Describing Technical Projects

Summary

For each project you complete, compile the following information:

- **Description**. One or two sentences summarizing your project. This should be more interesting than technical.
- **Motivation**. Two or three sentences describing the problem the project solves.
- **Results**. One or two sentences describing the end result—in other words,
- **Team Efforts**. Describe how you delegated work amongst your team members.
- **Individual Responsibilities**. Describe what *you* contributed to the project as an individual—in other words, the parts of the project that you were accountable for.
    -
        - **Challenges**. Describe any challenges you and/or your team encountered that you personally played an important role in resolving.
- **Improvements**. Describe changes you would make going forward. These can be changes you’d make for better scalability; additional features you’d add; improvements you’d make to the codebase; etc.

After compiling this information for each project, use it to rehearse your answer to the question: *“Tell me about a technical project you’ve worked on”.*

You should be able to cover the first three points in less than **45 seconds**, and summarize the last three in **approximately a minute**.

You should, however, be able to further elaborate on improvements; challenges; and individual responsibilities if prompted.

Some Words of Encouragement

Your interviewers will often ask about your contributions to technical projects you’ve participated in. When you’re looking for your first job, this can feel anything from harrowing to flat-out unfair—if you haven’t held a technical job, it’s only natural to feel like you dont’ *have* any relevant “previous technical projects.”

Fortunately, as Coding Boot Camp students, you *do* have relevant technical projects—namely, the group projects you’ve completed for class.

Your Technical Expertise is Real

First things first: Dispel any notion that your classwork somehow “doesn’t count” by virtue of its being classwork.

There is no stipulation that your “previous technical projects” be contracted or otherwise assigned by an employer. Your group projects are difficult assignments that present legitimate programming challenges: It takes real grit, real intellect, and real skills to work through them. You know, better than anyone, that solving those programming challenges was *anything* but easy. The fact that you did is sufficient to impress any potential employer.

While you should *never* stop coding, you shouldn’t focus on building a “more impressive” portfolio. Rather, you should focus on learning how to discuss the technical problems you have already encountered and solved.

Your Projects Are Products

Filling your portfolio with impressive projects isn’t sufficient to impress recruiters. The important piece is that you can *explain why* they’re impressive.

Instead of discussing your projects as assignments, discuss them as if they were **products**, for which your description is an elevator pitch.

Descriptions

For each project you complete, write:

- **Description**. One or two sentences summarizing your project.
- **Motivation**. Two or three sentences describing the problem the project solves. This

should answer the question: *Why would I want to use this product?*
- **Results**. One or two sentences describing the end result.

Consider this your abbreviated elevator pitch—it describes a problem, and the steps you took to solve it.

Technical Contributions

In addition to a compelling description of the product as a whole, you'll need a compelling description of your responsibilities:

- **Team Efforts**. Describe how you delegated work amongst your team members.
- **Individual Responsibilities**. Describe what *you* contributed to the project as an individual—in other words, the parts of the project that you were accountable for.
- **Challenges**. Describe any challenges you and/or your team encountered that you personally played an important role in resolving.
- **Improvements**. Describe changes you would make going forward. These can be changes you'd make for better scalability; additional features you'd add; improvements you'd make to the codebase; etc.

It's important to discuss your team's contributions, both so you look like a team player and so it's clear what knowledge your interviewer can reasonably hold you accountable for.

Your **Individual Responsibilities** are the most important piece of information on this list. Write these as a bullet list of features you implemented, or serious problems you fixed. Try to write at least 5 to 7 responsibilities for each project, and write a sentence or two further describing each one. Be ready to discuss these in-depth, if prompted.

The **Challenges** your team encountered that you played an important role in solving are also important. These should be substantial challenges—e.g., *our codebase was initiailly unorganized and quickly became buggy and unmaintainable*—and your contribution should clearly be relevant to resolving the problem—*I refactored the application to an MVC architecture to more cleanly separate concerns*.

Example

Imagine you worked on an application that allows people to search for the name of a weightlifting exercise and retrieve a description of how to perform it and a YouTube video demonstrating the exercise, and also allowed users to save “favorite exercises”.

Your “script” might read something like this:

- **Description**. “Our application helps gym-goers improve their form and safely expand the repertoire of exercises they do at the gym, without having to pay a personal trainer or ask strangers for help.”
- **Motivation**. “We had the idea because one of our teammates was just getting started in the gym, but couldn’t remember how to perform exercises she’d look up online before training, and didn’t feel comfortable asking people in the gym for help. She figured having a webapp where she could review instructions for exercises she’d save at home would solve the problem—so we decided to build one.”
- **Result**. “Using the MERN Stack; the YouTube API; and an online database of exercise instructions, we were able to create an app that allows users to search for instructions for, and demonstrations of, most major strength training exercises.”
- **Team Efforts**. “Andrew was responsible for writing the UI, and Joanna was responsible for writing the database code. I was responsible for writing the server-side code that communicated with the YouTube and Exercise Database APIs.”
- **Individual Responsibilities**. “In particular, I…”
    - Used Promises to write a helper library for API calls that Andrew could use on the front-end;
    - *Etc.*
- **Challenges**. “Eventually, we realized that the fact that JavaScript is asynchronous meant our UI wouldn’t work with the code we’d written to call the external APIs. So. I used Promises to wrap our API calls and simplify our refactor of the front-end.
- **Improvements**. “In the future, we’d like to add a way to ‘crowdsource’ exercise instructions—in particular, if a user searches for an exercise for which there are no instructions, we’d like to allow them to upload their own, which other users could see, later.”

Copyright