A GUIDE TO DEVELOPING A TECHNICAL PROJECT

Overview:

Why is it important to develop technical projects?

Developing technical projects outside of the classroom can add tremendous value to your resume and skill set. There are many reasons you should have technical projects – these include projects: help add technical experience to your resume if you don't have software engineering jobs/experience, allows you to express your interest, is the opportunity to show you were able to learn different tools/technologies that you may not have learned in coursework or other means. In addition, employers like to see side projects on your resume. It shows your passion to learn, that you can learn on your own, that you can be creative and that you are able to self-organize to accomplish a task. All of which are essential skills for success in the workplace.

Many students tend to want to develop a technical project, but are unsure where to begin or what to do. This document is here to help! Keep in mind that you want to be intentional about the projects you highlight and that projects should ALWAYS demonstrate your strengths, interests, and/or an ability to provide solutions to problems.

Career Pathways

Consider the Career You Would Like to Go Into:

When thinking about a project, make sure it aligns with the career path you are most interested in. In addition, make sure it highlights relevant skills needed for that career (technical and professional) and technical pieces you can speak to (trade offs, why you decided one thing after the other). Think about the career path you want to go into and use that to consider the skills you want to highlight in the project. These are just a few career paths and suggested skills, more skills/career paths can be found by using ONETonline.org. As you will notice, the complexity of skills and what is expected to be discussed in an interview varies by job title.

Web Developer

- Consider designing a project with features like a static vanilla JS, elementary logic, and/or popular website front-end clones
- The skills looked for in web developers include:
    - Front-end web components with React or equivalent
    - Knowledge of frameworks, but mostly comfortable with vanilla JS, HTML, and CSS
- Interviews - be comfortable talking about:
    - The project, and the technologies you used
    - Differences/trade-offs involving HTML, CSS and JS

Front-End Engineer

- Consider designing a project with features like sign-up/sign-in authorization flow, elementary API usage/routes, and/or popular website clone

- The skills looked for in front-end engineers include:
    - Building reusable, testable front-end web components with React/Redux or equivalent frameworks
    - Building routes and API endpoints using Node/Express or equivalent frameworks
    - Talking about differences/trade-offs involving React/front-end components
- Interviews - be comfortable speaking about:
    - The project,
    - The technologies you used/why you used those technologies,
    - The different decisions you made and,
    - The things you could do to improve the project

Back-End Engineer

- Consider designing a project with features like sign-up/sign-in authorization flow, popular website clones, multiple API integrations, and/or real-time messaging/video capabilities.
- The skills looked for in back-end engineers include:
    - building routes and API endpoints using Node/Express or equivalent frameworks
    - creating the DB schema and using relational and NoSQL databases
    - deployment with AWS/Heroku or equivalent technologies.
- Interviews - be comfortable with:
    - data structures and algorithms,
    - discussing the project,
    - the technologies you used/why you used those technologies,
    - the different decisions you made,
    - the things you could do to improve the project,
    - how to scale the project and,
    - a general system design question involving their project with concepts like - caching, storage, scaling, entity relationships, etc.

Full-Stack Engineer

- Consider designing a project with features like sign-up/sign-in authorization flow, popular website clones, multiple API integrations, and/or real-time messaging/video capabilities
- The skills looked for in back-end engineers include:
    - building reusable, testable front-end web components with React/Redux or equivalent frameworks
    - building routes and API endpoints using Node/Express or equivalent frameworks
    - creating the DB schema and using relational and NoSQL databases
    - building deployment with AWS/Heroku or equivalent technologies
- Interviews - be comfortable with:
    - navigating a technical conversation that describes the project,
    - technologies you used and why,
    - the decisions you made and why, and things you could do to improve the project

    - differences/trade-offs across the stack,
    - data structures and algorithms
    - how to scale the project and feel comfortable with a general system design question involving concepts like - caching, storage, scaling, entity relationships, etc.

PRACTICAL TIPS ON PROJECTS AND SELECTION:

Make sure the project 1) fits your career interest (see career section) and 2) uses modern technologies valued in the industry. If you develop a project that doesn't accomplish both items, you may be wasting your time. Here are some in demand technologies:

- **Web Development/Full-Stack/Back-End/Front-End:** JavaScript, Python, Node, HTML/CSS, React, Express, Angular, Vue
- **Mobile Development:** Swift, Kotlin, Java, React Native
- **Machine Learning:** TensorFlow, Hadoop/MapReduce, Pytorch, Jupyter
- **Developer Tooling**: Git/GitHub, Heroku, Netlify, Glitch, Firebase, Docker
- Use this link for a full listing of StackOverflow Popular Technologies

In addition, be sure to:

- Launch/deploy the project ensuring it is on the internet and usable (recruiters/hiring managers love when projects are deployed so they can actually see what you created)
    - Code should be posted to GitHub/GitLab/BitBucket
    - Application can be run on Heroku and Glitch (both have free tiers for web), OR blog post with screenshots
- Have complete features for a user
    - It is better to have a smaller thing complete, than a larger thing partially finished

Step 1 - Brainstorm

There are a LOT of things you can develop. Start by brainstorming! Keep in mind all good projects start with some sort of **PROBLEM** that you want to solve. Here are some questions you can ask yourself to think about when thinking of a project to develop.

- Are there things in your or your friends/families everyday life that could be solved by technology?
    - Could you automate something?
- Do you have your own website/portfolio?
    - You could build a blog or make your own version of a GitHub personal website
- Are there any open source projects that you think are interesting?
- What existing developer platforms or APIs exist?
    - What have people already built that you have an interest in cloning?
- What are prior hackathon prompts?
    - What have prior hackathon participants built?
- What are corporate hackathons asking for? "Call for startups"
- What have you already built? Is there something you wish you could have improved?

- Can you add onto an existing group or class project?
- What are silly ideas you've had?

Step 2 - Plan!

- Jot down all the ideas you have based on the above prompts, then think about which you'd like to start with
    - Make sure they are in line with your career interests!
- Do you need teammates? Can you do this alone?
    - If you need teammates, see who may be interested in working with you!
- Begin to break down the project into manageable chunks and goals.
    - A framework that may be helpful is Agile, which is used for developing and sustaining complex products https://www.atlassian.com/agile
    - Some tools that may help keep you on track include Asana, Jira, Trello

Sample Topics:

- Chrome extension to remind you about a website
- Budgeting app
- Send letter to future FutureMe.org
- Class sign-up automation
- Chrome extension that degrades a website the more often you visit it (eg each visit removes functionality or breaks it)
- Website that only one person can view at a time (this got built by someone else - exclusive.website)
- English to emoji translator
- What are ideas from movies/books/etc. that don't exist?
- Texting via spreadsheet (like Nelly/Kelly Rowland in Dilemma)
- Side Project Ideas
- Appointment Schedule Automation (Using Google App Script associated with Google Sheet to check and fill in availability, N-Queens, Nurse Scheduling Problem, Genetic Algorithms, Constraint Satisfaction Problems)
- 20 Exciting Software Development Project Ideas & Topics for Beginners
- 12 Mind-Blowing DIY Tech Projects
- 30 Truly Interactive Websites Built With CSS & JavaScript
- 40+ Best Website Ideas For Your Startup
- Android Projects & App Ideas
- Devpost Other Project Inspirations
- Website/Bio Inspirations
- Portfolio
- CODE AT HOME ACTIVITIES
- NASA Space App Challenge

Step 3 - Get to Work and Hold Yourself Accountable!

Resources to Help!:

Here are some resources that may help you work through your project:

- Alum Speaks About a Project that Stands Out
- Forage: which connects students with projects from different companies
- Stack Overflow: debug and ask for help when you need

- https://www.firsttimersonly.com/ - guide and list of issues that are good for first-time contributors
- Sites that will provide some information around building projects on your own:
    - Clone famous sites: look on Medium
    - MDN has many tutorials
    - Udemy Real Life Projects
    - Tuvtran
    - React tutorials
- Here's a list of APIs which could inspire you to work on a project:
    - MDN overview
    - Twitter docs
    - Spotify tutorials
    - Google Maps API
    - Weather APIs
    - Giphy API
    - Library of APIs
- **General tools to collaborate together**
    - **Github**
        - Git/GitHub Video tutorial: https://www.youtube.com/watch?v=MnUd31TvBoU
        - Team Collaboration with Github
        - https://www.freecodecamp.org/news/how-to-use-git-and-github-in-a-team-like-a-pro/
    - **Repl.it**
        - Video tutorial: How To Use Replit Python
        - https://www.freecodecamp.org/news/how-to-use-replit/
- **Web Development Tools/Resources**
    - **React**
        - **https://reactjs.org/docs/add-react-to-a-website.html**
        - React Website Tutorial - Beginner React JS Project Fully Responsive
        - **https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started**
    - **HTML/CSS/JavaScript**
        - HTML CSS and Javascript Website Design Tutorial - Beginner Project Fully Responsive
        - **https://www.pluralsight.com/paths/building-websites-with-html-css-and-javascript**
    - **MERN** (MongoDB, Express, React, Node) **Stack**
        - MERN Stack Tutorial #1 - What is the MERN Stack?
        - **https://www.mongodb.com/mern-stack**
        - **https://www.youtube.com/watch?v=WsRBmwNkv3Q&list=PL4cUxeGkcC9g8OhpOZxNdhXggFz2lOuCT**
        - **https://www.youtube.com/c/TheNetNinja/playlists**

- **App Development Tools/Resources**
    - **Flutter/Firebase**
        - **https://docs.flutter.dev/development/data-and-backend/firebase**
        - **https://www.youtube.com/watch?v=sfA3NWDBPZ4&list=PL4cUxeGkcC9j--TKIdkb3ISfRbJeJYQwC**
    - **Swift**
        - **https://docs.swift.org/swift-book/GuidedTour/GuidedTour.html**
        - **https://www.tutorialspoint.com/swift/index.htm**
        - **https://www.youtube.com/watch?v=Ulp1Kimblg0**
    - **Android App**
        - **https://developer.android.com/training/basics/firstapp**
        - **https://www.youtube.com/watch?v=FjrKMcnKahY**
        - **https://www.netsolutions.com/insights/android-app-development-tutorial-learn-basic-concepts/**
- **Cloud Technologies**
    - **https://learn.microsoft.com/en-us/training/azure/**
    - **https://aws.amazon.com/training**
    - **https://cloud.google.com/training**
- **Machine Learning Tools/Resources**
    - **https://serokell.io/blog/top-resources-to-learn-ml**
    - **https://towardsdatascience.com/beginner-friendly-resources-for-machine-learning-fd198f844dc3**
    - **https://medium.com/machine-learning-in-practice/my-curated-list-of-ai-and-machine-learning-resources-from-around-the-web-9a97823b8524**

  AI Tools/Resources for Web and Mobile applications
    - **https://hub.packtpub.com/7-ai-tools-mobile-developers-need-to-know/**
    - **https://appinventiv.com/blog/how-to-integrate-machine-learning-artificial-intelligence-into-apps/**

Sample Projects!

- https://github.com/connorskees/grass
- https://github.com/rust-lang/regex
- https://github.com/BurntSushi/ripgrep
- https://github.com/jonhoo/inferno

After You Finish Your Project - Add it to Github

Make sure your project is listed on your GitHub. Review our GitHub - What is it and how to stand out documents for tips and insights.

Add the Project to Your Resume!

Now that you completed your project, it is important to showcase it! It is essential that you add this project to your resume as you want to be sure employers know what you were able to develop. Make sure you include the following when you add it to your resume:

- Relevant Links - this includes the source code and live site/app
- A bullet that states what you made, the technologies you used and the problem it intends to solve?
- Bullets about some of the features?
- Your specific contributions (if you worked in a group it is essential your bullets explain what it did)
- Show the results or the intended result of the project

Get Ready to Talk About It!

You want to be comfortable speaking about your project. These are some things an employer may ask:

What does it do?

- Make sure this is a quick summary don't get too detailed

Why did you pick __ technology?

- Speak to why you picked one technology over another
- If you can speak to technical tradeoffs, do that!
- Otherwise, speak to benefits to your career (it's used at this company/widely used in the roles I want, wanted to learn something new)

What did you learn?

- Try to keep it technical - did you discover something new about a technology you were using? Did you learn how X and Y relate to each other, or how to use them together (e.g. React/Redux, why use both?)
- If you have real users, you can also speak to what feedback you got and any changes you made because of that

What was hard about it?

- Focus on technical aspects:
    - Were there slow/too many database queries?
    - Was the UX slow?
- Be prepared to talk about the solutions you tried and why they didn't work, in detail (use the shared doc/whiteboard if it's helpful!)
- Challenges working in a team or individually
- If nothing was hard, say why

What would you do differently/what's next?

- Focus on technical
    - Use a different data model
    - Found some new framework that was better for what I was doing
- OK to speak about new features, but be ready for how you would implement them

Misc Questions

- Were there any interesting bugs you found?
- Did you do any user testing? What were the reactions? What did you learn from it?

The Handout was Developed using the following resources:

- John Cline/Emilie Hsieh 2021-10 TTP Interview Workshop
- Github_Guide_Rubric_Deck
- https://handbook-for-cuny-hunter-cs-students.webnode.page/individual-projects/