📌 WEEK 1 — ENGINEERING MINDSET BOOTCAMP

🟦 DAY 1 — SYSTEM REVERSE ENGINEERING + NODE & TERMINAL (8 hrs)

New Exercises (Different from B01)

1️⃣Create a script **`sysinfo.js`** that prints:

- Hostname
- Available Disk Space (GB)
- Open Ports (top 5)
- Default Gateway (network)
- Logged-in users count

2️⃣Create **3 shell aliases**
Store in `.bashrc`/`.zshrc`:

```
alias gs="git status"
alias files="ls -lha"
alias ports="lsof -i -P -n | grep LISTEN"
```

3️⃣Run Node program + log runtime metrics:

```
process.cpuUsage()
process.resourceUsage()
```

Store in `/logs/day1-sysmetrics.json`

Deliverables

- `sysinfo.js`
- `.bashrc` or `.zshrc` snippet screenshot
- `logs/day1-sysmetrics.json`

🟩 DAY 2 — NODE CLI & CONCURRENCY (8 hrs)

New Exercises

Build a CLI tool:

```
stats.js --lines <file> --chars <file> --words <file>
```

✔ Count total:

- Characters
- Lines
- Words
  ✔ Process **3 files in parallel**
  ✔ Output performance report:

```
{
 "file":"data1.txt",
 "executionTimeMs": 51,
 "memoryMB": 14.3
}
```

Bonus:
Add flag to **remove duplicate lines** and write to:

```
output/unique-<filename>
```

Deliverables

- `stats.js` CLI
- `/logs/performance*.json`
- Output files with uniqueness processing

🟨 DAY 3 — GIT MASTERY (RESET + REVERT + CHERRY-PICK + STASH) (8 hrs)

New Exercises

1️⃣Setup repo with 10 commits:

- Introduce **syntax error** in commit 5
- Use **git bisect** to detect breaking commit → record

2️⃣Create a **release** branch from older commit:

```
release/v0.1
```

3️⃣Use **cherry-pick** to bring only **important changes** from main → release

4️⃣Use **stash** while switching branches + restore cleanly

5️⃣Document:

- Bisect command logs
- Cherry-pick result
- Stash scenario

Deliverables

- `bisect-log.txt`
- `cherry-pick-report.md`
- `stash-proof.txt`
- Commit graph screenshot

🟥 DAY 4 — HTTP / API FORENSICS (cURL + POSTMAN) (8 hrs)

New Exercises

1️⃣Use cURL to fetch **GitHub API**

```
curl -v https://api.github.com/users/octocat
```

Extract & log:

- Rate-limit remaining
- ETag
- Server header

2️⃣Pagination fetch:

```
https://api.github.com/users/octocat/repos?page=1&per_page=5
```

Document:

- Link headers
- Navigating pages

3️⃣ Create POSTMAN collection testing:

- GET GitHub user
- GET repos × 3 pages

4️⃣ Build HTTP Server with:

- `/ping` return timestamp
- `/headers` return request headers
- `/count` maintain counter in memory

Deliverables

- `curl-headers.txt`
- `pagination-analysis.md`
- POSTMAN exported collection `.json`
- `server.js`

🟪 DAY 5 — AUTOMATION & MINI-CI PIPELINE (8 hrs)

New Exercises

1️⃣ Build script: `healthcheck.sh`

- Pings your server every 10s
- Logs failures to `/logs/health.log`

2️⃣ Pre-commit validation:
✔ Ensure `.env` file **does not exist** in Git
✔ Ensure **JS is formatted**
✔ Ensure log files are ignored

3️⃣ Packaging:

```
bundle-<timestamp>.zip
```

Include:

- `src/`
- `logs/`
- `docs/`
- `checksums.sha1`

4️⃣ Cron/Task scheduling:
Every 5 min → run `healthcheck.sh`

Deliverables

- `healthcheck.sh`
- Husky pre-commit hook screenshot (failed & passed)
- `bundle*.zip`
- `checksums.sha1`
- Screenshot of scheduled cron job