Software Engineering Ethics

Software engineers make decisions that affect users, clients and society. This chapter introduces professional codes of ethics and the legal and social obligations (privacy, licences, accessibility, security) that shape everyday engineering choices, applied to ITC Club Hub.

ACM/IEEE-CS SE Code 5.2 · ACM Code 2018 | GDPR principles · WCAG 2.2 | MIT · Apache 2.0 · GPL-3.0 | C++20 · ITC Club Hub

AGENDA

What we will cover

1. Why ethics matters in software engineering; professional responsibility
2. The ACM/IEEE-CS Software Engineering Code of Ethics: its eight principles
3. The ACM Code of Ethics and Professional Conduct
4. Privacy and data protection: data minimisation, consent, retention (Cambodian regulations and GDPR as an international reference)
5. Intellectual property, copyright and open-source licences (MIT, Apache 2.0, GPL)
6. Accessibility and inclusive design (WCAG 2.2)
7. Security responsibilities and responsible disclosure
8. Ethics of AI and data: bias, transparency, dark patterns
9. Case studies: Therac-25, the Volkswagen emissions software, Cambridge Analytica
10. A framework for ethical decision-making and whistleblowing
11. Chapter Quiz
12. Lab-03

Click any item to jump directly to that topic.

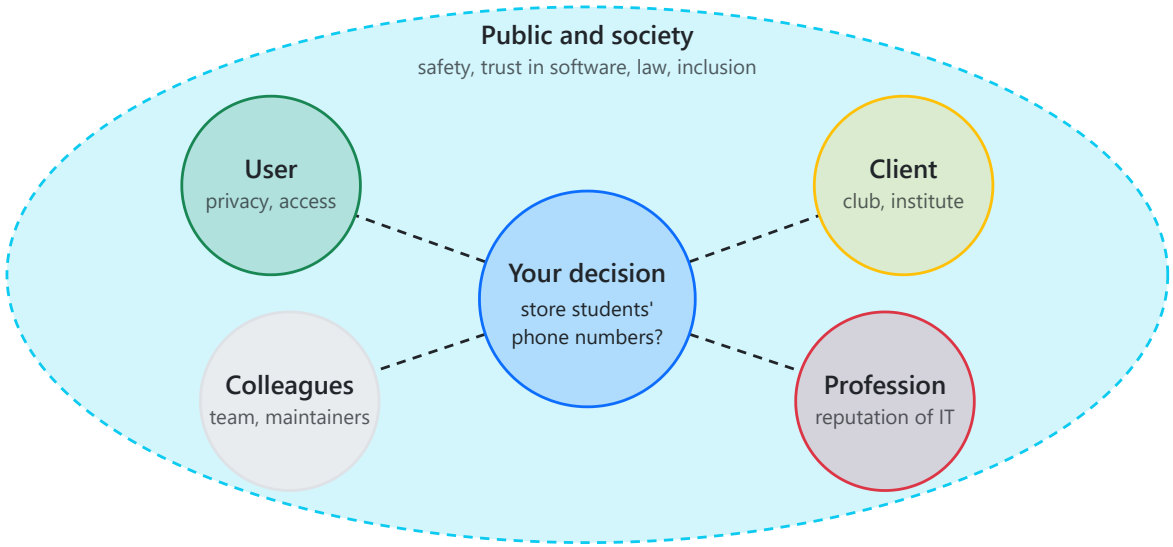
Ethics is not a separate phase: it hides inside ordinary tickets

A normal Club Hub task	The ethical question inside it	Who is affected	Topic
Add a phone field to registration	Do we need it? Is it optional? Who will see it?	Students	Privacy
Export attendance to CSV	May a sponsor receive it? Can the file run formulas?	Students, club	Privacy, security
Copy a date-parsing function from GitHub	What does its licence ask of us?	Original author, team	Licences
Show "full" events in red	Can a colour-blind student read the status?	Students with disabilities	Accessibility
Recommend events to students	Is the ranking fair and explainable?	New and part-time students	AI and data
Demo on Friday with a known bug	Do we tell the client?	Client, team, users	Code of ethics

🕮 **Ethics**: reasoned principles for deciding what is right when choices affect other people. **Professional ethics**: the obligations a profession accepts in return for the trust society gives it. SWEBOK Guide v4.0 places both in the knowledge area *Software Engineering Professional Practice*.

- Most ethical problems in software are not dramatic: they are small defaults, missing checks and silent shortcuts that add up.
- Codes of ethics, laws and standards (GDPR, WCAG 2.2, open-source licences) give you shared vocabulary so that "this feels wrong" becomes "this breaks principle 1.03".
- Today every rule is applied to **ITC Club Hub**, the system your team builds this semester.

One decision, many people: the stakeholders of a software choice



- **User** (students): lose control of their number, receive spam, may be harassed if a list leaks.
- **Client** (the club, the institute): liable and embarrassed after a leak; loses members' trust.
- **Colleagues**: must secure, back up and one day delete data they never needed.
- **Profession**: every leak makes people trust "IT people" a little less.
- **Public**: data habits of small apps shape what society accepts as normal.

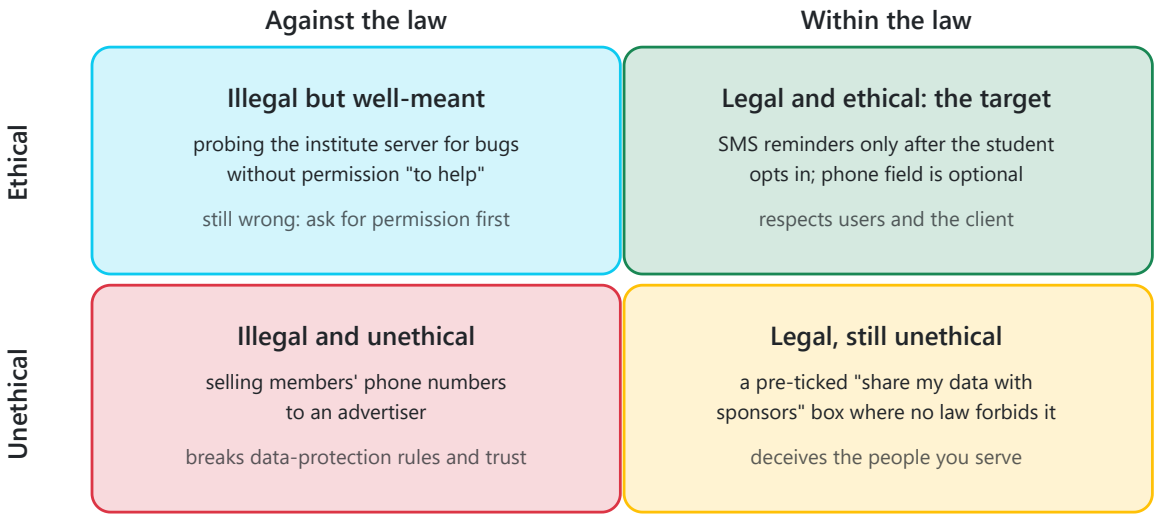
⚠ Software scales harm: one careless default in a form is copied to every one of the 2000+ students who register.

Professional responsibility: legal is the floor, not the ceiling

A **profession** has specialised knowledge that the public cannot check for itself, so it promises to use that knowledge responsibly. For a software engineer that means:

- **Competence**: accept only work you can do or can learn in time; say so when you cannot.
- **Due care**: test, review and think about misuse before you ship.
- **Honesty**: report bugs, risks and delays, even when the news is unwelcome.
- **Accountability**: your name is on the commit; "the client asked for it" is not a defence for harm.

> 💡 Law changes slowly and differs by country; ethics asks what is right *here and now*. Aim for the green quadrant.



The Software Engineering Code: eight principles version 5.2, 1999

- Written by a joint task force of the **ACM** and the **IEEE Computer Society**; it is the code aimed specifically at people who build software.
- A **short version** states the eight principles; the **full version** adds about 80 numbered clauses, cited like *1.03* (principle 1, clause 3).
- The principles are ordered: when obligations conflict, the **public interest** comes first. Serving your client never justifies harming the public.
- The code is a guide for judgment, not an algorithm: it tells you what to weigh, you still have to decide.

> ❝ Source: ACM/IEEE-CS Software Engineering Code of Ethics and Professional Practice, version 5.2. Wording on this slide is paraphrased.

Worked example: what each principle means for your Club Hub team

#	Principle	In one line (paraphrased)	Concrete Club Hub behaviour
1	Public	Act consistently with the public interest	Event capacity matches the room's real safety limit, not what the club wishes
2	Client and employer	Serve them well, as long as the public is not harmed	Keep a club's member list confidential; warn the Product Owner early when a deadline slips
3	Product	Meet the highest professional standards you can	Unit-test the 24-hour cancellation rule; never hide a known defect
4	Judgment	Keep integrity and independence	Do not approve a pull request you did not read or run
5	Management	Lead development ethically	The Scrum Master gives realistic estimates and never asks the team to hide bugs
6	Profession	Advance the integrity and reputation of the profession	Respect licences: credit GoogleTest; never paste code of unknown origin
7	Colleagues	Be fair to and supportive of colleagues	Review code, not people; credit teammates' commits in the report
8	Self	Keep learning; practise ethically	Read WCAG 2.2 before designing the web UI

Conflict example. The club (client, principle 2) wants 80 seats for a talk in a room that holds 50. Principle 1 wins: the system enforces 50 and offers a waiting list.

When you cite the code in a lab, name the principle and explain the link to your decision in one sentence: "Principle 3 Product: we add a test for the duplicate-registration rule because a double booking blocks another student."

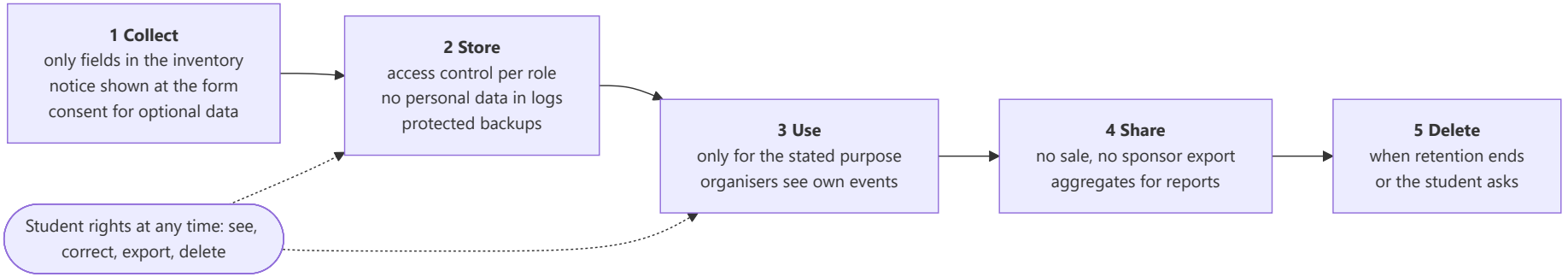
The ACM Code: four sections, for every computing professional

Section	Items	What it covers (paraphrased)	Club Hub example
1 General ethical principles	1.1 to 1.7	Contribute to society and human well-being; avoid harm; be honest and trustworthy; be fair and do not discriminate; respect the work behind ideas and software; respect privacy; honour confidentiality	1.6: collect only what the inventory justifies; 1.4: the web UI must work for blind students too
2 Professional responsibilities	2.1 to 2.9	Quality of work and process; competence; knowing the rules that apply; professional review; evaluating risks; working within competence; public awareness; authorised access only; systems that are robustly and usably secure	2.4: every change goes through a pull request review; 2.9: passwords hashed, CSV exports safe
3 Professional leadership	3.1 to 3.7	Public good as the central concern; social responsibility; quality of working life; policies that reflect the code; helping others grow; care when changing or retiring systems; special care for infrastructure	3.6: when Club Hub replaces the paper sign-up sheet, migrate and then destroy old lists
4 Compliance with the code	4.1 to 4.2	Uphold and promote the code; treat violations as incompatible with ACM membership	Raise a concern in the retrospective instead of staying silent

	SE Code	ACM Code
Audience	Software engineers	All computing professionals
Version	5.2 (1999)	2018 update
Shape	8 principles, ~80 clauses	4 sections, 25 items
Newer topics	Quality, management	Discrimination, security, infrastructure

The two codes agree on the core: **public good first, avoid harm, be honest, respect privacy, do competent work**. Use whichever gives the sharper sentence for your case.

Personal data has a lifecycle, and every step carries an obligation



Personal data: any information about an identifiable person. In Club Hub: student ID, name, email, phone, club membership, attendance records.

Data minimisation: collect and keep only what is necessary for a stated purpose. The safest data is the data you never stored.

Retention: a written limit on how long each field is kept, followed by real deletion (including exports and backups).

The legal landscape: Cambodia is evolving, GDPR is the reference

GDPR principle (Art. 5)	Meaning for Club Hub
Lawfulness, fairness, transparency	A plain-language notice at registration; no hidden uses
Purpose limitation	Emails collected for reminders are not reused for sponsor marketing
Data minimisation	No date of birth, no home address, phone only if SMS is wanted
Accuracy	Students can correct their name and email themselves
Storage limitation	Identified attendance kept 12 months, then aggregated
Integrity and confidentiality	Role-based access, protected files, redacted logs
Accountability	A written data inventory and retention rules in the repository

Lawful bases (Art. 6) include consent, contract, legal obligation, vital interests, public task and legitimate interests. **Valid consent** is freely given, specific, informed and unambiguous (an active choice, never a pre-ticked box) and as easy to withdraw as to give. Source: Regulation (EU) 2016/679 (GDPR).

Cambodia

- The data-protection landscape is **evolving**.
- The **Law on Electronic Commerce (2019)** contains provisions on protecting personal data held in electronic form, which covers a system like Club Hub.
- A dedicated, comprehensive personal data protection law has been in preparation.
- The institute's own rules on student records also apply.

Why GDPR as the reference?

It is the most widely copied model: many newer national laws reuse its principles. It binds organisations serving people in the EU; here we use it as a **design standard**: if Club Hub meets GDPR principles, it is well placed for whatever Cambodian law requires.

Lecturer note: check the current status of a dedicated Cambodian personal data protection law before this class and update this slide if it has been adopted.

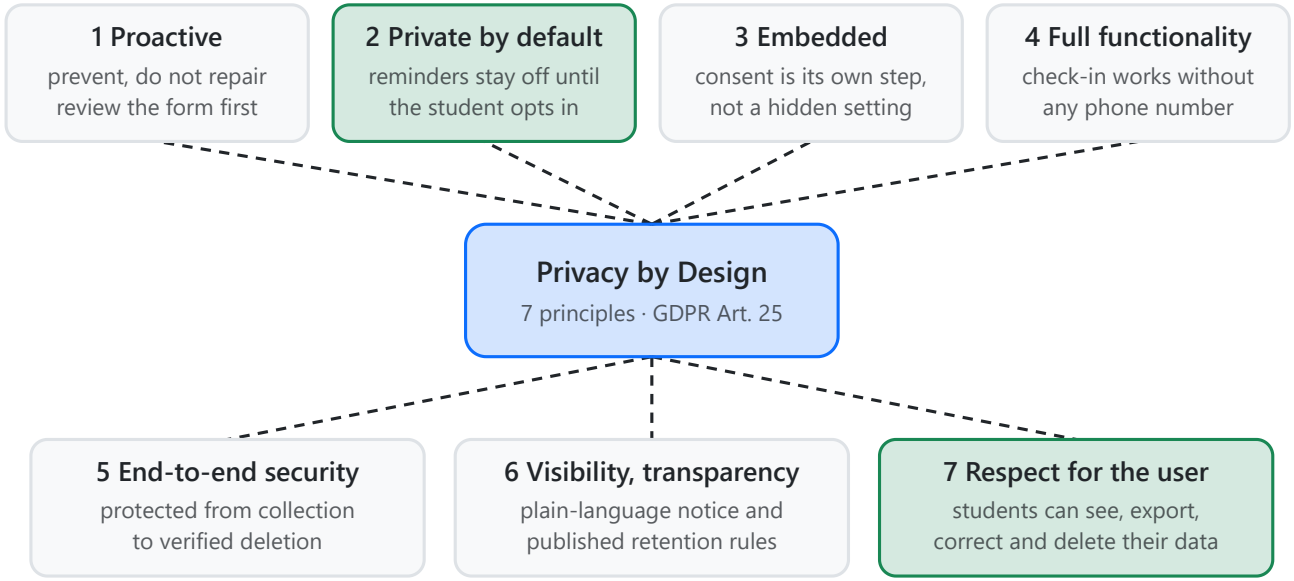
Worked example: a data inventory for ITC Club Hub

Field	Purpose	Legal basis / consent	Retention	Who can see it
`studentId`	Identify the student; block double registration	Necessary for the service	While the account is active, deleted 6 months after graduation	The student; admins; organisers see it **masked** (`e2******7`)
`name`	Check-in list at the door	Necessary for the service	As `studentId`	The student; organisers of events the student registered for; admins
`email`	Login; reminders	Necessary for login; **reminders by opt-in consent**	As `studentId`	The student; admins. Not organisers
`phone` (optional)	SMS reminder, only if chosen	**Consent**, withdrawable at any time	Deleted at once when consent is withdrawn	Nobody on screen; used only by the reminder sender
`clubMemberships`	Show "my clubs"; leaders manage members	Necessary for the service	Until the student leaves the club, + 30 days	The student; leaders of that club; admins
`attendance`	Check-in; monthly activity report	Legitimate interest of the institute (documented)	Identified 12 months, then aggregated to counts	Organisers (own events only); admins see **aggregates**
~~`dateOfBirth`~~, ~~`address`~~, ~~`photo`~~	No purpose found	None	Not collected	Nobody

💡 Build the inventory **before** you design the `Student` class: every member variable must have a row, and every row needs a purpose. A field without a purpose is deleted.

⚠ The inventory covers copies too: CSV files, exports, backups, log files and test fixtures. Test data uses invented names, never real students.

Privacy by design: build it in, do not bolt it on



- The seven principles come from Ann Cavoukian (Privacy by Design, 2009). GDPR Art. 25 turned the idea into a legal duty: *data protection by design and by default*.
- **By default** is the most powerful: most users never change settings, so the default *is* the policy.
- **Positive-sum** (principle 4): privacy and features are not a trade-off; the check-in feature works perfectly without phone numbers.

In code: optional data is `std::optional`, consent flags start as `false`, and logging functions only accept redacted text.

Data minimisation in code: store less, log nothing personal

```cpp
// include/clubhub/Student.h (excerpt)
namespace clubhub {

// Only the fields that the data inventory allows
struct Student {
    std::string id;      // institute ID, e.g. "e20230417"
    std::string name;
    std::string email;
    // present only if the student opted in to SMS
    std::optional<std::string> phone;
};

// "e20230417" -> "e2******7": recognisable, not reusable
inline std::string maskId(std::string_view id) {
    std::string out{id};
    if (out.size() <= 3) return std::string(out.size(), '*');
    for (std::size_t i = 2; i + 1 < out.size(); ++i)
        out[i] = '*';
    return out;
}

// The ONLY form of a Student that may reach a log file
inline std::string redacted(const Student& s) {
    return "Student{id=" + maskId(s.id) + "}";
}
}  // namespace clubhub
```

```cpp
// Monthly report row: a pseudonym, no name, no phone
struct AttendanceRow {
    std::size_t studentKey;   // keyed hash of the ID
    std::string eventId;
};

// Demo only: std::hash is NOT a cryptographic hash.
// Production: HMAC-SHA-256 from a crypto library.
inline std::size_t pseudonym(std::string_view id,
                             std::string_view secret) {
    std::string key{secret};
    key += id;
    return std::hash<std::string>{}(key);
}

// BAD: name and phone end up in a shared log file
// log << "registered " << s.name << " " << *s.phone;

// GOOD: masked ID and the event, nothing else
log << "registered " << redacted(s) << " for E-017\n";
```

```
registered Student{id=e2******7} for E-017
```

The admin report counts attendance per club with `AttendanceRow`: it needs to know that the *same* student came twice, not *who* the student is. Keep the secret out of Git (environment variable or local config file).

Worked example: a privacy notice in plain language, and a fair consent step

```
# How ITC Club Hub uses your data (v1.0, September 2026)

**Who we are.** Club Hub is run by the ITC student clubs office.
Contact: clubhub-privacy@itc.example (reply within 7 days).

**What we collect and why.**
- Student ID and name: to register you and check you in.
- Email: to log you in. Reminders only if you switch them on.
- Phone (optional): only for SMS reminders you ask for.
- Attendance: to run events and a monthly report per club.

**Who sees it.** Organisers see your name and a masked ID for
events you registered for. We never sell or give your data to
sponsors or advertisers.

**How long.** Attendance with your name: 12 months, then only
counts remain. Account: until 6 months after you graduate.

**Your choices.** See, correct, download or delete your data
at any time in *My profile*. Withdraw SMS consent in one click.
```

Registration screen: consent step (mock-up)

Your ID, name and email are needed to use Club Hub. Read how we use them.

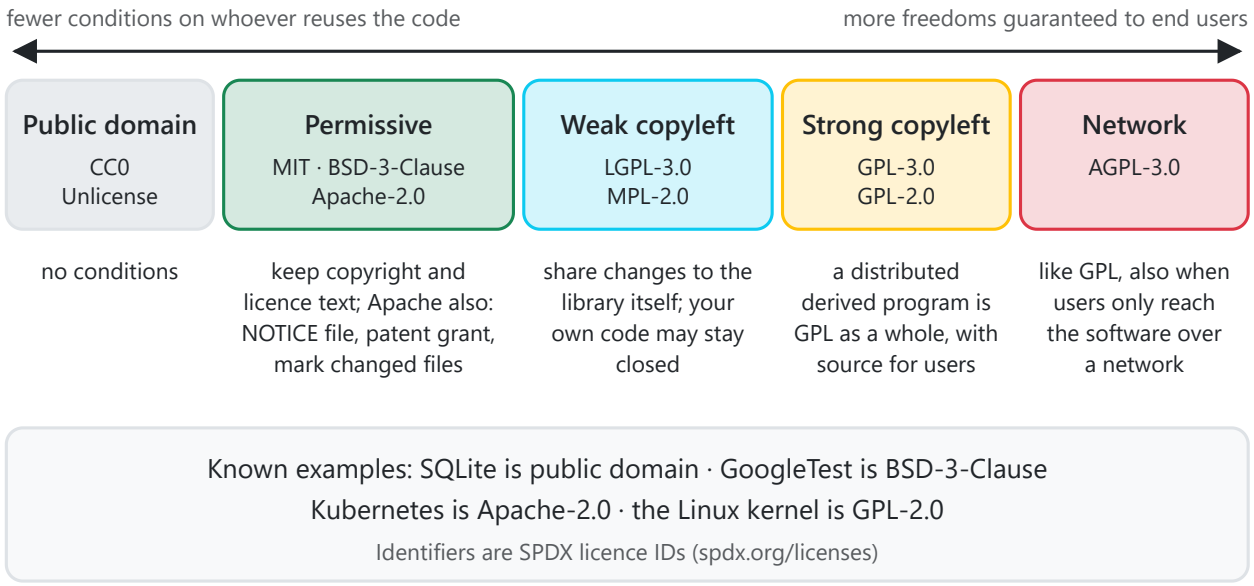
- ☐ Email me a reminder 24 hours before my events
- ☐ Also send SMS reminders (you will be asked for a phone number)

Both boxes start unticked. "Create account" works with neither ticked.

- **Layered**: a one-line summary on the form, the full notice one click away.
- **Separate**: one checkbox per purpose; no "I agree to everything".
- **Not a condition**: optional consent never blocks registration.
- **Readable**: short sentences, "we" and "you", no legal jargon; offer a Khmer version.

Copyright is automatic; a licence is the permission you give or receive

- **Copyright** protects the expression (the code you wrote) from the moment you write it; no registration needed. Ideas and algorithms are not covered.
- **No licence = no permission.** Public code on GitHub without a LICENSE file may be read, but not legally copied into your project.
- An **open-source licence** grants everyone the right to use, modify and share, under conditions.
- **Permissive**: few conditions (keep the notice). **Copyleft**: derived works must stay open under the same licence.
- Other IP: **trademarks** (names, logos), **patents** (inventions), **trade secrets**.



Known examples: SQLite is public domain · GoogleTest is BSD-3-Clause
Kubernetes is Apache-2.0 · the Linux kernel is GPL-2.0

Identifiers are SPDX licence IDs (spdx.org/licenses)

What a licence allows and requires, and how to apply it to a repository

	MIT	Apache 2.0	GPL-3.0
Commercial use, modify, distribute	yes	yes	yes
Explicit patent grant	no	yes	yes
Keep copyright and licence text	yes	yes (+ NOTICE file)	yes
State changes in modified files	no	yes	yes
Release source of derived work when distributed	no	no	**yes, same licence**
Can be combined into closed-source products	yes	yes	no
Warranty or liability	none: "as is" in all three		

💡 **Choosing.** MIT: simplest, maximum reuse. Apache 2.0: like MIT plus patent protection, common in companies. GPL-3.0: you want every improvement to stay open.

⚠ Compatibility runs one way: MIT and Apache-2.0 code may go *into* a GPL-3.0 project, but GPL code may not go into an MIT or closed project.

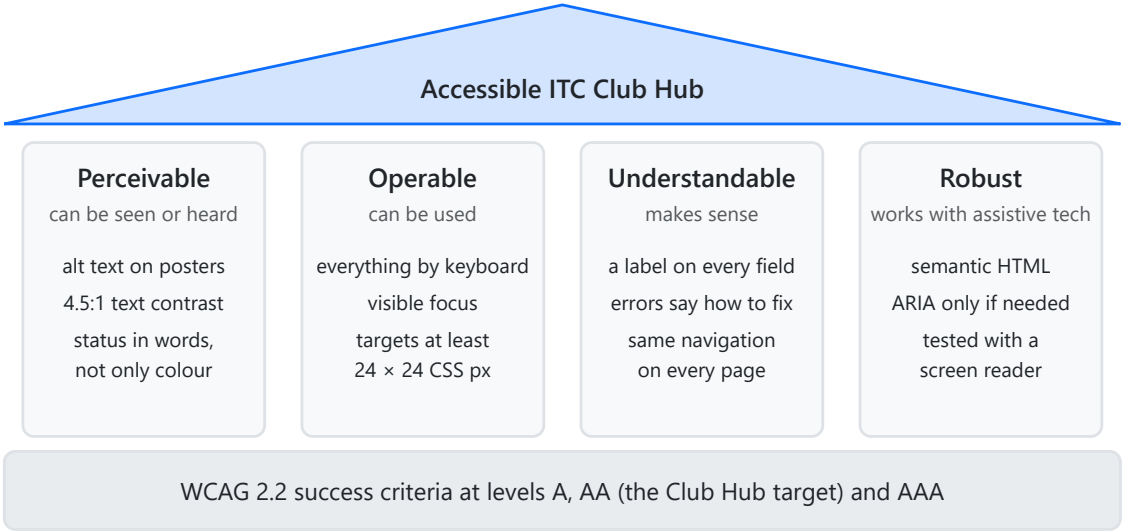
The first two lines of every source file, for example `src/RegistrationService.cpp`:

```
// SPDX-License-Identifier: MIT
// Copyright (c) 2026 Team Mekong, ITC Club Hub contributors
```

```
# Third-party notices

ITC Club Hub uses the following components.

| Component  | Version | Licence      | Used for | Shipped? |
|------------|---------|--------------|----------|----------|
| GoogleTest | 1.15.2  | BSD-3-Clause | tests    | no       |

## GoogleTest
Copyright 2008, Google Inc. All rights reserved.
Source: https://github.com/google/googletest
Full licence text: see licenses/googletest-LICENSE.txt
```

Repository files: `LICENSE` (full text of the chosen licence, at the root), the SPDX header in each source file, and `THIRD_PARTY_NOTICES.md` listing every dependency. GitHub detects `LICENSE` and shows the licence on the repository page.

WCAG 2.2: four principles, three conformance levels



- **Accessibility**: people with disabilities (vision, hearing, motor, cognitive) can use the product. **Inclusive design** widens this to age, language, slow networks and old phones.
- WCAG 2.2 (W3C, 2023) groups testable **success criteria** under the four **POUR** principles. Level **AA** is the usual legal and contractual target.
- New in 2.2, for example: *2.4.11 Focus Not Obscured*, *2.5.8 Target Size (Minimum)*, *3.3.8 Accessible Authentication* (no memory puzzles at login).

💡 The "curb-cut effect": captions, contrast and clear errors help everyone, including a student reading Club Hub on a phone in bright sun.

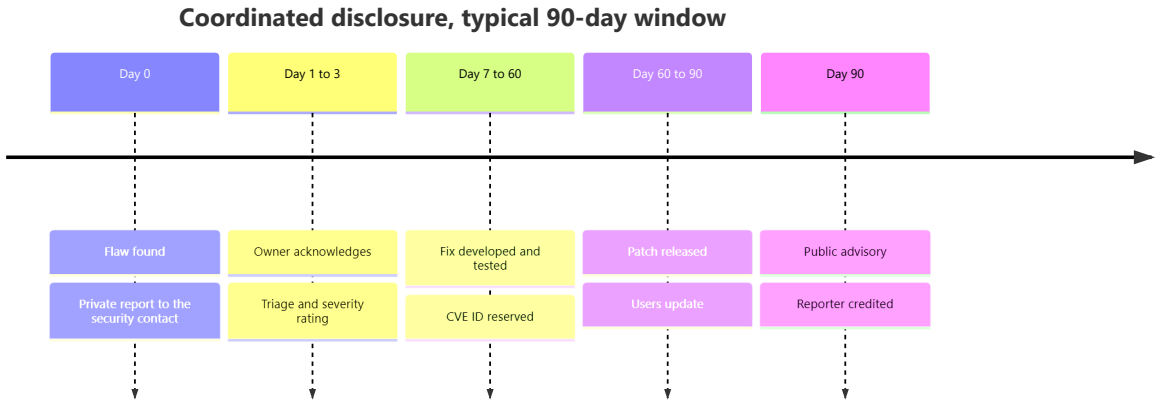
Worked example: accessibility checklist for the CLI and the planned web UI

Check	C++ command-line front end	Planned web / mobile UI	WCAG 2.2
Colour is never the only signal	Status as a word: `[FULL]`, `[CANCELLED]`; honour `NO_COLOR`	Icon + text for status; links underlined	1.4.1 (A)
Contrast	Use the terminal's default colours; no dark blue on black	Text 4.5:1, large text and UI parts 3:1	1.4.3, 1.4.11 (AA)
Keyboard	Menus accept typed numbers or commands, not only arrow keys	Tab order follows layout; visible focus not hidden under a sticky header	2.1.1 (A), 2.4.7, 2.4.11 (AA)
Labels and instructions	Prompts give the format: `Date (YYYY-MM-DD):`	Every input has a `<label>`; required fields marked in text	1.3.1, 3.3.2 (A)
Clear errors	Say what failed, why, and what to do next; non-zero exit code	Error next to the field, with a suggestion	3.3.1 (A), 3.3.3 (AA)
Target size	not applicable	Buttons at least 24 × 24 CSS px	2.5.8 (AA)
Login	Allow pasting the password	Password managers work; no puzzle	3.3.8 (AA)

```cpp
// The word carries the meaning; colour repeats it
std::string_view statusLabel(RegistrationStatus s) {
    switch (s) {
    case RegistrationStatus::Registered:
        return "REGISTERED";
    case RegistrationStatus::Waitlisted:
        return "WAITLISTED";
    case RegistrationStatus::Cancelled:
        return "CANCELLED";
    case RegistrationStatus::CheckedIn:
        return "CHECKED IN";
    }
    return "UNKNOWN";
}

// no-color.org: any NO_COLOR value disables colour
bool colourEnabled(bool noColorFlag) {
    return !noColorFlag &&
           std::getenv("NO_COLOR") == nullptr;
}
```

```
Error: cannot register for "Git Workshop" (E-017):
the event is full (40 of 40 seats).
Next step: clubhub waitlist E-017
```

Responsible (coordinated) disclosure: fix first, then tell the world



As a finder

Report privately to the owner with steps to reproduce. Access or copy no more data than needed to show the flaw. Do not publish before the agreed date.

As an owner

Publish a contact in `SECURITY.md`, acknowledge quickly, fix, credit the reporter, and never threaten people who report in good faith.

Deadlines and CVE

90 days is a widely used default (for example Google Project Zero); flaws already exploited get shorter deadlines. A **CVE** ID names one vulnerability so everyone talks about the same bug.

Testing a system you do not own without written permission is unauthorised access (ACM Code 2.8), even with good intentions. Responsibility starts with your own code: *robustly and usably secure* (ACM 2.9).

Security is part of the job: three duties in Club Hub's code and repository

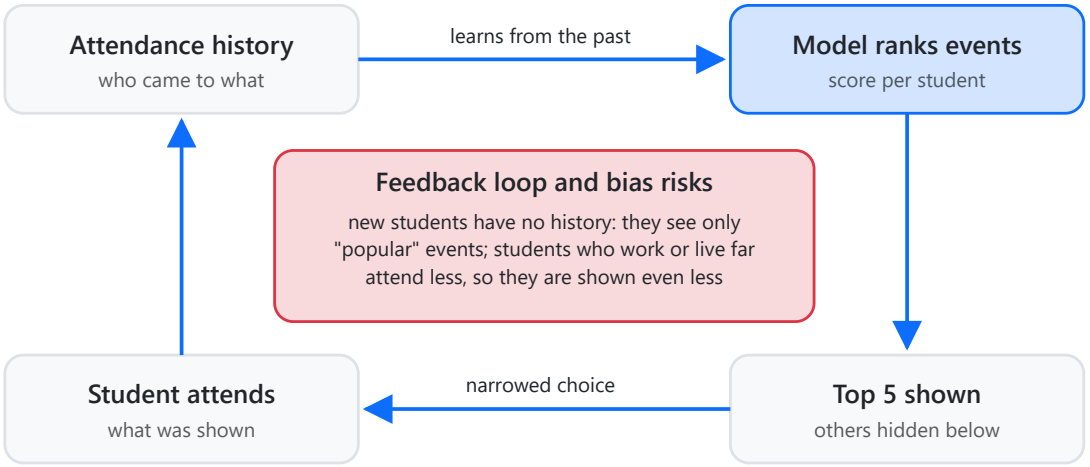
```cpp
// Attendance export: cells that start with = + - @ tab
// or CR run as formulas when an organiser opens the file
// in a spreadsheet (CSV injection). Neutralise them.
std::string csvCell(std::string_view raw) {
    constexpr std::string_view risky{"=+-@\t\r"};
    std::string out = "\"";
    if (!raw.empty() &&
        risky.find(raw.front()) != std::string_view::npos)
        out += '\'';
    for (char c : raw) {
        // RFC 4180: a quote inside a field is doubled
        if (c == '"') out += '"';
        out += c;
    }
    return out + '"';
}
```

```
csvCell("Git Workshop")      -> "Git Workshop"
csvCell("=HYPERLINK(\"x\")") -> "'=HYPERLINK(""x"")"
```

```
# Security policy (SECURITY.md)

## Reporting a vulnerability
Please do not open a public issue. Email
clubhub-security@itc.example with steps to reproduce.
We reply within 3 working days and aim to fix
within 30 days. We credit reporters who wish it.

## Scope
The C++ core and CLI in this repository. Do not test
against real student data or systems you do not own.
```

Duty	Club Hub practice
Validate all input	Reject malformed IDs and dates; escape CSV output
Protect secrets	No passwords or keys in Git; `.gitignore` the local config
Keep dependencies current	Pin GoogleTest to a release; update when advisories appear

Bias and transparency: a harmless-looking recommender can lock students out



- **Bias**: systematic unfairness to a group. It enters through *data* (who is missing), *design* (what is optimised) and *feedback* (outputs become new inputs).
- **Transparency**: people can tell that a system decides, and why. Club Hub: a "Why am I seeing this?" line on each recommended event.
- **Contestability and choice**: a switch to "show all events by date" and an opt-out of profiling.
- Ask before building: does this need personal data at all? A "most popular this week" list may be enough.

The lab challenge asks your team for a go/no-go decision on exactly this feature.

Dark (deceptive) patterns: interfaces that trick users against their interest

Pattern	What it does	Club Hub temptation	Fair alternative
Pre-ticked consent	Assumes "yes" unless the user notices	"Share my profile with sponsors" ticked by default	Unticked, separate box; not needed to register
Confirmshaming	Guilt-trips the "no" option	"No thanks, I don't care about my club"	Neutral: "Not now"
Roach motel	Easy to get in, hard to get out	One-click sign-up, deleting the account needs an email to an admin	"Delete my account" in *My profile*, same effort as sign-up
Nagging	Repeats a request until the user gives in	A phone-number pop-up on every login	Ask once; "Don't ask again" is respected
Fake urgency or scarcity	Invents pressure	"Only 2 seats left!" when 30 are free	Show the real count: "12 of 40 seats left"
Disguised ads	Advertising dressed as content	A sponsor promotion styled as a club event	Clearly labelled "Sponsored"
Trick questions	Confusing wording, double negatives	"Untick to not stop receiving reminders"	Plain positive wording: "Send me reminders"

🕮 The term "dark patterns" was coined by UX designer Harry Brignull in 2010; many regulators now say "deceptive patterns". Under GDPR, consent obtained through such tricks is not valid consent.

⚠ Dark patterns usually improve a metric (sign-ups, consent rate) for a sprint and destroy trust for years. SE Code principle 1 and ACM 1.3 (be honest) rule them out.

Three failures, three violated principles

Therac-25 (1985 to 1987)

A computer-controlled radiation therapy machine gave massive overdoses in six known accidents in Canada and the United States; several patients died. The new model had dropped the hardware safety interlocks of its predecessors and trusted software reused from them. A race condition, triggered when a fast operator edited the treatment settings, let the beam fire in the wrong mode. Error messages such as "MALFUNCTION 54" were cryptic, and early reports from hospitals were dismissed.

Principle 3 Product · Principle 1 Public

Lesson: safety-critical software needs independent safeguards, testing of timing, and honest follow-up of incident reports. (Leveson and Turner, 1993)

Volkswagen emissions software (2015)

US regulators revealed that diesel cars sold from 2009 contained engine-control software that recognised the conditions of an official emissions test and switched on full exhaust treatment only then. On the road, nitrogen-oxide emissions were many times the legal limit. About 11 million cars worldwide were affected; the company paid tens of billions of dollars in fines, recalls and settlements, and an engineer was sentenced to prison in the US.

Principle 1 Public · Principle 4 Judgment

Lesson: "my manager asked for it" does not transfer responsibility. Engineers wrote and maintained code whose only purpose was to deceive.

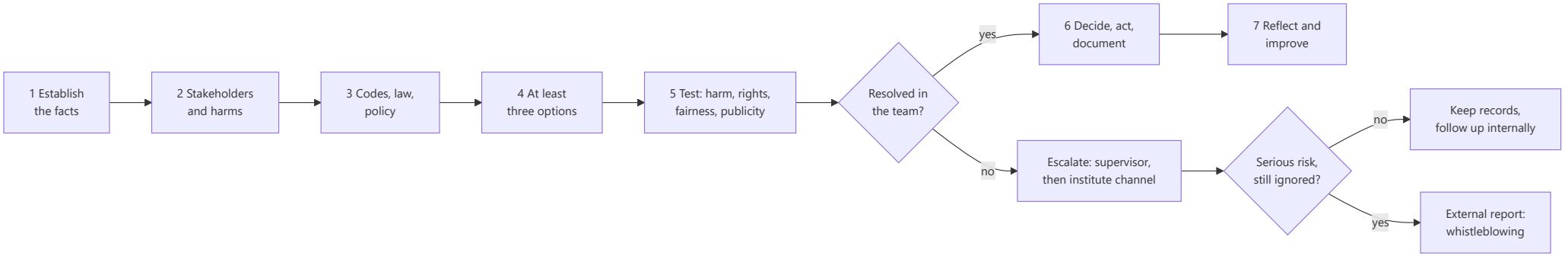
Cambridge Analytica (revealed 2018)

A personality-quiz app on Facebook collected data from the roughly 270,000 people who used it and, through the platform's friends feature, from their friends: up to 87 million people in total, almost none of whom had consented. The data was passed to the political consultancy Cambridge Analytica and used to profile voters for targeted political advertising. In 2019 the US Federal Trade Commission fined Facebook 5 billion dollars.

Principle 1 Public · ACM 1.6 Respect privacy

Lesson: purpose limitation and consent matter; an API that exposes friends' data is an ethical design decision.

A seven-step framework, with an escalation path



Publicity test

Would you be comfortable if the decision were on the front page, or explained face to face to the student it affects?

Third option

It is rarely just "obey or refuse". Look for the option that meets the legitimate need without the harm.

Whistleblowing

Reporting serious wrongdoing to someone who can act. Internal channels first, dated written records, facts not rumours, and no leaking of personal data while doing it.

SE Code 6.12 and 6.13

Paraphrased: first raise a concern with the people involved; report a significant violation to the appropriate authorities when that is impossible, counter-productive or dangerous.

Worked case: "Export all members' phone numbers for our sponsor"

Step	Analysis
1 Facts	The Robotics Club president asks the developer on duty for a CSV of all 120 members' phone numbers so that a phone-shop sponsor can send them offers. 35 members gave a number, for SMS reminders only.
2 Stakeholders	Members (spam, loss of control), president and club (sponsor money, reputation), sponsor, institute (liability), developer (responsibility).
3 Codes and rules	SE Code 1 Public and 2 Client (serve the client only within the public interest); ACM 1.6 Respect privacy; GDPR principle *purpose limitation*: numbers were collected for reminders, not marketing; our own privacy notice says "never given to sponsors".
4 Options	(a) export as asked · (b) refuse and stop there · (c) the club sends the sponsor's offer itself in the newsletter, members contact the sponsor if interested · (d) new opt-in "sponsor offers" box, export only those who tick it
5 Tests	(a) fails harm, rights and publicity tests; breaks our notice. (b) protects members but ignores the club's legitimate need. (c) and (d) meet the need without harm; (d) needs a notice update and a new consent.
6 Decide, act	Decline (a) politely in writing, citing the privacy notice; propose (c) now and (d) as a backlog item for the Product Owner. Record the decision in `docs/decisions/0003-sponsor-export.md`.
7 Reflect	Add "no bulk export of contact data" to the admin role design; add a test that the CSV export never contains a phone column.

✓ **Reply to the president** (draft): "We can't export members' phone numbers: they were given only for reminders, and our privacy notice promises they never go to sponsors. We can put the sponsor's offer in your next club newsletter today, and we'll add an opt-in 'sponsor offers' option in the next sprint."

⚠ If the president insists or asks an admin to do it quietly, escalate: the teaching assistant (acting client) first, then the clubs office. Do not export "just this once".

Habits of an ethical engineering team

1 Inventory before code

Every stored field has a purpose, a legal basis, a retention limit and a reader list. Mistake: "let's collect it, it might be useful later".

2 Private by default

Optional features start off; consent boxes start unticked. Mistake: pre-ticked boxes and "agree to all" buttons.

3 Redact every log

Only `redacted()` output reaches logs; test data is invented. Mistake: printing a whole `Student` while debugging and committing the log.

4 Licence on day one

`LICENSE`, SPDX headers and a third-party notice from the first commit. Mistake: copying Stack Overflow or GitHub code without checking its licence.

5 Accessible from the first mock-up

Words plus colour, labels, keyboard paths, 4.5:1 contrast. Mistake: "we'll add accessibility at the end": it is then a redesign.

6 Speak up, in writing

Raise concerns early, use the seven steps, record decisions. Mistake: staying silent about a known bug before a demo, or leaking data to "prove a point".

Chapter quiz 10 questions

1. The club wants 80 seats for a talk in a room whose safety limit is 50. Which principle of the ACM/IEEE-CS SE Code settles the conflict?

- Principle 2 Client and employer: the club is the client
- Principle 1 Public: the public interest comes first
- Principle 7 Colleagues: ask the team to vote
- Principle 8 Self: the engineer decides alone

2. Which of these is NOT one of the eight principles of the SE Code?

- Product
- Judgment
- Profit
- Self

3. A teammate writes `log << "registered " << s.name << " " << *s.phone;`. What is the best fix?

- Nothing: logs are private to the team
- Log `redacted(s)` and the event ID instead
- Encrypt the log file and keep the line
- Print the same line to the console instead of the log

4. What does data minimisation mean for the Club Hub registration form?

- Compress the CSV files to save disk space
- Collect only fields needed for a stated purpose, such as no date of birth
- Store all fields but hide them from organisers
- Delete the whole database at the end of each semester

5. Which statement about the Cambodian legal situation matches this chapter?

- Cambodia has no rule that touches personal data
- The Law on Electronic Commerce (2019) contains data-protection provisions and the landscape is evolving
- Cambodia applies GDPR directly as national law
- Only paper records are covered by Cambodian law

6. Your team wants anyone, including companies, to reuse the Club Hub core freely, with an explicit patent grant. Which licence fits best?

- GPL-3.0
- No licence at all
- Apache 2.0
- AGPL-3.0

7. Normal-size body text in the planned web UI has a contrast ratio of 3.2:1 against its background. Does it meet WCAG 2.2 level AA?

- Yes, 3:1 is enough for all text
- No, normal text needs at least 4.5:1
- Yes, contrast is only a level AAA criterion
- It depends only on the font family

8. While using the institute's registration website you notice that changing a number in the URL shows another student's record. What should you do?

- Download a few records as proof and post them online
- Report it privately to the site owner with steps to reproduce, and access nothing more
- Keep testing other URLs to measure the damage
- Ignore it, it is not your system

9. A sign-up screen shows "Share my profile with sponsors" already ticked. Which dark pattern is this?

- Confirmshaming
- Roach motel
- Pre-ticked consent
- Disguised ads

10. Engine software that switched on full emission controls only when it detected an official test is the core of which case?

- Therac-25
- Volkswagen emissions software
- Cambridge Analytica
- Ariane 5

WRAP-UP

Summary

- Software decisions affect users, clients, colleagues, the profession and the public; legal is the floor, not the ceiling.
- The SE Code has eight principles; the public interest comes first when obligations conflict.
- The ACM Code (2018) adds explicit duties on discrimination, privacy and robust, usable security.
- A data inventory gives every field a purpose, a basis, a retention limit and a reader list; minimise, redact logs, delete on time.
- Cambodia's rules are evolving (E-Commerce Law 2019); GDPR principles are the design reference; consent is active, specific and withdrawable.
- No licence means no permission; MIT and Apache 2.0 are permissive, GPL-3.0 is copyleft; ship LICENSE, SPDX headers and third-party notices.
- WCAG 2.2 POUR, level AA: words not only colour, 4.5:1 contrast, keyboard access, labels and helpful errors, also in the CLI.
- Report vulnerabilities privately and coordinate disclosure; avoid bias feedback loops and dark patterns.
- Use the seven decision steps, look for the third option, escalate internally first, and write the decision down.

Open Lab-03: 5 tasks + 1 challenge

Next chapter: 04 · Software Development Processes