# Foundations of Software Automation in Healthcare

*What automation is, how to find it in a workflow, and the tools you will use.*

This self-study guide accompanies Module 1 of the ARIVE course Software Automation in Healthcare. It is designed for independent reading alongside the recorded lectures, and it can also be studied on its own as a stand-alone introduction. The guide develops the conceptual foundations of the module in considerably more depth than the slides allow, adds worked examples and short pieces of illustrative code, and draws on peer-reviewed literature so that its claims can be traced to evidence. Wherever a statement rests on published work, a numbered citation in IEEE style points to the reference list at the end.

The central argument of the module is easy to state and worth absorbing early: automation in healthcare is valuable, but only when it is applied deliberately to the right tasks and kept under human oversight where the stakes are high. The cross-industry experience of automation, systematically reviewed for healthcare by Zayas-Cabán and colleagues, shows that the benefits depend closely on how well a task is defined, how repetitive it is, and how much human judgement it requires [1]. Those same factors reappear in every later module of the course, so the habits of thought you build here will pay off repeatedly.

It is also worth being clear about why automation matters so much in this particular field. The widespread adoption of electronic health records, while beneficial in many respects, has introduced a substantial documentation and administrative burden on clinicians. A meta-analysis of hospital studies found that physicians' share of time spent on documentation rose from roughly sixteen to twenty-eight percent after electronic-record implementation, with comparable increases for nurses [5]. A scoping review has linked much of this burden to usability problems that force clinicians into workarounds, duplicate entry, and constant task-switching [4], and a systematic review of the measures used to quantify the burden confirms that it is now a recognised contributor to professional stress and burnout [6]. Well-targeted automation of the repetitive, rule-based parts of this work is one credible response, which is exactly what this course teaches you to design.

## Learning outcomes

After working through this guide you will be able to:

- Explain the administrative and documentation burden that motivates automation in healthcare.
- Define software automation and place a task on the spectrum from manual to fully automated.
- Explain the principal benefits of automation in clinical and research settings, and the single condition attached to them.
- Describe a workflow as steps, inputs, and outputs, and identify candidates for automation.
- Score candidate steps and select a safe first automation using value and effort.
- Name the core tools of the automation toolchain and explain the role of each.
- Recognise where automation should stop and human judgement must remain.

# 1. Why automation, and what it is

## 1.1 The problem automation addresses

Before defining automation it helps to understand the problem it is meant to solve. Modern healthcare generates enormous quantities of routine, rule-governed work: transcribing results, moving data between systems, generating reports, managing inboxes, and reconciling records. Much of this work does not require clinical judgement, yet it consumes the time of highly trained staff. The evidence that this burden is real and consequential is now substantial. Baumann and colleagues, pooling twenty-eight hospital studies, found that the proportion of physicians' time spent on documentation increased markedly after electronic-record implementation, though with some suggestion that familiarity eventually improves workflow [5]. A scoping review by Olakotan and colleagues traced these disruptions to interface and usability flaws that misalign systems with clinical workflow and drive error-prone workarounds [4].

This context matters because it tells us where automation is most useful and most welcome: not in replacing clinical decisions, but in relieving the repetitive administrative load that surrounds them. A recent review of automation in clinical settings identifies prior authorisation, inbox management, scheduling, billing, and report generation as workflows that commonly benefit, while cautioning that value is realised only through a deliberate implementation strategy rather than automation for its own sake [2].

## 1.2 A working definition

Software automation is the use of software to carry out a task that a person would otherwise perform by hand, following fixed rules and requiring little or no intervention once it is running. Two elements of this definition deserve emphasis. First, the task must be governed by rules explicit enough to be written down; a task that depends on tacit judgement cannot be automated without first being made explicit, and often should not be. Second, once started, the process should be able to complete on its own, which is what distinguishes automation from mere software assistance.

## 1.3 Automation, augmentation, and the human role

It is useful to separate two related but distinct ideas. Automation, in the strict sense, replaces a human action with a software action. Augmentation, by contrast, leaves the action with the person but makes it easier, faster, or less error-prone. Much of what succeeds in healthcare is augmentation rather than full automation: a system that pre-fills a form, surfaces the right record, or flags an out-of-range value still leaves the clinician in control, yet removes friction. This distinction matters for expectation-setting, because promising full automation where only augmentation is safe or feasible is a common cause of disappointment and mistrust. Throughout this course, when we say automate, we mean moving a task as far along the spectrum as is safe and useful, which is frequently short of the fully automated extreme.

A second clarification concerns what automation is not. It is not, in most cases, artificial intelligence: the majority of useful automation follows fixed rules that a person wrote, with no learning involved, and we keep that distinction sharp until Module 5, where learned models are introduced deliberately. Nor does automation remove people; it removes repetitive steps and redirects human attention toward the

judgement that machines do not perform well. Keeping these boundaries clear prevents both the fear and the over-optimism that get in the way of sound decisions.

> **Key idea.** Automation is most valuable where routine, rule-based work surrounds clinical judgement. The goal is to relieve the administrative burden that the evidence links to clinician stress, not to replace professional decisions.

## 2. The automation spectrum

It is a common mistake to treat automation as a binary property, something a task either has or lacks. In practice it is a spectrum, and locating a task correctly on that spectrum is the first analytical skill of the course. At one extreme a task is fully manual; a person performs every step. One step along, the work is assisted, meaning software helps but a person still triggers each action and checks the result. Further still, the task is scripted, so a single command runs the whole sequence at once. At the far end it is scheduled or event-driven, running with no human trigger at all, for example every night or whenever a new file arrives.

The systematic review by Zayas-Cabán and colleagues found precisely this range across industries and, crucially, observed that the appropriate level of automation for a given task is linked to how well the task is defined, how repetitive it is, and how much human decision-making it requires [1]. The practical implication is that most healthcare tasks should move only one or two steps along the spectrum rather than leap to the fully automated end. A results-transfer task might reasonably become scheduled; a decision about a patient's care should remain, at most, assisted.

### 2.1 A five-level view

It is often useful to name intermediate levels explicitly when discussing a workflow with colleagues:

- Manual: every step performed by a person.
- Assisted: software supports the person, who still triggers and checks each step.
- Scripted: a single command runs the sequence when a person starts it.
- Scheduled: the sequence runs automatically at set times.
- Event-driven: the sequence runs automatically in response to a trigger, such as a new record.

> **Key idea.** Automation is a matter of degree, not a switch. Deciding how far along the spectrum a task should move, given its definition, repetitiveness, and the consequences of error, is the judgement this module builds.

## 3. Why automation matters, and its one condition

Three benefits recur throughout the literature, and it is worth understanding each precisely rather than as a slogan.

### 3.1 Accuracy

Software applies the same rule identically on every record and does not tire at the end of a shift, so it removes a class of transcription errors that manual data entry invites. This is not a claim that people are careless; it is a recognition that repetitive manual transcription is intrinsically error-prone, and that removing the manual step removes the error with it. The documentation-burden literature reinforces the point, noting that workarounds such as duplicate entry actively increase the risk of data-entry errors [4].

### 3.2 Reproducibility

Because a script is an exact, executable record of what was done, the same input yields the same output on every run. This property is foundational to research, where a result that cannot be repeated is difficult to trust, and it becomes a central theme of Modules 4 and 6. Automation thus delivers two things at once: the work itself, and a precise description of how the work was done.

### 3.3 Returned capacity

Routine transfers and report generation consume hours of skilled staff time that rarely require human judgement, and returning those hours to clinical and research staff is a central motivation for automation in clinical settings [2]. Given the documented scale of the administrative burden [5], [6], even modest automations can free meaningful amounts of time.

### 3.4 The condition

These benefits come with a single, non-negotiable condition: the automated step must be correct and monitored. Automation multiplies whatever behaviour it encodes, applying it to every record without pausing to doubt. A fast but wrong process is therefore more dangerous than a slow manual one, because it produces errors at scale. The case-study literature is consistent on this point, stressing that automation succeeds only when goals are defined in advance and outcomes are monitored over time [1].

> **Key idea.** Accuracy, reproducibility, and returned capacity are the rewards of automation, but each appears only if the automated step is correct and watched. Speed is a benefit on top of correctness, never a substitute for it.

## 4. Mapping a workflow

Before any task can be automated it must be understood, and the practical instrument for understanding is the workflow map. A workflow is an ordered set of steps that turns a starting input into a finished output, where each step has its own inputs and produces something the next step uses. Describing every step in these terms, as an input, an action, and an output, converts a vague routine into something that can be inspected and questioned.

### 4.1 The method

A short, repeatable method serves for almost any process:

- List the steps in the order they actually happen, in plain language.
- Mark the input and the output of each step, noting where inputs come from and who uses outputs.
- Score each step for how repetitive and rule-based it is.
- Choose one strong, safe candidate to automate first.

The discipline of writing down what actually happens, rather than the idealised official process, is what exposes hidden steps, silent dependencies, and duplicated effort. This is not a merely academic exercise: the documentation-burden literature shows that undocumented workarounds are pervasive in real clinical workflows [4], so a map built from the official procedure alone will usually be wrong in important ways.

### 4.2 A worked example

Consider a workflow used throughout the module: moving laboratory results into a research dataset. A plausible map runs: (1) an instrument produces a results file; (2) a technician saves it to a shared folder; (3) someone opens it and checks the format; (4) values are copied into the dataset; (5) a second person spot-checks the entries. Reading this map, steps two and four are repetitive and rule-based and therefore candidates, while steps three and five involve judgement and remain with people; step one is already automated by the instrument. In a single pass, the map has told us where to act and where not to.

### 4.3 Common mapping mistakes

Three mistakes recur when people first map workflows, and each is worth guarding against. The first is mapping the ideal process rather than the real one; the official procedure is always tidier than practice, and automating an idealised map produces automation that fails on the messy reality. Walking through a specific, recent case rather than describing the process in the abstract is the reliable antidote. The second is recording only actions while skipping inputs and outputs; most hidden complexity, and most of the surprises that break an automation later, live in what a step needs and what it produces. The third is attempting to automate everything at once; starting with a single, clear win and expanding from there is far more likely to succeed than a large, all-at-once project.

> **Practical method.** List the steps in real order; mark the input and output of each; score each step for how repetitive and rule-based it is; then choose one strong, safe candidate to automate first rather than attempting the whole workflow at once.

## 5. Scoring and selecting candidates

Once a workflow is mapped, candidate steps can be scored against criteria drawn from the cross-industry evidence: a step is a strong candidate when it is repetitive, rule-based, high in volume, stable over time, and error-prone when done by hand [1]. Steps that require clinical judgement, occur only once, or rest on unclear or shifting rules are poor candidates and should remain with people.

Scoring potential, however, is not the same as deciding what to automate first. A second consideration, the balance of value against effort, governs that choice. The best first project is high in value and low in effort, because it proves the approach quickly and earns the confidence and trust needed for harder work later. Aligning automated workflows with structured, standards-based data has been shown in hospital case studies to reduce waiting times and errors while producing higher-quality structured data as a by-product [3], which makes such data-centred steps attractive early targets.

### 5.1 A worked scoring of the laboratory workflow

Returning to the laboratory example, we can score its steps informally to see selection in action. Saving the file to a shared folder is repetitive and rule-based, with well-defined inputs and outputs, and is low in effort; it is a reasonable candidate. Checking the format is rule-based but includes some judgement about borderline cases, so it is a partial candidate at best. Copying values into the dataset is repetitive, high in volume, and error-prone by hand, and its inputs and outputs are clear; it is both high in value and safely checkable, which makes it the strongest candidate. Spot-checking entries is judgement work and should

remain with a person. Putting potential and value together, the copy-to-dataset step is the natural first automation:

- Save file to folder — repetitive, low effort: a possible early win.
- Check format — rule-based but needs judgement: partial candidate.
- Copy values to dataset — high value, well-defined, checkable: the strongest choice.
- Spot-check entries — judgement: keep with a person.

This kind of quick, explicit scoring turns an intuition about where to start into a defensible decision that colleagues can review, which matters when automation touches shared data.

> **Key idea.** Score steps for repetitiveness, rule-basis, volume, stability, and error-proneness; then, among strong candidates, choose one that is high value and low effort and whose output a person can still check.

## 6. The automation toolchain

The remainder of the course rests on a small, stable toolchain, and it is worth meeting the components in advance. A code editor is the workspace in which instructions are written and read; a good editor highlights likely mistakes as they are typed and keeps a project's files together, so that errors are caught before code ever runs. Python is the programming language used for most examples, chosen for its readability and its large ecosystem of libraries for data and healthcare work. Version control, provided by Git, records every change and keeps a reviewable history, a capability later modules show to be valuable both for collaboration and for the accountability that regulated healthcare work requires.

### 6.1 The four tools in a little more detail

The code editor is more than a place to type. A good editor lists a project's files for quick navigation, colours code so its structure is visible at a glance, underlines likely mistakes before the code is ever run, and provides an integrated place to run and inspect a script. None of this is essential to understand deeply now; the point is that the editor is designed to catch problems early, which is especially valuable when code touches health data.

Python is favoured here because it is readable and widely used in healthcare and research, with a large ecosystem of ready-made libraries. You will meet a small number of building blocks repeatedly: variables that hold values, functions that take input and return a result, libraries of code written by others, and scripts that run from top to bottom. Familiarity with these ideas, rather than mastery, is all that is needed to begin, and later modules introduce specific libraries such as pandas for tabular data and a FHIR client for healthcare records.

Git, the version-control system, records every change as a commit with a short note, keeps an ordered history you can revisit, and lets several people work in parallel on branches without overwriting one another. In regulated healthcare work, the ability to show exactly what changed, when, and by whom is often required, and Git produces that record as a by-product of ordinary work. Module 4 returns to Git in practical detail; here it is enough to recognise its role as a safe history and an undo button that never forgets.

These tools converge on a single organising idea, the pipeline: a sequence of small steps joined so that they run in order and automatically. A minimal example, which reads a results file, removes rows with missing values, and writes a clean file, shows how little code real automation can require:

```
import pandas as pd

# read the raw results
data = pd.read_csv("results.csv")

# keep only complete rows
clean = data.dropna()

# write a tidy dataset
clean.to_csv("dataset.csv", index=False)
```

*A minimal data pipeline in Python. Four working lines read, clean, and write the data; the rest is explanatory comment.*

Standards-based automation of hospital workflows has demonstrated the practical value of turning ad hoc, manual sequences into structured, automated pipelines that generate clean data and reduce delays [3]. The pipeline is the shape of nearly all the automation built in this course, and Module 2 constructs one in detail against real healthcare data.

> **Key idea.** Editor, Python, Git, and the pipeline idea are the whole toolchain. A pipeline joins small, understandable steps into one dependable flow, and it is the destination of almost everything you will build.

## 7. Where automation should stop

A recurring theme of the module, and of the responsible-automation literature, is that knowing when not to automate is as important as knowing how. Automation is a poor fit for decisions that require clinical judgement, for one-off tasks that will not recur, for processes built on rules that are vague or frequently changing, and for any situation in which a wrong output could cause harm and would pass unchecked. The reviews cited here consistently frame automation as a tool that supports human work, embedded within processes that people continue to design, monitor, and correct, rather than as a replacement for professional judgement [1], [2].

The guiding rule for the whole course follows from this: automate the repetitive, rule-based work, and keep a human in the loop wherever the output carries clinical risk. This principle recurs, in progressively more technical forms, through the modules on testing, artificial intelligence, and data protection. Holding it clearly now will make the later material easier to place.

> **The enduring rule.** Automate the routine, and protect the decisions. The higher the stakes for a person, the more a human must remain in the loop.

## 8. Further reading and practice

To consolidate this module, work through the following short exercises before starting Module 2. Each is designed to take under an hour and to produce something you can reuse in the module assignment.

- Choose a routine task from your own work and place it on the five-level automation spectrum, justifying your choice.
- Map that task as steps with inputs and outputs, deliberately including any informal workarounds.
- Score each step for repetitiveness, rule-basis, stability, volume, and error-proneness, and circle your strongest candidate.
- Write one paragraph arguing whether that candidate should be automated, and to what level, referring to the consequences of an error.
- Read the abstract of reference [1] and note two criteria it uses for automation potential that also appear in your own analysis.

The reviews cited in this guide are all openly summarised and provide accessible entry points to the wider literature on automation, documentation burden, and clinical workflow [1], [2], [4], [5], [6].

## Key terms

**Software automation —** Using software to perform a rule-based task with little or no human intervention once it is running.

**Documentation burden —** The administrative and record-keeping workload placed on clinicians, linked in the literature to workflow disruption and burnout.

**Automation spectrum —** The range from fully manual work, through assisted and scripted work, to scheduled or event-driven automation.

**Workflow —** An ordered set of steps that turns a starting input into a finished output, each step having its own inputs and outputs.

**Workflow map —** An explicit written description of a workflow's steps, inputs, and outputs, used to find automation candidates.

**Automation candidate —** A step that is repetitive, rule-based, stable, high in volume, and error-prone by hand, and thus suited to automation.

**Pipeline —** A sequence of small automated steps joined so that they run in order and on their own.

**Toolchain —** The set of tools used to build automation: a code editor, a programming language, version control, and the pipeline idea.

## Self-check questions

1. Summarise, with reference to the evidence, why the administrative burden of electronic records makes targeted automation attractive in healthcare.
2. Place the following on the automation spectrum and justify each: a nightly data transfer, a clinician's diagnosis, a script a technician runs by hand.
3. State the three benefits of automation and the single condition on which all three depend. Why does the condition matter more in healthcare than in many other fields?
4. Take a routine task from your own work. Describe it as steps with inputs and outputs, and score each step for automation potential.
5. Explain, in your own words, why automating an inaccurately mapped process can be worse than not automating it at all.
6. Distinguish scoring a step's automation potential from selecting it as a first project. What does the value-versus-effort balance add?
7. Name the four tools of the toolchain and state the single job of each.
8. Give two tasks in a research or clinical setting that should not be automated, and explain what makes each a poor candidate.

# References

*Peer-reviewed sources located via the Consensus academic search tool; standards cited from their issuing bodies. Formatted in IEEE style.*


**[1]** T. Zayas-Cabán, K. Marquard, and E. Radhakrishnan, "Identifying opportunities for workflow automation in health care: lessons learned from other industries," Applied Clinical Informatics, vol. 12, no. 3, 2021. [Online]. Available: https://consensus.app/papers/details/9c7a21d887485050b87bd465c8713460/

**[2]** N. Q. Farrukh et al., "Automation and workflow efficiency in clinical settings: implementing automation to reduce administrative burdens," The Medical Clinics of North America, 2026. [Online]. Available: https://consensus.app/papers/details/8f0a16d173145cb0b50c7015bce58881/

**[3]** C. Neto et al., "Automation of hospital workflows using international standards: a case study with imaging exams," Computer Methods and Programs in Biomedicine, 2025. [Online]. Available: https://consensus.app/papers/details/9e4638f0aa6059db9732293ba5d911a1/

**[4]** O. Olakotan et al., "Usability challenges in electronic health records: impact on documentation burden and clinical workflow: a scoping review," Journal of Evaluation in Clinical Practice, 2025. [Online]. Available: https://consensus.app/papers/details/0674143f06095fb385d8f5348e8d6c37/

**[5]** L. A. Baumann et al., "The impact of electronic health record systems on clinical documentation times: a systematic review," Health Policy, 2018. [Online]. Available: https://consensus.app/papers/details/72065e4af52557f18b0c0469667d602d/

**[6]** M. H. Murad et al., "Measuring documentation burden in healthcare," Journal of General Internal Medicine, 2024. [Online]. Available: https://consensus.app/papers/details/48ab3a82bf125f63a530525730338683/