The 4 Pillars — Clear instructions (follow exactly)

Pillar 1 — Aptitude (screening & speed)

Why it matters: Aptitude (TCS NQT style) is the common first filter. Fast wins here get you to the technical rounds.

Action steps (do this now): 1. Open the TCS Numerical Ability video and watch once. Note down 6 common shortcuts/formulas.
2. Solve 10 timed numerical problems (30–40 minutes). Repeat until accuracy improves.
3. Watch the TCS Reasoning video; practise 2 reasoning sets daily for 7 days.
4. Watch the TCS Verbal video; practise para-jumbles and one RC daily for 10–15 minutes for two weeks.
5. Do one full timed mock each week and reduce the allowed time per section slowly.

Daily time: 20–45 minutes.
Resources (open & use): - TCS Numerical Ability — https://youtu.be/3obEP8eLsCw - TCS Reasoning Ability — https://youtu.be/dl00fOOYLOM - TCS Verbal Ability — https://youtu.be/IPvYjXCsTg8

Pillar 2 — CS Fundamentals (DBMS | OS | CN | OOPs)

Why it matters: Interviewers test fundamentals to evaluate system design choices and troubleshooting ability.

Action steps (do this now): 1. Watch one focused "one-shot" video for each subject. Pause and write 3–5 flashnotes per topic (commands, key SQL queries, protocol names, OOP principles).
2. Convert notes into one-page cheat-sheets (one page per subject) and save them as images or a single PDF. Keep this on your phone for quick revision.
3. Do two small applied tasks per subject (example: write a sample SQL query for a use-case; draw a process state diagram and explain transitions).
4. Weekly micro-review: 20 minutes per subject per week.

Daily time: 30 minutes (light) or 1 hour, twice weekly for deeper study.
Resources (open & use): - OS One Shot (Love Babbar) — https://youtu.be/7-lK9EpBS_Y - DBMS One Shot (Love Babbar) — https://youtu.be/S-Ji7aayH3A - Computer Networks (Kunal Kushwaha) — https://youtu.be/sWJfscVkhLI - Java OOPs One Shot — https://youtu.be/GDAVuWg0G8M

Pillar 3 — DSA (Pattern recognition — the game)

Why it matters: DSA determines most hiring outcomes. It's not speed alone — it's recognising problem patterns and mapping them to known techniques.

High-level rules: pick one language, pick one sheet, set a guaranteed daily minimum, study editorials deliberately, and cluster learning by weekly themes.

Action steps (do this now): 1. One-shot your coding language: list 10 library functions you'll use frequently (sort, heap operations, deque, bisect/binary_search, hashmap helpers).
2. Choose **one** problem sheet — NeetCode 250, Grind 169, or Striver A2Z — and stick with it.
3. Daily routine: attempt **5 problems/day**. If you're busy, at least 2 problems/day — consistency > volume.
4. Per-problem flow (mandatory):
- Blind attempt for 15–30 minutes.
- If stuck, read the editorial fully and understand the pattern.
- **Copy the solution by hand** (or type it while annotating) and write a one-line summary of the pattern used.
- Re-solve a related problem within 24–48 hours without looking at the solution.
5. Weekly themes: dedicate a whole week to arrays/two-pointers, another to graphs, another to DP, binary search week, etc.
6. After ~100 problems, start timed mock interviews and small contests to build pressure handling.

Daily time: 1.5–4 hours depending on your target.
Measure progress: maintain a GitHub repo with solved problems and a short study log (date, problems, pattern learned).
Resources (open & use): - DSA Playlist (Python) — https://youtu.be/OtYEY2htIjM - NeetCode — https://neetcode.io/ - LeetCode problem list (NeetCode) — https://leetcode.com/problem-list/rabvlt31/ - Striver A2Z — https://takeuforward.org/strivers-a2z-dsa-course/strivers-a2z-dsa-course-sheet-2/

Pillar 4 — Projects (impact & value — don't let it gather dust)

Why it matters: Flagship projects show you can design, ship, and measure impact. Prefer projects that either make money or positively impact users.

Action steps (do this now): 1. Choose at least **2 flagship projects** that solve real problems (campus automation, small business tool, accessibility, monetizable idea).
2. Follow a sprint-based approach:
- Sprint 0 (2–3 days): plan, wireframe, choose stack.
- Sprint 1 (1 week): implement core features + tests.
- Sprint 2 (1 week): polish, deploy, record 2–3 min demo.
- Sprint 3: gather feedback and add one meaningful improvement.
3. Document decisions and tradeoffs: create a one-slide summary (problem, approach, tradeoffs, impact) per project.
4. Put code on GitHub and add a README that tells the interviewer how to run the project.
5. If possible, monetize a feature (paid subscription, small ads, or a paid plugin). If not monetizing, ensure your project helps people — join a club/cell that ships tools that users rely on.

Daily time: depends on sprint; aim to ship an MVP within 2–3 weeks.

Study plan & rituals (daily / weekly / monthly)

Daily (guaranteed): - DSA: 5 problems (or 2 if busy).
- CS fundamentals / Aptitude: 15–20 minutes review.
- Update GitHub and short study log.

Weekly: - One mock interview / timed contest.
- One topic deep-dive week (graphs, DP, etc.).

Monthly: - Ship a mini feature or record a demo for a project.
- Update resume and portfolio.

Join PlacePrep WhatsApp group

Join the student-run **PlacePrep** WhatsApp group where seniors share tips, resources, live problem sessions, and schedule updates. Join here: https://chat.whatsapp.com/J0tZj5JqyXTBjc09AGzFHr

---

Quick checklist (export-ready)

- [ ] Add two flagship projects with README + demo link.
- [ ] Maintain GitHub with solved DSA problems and timestamps.
- [ ] Create a 1-page CS fundamentals cheat-sheet.
- [ ] Record 2–3 minute demo videos for each flagship project.

Useful links (from your material)

TCS Numerical Ability — https://youtu.be/3obEP8eLsCw
TCS Reasoning — https://youtu.be/dl00fOOYLOM
TCS Verbal — https://youtu.be/IPvYjXCsTg8
OS One Shot — https://youtu.be/7-lK9EpBS_Y
DBMS One Shot — https://youtu.be/S-Ji7aayH3A
Networks (Kunal Kushwaha) — https://youtu.be/sWJfscVkhLI
Java OOPs One Shot — https://youtu.be/GDAVuWg0G8M
DSA Playlist (Python) — https://youtu.be/OtYEY2htIjM
NeetCode — https://neetcode.io/
NeetCode list on LeetCode — https://leetcode.com/problem-list/rabvlt31/
Striver A2Z — https://takeuforward.org/strivers-a2z-dsa-course/strivers-a2z-dsa-course-sheet-2/
