NAME: CHAABILO LUBOBYA

COMPUTER NUMBER: 2021418235

LAB 4 DELIVERABLES

1. Assessment summary of both systems

CRITERIA	SYSTEM A	SYSTEM B
System Complexity	High	Moderate
Risk factors	High (no docs, outdated PHP 5, no version control)	Low (good docs, CI/CD, active repo)
Maintainability	Poor (hard to modify, high technical debt)	Good (modular, testable)
Resource Availability	Limited (less PHP 5 experts available)	High (Node.js/React developers prevalent)
Technology Familiarity	Low (except team has PHP experience)	High (new stack, widely adopted)

2. Expert Judgment and Analogy estimates for each

Expert Judgment Estimates

System A (Legacy PHP)

Estimated Effort: 400-500 developer-hours

Reasoning:

- Steep learning curve due to undocumented legacy code.
- Immediate changes require extensive debugging.
- Lack of CI/CD doubles testing time manually.

Assumptions:

- Team has some experience with PHP.
- Risk buffer of +100 hours for unexpected issues.

System B (Node.js/React)

Estimated Effort: 200-300 developer-hours

Reasoning:

- Good documentation reduces onboarding time.
- Modular design allows parallel work.
- CI/CD speeds up testing/deployment.

Assumptions:

- Team is already familiar with Node.js/React.
- Risk buffer of +50 hours for small adjustments.

Analogy-Based Estimation

System A (Legacy PHP)

Comparison:

- Similar to maintaining an old WordPress website with custom plugins (~350 hours in past experience).

Adjusted Estimate: 450 hours (because of added complexity).

System B (Node.js/React)

Comparison:

- Similar to a startup MVP using React frontend (~250 hours on past projects).

Revised Estimate: 275 hours (due to additional features).

3. Team recommendations

What system would you rather keep around and why?

- We would recommend System B (Node.js/React) because there is less risk with better docs and current practices, longer development cycle with CI/CD and More maintainable in the long run.

What assumptions had the most influence over your estimates?

- System A: Assumed some PHP experience in assumed team; otherwise, effort would be higher.
- System B: Assumed no major architectural changes.

Which estimation method appeared more useful or believable?

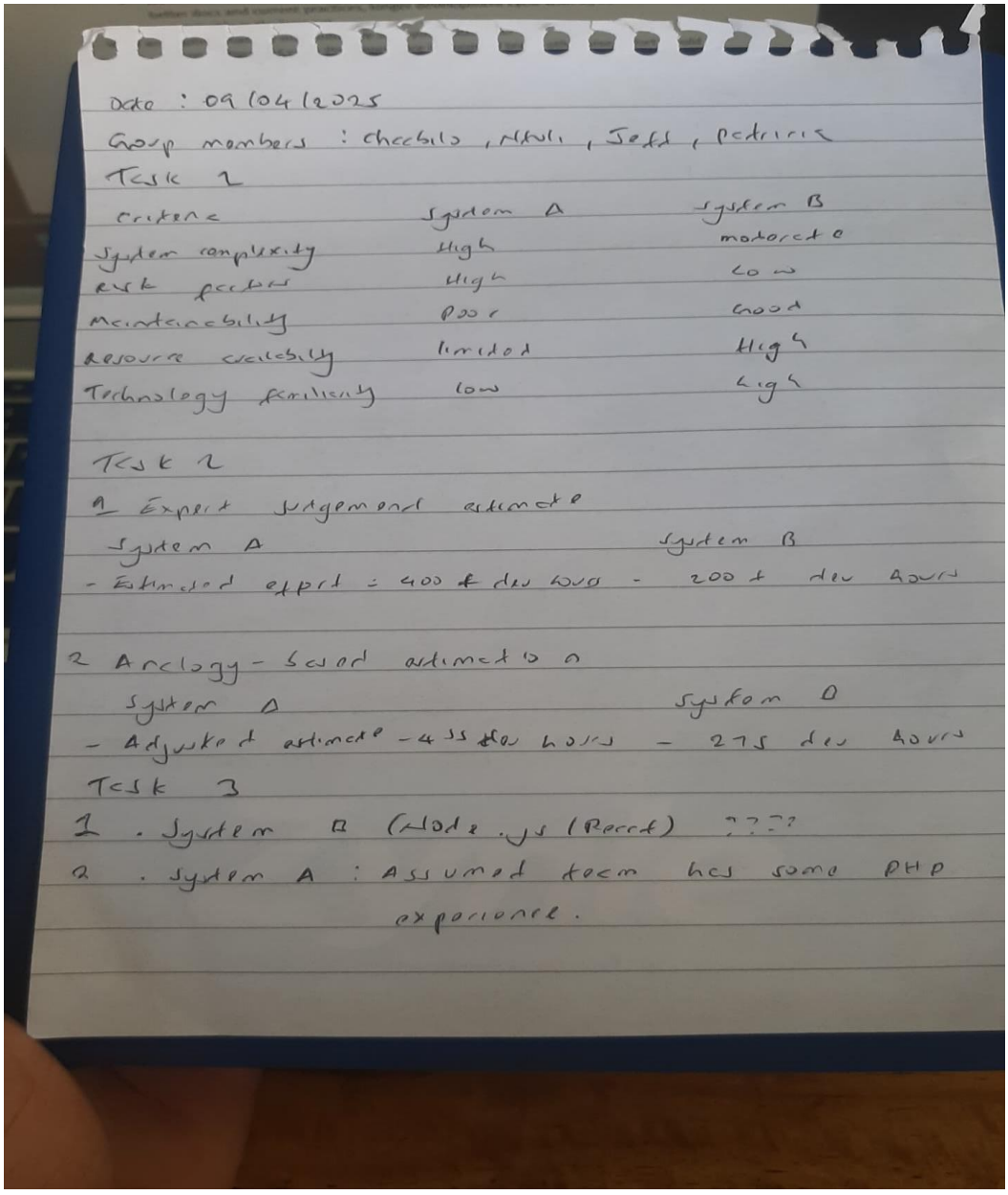
- Expert judgment was more accommodating to deal with unknowns.

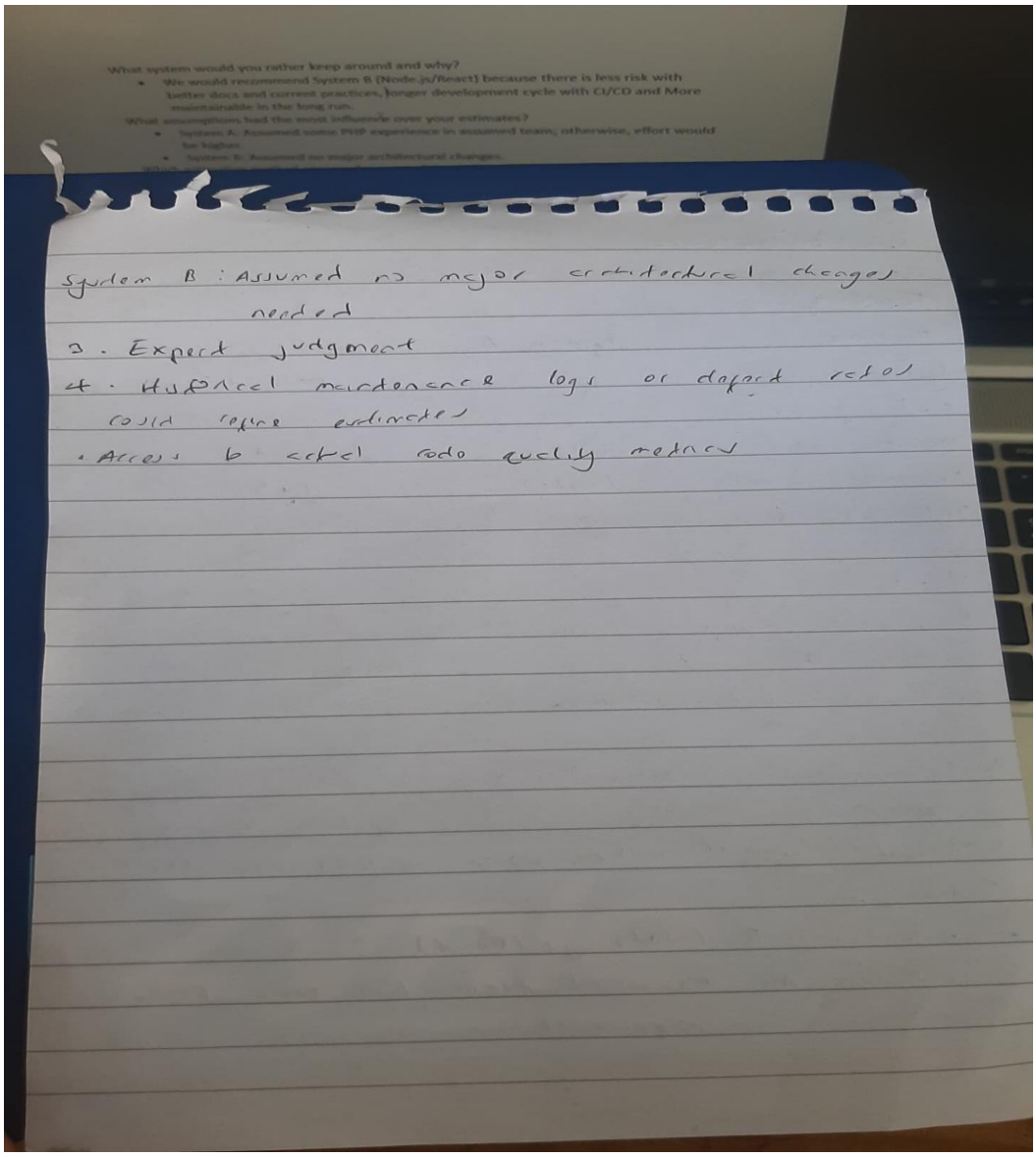
Analogy-based was useful but based on similar previous projects.

How would estimates become better with additional information?

- Historical maintenance logs or defect rates would make estimates more accurate.
- Access to actual code quality metrics (e.g., cyclomatic complexity) would improve accuracy.

4. Photos of planning process




What system would you rather keep around and why?
We would recommend System B (Node.js/React) because there is less risk with
better docs and current practices, longer development cycle with CI/CD and More
maintainable in the long run.
What assumptions had the most influence over your estimates?
System A: Assumed some PHP experience in assumed team; otherwise, effort would
be higher
System B: Assumed no major architectural changes.
System B : Assumed no major architectural changes
needed
3. Expert judgment
4. Historical maintenance logs or defect rates
could refine estimates
• Areas to track code quality metrics