FAANG Career Preparation Plan

Overview

This document outlines a comprehensive plan to prepare for a software engineering role at top-tier tech companies (FAANG: Meta, Google, Amazon, Netflix, etc.). It targets a computer science undergraduate with a 3.4 GPA, skilled in competitive programming, C, Rust, C++, Java, Python, Linux, React, and Tauri. The focus areas include Data Structures and Algorithms (DSA)/Coding, System Design, Projects/Portfolio, Behavioral Interviews, and Networking/Applications.

1 DSA/Coding

Mastering Data Structures and Algorithms (DSA) is critical, accounting for 70% of technical interviews. Aim to solve 500+ problems on platforms like LeetCode (200 easy, 200 medium, 100 hard), focusing on patterns. Practice explaining solutions aloud in Python or C++ for speed, as Rust is niche unless specified.

- **Arrays & Strings**: Manipulation, two-pointers, sliding window, prefix sums, longest substring, anagrams, palindromes.
- **Linked Lists**: Reversals, cycle detection, merging, fast/slow pointers, cloning with random pointers.
- **Stacks & Queues**: Next greater element, monotonic stacks, parentheses validation, LRU cache.
- **Trees & Binary Trees**: Traversals (in/pre/post-order, level-order), height, diameter, LCA, serialization, balanced checks, path sums.
- **Binary Search Trees (BSTs) & Balanced Trees**: Insert/delete/search, inorder successor, AVL rotations.
- **Graphs**: BFS/DFS, shortest paths (Dijkstra, Bellman-Ford), topological sort, connected components, union-find, MST (Kruskal/Prim).
- **Heaps & Priority Queues**: Kth largest/smallest, median in stream, merge k sorted lists.
- **Hashing & Maps**: Collision handling, custom hash functions, subarray sums, group anagrams.
- **Sorting & Searching**: Quick/merge/heap sort, binary search variants (rotated array, first/last occurrence).
- **Dynamic Programming (DP)**: 1D/2D arrays, knapsack, LIS, coin change, matrix paths, state compression, digit DP, probability DP.
- **Greedy Algorithms**: Interval scheduling, jump game, fractional knapsack.
- **Bit Manipulation**: XOR tricks, bit masks, counting bits, single number variants.
- **Recursion & Backtracking**: Subsets, permutations, N-queens, sudoku solver.

- **Advanced Strings**: KMP, Rabin-Karp, suffix arrays.
- **Advanced Data Structures**: Tries, segment trees, Fenwick trees, monotonic queues.
- **Math & Misc**: Prime sieves, modular arithmetic, reservoir sampling, randomized algorithms.

Resources: NeetCode 150, Grokking the Coding Interview, Striver's sheet. Practice 3 hours daily, use Pramp/Interviewing.io for mocks.

2 System Design

Entry-level roles require basic system design knowledge, emphasizing scalability, reliability, and trade-offs. Practice low-level designs (e.g., URL shortener, chat app) and deploy one on AWS/GCP to demonstrate practical skills.

- **Requirements Clarification**: Functional (features, users) vs. non-functional (scale, throughput, security).
- **High-Level Design**: End-to-end flow, 5-6 components (clients, APIs, services, databases).
- **APIs & Communication**: REST/GraphQL, RPC, WebSockets for real-time.
- **Databases**: SQL vs. NoSQL, sharding, replication, ACID vs. BASE.
- **Caching**: Strategies (CDN, Redis), eviction policies (LRU/LFU).
- **Load Balancing & Scaling**: Horizontal/vertical, auto-scaling, microservices.
- **Message Queues**: Kafka/RabbitMQ for async processing.
- **Security & Reliability**: Authentication (OAuth/JWT), encryption, fault tolerance, backups.
- **Common Systems**: TinyURL, Twitter feed, chat app, recommendation engine, autocomplete.
- **Trade-Offs**: Monolith vs. microservices, consistency vs. availability (CAP theorem).
- **Monitoring & Logging**: Metrics, alerting (Prometheus, ELK stack).

Resources: Grokking the System Design Interview, ByteByteGo (YouTube). Study 1 hour daily, integrate Linux/React skills in examples.

3 Projects/Portfolio

A strong portfolio compensates for a 3.4 GPA. Build 3-5 impactful projects on GitHub with clean code, documentation, and metrics (e.g., "40% latency reduction"). Tailor to Rust, Tauri, React, Linux skills.

- **Full-Stack Apps**: MERN/Tauri desktop app with React frontend (e.g., real-time chat with WebSockets, auth, database).
- **CLI Tools**: Rust-based Linux sysadmin tool (e.g., performance optimizer, log analyzer).
- **Games/Engines**: Pygame/Rust game (e.g., chess AI with minimax), or custom engine component.
- **Data/Algo Projects**: Competitive programming visualizer (graphs, trees in React), ML app with Torch/NetworkX.
- **Open-Source Contributions**: Fix bugs in Rust/Python repos, add features to Tauri/Linux tools.

- **Scalable Systems**: Rust-based distributed simulator, AWS-deployed app with caching/load balancing.
- **Portfolio Site**: HTML/CSS/JS site showcasing projects with case studies (problem, solution, impact).

Action: Dedicate 1 hour daily. Open-source projects or fake metrics if no users. Host on GitHub.

4 Behavioral

FAANG emphasizes cultural fit through behavioral interviews. Prepare 10-15 stories using STAR (Situation, Task, Action, Result) tied to company principles (e.g., Amazon's Leadership Principles, Google's Googliness).

- Tell me about a time you failed and what you learned.
- Describe a conflict with a teammate and how you resolved it.
- When did you take a risk that paid off?
- How have you innovated or improved a process?
- Tell me about a time you led a project/team.
- Describe handling a tight deadline.
- When did you disagree with a superior?
- How do you prioritize tasks?
- Tell me about mentoring someone.
- Why [Company]? (Align with company values, e.g., Customer Obsession for Amazon).

Action: Practice aloud 30 minutes daily, use competitive programming/Linux experiences. Be honest; faking is obvious.

5 Networking/Applications

Networking bypasses GPA filters; referrals triple your odds. Apply to 100+ roles via company portals, LinkedIn, Handshake. Build genuine connections, not spam.

- **LinkedIn Mastery**: Optimize profile, comment on posts, DM alumni/recruiters (10 personalized outreach/week).
- **Events & Conferences**: Attend tech meetups, hackathons, job fairs (1/month, follow up same day).
- **Alumni & Communities**: Connect via school networks, Reddit (r/cscareerquestions), Rust/Python Discord.
- **Informational Interviews**: Ask for advice, not jobs, to build rapport.
- **Open-Source & Groups**: Contribute (5 PRs/month), join IEEE or similar.
- **Internal Referrals**: Leverage ex-coworkers moving to big tech.
- **Application Spam**: Track rejections, iterate resume for company portals.

Action: Spend 30 minutes daily networking/applying. Target Google STEP, Meta University, Amazon SDE Intern.

6 Daily Plan

Focus Area	Daily Time	Resources	Why It Matters
DSA/Coding	3 hours	LeetCode, NeetCode, Grokking	Core of technical rounds.
System Design	1 hour	Grokking, ByteByteGo	Shows big-picture thinking.
Projects/Portfolio	1 hour	GitHub, AWS free tier	Proves skills over GPA.
Behavioral	30 min	Company principles, mocks	Ensures cultural fit.
Networking/Apps	30 min	LinkedIn, Handshake	Gets you in the door.

Table 1: Daily preparation schedule for FAANG roles.

Conclusion

Execute this plan daily for 3-6 months to maximize FAANG interview chances. Focus on DSA first, then system design, projects, behavioral prep, and networking. Your 3.4 GPA is a hurdle, but impactful projects and referrals can overcome it. Track progress, iterate on rejections, and leverage your Rust, React, Linux, and competitive programming skills.