Carnegie Mellon University
Silicon Valley

SOFTWARE ENGINEERING CAREER GUIDE

Student Affairs at Carnegie Mellon University Silicon Valley

CONTENTS

INTRODUCTION

Acknowledgment section

These career guides are adapted each year, and the career services team would like to sincerely thank the 2019 cohort of Peer Career Consultants for creating the first iteration. We appreciate the hard work of Chaitra Dinesh Pai, Nursultan Dyussebayev, Arpit Gupta, Sweta Hari Kumar, and Joseph Konan who researched specific fields of interest and compiled helpful tips and resources for students to enable them to make the most of their CMU-SV experience and navigate the internship and job search process in Silicon Valley. Their valuable contributions made these career guides possible. Additionally, the example resumes are compiled from recent alumni who also wanted to help our future students at CMU-SV. Thank you to everyone who contributed their time and effort.

Many software engineering students use C and C++ as their primary programming languages, and this guide aims to help students leverage these skills to attain job offers at their dream companies. C/C++ programmers are required in multiple arenas of software development, ranging from desktop applications to native mobile applications, and embedded systems. If you know C, be very proud because this means you can pick up any language with some effort and time! C is a language that stays in demand despite the development of many new languages, as it is paramount to developing operating systems, device drivers, networking protocols and even in developing other programming languages. C++ is also highly utilized in the development of gaming engines. If some of these options interest you, please read on ahead to get tips on what skills to hone so as to shine at technical interviews.

This next section will define some generic expectations that most companies have from C/C++ programmers:

- Strong proficiency in C and C++
- Deep understanding of various Data Structures & Algorithms
- Thorough knowledge about standard library and STL containers
- Strong grasp of OOPS concepts
- Familiarity with multi-threaded and concurrent programming, which can be shown through work experience or academic projects
- Familiarity with system call wrapper library functions
- Familiarity with embedded systems design, low-level hardware interactions
- Knowledge of language tools such as Valgrind
- Proficient understanding of version control systems, such as Git, Mercurial
- Good understanding of memory management in non-garbage collected environments
- Implementation of unit testing
- Good debugging and testing skills

It is important to note that even though this next requirement may not be listed explicitly on job descriptions (they mostly are), this is a crucial requirement to most companies

- Expertise in at least one scripting language (Python, Javascript, Ruby, PHP)

EXAMPLE RESUME

To give you an idea of what a software engineering resume can look like before it's tailored to the job description, please see the example below. *Please note, these roles on the resume may not have occurred between 2018-2020. This is a resume example from a real CMU student, however, other data points are edited each year. Please recognize that this is not a real candidate. Please email career-services@sv.cmu.edu if you have any questions about this example resume or would like to schedule a resume appointment.

CMU Tartan

cmu.tartan@sv.cmu.edu
https://linkedin.com/in/cmu-tartan
123-456-7891
https://github.com/CMUtartan
Mountain View, CA

Education

- **Carnegie Mellon University Silicon Valley** — Mountain View, CA
  Master of Science in Software Management — *Aug 2019 - Dec 2020*
  Courses: Foundation of Computer System, Architecture and Programming Principles, Cloud Computing
- **Tongji University** — Shanghai, China
  Bachelor of Engineering in Software Engineering — *Sep 2015 - Jul 2019*
  Courses: Object-Oriented Programming, Data Structures, Algorithms, Operating Systems, Computer Network, Software Engineering

Skills

- **Programming Languages**: Java, Python, Scala, Javascript, Typescript, C, C++, Ruby, Swift
- **Databases**: MySQL, PostGreSQL, MongoDB, DynamoDB, Neo4j, HBase, Redis
- **Frameworks and Tools**: Java SpringBoot, Node.js, React, Vue, Ruby on Rails, Hadoop, Spark, Git, Docker, Kubernetes, Kafka

Internships

- **Amazon** — Sunnyvale, CA
  Software Development Engineer Intern — *May 2020 - Aug 2020*
  - Designed and implemented ETL pipelines in a Spark cluster which performed transformation and aggregation on a 3 TB raw billing dataset and processed them into DynamoDB using Scala
  - Reduced current product's API query latency by 8 times (from 25 seconds to 3 seconds) by constructing a NoSQL schema and building a new Java-based API service on AWS ECS which was suitable for serving complex billing data queries and scaling up for larger-sized upstream datasets
  - Shortened billing data lag time from 1 day to 5 seconds by building an AWS SQS message consumer which updated relating rows in NoSQL tables once billing data change according to organizational structure change using Java
- **SAP** — Shanghai, China
  Software Development Engineer Intern — *Jul 2018 - Apr 2019*
  - Built 20+ SAP Jam public front-end components with high reusability and cross-browser compatibility using React
  - Implemented an efficient front-end state management method to handle real-time customer data with Redux
  - Designed microservices and implemented SAP Jam back-end features & RESTful APIs using Ruby on Rails
  - Developed and deployed an integration tool which connects GitHub and Jira by updating Jira status once pull requests were published to SAP GitHub repository using Ruby, succeeded in improving teamwork efficiency

Projects

- **Twitter Analytics Web Service** — Mountain View, CA
  Carnegie Mellon University — *Mar 2020 - May 2020*
  - Built high-performant and reliable (Rank 3rd place among all 40 teams during live test) web service on AWS which analyzed a large Twitter dataset stored in MySQL & HBase and served complex CRUD queries using Java
  - Designed and programmed MapReduce jobs on a 1TB raw dataset from Twitter on Hadoop clusters using Java
  - Optimized service throughput to up to 50,000 per 10 minutes by auto-scaling the web server instances using AWS load balancers, implementing database connection-pooling using HikariCP, and distributing requests to replicated AWS RDS databases
- **WeCloud Chat - A Social Network Chatting Application** — Mountain View, CA
  Carnegie Mellon University — *Feb 2020*
  - Built a scalable real-time chatting web application with a loosely-coupled microservice architecture deployed on Google Cloud Platform and Azure
  - Containerized profile, chat and login services with Docker and Kubernetes, leveraged Horizontal Pod Autoscaler to scale up the services for circumstances with high volume of requests
- **Research on User Profiling Based on Transfer Learning** — Shanghai, China
  Tongji University — *Feb 2019 - Jun 2019*
  - Collected 20,000 users' profiles and reviews data by a Python web-crawler from Dianping for training
  - Extracted different dimensions of language features with designed matching strategy from Dianping and Yelp user reviews using Python
  - Performed topic distribution analysis by understanding users' review texts using LDA model
  - Acheived a 70% accuracy in predicting user gender by implementing a transfer learning model based on Random Forest with a pre-trained text to vector embedding using Word2Vec model

GETTING READY FOR INTERVIEWS

Although the above requirements state the required skills, here is a bunch of material to be covered and helpful links and books that will help achieve the above requirements!

Cracking the Coding Interview By Gayle Laakmann McDowell - This is a must have book when you are starting out preparing for interviews. It covers all the basic data structures and has some really good programming questions.

Elements of Programming Interviews By Adnan Aziz, Tsung-Hsien-Lee, and Amit Prakash - Once you are familiar with data structures (Recommend starting with Cracking the Coding Interview book), move on to EPI, which dives more deeply into data structures, and becomes slightly more complex.

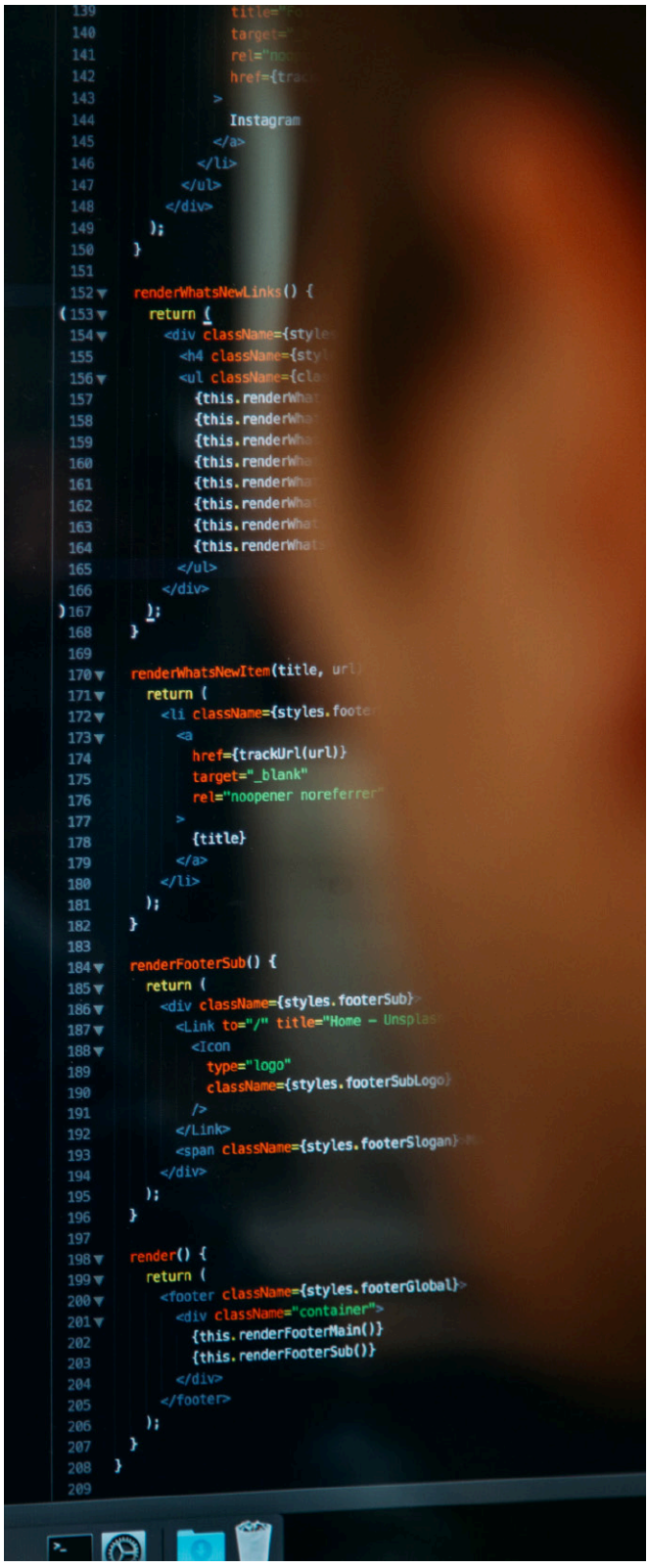
While preparing with the two books, keep practicing online on LeetCode.com, HackerRank.com, and CodeSignal.com on the data structures that you covered to gain expertise. Some students also buy the premium version and practice questions specific to their dream companies. While it is difficult to recommend a certain number of questions one should do to be able to ace coding interviews, continuity in practice of algorithm questions is the key. One must exercise the data-structures and algorithms skills repeatedly at regular intervals for the interview prep to be most effective. There are several different study-guides/resources available online. I consulted many video tutorials to understand and reinforce several concepts. There are many great Youtubers who share explanations to questions, interview tips, and preparation strategies. I recommend subscribing to your favorite technical interview channel so that you are always reminded of the way they approach problems. I would recommend the following study material:

- Overall study guide: https://workflowy.com/s/study-guide/RD5kZ682pWX5oxiE
- 12 Week Leetcode based program: https://docs.google.com/document/d/1wUCqhVHydWiDk6FJdFLSMpgigNrGcs4OFZg0Wa7JGEw/edit
- Video playlist of solved leetcode questions: https://www.youtube.com/playlist?list=PLAE-zml3hxQvdC8iD61W9-lqeUd4RWC45
- Individual Topic/Algorithm Explanations by Tushar Roy https://www.youtube.com/channel/UCZLJf_R2sWyUtXSKiKlyvAw

More Youtube Explanation about algorithms and leetcode questions:

- Back To Back SWE - YouTube: https://www.youtube.com/channel/UCmJz2DV1a3yfgrR7GqRtUUA
- Irfan Baqui - YouTube: https://www.youtube.com/channel/UCYvQTh9aUgPZmVH0wNHFa1A
- Clément Mihailescu - YouTube: https://www.youtube.com/channel/UCaO6VoaYJv4kS-TQO_M-N_g

Interview Tips & Tricks: Programming Interview Questions + Help Getting Job Offers | Interview Cake: https://www.interviewcake.com/



Make sure to brush up on the below concepts:

BIG O NOTATIONS - Try to develop a good understanding of this, as you will be asked to explain the time and space complexity of every algorithm you write down in front of your interviewer.

https://www.bigocheatsheet.com/ - This is a good link to brush up on BIG O, **once you are done studying all major algorithms**.

Following are the Top 10 Algorithm topics (according to TechLead - YouTube (https://www.youtube.com/channel/UC4xKdmAXFh4ACyhpiQ_3qBw) and Clément Mihailescu - YouTube (https://www.youtube.com/channel/UCaO6VoaYJv4kS-TQO_M-N_g):

1. DFS - Depth First Search
2. BFS - Breadth First Search
3. Matching Parenthesis - Best way to solve this is stacks
4. Hash Tables - Memoization, Caching
5. Multiple Pointers/Variables - Ex. Longest Palindrome Substring
6. Reversing a Linked List - Ex. Checking for cycles, removing duplicates
7. Sorting Fundamentals - Know runtimes, applications
8. Recursion
9. Custom Data Structures - Ex. Suffix tree like data structure, OOA
10. Binary Search

Here are some of the data structures you absolutely must practice:

- ARRAYS
- STRINGS
- LINKED LISTS - SINGLY AND DOUBLY https://www.hackerearth.com/practice/data-structures/linked-list/singly-linked-list/tutorial/
- TREES - BINARY , BST and NARY https://www.hackerearth.com/practice/notes/trees/
- GRAPHS https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
- STACKS
- HEAPS https://www.geeksforgeeks.org/heap-data-structure/

The next few are for more advanced preparation, before you start, make sure that you are an expert in the above given structures.

- TRIES - https://www.geeksforgeeks.org/trie-insert-and-search/
- QUAD TREE
- RED BLACK TREE

Along with these data structures, here are some algorithms you absolutely must know:

- SORTING - (bubble, selection, merge, heap, quick, Insertion) - https://www.hackerearth.com/practice/notes/sorting-code-monk/
- Binary Search and other searching algorithms
- Hashing using maps
- Graph traversal algos - Breadth-first-search, Depth-first-search, Dijkstras (Use Hackerearth links for these)
- Dynamic Programming - https://www.youtube.com/watch?v=OQ5jsbhAv_M
- Greedy Algorithms - https://www.geeksforgeeks.org/greedy-algorithms/

All of these topics have really good material online, sometimes it is helpful to use multiple links online (hackerearth, geeksforgeeks, tutorialpoint) to get a good grasp.

This next section lists a number of extremely helpful conferences!

- **CPPCon**
- **TAPIA**
- **DeveloperWeek**
- **GoToConference**
- **TOC + Converge every Fall**
- **Grace Hopper Conference**

If you are a woman, make sure to attend the **GRACE HOPPER CONFERENCE**. There are many graduate students who get multiple calls from various companies at this conference.

INTERVIEWS

The timeline for interviews differ for each company. The whole process could last anywhere between 2 weeks to 3 or 4 months, depending on how fast the company moves onto the next step in the interview. Start early! Make an appointment with your go-to career consultant at CMU-SV if you have questions about this.

The Online Assessment (OA)

The first step for new grads is usually an online coding assessment. (Many of these are either hosted by hackerrank or codesignal). Typically these are leetcode style questions and your aim is to pass all the test-cases in the given amount of time. Practicing with a timer helps you prepare for such scenarios. Take the OA seriously and make sure that you are not disturbed for the duration of the OA. It is really helpful to do a google search about the kinds of questions a company asks in its OA so that you can prepare accordingly. You can find such information on Leetcode discussions, Aonecode.com, Blind.com, Glassdoor, or on Reddit. It is however important to know that the information of these forums might be inaccurate or outdated so you must take it with a grain of salt.

These usually contain computer science fundamentals or math questions(depending on the company). For example, Pure Storage sends out hackerrank coding challenges with questions about caching, and synchronization mechanisms (semaphores and mutexes). In addition to these, they also have some coding challenges that would require the knowledge of data structures and practice on websites like leetcode. Mathworks sends out a hackerrank OA with questions related to probability, and basic language fundamentals of two languages you can pick out from - C,C++,Java,Python, Matlab. You would need to pick out one from the first three, and one scripting language - (Matlab or Python).

Some coding challenges:

- https://app.codesignal.com/company-challenges
- https://www.hackerrank.com/interview/interview-preparation-kit
- https://leetcode.com/interview/

The Phone Interview

The next step is usually one or more phone interviews. Here, you are expected to code one or more questions on some platform during a video call. The important thing to remember is to walk the interviewer through the thoughts in your head. Which data structure you think will be fit for this problem, which algorithm, it is probably best to start with explaining the brute force solution(the solution with worst time complexity). From there, you need to start explaining another algorithm with better time/space complexity. You can then ask the interviewer which one they would like you to code. Then move on to code in proper syntax.

Another major part of phone interviews may be asking you about the projects and internships you have listed on your resume, and remember! Some companies ask about your resume in depth. So make sure you brush up on all the details of the projects and internships you have listed on your resume!

The Technical Interview

In order to prepare for technical coding questions, leetcode, and hackerrank are very good places to start.

- https://leetcode.com/problemset/all/
- https://www.hackerrank.com/dashboard

But, simply solving these problems may not be enough. It is a really good practice to practice mock interviews with friends, and **practice on a whiteboard**! It is a different experience to code on a whiteboard while compared to coding on leetcode or hackerrank. White boards do not have autocomplete! We need to be syntactically correct, and very sure about the quirks of the particular language we are using. Practicing on a whiteboard helps a lot! Also, make sure to assess the time and space complexity in **Big O notation** every time you solve a problem.

The best way to practice for coding interviews is to interview with as many companies as possible. This will calm your nerves by the time you are doing your third or fourth interview, and will also give you opportunities to work on different aspects of your interviewing skills (needless to mention that a couple of offers will help you negotiate better). TripleByte is a place where you can interview and can skip a couple of application steps at some of the companies.

Usually, the main difference between internship interviews and full time job interviews is that internship interviews are usually easier. They have less coding rounds, and comparatively easier questions. But they still require good knowledge of data structures and algorithms, and a significant amount of practice on leetcode.

Given below are some questions for getting started:

- https://leetcode.com/problems/two-sum/
- https://leetcode.com/problems/reverse-linked-list/
- https://leetcode.com/problems/delete-node-in-a-linked-list/
- https://leetcode.com/problems/same-tree/
- https://leetcode.com/problems/binary-tree-preorder-traversal/
- https://leetcode.com/problems/binary-tree-inorder-traversal/
- https://www.geeksforgeeks.org/commonly-asked-c-interview-questions-set-1/

After these, try to get into a schedule and do at least one or two leetcode or hackerrank problems a day. And very important - assess the time and space complexity of each question.

SYSTEM DESIGN

This next point is for students interested in pursuing a career in systems engineering - Make sure to read the first 10 chapters of the Operating Systems Dinosaur book - You can get this from Amazon. It offers a very solid introduction to OS fundamentals.

CO-CURRICULARS

Internships - If you lack work experience, make sure to try your best to get an internship with a company you are interested in, as while searching for a full time role, these internships will give a lot of value over academic projects. It is often easier to get an internship, by learning basic OOPS concepts, and practicing easy and medium questions on Leetcode. Do at least a 100 questions before on-site interviews.

Research Assistantship - CMU SV has many professors willing to give you RA for credit, and some for pay as well. If you need to build some work experience, please use the Student Project Tracker, and apply to positions that interest you! RA on your resume looks really good, and recruiters will be excited to ask you about what you implemented. If you are really interested in a particular project/field, you can approach a professor in that field and propose a semester-long research project for credit. It would help if you have taken a course with that professor, but irrespective of that fact, you should not shy away from approaching any professor for a possible research opportunity.

Coursera certifications - Coursera is a really good website which lets you explore many technical topics. It gives you a choice to audit many courses without paying. If you finish auditing a course that you did well on, and if the course adds value to the role you are looking for, get the certification and put it on your resume!

Teaching Assistantship - This is a really fun way to excel at a course, and often impresses recruiters, as it proves your proficiency in a course. If you are interested in TAing a course, make sure the course is relevant to the role you are searching for. Being a TA for a course helps you showcase not only your technical skills but also the soft skills that are required for teaching.

Academic Projects - While at CMU, try to balance out courses, some with exams and some with projects as the major component, as they offer very different experiences crucial to performing well at your job. Timed examinations make you practice timeboxing yourself which is often tested during interviews, and academic projects add a certain value to you as a candidate, so make sure that you give it your all! Additionally, the class projects can really add value to your candidacy as they underscore the application of possibly required skills for a job.

Courses at CMU

If you want to improve your C skills, and get some highly useful knowledge about systems in the process, we recommend taking **FOUNDATIONS OF COMPUTER SYSTEMS**.

If you have some time before this course starts(like the summer or winter break) , please brush up your C skills, and practice dynamic memory allocation, pointer manipulation, X86 assembly language, GDB etc before you take the course, it will be extremely helpful!

If interested in cloud, but cannot take the very time consuming Cloud computing course due to other course requirements, a very good alternative is the Cloud infrastructure course offered by **INI**. I particularly enjoyed the projects offered by this course.

I would also suggest to spend your time at CMU exploring different aspects of Software Engineering. This might well be the last time you have the liberty of choosing a diverse set of courses in areas that you have fancied but never got around to. The special topic mini's are a good elective option as they help you network with notable individuals from the field and with peers from other disciplines.

TIP: If you see a course offered in Pittsburgh that you would really like to take in the next semester, try to gather other students who are interested in the course, and talk to the academic advisor. They may be able to arrange a session for you guys! Do this early and during the previous semester, so that they have enough time to arrange it!