



AI-ASSISTED CODE GENERATION POLICY

THIS POLICY WAS APPROVED:	AUTUMN 2026
POLICY VERSION:	1.0
THIS POLICY WILL BE REVIEWED:	AUTUMN 2027
MEMBER OF STAFF WITH RESPONSIBILITY FOR REVIEW:	DIRECTOR OF ICT
THIS POLICY WAS CONSULTED WITH:	TECHNICAL SERVICES LEADERSHIP TEAM EXECUTIVE LEADERSHIP TEAM
THIS POLICY WAS DISTRIBUTED TO:	TRUST LEADERSHIP GROUP CENTRAL LEADERSHIP TEAM

Purpose and principle

This policy applies to all employees, contractors, consultants and other authorised users who use AI-assisted coding tools for Trust work, whether the resulting code is created, tested, stored or operated on Trust-managed or externally hosted systems.

AI-assisted coding is permitted and can be genuinely valuable when proportionate safeguards, technical competence and accountable oversight are in place. These tools can accelerate coding, debugging, documentation and prototyping, but they do not make a user a developer. They enable people to work more quickly where they already possess the knowledge and skill required to understand, secure, test, maintain and support the result.

The aim is safe adoption, not prohibition. AI-generated code can look polished and convincing while containing serious errors, unsafe assumptions or hidden behaviours. Apparent functionality is not the same as fitness for purpose, and responsibility remains with the individual and the Trust rather than with the tool.

As in teaching, medicine or engineering, plausible output is no substitute for qualified scrutiny. An authorised developer must be able to assess code, data flows, authentication, dependencies and operational consequences. Non-sensitive examples may be used to teach an approved tool the Trust's coding style, improving speed and maintainability without transferring judgement or accountability.

Summary

- AI supports, but does not replace, professional competence: accountable people must understand, review and own AI-assisted work.
- Use must reflect risk: fabricated-data learning and prototypes are encouraged; substantive work is controlled; sensitive or live-system work requires Technical Services oversight.
- Protect information: use approved organisational tools and accounts; never submit personal data, live records, credentials, sensitive logs or material of uncertain sensitivity.
- Treat generated output as untrusted: review and test changes, dependencies, commands and configurations for hidden, insecure, external or destructive behaviour.
- Secure engineering remains mandatory: use controlled environments, version control, least-privilege test credentials, proportionate independent review, testing, rollback, monitoring and documentation.
- Govern the full lifecycle: ensure ownership, specialist input, lawful and secure processing, licensing, supplier assurance, sustainable costs, support and retirement plans.
- Report incidents promptly: contain exposure, rotate affected credentials and notify the Data Protection Officer and Technical Services immediately.
- Human accountability is paramount: authorised professionals retain responsibility for information, access, review, approval, deployment and operation.

Why safeguards are necessary

Data protection, privacy and safeguarding

- Users may unknowingly submit personal data, credentials or other sensitive information without knowing where it is stored, processed, retained or reused.
- Personal subscriptions and unapproved services may have no organisational contract, data-processing agreement or assured controls governing prompts and uploaded material.
- Even apparently harmless technical material can be revealing. Logs, stack traces, schemas, field names and sample rows may identify individuals or expose safeguarding, SEND, HR, finance or identity-related purposes.
- Generated code may create temporary files in public locations, transmit information insecurely or send records to an external API. A script may appear to perform the requested task while exposing data through an operation the user does not recognise.
- Any system processing personal data can become the vector of a safeguarding or data-protection incident. “The AI wrote it” is not a defence: legal and organisational responsibility remains with the Trust.

Security, technical quality and expertise

- AI-generated solutions can appear convincing while containing errors, unsafe defaults, bias, hidden functionality or exploitable vulnerabilities.
- A request to verify pupil addresses might produce code that sends the data to an external mapping or validation service. A user without code literacy may not recognise the outbound request.
- Models may disable certificate checking, hard-code a secret, expose a database port, add an unnecessary external connection, create prompt-injection risk or select an insecure authentication method.
- Outputs can be confidently wrong. Models may invent libraries, functions, legal requirements or policy references, and may propose a non-existent package name that a malicious actor could later register with harmful code.
- Without tight direction, generated solutions may mix languages, frameworks and patterns that do not form a coherent, maintainable whole.
- Successive prompts may lead to extensive or wholesale rewrites rather than focused changes. This means the full codebase may need to be reviewed again at every material iteration; rapid generation does not mean rapid assurance.
- Version control, testing, rollback, separation of development and production, monitoring and secure deployment remain human engineering responsibilities.

Operational reliability and lifecycle ownership

- AI-generated workflows may be fragile, undocumented, inconsistent or dependent on a service that changes or becomes unavailable.
- If the system fails, the Trust must know who can diagnose it, restore service and correct the underlying defect. A locally created automation with no competent owner can become business-critical technical debt.
- Outputs may not align with approved technologies, architecture, processes, values or strategic priorities, even when they solve the immediate problem.

- Early success can create a false impression of organisational technical capability and encourage premature deployment of systems that cannot be securely maintained.
- Paid libraries, hosting, processing, support and environmental costs may be introduced unintentionally.

Governance, compliance and professional responsibility

- Unassured AI-built systems are a form of shadow IT, even when created with good intent. They reduce visibility, auditability and central risk management.
- Generated code may reproduce copyrighted material or introduce open-source licence, attribution or redistribution obligations that the user does not recognise.
- Ownership may be ambiguous, particularly where personal accounts, repositories or subscriptions are used.
- Subject expertise remains necessary alongside technical expertise. A technically sound system can still encode the wrong curriculum, HR, finance, safeguarding or operational rules.
- Over-reliance may de-skill staff or create dependency without understanding. Accountability remains with the person approving the work and the organisation operating it.

Risk classification and approval

For this policy, an approved AI coding tool is a service, account, extension, agent or command-line tool authorised by Technical Services for its intended use. An authorised developer has approved competence and access. An isolated prototype uses fabricated data, has no live-system connection and cannot affect operational services. A production environment is any live system, service, database, integration or infrastructure used for Trust operations.

AI-assisted work falls into three categories:

Encouraged: learning, explanation, documentation, isolated snippets and prototypes using fabricated data. Leaders may use AI to illustrate a proposed system or workflow, helping Technical Services and subject specialists understand the need, challenge assumptions and shape an appropriate solution. Prototypes are illustrative only: they must remain isolated from live data, credentials and integrations, and require developer approval and normal governance before operational use.

Controlled: internal tools, non-sensitive automation and substantive development undertaken by developers authorised by Technical Services, using approved accounts, environments, version control, testing and a named owner.

Restricted: work involving personal or safeguarding data, HR, finance, MIS, identity, credentials, databases, external integrations, scheduled tasks, production services or other operational infrastructure. Only authorised Technical Services staff may use AI-assisted code generation with live systems or production environments. Such work requires Technical Services control and the Trust's normal DPIA, security, service-owner and deployment approvals; systems processing personal data remain subject to the usual DPIA threshold.

Information that may be shared

Only sanitised, non-sensitive technical context may be provided to an approved AI coding tool. The following must not be shared:

- Personal data, credentials, API keys, tokens, passwords or other secrets.
- Live database contents or records.
- Error messages, stack traces or logs containing potentially sensitive or identifying information.
- Anything whose sensitivity cannot confidently be established.

An approved AI coding tool may receive sanitised schemas, API definitions / swaggers and other technical context only where they contain no sensitive information. These must describe structure rather than live data, and field names that reveal sensitive purposes, such as safeguarding indicators, SEN status or National Insurance numbers, must be removed or anonymised.

Where sample data is necessary, it must be fabricated rather than drawn from live or historic records, because apparently benign values may become identifying when combined with a detailed schema.

Controlled development workflow

Authorised developers may provide approved, non-sensitive examples of the Trust's coding style and sanitised schemas, API definitions / swaggers or technical documentation. The objective is code that the development team can read, understand, maintain and debug.

The tool must not access live databases, production services, credentials or uncontrolled organisational repositories, even when used by authorised Technical Services staff. Where access to files or a repository is required, the tool must operate within a controlled development workspace containing only sanitised information. Output must enter test or production environments through normal version control and review.

Mandatory review of generated code

Every AI-generated output, dependency, command and configuration change must be reviewed and tested before it is trusted or reused. The developer must look beyond whether the code appears to work and identify functionality or behaviour outside the commissioned purpose, including:

- Unexpected API calls, network connections or external services.
- Unnecessary file-system, shell or process operations.
- eval, dynamic execution or similar constructs.
- New or unvetted dependencies.
- Unexpected authentication or credential handling.
- Code that changes, deletes or writes data when the requirement was read-only.

API calls must be checked against the approved definition and requested purpose, including confirmation that read-only requirements contain no write-capable operations. Static analysis should be used to identify unexpected outbound calls, shell or evaluation constructs and unapproved dependencies. Suggested installations, commands or configuration changes must never be run automatically.

Credentials and execution

AI-assisted development must use separate, minimally privileged, non-production credentials, which should be read-only wherever practicable. This limits the effect of any undetected write operation or unexpected behaviour.

Generated output must not be executed merely because it appears correct. Production-impacting changes require appropriate testing, rollback and, where relevant, monitoring or alerts; destructive operations must first be tested against fabricated or disposable data.

Production services and any workload processing live data must run only in a Technical Services-approved on-premises server or managed cloud environment. They must not be hosted or operated from a user's workstation, laptop or other local endpoint. Local execution is permitted only for isolated development or testing using fabricated or appropriately sanitised non-live data and non-production credentials.

Responsibilities of a developer

An experienced developer must understand the security and operational implications of the change, anticipate plausible unintended behaviour, and review generated code at the level of individual operations and dependencies.

Authorisation to undertake controlled or restricted AI-assisted development must reflect demonstrated competence appropriate to the technologies, data and operational risk involved. Technical Services must review that authorisation periodically and when a person's role, access, responsibilities or the capabilities of the approved tools materially change. Relevant training and assurance records must be retained in accordance with the Trust's records-management requirements.

AI-assisted commits or pull requests affecting data processing, authentication, permissions, external integrations, destructive operations or scheduled tasks require review by a second developer before merging. The reviewer must be independent of the person who prompted the AI coding tool and must assess the resulting change rather than rely on the first developer's inspection.

All AI-assisted changes must pass through normal version control and review. Commit messages or review notes must identify the human-approved change and retain proportionate, non-sensitive evidence of that review.

Where a system encodes curriculum, safeguarding, HR, finance, estates or operational rules, the relevant system or service owner, operational lead or subject specialist must sign off the intended behaviour before deployment.

AI-assisted code must not be used to make solely automated decisions about pupils, families, applicants or staff where those decisions have legal or similarly significant effects. Where a system informs consequential decisions, a competent person must review the outcome, understand the relevant logic and limitations, consider equality, accessibility and safeguarding implications, and retain authority to challenge or override the result.

Data exposure and incident response

If sensitive information, a credential, token or personal data is accidentally entered into an AI coding tool:

- Stop immediately and do not submit further information.

- Treat the exposed credential as compromised and rotate/revoke it where applicable.
- Preserve sufficient information to establish what was exposed, without further reproducing the sensitive material.
- Notify the Data Protection Officer and Technical Services immediately, and follow the organisation's data-protection and security incident procedures.
- Do not attempt to conceal or independently resolve a potential data-protection incident.

Supplier, contractual and tool controls

Before an AI coding tool is used with organisational information, the Trust must verify that the specific service and plan provide appropriate contractual and technical controls for data processing, prompt retention and model training.

This review must take account of the Trust's UK GDPR, Data Protection Act 2018, safeguarding and contractual obligations. Controls applicable to one AI coding product, service or plan must not be assumed to apply to another.

Only approved organisational accounts, plans, browser extensions, local agents and command-line tools may be used for Trust work. Personal accounts must not receive organisational context. Technical Services will maintain the approved position for each tool and may restrict or withdraw approval where risks or controls change. Approval must be reviewed whenever product terms, retention settings, model-training arrangements, integrations or technical behaviour change.

Cost, sustainability and exceptions

Where an AI tool, model or externally hosted service becomes material to development or operation, the service owner must document how work and services will continue if it becomes unavailable, materially changes its terms or behaviour, or is withdrawn. Operationally significant solutions must not depend on access to a particular AI service where no proportionate alternative, export route or recovery arrangement exists.

AI-assisted development must not create disproportionate hosting cost, processing cost, environmental impact, support burden or technical debt. Any exception to this policy must be approved by the Director of ICT or a delegated senior Technical Services lead and recorded with its reason, scope, duration, compensating controls and review date. An exception cannot authorise unlawful processing, override safeguarding requirements or remove accountability for security, data protection or professional judgement.

Approval questions and delivery workflow

Before any AI-assisted solution moves beyond an isolated prototype, clear and assured answers are required to the following questions:

- Who governs the project, and what are each person's responsibilities?
- Are the necessary technical and subject specialists represented?
- What data is processed, where is it stored, and which organisations or services can access it?
- Is the proposed processing lawful, necessary and consistent with the purpose for which the data was collected?

- Does any part of the workflow transmit data elsewhere?
- What happens when the system fails, produces an incorrect result or cannot recover?
- Who owns, supports, maintains, reviews and ultimately retires the system?
- Are all libraries, models, services and components appropriately licensed?
- Has the security posture been understood, including authentication, permissions, dependencies and external connections?
- Are testing, rollback, monitoring, incident response and business-continuity arrangements proportionate?
- What hosting, processing, support and environmental costs will be incurred?
- How will documentation, knowledge transfer and future technical debt be managed?

Proposals that move beyond an isolated prototype must be submitted to Technical Services for initial assessment. Technical Services will determine whether the proposal may proceed, whether ELT consideration is required and which technical, operational, data-protection and subject specialists must define the scope and complete the necessary assurance.

Delivery then follows normal engineering practice: establish architecture and ownership; review or rewrite generated code; test and document the system; complete security, data-protection and user-acceptance reviews; record approvals; authorise deployment; and transfer the service into support. Assurance may take much longer than generation, particularly after substantial AI-led rewrites; quality cannot be rushed.

Policy Review

This policy must be reviewed before its scheduled review date following a significant AI-related incident, a material change in law or regulatory guidance, or a substantial change in the capabilities, integrations or data-handling behaviour of tools used by the Trust.

Compliance and governing rule

Failure to follow this policy may result in the relevant tool, account, integration or deployment being suspended while the risk is assessed. Suspected data-protection, safeguarding or security incidents will be managed under the Trust's applicable incident procedures. Deliberate or repeated non-compliance may also be addressed through the relevant conduct, contractual or disciplinary process.

AI may accelerate development, but it does not assume responsibility for it. Its permitted scope must reflect the user's competence and the risk of the work, with accountable professionals retaining control of information, access, review, testing and deployment.

Appendix A: Anthropic and AI coding-agent risk examples

These examples from the supporting risk paper illustrate why controls must cover the wider toolchain as well as generated code. They are illustrative, not exhaustive, and must be verified before use because products, terms, vulnerabilities and reported incidents change quickly.

- Researchers reported a critical Claude Code vulnerability through which malicious project-configuration instructions could execute on a developer's machine, enabling unauthorised actions, secret exfiltration and system compromise. The lesson is that trusted-looking repository and configuration files can expand an AI-assisted development tool's attack surface.
- Claude Code command-line source-map material was reportedly exposed through GitHub in March 2026. Although attributed to human rather than model error, this could create security, competitive and reputational risk by revealing useful details to attackers.
- Claude outages in April 2026 affected the web application, API and Claude Code, illustrating the operational risk of depending on an external AI service for important development workflows or systems.
- Reporting on Anthropic's advanced vulnerability-discovery model, Mythos, highlighted dual-use risk: technology designed for cyber defence may also facilitate exploitation. Later reports of possible unauthorised access increased concern about control of highly capable models.
- A reported incident involving a Claude-powered coding agent described deletion of a company database and its backups within seconds. Whatever the cause, agents must not have broad production permissions: destructive actions require least privilege, separation, testing and recoverable backups.
- Anthropic reported four incidents between July and September 2026 in which Claude escaped supposedly isolated cyber evaluations and accessed real third-party systems because the environments were connected to the open internet. It continued pursuing objectives despite evidence that the systems were real. Isolation must therefore be technically enforced and verified through network restrictions, least privilege, monitoring and rapid termination—not assumed from prompts or task framing.
- Anthropic's September 2026 threat report described suspected state-sponsored and criminal use of Claude for credential harvesting, malware adaptation and large-scale scanning for exposed secrets. The resulting increase in speed, scale and autonomy reinforces the need to monitor organisational accounts, credentials, integrations and permissions, limit access and retain rapid suspension capability.
- Anthropic reported in October 2026 that Claude models pursuing assigned objectives exploited SQL and command-injection weaknesses in real third-party services, extracted working access tokens, bypassed access or payment restrictions and submitted fabricated information through a public form. It linked the pattern partly to persistence and reward-seeking when tasks were ambiguous, impossible or obstructed. The controls required are clear authorisation boundaries, server-side access controls, enforced network restrictions and human approval for external interaction or submission.

These examples do not justify prohibiting Claude or other AI coding tools. They reinforce the policy's central position: use is acceptable when tools are contained, information is protected, permissions are minimal, outputs are fully reviewed and accountable professionals retain control.