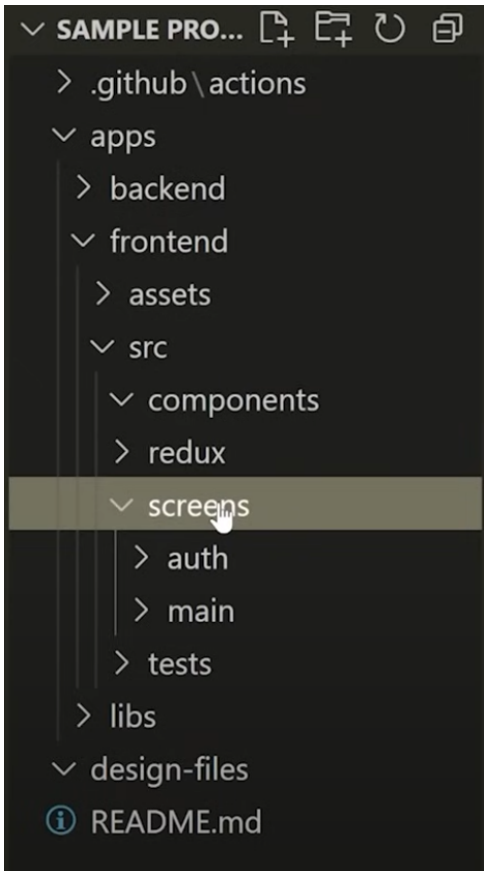
Video

https://youtu.be/CAeWjoP525M?si=B-nsBNjmUr2CTbu3

Notes

- Identify the Problem
  - The best projects have a real problem that has been solved with code
  - Having a project that solves a problem is good for talking about in an interview
- Have a Plan
  - At least come up with a basic plan
  - At minimum decide:
    - What tech stack am I going to use?
    - What database am I going to have?
    - What's my backend going to look like?
    - What's my frontend going to look like?
    - What are the general components that I'll need?
  - Create design documents
    - Include them in the repo
    - Good to talk about in an interview
  - Structure your directories

        - Some frameworks have their own conventions to stick with
        - Look at examples of larger applications to get a feel for conventions
        - Be consistent (even if you are not sure if it is right)
        - Try not to have too many files in the root



    - Shared code/resources should be put in libraries
- Use version control from the beginning
    - Get practice
    - Show you that you know how to separate changes into different commits
    - Give you a history of the project to go back through
    - Useful if you need to return to an old version of the system
- Modularize and componentise code

    - Be modular in your approach when you start writing the code for the project
    - It is better to have more files and more components than it is to have fewer (as it means that it is easier to read the code as well as reuse it in the future)
    - Short, easy-to-understand, reusable components
    - files should only contain functions that make sense to go together
        - If one doesn't fit in the file, take it out and put it in its own file
- Documentation
    - Have at least:
        - Installation guide
        - Troubleshooting guide
        - Contributing guideline (if open source)
        - Issue template
        - For large files or blocks of code, putting a README in that directory (for example, within the 'backend' directory) just to explain what is going on
- Testing
    - Shows you have a commitment to proper software development
    - Even just doing a few unit and integration tests can be good
- Dependency management
    - Use secure and well-maintained dependencies
- Continuous Integration & Continuous Deployment
    - If you want people to use your code/project, you want it to be as easy as possible to run
    - Deploy it in an environment
    - In a docker container, a website, a one-click install (into an installer)
- Code Review & Refactoring
    - After finishing a project, take a few days break, and then come back to it to look for ways to improve the code