ChatGPT

Public Background of Aakarshit Srivastava

Aakarshit Srivastava is an Indian computer science student and developer noted for wins in national-level competitions and research contributions. He currently serves as an intern at Volkswagen Group Technology Solutions India [1]. His LinkedIn profile describes him as *"Founder @PerfectCube"*, a machine learning expert and technical writer, and highlights that he was a Flipkart Grid 6.0 semifinalist, Volkswagen i.mobilothon 4.0 winner, and an "Unstoppable Leader 2025" [2]. For example, as part of a team from PSIT Kanpur he won first prize at VW's "i.mobilothon 4.0" hackathon (₹1.5 Lakh prize) in early 2024 [3]. He co-founded the **PerfectCube** group (with Ayush Verma and Bhaskar Banerjee) to work on AI and cloud projects [4]. Aakarshit is proficient in many programming languages and frameworks (Python, Java, C#, C++, Kotlin, Dart, Flutter, Django, Flask, etc.) [5]. He is actively writing technical articles and has co-authored peer-reviewed research papers. Notably, he co-authored an IJFMR paper on hybrid RL scheduling for energy-efficient cloud computing [6] and another on a post-quantum cybersecurity framework (Crystal Quantum Shield) [7].

- **Key Achievements:** Winner of Volkswagen i.mobilothon 4.0 (EV battery optimization) [3]; **Unstoppable Leader 2025** recognition (Unstop talent awards) [2]; Flipkart Grid 6.0 semifinalist [2]; 13x national hackathon finalist and winner (via PerfectCube projects) [2] [3]; Intern at VW Group Tech Solutions [1].
- **Research & Publications:** Co-author on IJFMR papers ("QWhale and SARSAWhale: Energy-Efficient... Cloud Environments" [6], "Crystal Quantum Shield (CQS): Post-Quantum API Security" [7], etc.). He maintains a GitHub blog and Medium posts on topics like QPUs, neural nets, and AI frameworks (see GitHub profile).
- **Technical Profile:** Describes himself as a *"seasoned machine learning expert, cloud architect, and competitive coder"* [5]. His GitHub README lists expertise in cloud platforms (AWS Athena/SageMaker), blockchain, cybersecurity, and daily practice in algorithms.

Public Profiles: Aakarshit's GitHub handle is **ArkS0001**, and his personal website (PerfectCube) and Medium blog (`@arks0001`) contain much of his work. His LinkedIn is at in/arks0001 [8]. The Perfect-Cube GitHub page (https://github.com/Perfect-Cube) lists him by name along with teammates [4]. PSIT Kanpur's LinkedIn post also highlights his hackathon win [3].

Go Learning Roadmap for Aakarshit Srivastava

Given Aakarshit's strong background in Python, C++, C#, machine learning and cloud, the following Go (Golang) learning plan builds from fundamentals to advanced topics, with practice exercises, project ideas, and Google-specific interview prep. The official Go language site notes that Go is a Google-supported, open-source language designed for simplicity and built-in concurrency [9]. This roadmap emphasizes idiomatic Go, concurrency patterns, and common Google-style software engineering tasks.

1. Go Fundamentals (Beginner)

- **Basic Syntax & Data Types:** Learn Go's syntax, basic types (int, string, slices, maps), and control flow (`for`, `if`, `switch`). Use the official *"Tour of Go"* tutorial and Go by Example to cover basics. Go's documentation (on go.dev) is authoritative [9].
- **Functions, Methods, and Interfaces:** Study function declarations (including multiple return values and named results), pointers, and methods on `struct` types. Learn Go interfaces (duck typing) and embedding. This is key to Go's style – translating Java/C++ code directly is discouraged; rather learn Go idioms (Effective Go advises writing *"the problem from a Go perspective"* for best results) [10].
- **Error Handling:** Practice Go's explicit `error` type. Read about idiomatic error checks (e.g. `if err != nil`). Use `panic` / `recover` sparingly.
- **Packages and Modules:** Understand Go's project structure and workspace. Use Go modules (introduced in Go 1.11+) for dependency management. The Go team emphasizes that *"Go modules are the future of dependency management in Go"* [11], so practice `go mod init` / `go get`.

2. Concurrency and Intermediate Concepts

- **Goroutines & Channels:** Go's standout feature is lightweight concurrency. Learn to launch goroutines (`go func()`) and coordinate with channels (`chan`) and `select`. Explore patterns like worker pools. Go documentation highlights its built-in concurrency and robust std library [9].
- **Synchronization:** Learn the `sync` package (Mutex, WaitGroup) and `context` for cancellation. Practice controlling shared data.
- **Standard Library:** Familiarize with common packages: `net/http` (building web servers/clients), `encoding/json`, `database/sql` (with a driver like Postgres or SQLite), and `fmt`, `io/ioutil`, etc. Go's batteries-included library lets you build many tools without third-party code.
- **Testing and Tooling:** Use Go's built-in testing (`go test`) and benchmarking. Apply `go fmt`, `go vet`, and lint tools to write clean code. Follow *Effective Go* conventions for formatting (see `gofmt` advice in documentation) [10].

3. Advanced Topics

- **Generics (Go 1.18+):** Study Go's new generics (type parameters) to write reusable data structures.
- **Performance & Profiling:** Learn to use the built-in profiler (`pprof`) and benchmarking (`testing.B`). Understand Go's garbage collector and how to write memory-efficient code.
- **Networking & RPC:** Implement higher-level services: explore `net/http/httprouter`, gRPC (protobufs), and microservices patterns. Since many Google systems use gRPC/Kubernetes, this is valuable.
- **Internals and Concurrency Patterns:** Read about Go runtime and scheduling if time allows. Learn advanced concurrency patterns (pipelines, fan-in/fan-out).
- **Cross-language Integration:** (Optional) For his C++ background, try using `cgo` to call C code from Go.

4. Practice Problems & Project Ideas

- **Coding Challenges:** Solve algorithm/data-structure problems in Go on sites like LeetCode or HackerRank. Focus on trees, graphs, sorting, and dynamic programming. This builds fluency in Go syntax for interviews.
- **Project Ideas:**

- **Web Service API:** Build a simple RESTful API (e.g. a TODO list or URL shortener) using `net/http`, Gorilla Mux or Gin, and a database. Covers JSON, HTTP, and data storage.
- **Concurrent Data Processor:** Write a program that fetches or processes data in parallel (e.g. web crawler, log processor) using goroutines and channels.
- **CLI Tool:** Create a command-line utility (e.g. file search, text parser, or a simple static site generator) using Go's flag parsing and file I/O libraries.
- **Kubernetes Exercise:** (Bonus) Write a simple Kubernetes controller or operator in Go, since Kubernetes is written in Go (demonstrates advanced Go use in cloud).

5. Google Interview Preparation

- **Data Structures & Algorithms:** Google interviews emphasize algorithms. Practice coding problem sets in Go, focusing on efficiency and clarity. Leverage Go's testing to verify solutions.
- **System Design:** Be prepared to discuss design of scalable systems. With Go's ecosystem, mention how you might use microservices, gRPC, and cloud services (e.g. Google Cloud).
- **Go-specific Skills:** Study common Go interview topics (e.g. how Go handles concurrency, channels vs shared memory, Go's model of error handling). Understand Go's memory model (slice internals, garbage collection).
- **Resources:** Use books like *"Elements of Programming Interviews in Go"* or online guides. Google's own advice is to code legibly and efficiently in the chosen language, and follow Go style (the official docs encourage idiomatic naming and formatting) [10].

6. Learning Resources

- **Official Go Documentation (go.dev):** The authoritative site (includes *"Get Started"* guides, spec, and a blog). Go's homepage notes Go's ease of learning and strong std library [9].
- **Tour of Go (tour.golang.org):** An interactive tutorial covering syntax and core concepts from basics to concurrency.
- **Effective Go (go.dev/doc/effective_go):** Official guide to idiomatic Go programming [10].
- **Go by Example:** Clear, annotated examples of Go idioms and patterns.
- **Exercism – Go Track:** Community-driven Go exercises with mentor feedback.
- **Books:** *The Go Programming Language* (Donovan/Kernighan) for in-depth study; *Go in Action* or *Go Programming Cookbook* as alternatives.
- **Online Courses:** Reputable courses on Udemy, Coursera or Pluralsight.
- **Community:** Go's Gopher Slack, StackOverflow, and Reddit (r/golang) for Q&A.

Each learning resource complements Aakarshit's existing skills (e.g. building web/cloud apps in Go to leverage his cloud and ML knowledge) and prepares him for a Google SWE role. The official docs and guides are recommended starting points [9] [10], followed by coding practice and project work as outlined above.

---

[1] [5] [8] ArkS0001 (Aakarshit Srivastava) · GitHub
https://github.com/ArkS0001

[2] The Grandmasters are here! | Unstop
https://www.linkedin.com/posts/unstop_unstoppable-e-school-leaders-2025-activity-7309081508219166720-yb6s

3 When innovation meets excellence, it’s called brilliance! | PSIT Kanpur (Pranveer Singh Institute of Technology)
https://www.linkedin.com/posts/psitkanpur_driventoinnovate-psitproud-futureofev-activity-7275168893164244994-tT95

4 Perfect-Cube · GitHub
https://github.com/Perfect-Cube

6 QWhale and SARSAWhale: Energy-Efficient and Energy-Aware Algorithms for High-Load Cloud Environments - IJFMR
https://www.ijfmr.com/research-paper.php?id=34840

7 Crystal Quantum Shield (CQS): A Post-Quantum Cybersecurity Framework for API and Data Protection - IJFMR
https://www.ijfmr.com/research-paper.php?id=37652

9 The Go Programming Language
https://go.dev/

10 Effective Go - The Go Programming Language
https://go.dev/doc/effective_go

11 Using Go Modules - The Go Programming Language
https://go.dev/blog/using-go-modules