INSTITUTIONAL AUTOMATION

EduSphere AI: A Context-Aware and Policy-Constrained Approval Workflow Automation Framework for Higher Education Institutions

Rutvik Burra, Saurabh Yadav, Sumit Tambe, Sakshi Shinde, Chetana Bhagat, Madhuri Kawarkhe

Department of Computer Science and Engineering, MGM University, Chhatrapati Sambhajinagar, Maharashtra, India

Published November 2026

Abstract

Approval paperwork in universities is slow for a simple reason: nobody agrees on where a document should go next. An examination schedule, a timetable change and a departmental notice all need sign-off, but from different officials, and the route is usually decided by whoever is holding the file. Fixed digital workflows add tracking but send every document down one path, which burdens simple documents and can bypass authorities on complex ones. This paper presents EduSphere AI, a framework that reads a document's context (type, creator role, department, scope and impact), proposes an approval route, and checks that route against encoded institutional policy before any task is assigned. The system only recommends; authorised officials still make every decision. We formulate route selection as minimisation under policy constraints, describe the validation procedure and a policy-record schema, and design an evaluation against manual and fixed-route baselines that reports efficiency and compliance together. An analysis of five expected routes shows that a single fixed route would add unnecessary approvals in four document types and skip a required authority in one, while context-aware routes need one-third fewer approval steps (10 versus 15). A partial prototype covers role-based login and rule-based routing; the experiment is still pending.

Keywords: approval workflow; higher education; policy validation; context-aware routing; business process compliance; document management

1 Introduction

Take a faculty member who wants to move an examination slot. The change cannot take effect until the Head of Department (HOD) signs it and, because examinations are a college-level matter, until the Controller of Examinations (COE) and the Principal have signed as well. In practice the document travels by email, by hand or through a messaging group, and each person who receives it decides what happens next. At our institution this produced familiar problems: files sitting in inboxes for days, documents pushed through stages they did not need, and rejections that came back without saying what to fix. Nobody could say with confidence how far an approval had got.

A fixed digital workflow solves the visibility problem but not the routing problem. When every document follows the same sequence, a departmental notice waits on officials who have no reason to see it, while a proposal that needs an academic committee may never reach one. The route has to depend on the document.

Letting software choose the route brings its own hazard. A generated route can omit a mandatory approver, put stages in the wrong order or assign a task to someone without the required role, and once executed it looks complete. EduSphere AI is built around this tension. It generates a candidate

route from the document's context, then tests the candidate against institutional policy before anything runs. The system proposes; the institution decides. No approval is automated.

Our central question is whether context-aware routing with explicit policy enforcement can cut unnecessary approval effort without weakening mandatory authorisation. The paper makes three contributions:

- A pipeline that separates route generation from route acceptance, so a generated route is never trusted until it passes a policy check (Section 4).
- A policy-record schema and validation procedure whose failures are explained by naming the missing role, the rule that requires it and the fix (Section 4.4).
- An evaluation design in which efficiency counts only when mandatory approvers remain, plus an analysis of expected routes showing what fixed routing costs (Sections 5 and 6).

2 Related work

Workflow foundations. A workflow can be modelled as a directed graph of activities and permitted transitions, and recurring control-flow constructs have been catalogued as workflow patterns [1, 2]. One fixed graph is adequate when an institution handles a single kind of document. It breaks down when types, scopes and importance vary, which is the normal situation in higher education.

Approval in higher education. Velásquez Angamarca et al. describe a formal document-approval workflow introduced during a private university's digital transformation in Ecuador, and show that structured workflows bring traceability and accountability [3]. We share that motivation but ask a different question: how to generate routes that respond to document context, and how to enforce them.

Compliance. Business-process compliance research specifies constraints formally and checks processes against them. Elgammal et al. map compliance patterns to temporal-logic rules [4], and Ly et al. survey what compliance-monitoring tools can do [5]. EduSphere AI is deliberately lighter: it stores institutional rules as structured policy records and checks a candidate route against them, without temporal-logic verification.

Route generation. Classification and extraction can recover routing-relevant properties such as document type, named authorities and referenced departments, and process mining can suggest routes from historical logs [6]. Neither guarantees that a suggested route matches current policy, and logs may record unofficial shortcuts. We therefore treat every generated route as a candidate only.

Gap. Workflow automation, electronic signatures and document management are each well studied. The gap we address is their combination with document-context analysis and pre-execution policy checks in a higher-education setting, evaluated against manual and fixed-route baselines on compliance as well as speed.

3 Problem formulation

A document is $d = (x, m, c)$: content x, metadata m (creator, department, upload date) and context c extracted from both. A candidate workflow G is a directed graph of approval stages for d, and P is the set of active institutional policies. G may execute only if it satisfies every applicable policy, written $G \models P$. Among the valid candidates the system prefers the cheapest:

$$G^{*} = \underset{G \vDash P}{\operatorname{argmin}} \left[\alpha \cdot T(G) + \beta \cdot S(G)\right] \quad (1)$$

where $T(G)$ is expected processing time, $S(G)$ the number of approval stages, and α, β are configurable weights. Constraints come first; cost is compared only among valid routes. The prototype approximates Equation (1) with rules and does not run an optimiser.

Hypotheses. H1: context-aware routing with validation lowers mean processing time relative to manual circulation. H2: it sends documents through fewer unnecessary stages than a fixed digital route. H3: validation gives full compliance with mandatory approval requirements on the tested set. H4: validation improves workflow validity compared with context-based routing without it. A fifth question is descriptive: which document types are ambiguous enough that a person must review them?

4 The EduSphere AI framework

4.1 Pipeline

Figure 1 shows the flow. A user uploads a PDF or DOCX file; the system records identifier, creator, department, declared category, version and timestamp, and keeps the original apart from any extracted text. The context engine fills the fields described below, the generator proposes a route, and the policy engine checks it. Only a validated route reaches execution. A route that fails, or a document whose classification confidence is low, goes back for correction or to a human reviewer. The prototype uses React, FastAPI and PostgreSQL in Docker containers; the target design splits the work into seven services, shown in Figure 2.

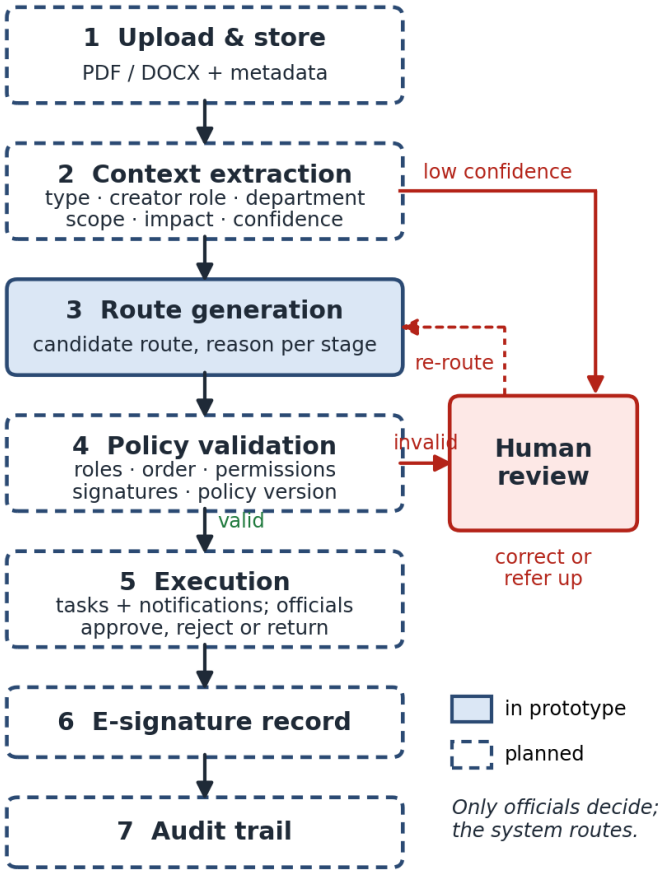


Figure 1. EduSphere AI pipeline. The shaded stage is what the prototype currently implements (basic rule-based routing); role-based login and the front-end sit alongside it. Dashed stages are planned.

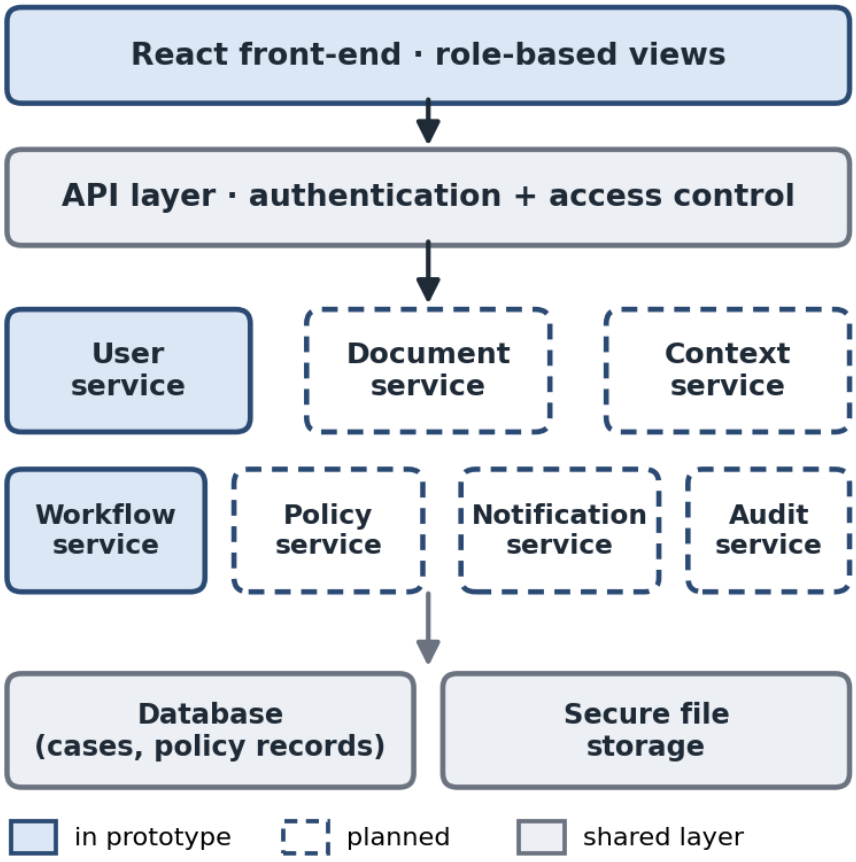


Figure 2. Target architecture. Only the front-end, the user service (role-based login) and basic workflow routing exist in the prototype; everything dashed is still to be built.

4.2 Context extraction

For each document the engine tries to fill ten fields: document type, creator role, department, institutional scope, purpose, impact level, deadline, current responsible authority, referenced offices, and a classification confidence between 0 and 1. An examination schedule from a computer-engineering faculty member might come out as type = Examination Schedule, scope = College, impact = High, confidence = 0.91. The confidence value only decides whether a person should look at the classification. It says nothing about policy approval. Documents that fall below a threshold τ skip route generation and go straight to a person; τ will be set from errors on the development set rather than guessed.

4.3 Route generation

Every stage in a candidate route carries the role assigned to it and the reason it is there. A departmental notice from a faculty member goes to the HOD. An examination schedule goes HOD → COE → Principal, with the COE stage recorded as examination-related academic review and the Principal's as final institutional authorisation. An institutional notice from an HOD goes straight to the Principal. Recording reasons keeps the route inspectable and gives both the validator and the officials receiving tasks something to check against.

4.4 Policy validation

Institutional rules live in policy records (Table 1). Before execution the engine checks that (i) every mandatory role appears, (ii) roles appear in the required order, (iii) each assigned user holds the

permission for the stage, (iv) workflow scope matches document scope, (v) no stage is duplicated, (vi) required signatures are included, and (vii) the policy version applied is the active one.

Table 1. Policy record fields and an example.

Field	Example (examination schedule)
Policy ID	POL-EXAM-001
Document type / scope	Examination schedule / College
Required roles	HOD, COE, Principal
Order constraint	HOD → COE → Principal
Action	Approve and sign
Exception	Conditions allowing a modified route
Effective date / version	Held per record; only the active version applies
Status	Active

Algorithm 1 (Table 2) gives the logic. In the prototype design it is Python rule evaluation against a policy database; database queries, graph traversal or a dedicated rule engine are alternatives if performance demands it.

Table 2. Algorithm 1: pre-execution validation.

Algorithm 1. Pre-execution validation
in: document d, candidate route G, policies P
$c \leftarrow$ extract_context(d)
if confidence(c) < τ: **return** HUMAN_REVIEW
$P_c \leftarrow$ active policies in P matching c
reasons ← [ ]
for each policy p in P_c:
if role_missing(G, p): add reason
if order_violated(G, p): add reason
if user_unauthorised(G, p): add reason
if signature_missing(G, p): add reason
if reasons is empty: execute(G)
else: return_for_correction(G, reasons)

When a route fails, the engine does not return a generic error. It names the missing role, the policy that requires it and the correction needed, and shows this to the person who submitted the document (Figure 3). Because the checks read from records, the design lets rules be revised as data; Section 8 explains what that does and does not guarantee.

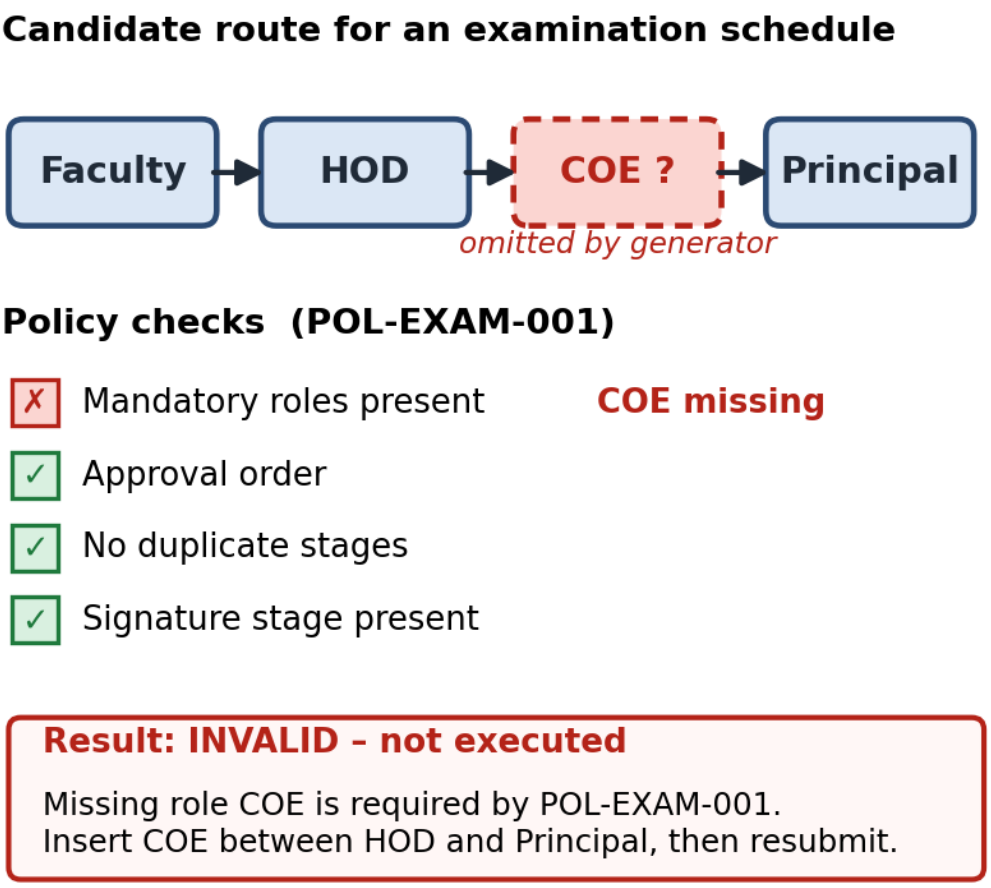


Figure 3. Illustrative example on a hand-built candidate route: a generated route that omits the COE is stopped before execution and the submitter is told why.

4.5 Execution, signature and audit

After validation, the execution service creates one task per approver, in sequence; a stage opens only when earlier required actions are complete. Each task links the document, the required action, the deadline, the status of earlier stages and the policy reason for that stage, and offers approve, reject, return for correction, or sign. The next approver is notified whenever the workflow advances. The signature service records signer, role, timestamp, document version and verification status. This is electronic-signature support, not a legally certified signature. The audit service logs every event, from document creation and context extraction through validation, each approval and final completion, with actor, timestamp, previous and new status, policy result and any comment. Figure 4 shows the event sequence for a completed examination schedule.

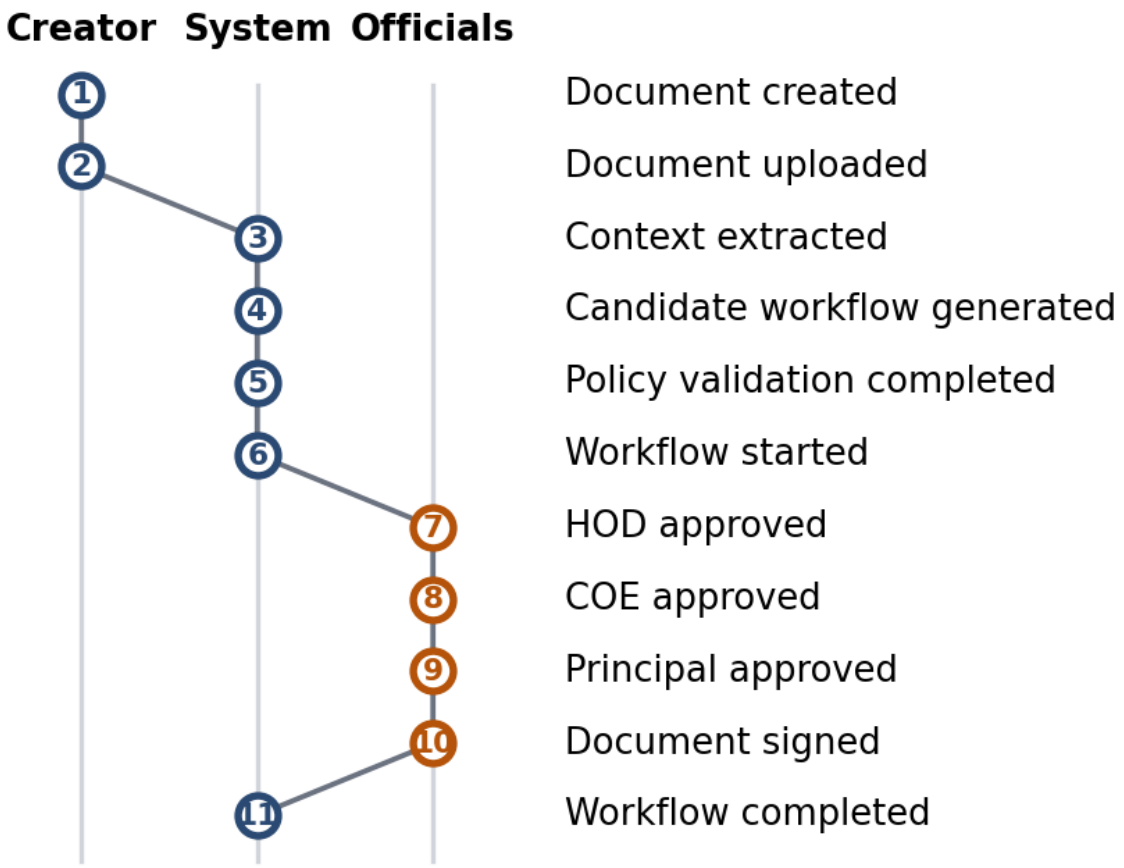


Figure 4. Audit trail of a completed examination schedule, by who acts. Every event also stores a case identifier, previous and new status, and policy result.

5 Evaluation design

5.1 Conditions and dataset

Three conditions run on the same cases against the same official requirements: manual circulation (email, paper, informal channels), a fixed digital route (one sequence for every document), and EduSphere AI. The dataset spans eight categories: department notice, timetable change, examination schedule, institutional notice, academic proposal, event permission, resource request and administrative circular. It includes correctly routed cases, correctly rejected cases, and cases where the right route differs from the fixed one. Expected routes are confirmed with authorised officials before they serve as ground truth. Each case records its policy version, exceptions are kept apart from standard cases, and all documents are anonymised and split into development and evaluation sets.

5.2 Metrics

Table 3 lists the metrics. Processing time is reported as mean and median because a few slow cases can skew the mean. The stage-reduction figure counts only when mandatory approvers remain, since a shorter route that drops a required authority is not a valid result. A non-zero human-intervention rate is not automatically a failure; what matters is whether the cases that trigger review are the ones that should.

Table 3. Evaluation metrics.

Metric	Definition	Tests
Processing time	$T = t_{\text{final approval}} - t_{\text{creation}}$; mean and median	H1
Stage reduction	$R_s = (S_b - S_p)/S_b \times 100$; valid only if mandatory approvers remain	H2
Policy compliance	C = compliant workflows/tested workflows × 100	H3
Workflow validity	V = correctly routed/tested × 100 (roles, order, users, end state)	H3, H4
Human intervention	H = cases needing manual correction/tested cases × 100	Fifth question

5.3 Procedure

We prepare the labelled dataset and confirm expected routes with officials, fix the start and end points for timing, run every case through the three conditions, log times, stage counts, validation outcomes and every rejected, returned or manually reviewed case, and compute the metrics. Results will be read within the limits of dataset size, document categories and assumptions.

5.4 What counts as evidence

Each hypothesis has a stated condition for support and one that counts against it, fixed before any data are collected (Table 4).

Table 4. Decision rules for the hypotheses.

Hypothesis	Supported if	Counts against if
H1	Mean and median time are lower than manual in most categories	No difference, or gains appear only in skipped stages
H2	Fewer unneeded stages than the fixed route, all mandatory approvers still present	Shorter routes only because a required approver was dropped
H3	C = 100% on the tested set	Any executed workflow lacks a mandatory approver
H4	V is higher with validation than without	No gain, so the validator adds nothing

6 Preliminary analysis and status

6.1 What a fixed route costs

Before any experiment we can say something about the two routing approaches from the expected routes alone. Assume the fixed baseline is one sequence, HOD → COE → Principal, sized for the most demanding common case, and count approval stages excluding the creator. Table 5 lists the expected routes, and Figure 5 compares them with the fixed one.

Table 5. Expected approval routes (to be confirmed).

Document	Creator	Scope	Expected route
Department notice	Faculty	Dept.	Faculty → HOD
Timetable change	Faculty	Dept.	Faculty → HOD
Examination schedule	Faculty / COE	College	HOD → COE → Principal

Document	Creator	Scope	Expected route
Institutional notice	HOD	College	HOD → Principal
Academic proposal	Faculty	College	HOD → Academic Committee → Principal

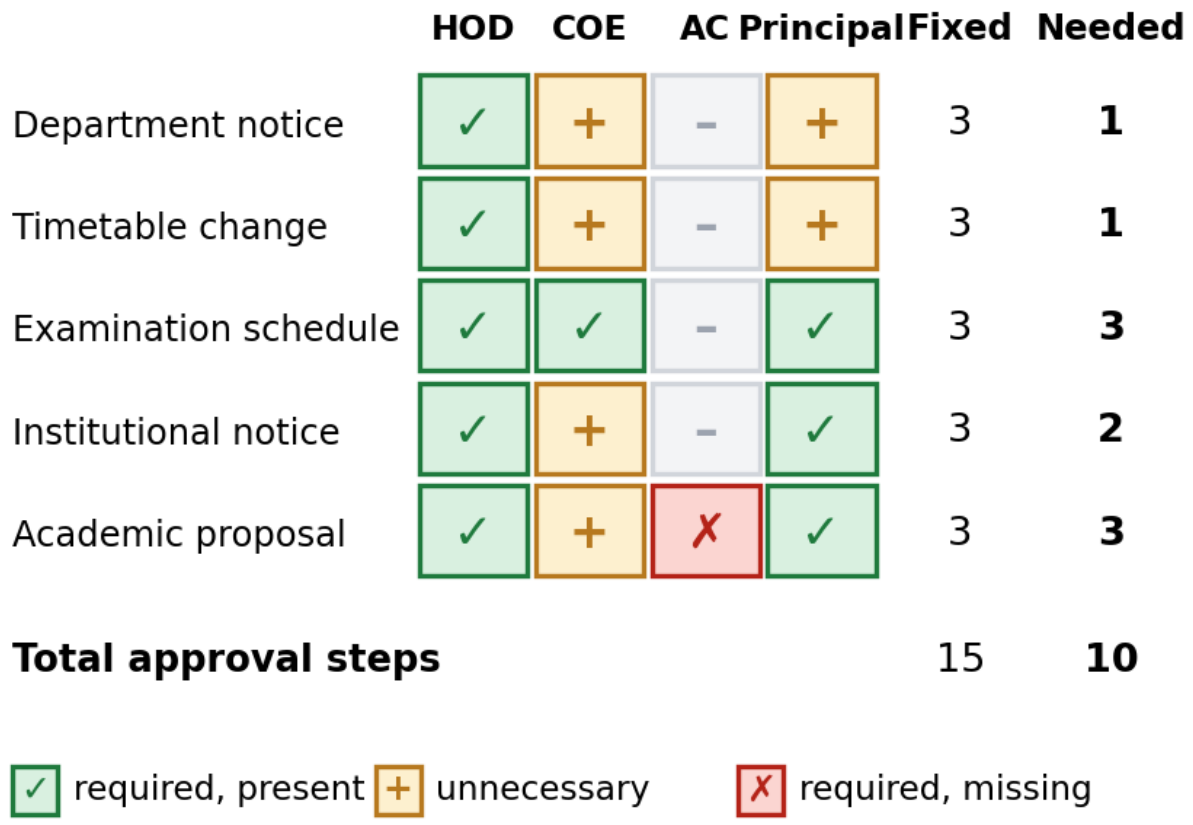


Figure 5. Approval roles required for five document types versus the roles a single fixed route (HOD → COE → Principal) would use. AC = Academic Committee. Analysis of expected routes, not measured data.

The fixed route adds an unnecessary approval in four of the five types (two each for department notices and timetable changes, one each for institutional notices and academic proposals), and for academic proposals it never reaches the Academic Committee, a required authority. Over the five types it uses 15 approval steps against 10 needed, a reduction of 33% (R_s = 5/15) that passes the validity rule because every mandatory approver remains.

This is an analysis of the design, not a measured result. It gives each document type equal weight, so real savings will depend on the institution's document mix, and the fixed baseline is our own assumption. The expected routes still await confirmation by authorised personnel, and nothing here speaks to time, which only the experiment can measure.

6.2 Prototype status

The prototype currently has role-based login, basic routing logic that maps document categories to expected approval sequences, and a front-end that displays mock workflows. Context extraction, policy validation, notifications, e-signature and audit logging are not yet implemented. Results will therefore be reported after the experiment, including any hypothesis the data does not support.

7 Where the design could fail

Separating proposal from acceptance does not make the system infallible; it moves the risk to places we can name. Table 6 lists the failure modes we expect and how the design responds. The first two matter most: a confident misclassification selects the wrong policy, and an incomplete policy record lets a bad route pass. Neither can be removed by software alone, which is why the pipeline keeps officials in the loop and requires them to confirm the policy records.

Table 6. Expected failure modes and design responses.

Failure mode	Effect	Design response
Confident misclassification	Wrong policy applied	Reason shown per stage; officials can return the route
Incomplete policy record	Violation passes unnoticed	Records confirmed by officials before use
Outdated policy version	Route valid under a retired rule	Check that the applied version is active
Unusual or exceptional case	Standard route unsuitable	Exception field; low-confidence cases go to a person
Approver unavailable	Stage stalls	Reminders and escalation notices

8 Limitations and ethics

Limitations. The data will come from one institution, so results should not be carried to institutions with different approval cultures without further study. Validation is only as good as the encoded policies: a constraint missing from a record is a violation the engine cannot catch, so records must be confirmed by authorised personnel rather than inferred from informal descriptions. Document interpretation is unreliable when a document has several purposes, ambiguous wording or no identifying metadata, and the confidence threshold for human review must be calibrated empirically. Historical routes may include shortcuts and outdated stages, so ground truth will come from active policy, not past practice. The signature module does not meet certified-signature requirements in any jurisdiction and cannot be used where those are legally required until it is replaced. Timing will be affected by user response speed, notification latency and workload, and these will be documented.

Ethics and security. The documents may contain information about staff, students or academic decisions. Access is role-based [7]: users see only the tasks assigned to them and the documents those tasks concern. Every action requires authentication and is logged, transfers are encrypted, and earlier versions are preserved. Experimental documents will be anonymised, and real ones used only with explicit permission. The system never approves anything; every approval, rejection and signature is a human act.

9 Conclusion and future work

EduSphere AI separates proposing a route from accepting it: context analysis suggests, policy validation verifies, and officials decide. Our analysis of expected routes shows that a single fixed route both over-routes simple documents and under-routes complex ones, which is the gap the framework targets. Next we will finish context extraction, policy validation, notifications and audit, build the labelled dataset with institutional officials, run the three-way comparison, calibrate the review

threshold, and swap in a certified signature service if needed. We will report the findings whether or not they support our hypotheses.

References

[1] van der Aalst WMP, ter Hofstede AHM, Kiepuszewski B, Barros AP. Workflow patterns. *Distrib Parallel Databases*. 2003;14(1):5–51. https://doi.org/10.1023/A:1022883727209

[2] Dumas M, La Rosa M, Mendling J, Reijers HA. *Fundamentals of business process management*. 2nd ed. Berlin: Springer; 2018. https://doi.org/10.1007/978-3-662-56509-4

[3] Velásquez-Angamarca V, Arce Cuesta DC, Oyola-Flores C, García-Pesántez A, Pesántez-Avilés F, Cárdenas Tapia JA. A formal document approval workflow for business process in higher education institutions. In: Rocha A, et al., editors. Proceedings of the 19th Iberian Conference on Information Systems and Technologies (CISTI 2024), vol. 2. Cham: Springer; 2026. p. 74–83. https://doi.org/10.1007/978-3-032-12879-9_8

[4] Elgammal A, Turetken O, van den Heuvel WJ, Papazoglou M. Formalizing and applying compliance patterns for business process compliance. *Softw Syst Model*. 2016;15(1):119–146. https://doi.org/10.1007/s10270-014-0395-3

[5] Ly LT, Maggi FM, Montali M, Rinderle-Ma S, van der Aalst WMP. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Inf Syst*. 2015;54:209–234. https://doi.org/10.1016/j.is.2015.02.007

[6] van der Aalst WMP. *Process mining: Data science in action*. 2nd ed. Berlin: Springer; 2016. https://doi.org/10.1007/978-3-662-49851-4

[7] Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. *Computer*. 1996;29(2):38–47. https://doi.org/10.1109/2.485845