Module 1 — Introduction to Software Engineering

Generated: 2025-11-20

Module 1 — Introduction to Software Engineering

Generated: 2025-11-20

Duration: 2–3 hours (self-study)

Learning objectives:
1. Explain what software engineering is and why discipline matters.
2. Describe the common software development lifecycle stages.
3. Identify core practices and tools used in modern engineering teams.
4. Apply a simple checklist to start a small project.

Module overview:
- What is software engineering?
A disciplined, engineering approach to designing, building, testing, deploying, and maintaining software reliably at scale.

- Key concepts (short):
• Requirements gathering and communication
• Design & architecture (abstraction, modularity)
• Implementation (coding standards, readability)
• Testing (unit, integration, end-to-end)
• Deployment & operations (CI/CD, infra as code)
• Monitoring & maintenance (observability, incident handling)
• Security and privacy-by-design

Lessons
1. Why discipline matters
- Costs of poor engineering: technical debt, outages, slow delivery.
- Benefits: predictability, maintainability, faster iteration.
- Quick activity: list 3 failure modes you've seen or can imagine.

2. The lifecycle (concise)
- Stages: Requirements → Design/Architecture → Implementation → Testing → Deployment → Maintenance.
- Common methodologies: Agile/Iterative, DevOps culture, SRE ideas.

3. Core practices & tools
- Version control (git): branching, commits, PRs, code review.
- Automated testing: unit tests, integration tests, test coverage.
- CI/CD pipelines: run tests, build artifacts, deploy to staging/production.
- Infrastructure as Code: declarative infra, reproducible environments.
- Observability: logs, metrics, traces, alerting.

4. Intro to system design (bite-sized)
- Decompose features into services/modules.
- Think about data flow, contracts (APIs), scalability, failure modes.
- Sketch a simple 3-component diagram for any feature.

Exercises (recommended)
- Exercise A: Write a 1-paragraph requirement for a tiny feature (e.g., "save drafts in a note app").
- Exercise B: Create a git repo, add a README with the requirement, commit a basic project scaffold.
- Exercise C: Write two unit tests for one small function; run them locally or in a simple CI (GitHub Actions example).

Quick checklist to start a small project
- Define a single, testable user story.
- Create a repo and add README + license.
- Add a basic CI that runs tests.

- Add at least 1 unit test and 1 integration test.
- Add monitoring/health endpoint for the service.
- Schedule a 1-hour design review with a teammate.

Further reading (concise)
- "Clean Code" — Robert C. Martin (practical coding practices)
- "Designing Data-Intensive Applications" — Martin Kleppmann (systems thinking)
- Official Git docs and GitHub/GitLab CI guides

End of Module 1 — concise introduction.