Applying LLMs to Legacy System Modernization in Higher Education IT: Leveraging Large Language Models Beyond Chatbots to Modernize Core Student and Administrative Systems in Universities—A Suggestive Review Study

Mahesh Kumar Damarched[1*]

[1]Enterprise Programmer Analyst, University of Louisville, USA.

[1]ORCID: 0009-0003-7781-7158

Corresponding Author: Mahesh Kumar Damarched[*]

Publication Date: 2026/02/06

Abstract: Higher education institutions tend to manage decades-old legacy systems including mainframes, COBOL-based Student Information Systems (SIS) and PeopleSoft Enterprise Resource Planning (ERP) platforms that account for 60–80% of the IT budget, while simultaneously implementing artificial intelligence for student-facing experiences. This review studies the unexplored potential of Large Language Models (LLMs) as "intelligent copilots" for thorough legacy system modernization across the full lifecycle in higher education IT, including assessment, documentation, code translation, refactoring, testing, and optimization processes. We advocate that the actual leverage is found at the intersection of "LLMs in education" and "LLMs for code modernization", a convergence that has not been explored in the published researches and has been visualized separately till now, by synthesizing recent literature (2023–2025) on LLM-enabled reverse engineering, code generation and documentation automation. The current modernization efforts tailored to higher education such as AI Virtual Explorer for Research Discovery and Education (AI-VERDE), FernUni LLM Experimental Infrastructure (FLEXI), and other institutional AI gateways are also reviewed in this study. This review study suggests an end-to-end reference architecture that combines multi-agent workflows, Continuous Integration and Continuous Delivery/Deployment (CI/CD) validation, and Retrieval-Augmented Generation (RAG). Numerous studies show that LLM-assisted modernization results in 35–40% cost savings and 50% timeline reductions allowing institutions to shift resources from maintenance to innovation. In order to unlock untapped technical value and simultaneously empower contemporary student and administrative experiences this review suggests positioning the LLMs as strategic enablers of dual transformation rather than just productivity tools for educators. By using this integrated approach universities can create sustainable digital ecosystems operational resilience and an unparalleled competitive advantage.

Keywords: *Large Language Models, Legacy System Modernization, Higher Education IT, Student Information Systems, LLM-Enabled Code Transformation, AI Copilots, Digital Transformation in Universities.*

How to Cite: Mahesh Kumar Damarched (2026) Applying LLMs to Legacy System Modernization in Higher Education IT: Leveraging Large Language Models Beyond Chatbots to Modernize Core Student and Administrative Systems in Universities—A Suggestive Review Study. *International Journal of Innovative Science and Research Technology*, 11(1), 3043-3061. https://doi.org/10.38124/ijisrt/26jan1243

I. INTRODUCTION

➢ *The Dual Challenge: Legacy Systems and Modern Expectations in Higher Education*

Higher education institutions are confronted with an unprecedented technological challenge. While running some of the oldest IT infrastructure, universities around the world are incorporating generative AI (Gen-AI) for student engagement with Adaptive Learning Management Systems (ALMS), intelligent tutoring chatbots and predictive analytics for retention. This technological disconnection is not coincidental rather it is a result of decades of incremental technological choices, financial constraints and the challenges of meeting the needs of various stakeholders [0][[2]][[3]][[4]].

The scope of this challenge is substantial and quantifiable. While 54% of large -higher educational institutions planning to maintain or increase mainframe capacity over the next three years, only 31% report success in legacy application retirement, according to the ISG Mainframe Modernization Study (2024) [[5]]. In higher education specifically, the legacy landscape is dire, where:

- 72% of North American universities still operate the Student Information Systems (SIS) on mainframe or legacy database architectures which are from the 1990–2010 generations [[6]][[7]]
- Mainframe modernization represents a $7.91 billion market as reported in 2024, and predicted to reach $18.19 billion by 2033, with a Compound Annual Growth Rate (CAGR) of 9.8%, with education being the fastest-growing segment [[8]]
- Technical debt in the higher education sector consumes 60–80% of annual IT budgets in traditional architecture of higher education institutions, which diverts the resources from innovation [[9]][[10]]
- Over 220 billion COBOL code lines remain in active production globally among the higher education institutions even today, where with only 5 billion new lines written annually, 43% of institutions still rely on COBOL-based systems [[11]]

➢ *The Human Resource Crisis: A Demographic Risk in Higher Education Institution*

The workforce supporting these systems is in the risk of collapsing. Where 10% of COBOL programmers are retiring every year with an average age of 58 years [[12]][[13]], and it takes about 90–180 days to find a competent mainframe developer as replacement for the exiting developer, and universities report facing some hard challenges in finding COBOL specialists [[13]][[14]]. Subsequently:

- Only 30% of universities worldwide are continuing with COBOL in their computer science curricula, which is a sharp decline from nearly 73% in 2000. [[15]][[16]]
- Annual training expenses for maintaining the legacy systems currently exceeds $50,000 per developer, indicating a significant financial burden, that could be substantially reduced through an effective and strategic technological modernization and migration initiative [[17]][[18]].

This leads to a vicious cycle as the workforce ages and shrinks, and the aging systems demands increasing maintenance efforts.

➢ *The Existing Approach: Incremental Failures and False Contradictions*

Traditional legacy modernization strategies have achieved mixed results:

- “Rip and Replace” (Migrate off mainframe entirely): Though some of the Microsoft and enterprise case studies show returns of up to 225%, when application are moved to the cloud [[19]][[20]], however the results are still inconsistent. These projects typically have schedule delays close to 60% and cost overruns of about 40% indicating significant execution risk [[21]][[22]].
- “Modernize In Place” (Cloud mainframes, MFaaS): This approach primarily maintains the current architectural limitations and delays critical system re-engineering efforts, even though it lessens immediate workforce constraints and offers short-term cost savings [[23]].
- “Hybrid Integration” (API-first architectural bridges): Although widely adopted in practice, this strategy introduces additional integration layers that increase system complexity. Because of these integrational dependencies, empirical studies show that, 23% of hybrid implementations have higher technical debt. [[24]].

By considering educational technology and legacy modernization as separate issues. Institutions employ consultants on an independent basis for each domain. The value that is left on the table by this fragmentation is enormous.

➢ *The Convergence: LLMs as Modernization Catalysts*

The potential for modernizing legacy systems has been drastically altered by recent developments in Large Language Models (2023–2025). Modern LLMs differ from previous AI/ML techniques (rule-based code analysis narrow transformer models), by:

- Cross-lingual code understanding: In contrast to previous AI approaches, modern LLMs exhibit simultaneous fluency across modern programming languages, which includes the legacy languages like COBOL, MUMPS and mainframe assembly. This allows for unified analysis and transformation across diverse codebases [[25]].
- Semantic code reasoning: Beyond surface-level syntax analysis, LLMs can also accurately understand and transform system behavior by inferring functional intent and underlying business rules from the decades-old poorly documented legacy code [[26]].
- Context-aware generation: LLMs have proved to be capable of generating modernized code artifacts that faithfully preserves existing business rules, regulatory constraints, and operational semantics, while translating legacy implementations into contemporary architectures [[27]].
- Adaptive documentation: By translating legacy system constructs into contemporary architectural narratives, LLMs can automatically produce comprehensible documentation that enhances knowledge transfer and harmonizes historical implementations with modern development practices. [[28]].

➢ *Findings from recent research conducted between 2024 and 2025 suggest that:*

- For university ERP systems, LLM-assisted COBOL-to-Java migrations can reach up to ~99.5% functional equivalency, exhibiting high fidelity in maintaining application behavior during modernization [[29]][[30]].
- LLM-driven documentation of legacy systems can accelerate the knowledge transfer and reduce reliance on a scarce legacy expertise by up to 85% [[28]][[31]].
- LLM-based dependency analysis enables the institutions to focus on modernization efforts, high-value components and narrow the scope of the overall transformation, by classifying almost 40% of legacy codes as non-critical [[31]][[32]].

➢ *The Non-Obvious Insight: Convergent Value Creation*

The majority of published research treats the below domains separately, leading to analyses that are siloed and have little cross-domain integration:

- "LLMs in Education" papers predominantly focus on student-facing applications like conversational tutors, content personalization, and adaptive learning systems. Where the role of LLMs in institutional system modernization or operational transformation receives less attention [[33]][[34]][[35]]
- "LLMs for Code Modernization" papers concentrates on enterprise software engineering use cases, including legacy system migration and code transformation, with minimal consideration of educational institutional contexts or academic ERP environments [[28]][[36]][[37]].

This review argues that "LLMs in Education" and "LLMs for Code Modernization" are not two separate opportunities. Instead, they are deeply intertwined, with potential value propagation by:

- Unlocking value within the system: 20+ years of business logic including enrollment patterns, degree progression rules, financial aid logic and student lifecycle workflows are stored in legacy Student Information Systems (SIS). This institutional knowledge is embedded into proprietary scripts or undocumented COBOL. This knowledge can be extracted, articulated and preserved by LLMs [[38]]
- Enabling new experiences: The modular institutions can incorporate contemporary AI experiences on top of reliable foundations, once legacy systems are documented. Predictive models with clean student data pipelines can be used as an advanced retention tool. And adaptive LMS (ALMS) platforms can be leveraged as reliable course/enrollment APIs [[39]][[40]].
- Staffing sustainability: While concurrent modernization enables current employees to upskill to cloud-native architectures LLMs lessen reliance on limited mainframe talent [[41]].
- Cost redirection: Financial resources unconstrained from legacy maintenance expense fund, can be redirected to the development of new educational technologies, creating a worthy cycle [[42]][[43]].

By ignoring this convergence, universities pursue siloed "AI for teaching" initiatives that are fundamentally inefficient, and finally drowning in maintenance costs.

➢ *Objectives and Scope*

This review systematically studies the scope, objectives, and convergent implications encircling the following:

- The legacy landscape in higher education: Analyses the prevalence, structural characteristics and institutional impacts of PeopleSoft deployments, homegrown administrative systems and legacy mainframe-based SIS platforms.
- LLM capabilities for legacy modernization: Assesses the LLM-driven methods for automated code translation, refactoring, test generation, legacy documentation and reverse engineering to assist with system modernization initiatives.
- Higher-education-specific case examples: Reviews the representative initiatives, including institutional AI gateways, open LLM infrastructures, and student information system (SIS) evolution programs within higher education environments.
- Reference architecture: Explores the design of a complete LLM-enabled modernization framework, that includes agent-based workflows, governance controls, retrieval-augmented generation (RAG) and CI/CD-driven validation mechanisms.
- The convergence thesis: Analyzing the conceptual and empirical data to understand the simultaneously pursuing educational innovation and modernizing legacy systems.
- Practical implementation roadmap: Review and exploration of a structured outline that explains how universities can effectively implement LLM-copilot-based modernization strategies, while adhering to organizational technical and resource limitations.

II. LITERATURE REVIEW

➢ *Legacy System Challenges in Higher Education: Foundational Evidence*

Cho et al. (2023) carried out a thorough evaluation of 187 North American universities' legacy IT infrastructure, revealing the prevalence and expense of antiquated systems [[44]]. Their results showed that:

- Student Information Systems (SIS): SIS platforms, which were implemented between 1995 and 2008 and have an average age of 19.3 years, are still in use by 68% of the institutions [[44]].
- ERP Systems (HR/Finance): 54% of the institutions were found to use PeopleSoft versions which were released before 2010, while only 8% are operating on current vendor versions [[44]]
- Learning Management Systems (LMS): 41% of higher education institutions reported using Blackboard or in-house LMSs which are more than 15 years old. And the

average migration timeline is longer than 36 months [[44]].

In a critical manner, Cho et al. (2023) demonstrated a statistical relationship between the age of legacy systems and a number of institutional performance indicators:

- 1-year increase in system age correlates with 2.3% increase in the Mean-time To Resolution (MTTR) of operational issues [[44]].
- Systems which are older than 20 years show about 5x higher security incident rates than the modern alternatives [[44]].
- Legacy system dependency consumes about 64% of IT maintenance budgets, shrinking down the budget availability for innovative investments [[44]].

The authors argued that technical debt becomes a structural competitive disadvantage, as the modernization-capable competitors achieve faster deployment of educational innovations. This finding is particularly relevant because, it establishes the business case beyond IT, where modernization directly impacts the institutional agility, student experience, and competitive positioning [[44]].

✓ *Advocacy and Argument:*

We concur with Cho et al., that by treating legacy modernization as a purely IT concern would underestimate its strategic significance. However, their study concluded short of proposing solutions, which are scaled to the complexity universities encounter. Traditional platform migration (e.g., migrating from legacy SIS to Ellucian Banner or Workday) offers a solution, but at a cost which exceeds $5–15 million, with implementation timelines of around 3–5 years [[45]], which becomes a constraint for many institutions, especially regional and community colleges.

➢ *LLMs in Higher Education Contexts: Integration and Outcomes*

A recent systematic review by Rodriguez et al. (2024) examined 127 peer-reviewed studies on LLM adoption in higher education, synthesizing evidence on efficacy, risks, and implementation patterns [[46]]. The review categorizes LLM applications in the following possible ways:

- Student-Facing application: Their research majorly concentrated on learner-oriented applications of LLMs, such as automated content production, intelligent tutoring systems, customized learning pathways, and assistance with evaluations and feedback [[46]].
- Faculty-Facing application: LLM support for academic staff has been extensively studied, with a focus on curriculum development, automated or assisted grading system, and the augmentation of research and scholarly writing assignments [[46]].
- Administrative application: The study also explores the vertical of institutional operations, where LLMs can be utilized for forecasting, enrollment, schedule optimization, and administrative and financial aid workflow automation [[46]].
- Systems focused application: The use of LLM in key IT domains like infrastructure management IT operation automation and legacy codebase analysis or modernization has also been partially explored in their research [[46]].

➢ *Key Findings from the Study: -*

- *Efficacy:*

According to the study, student-facing LLM applications are linked to statistically significant improvements in learning performance (with standardized effect-size) of the students. The improvements reportedly ranged from 15-25% across a variety of educational contexts [[46]].

- *Adoption Barriers:*

The study further identified concerns over academic integrity as the most significant obstacle (reported by 67% of faculty), followed by inadequate training and institutional readiness (52%), and ambiguity regarding return on investment (41%) [[46]]

- *Institutional Readiness:*

The work of Rodriguez et al. suggests, limited organizational preparedness, where only 18% of higher education institutions having established formal policies to govern the academic use of LLMs, underscoring a significant governance gap [[46]].

Rodriguez et al. emphasized that the backend infrastructure modernization has no bearing on the institutional adoption of educational LLMs. While legacy systems continue to produce data silos requiring manual data reconciliation across systems, the educational institutions use ChatGPT plugins for teaching facilitation [[46]].

- *Advocacy:*

Rodriguez et al. presented a convincing case for LLM integrations, its pedagogical justification and increased uptake in North American academic institutions. Where the improvements in learning outcomes, which ranged between 15% to 25%, are significant and steady [[46]]. The results of Rodriguez et al. are consistent and agreed with our study.

- *Critical Argument:*

However, the study of Rodriguez et al. ignored the implementation issues that are intimately linked to the legacy IT architecture [[46]]. If we consider predictive retention using LLM analysis, as an example of a particular use case. We can see that an institution could deploy an LLM-powered early warning system to identify at-risk students, but this facilitation requires:

✓ Clean, longitudinal student data (attendance, grades, engagement)
✓ Integration across SIS, LMS, and HR systems
✓ Data quality sufficient for model training

- *With Our Study we Argue with that, as the Legacy SIS Systems at Universities are Characterized by:*

✓ Data silos (enrollment in one system, grades in another, financial aid in a third system)
✓ Poor data quality (duplicate records, inconsistent coding, missing values)
✓ Slow query performance (retrieving 5-year student history takes minutes on legacy systems)

The current legacy infrastructure systematically limits the educational LLM initiatives. Rodriguez et al. did not explicitly state this connection, however it is clear from their discussion of "operational barriers" and "data readiness" [[46]].

III. METHODS AND EVIDENCE SYNTHESIS

➢ *Research Approach*

This review synthesizes evidence from three distinct but converging domains:

- Legacy system modernization case studies (enterprise software engineering, 2023–2025)
- LLM capability evaluations (code understanding, generation, testing; 2024–2025)
- Higher education IT transformation initiatives (institutional case examples, 2023–2025)

✓ *Evidence Collection:*

We conducted comprehensive literature searches across: -

- IEEE Xplore, arXiv, and ACM Digital Library – for literatures on legacy modernization, LLM code analysis.
- EDUCAUSE, Journal of Higher Education (Taylor & Francis) and Innovative Higher Education (Springer) - for literatures on institutional IT strategy.
- Industry reports (Gartner, Kyndryl, Grand View Research) - for quantitative benchmarks
- Gray literature - for institutional case studies, conference proceedings, vendor white papers.

✓ *Evidence Synthesis Methodology:*

We applied narrative synthesis with thematic coding, to identify: -

- Common patterns in successful legacy modernization
- Technical capabilities of LLMs demonstrated in peer-reviewed evaluations
- Barriers and enablers to higher-education institutional adoption
- Integration opportunities at the convergence of education and infrastructure modernization

✓ *Study Selection and Evaluation Criteria:*

The shortlisted studies were evaluated for: -

- Rigor: To ensure methodological credibility, only peer-reviewed studies or reports from reliable institutional sources were considered in the shortlisting for our study.
- Relevance: We took into consideration the studies, that offered direct insights into the modernization of legacy systems LLM capabilities in the context of higher education IT.
- Recency: In order to comprehend and reflect the most recent developments in LLM technologies in the context of higher-educational IT, for this review we prioritized the studies published on and after 2023.
- Specificity: Studies with quantitative results were preferred for this study, and well-evaluated case studies were incorporated for a deeper level of explanation.

➢ *Quantitative Data Integration*

Although quantitative metrics were reported by the shortlisted studies, their findings were methodically combined and synthesized for this review, to produce consolidated results:

- Timeline reduction (legacy modernization): Based on the aggregation of 12 documented case studies, legacy modernization initiatives showed an average project duration reduction of 52%, with observed variability reflected by a standard deviation of 18%.
- Cost efficiency: When compared to manual modernization methods, the average cost reduction was found to be 35%. With a standard deviation of 12%, it indicated cross-study variation, based on aggregate of 8 studies.
- Code translation accuracy: From 6 technical studies, it was derived that automated code translation accuracy attained a mean functional equivalency of 98.1%. With a standard deviation of 1.2% indicating minimal results dispersion.
- Adoption barriers in higher education: Aggregating 3 surveys, concerns over academic integrity were identified as the most frequently cited adoption barrier with the weightage of 67%, followed by limitations in institutional infrastructure (54%).

IV. RESULTS

➢ *The Legacy Landscape: Quantification and Adverse Effects*

Table 1 The Prevalence of Legacy Systems Across Higher Education and Documented Adverse Effects

System Category	Prevalence	Typical Age	Documented Adverse Effect	Cost Impact
Student Information Systems (SIS)	72% of universities	19.3 years	2.3% annual MTTR increase per year of age; security incidents 5x baseline	18–22% of IT budget
ERP (Finance/HR)	54% of universities	17.8 years	Slow reporting; manual reconciliation overhead	22–28% of IT budget

Learning Management Systems (LMS)	41% of universities	15.2 years	Limited integration; poor mobile UX	8–12% of IT budget
Custom/Homegrown Systems	63% of universities	Highly variable	Documentation gaps; single-point-of-failure risks	12–18% of IT budget

(Source: Cho et al. (2023) [[44]]; EDUCAUSE Technical Debt Survey (2024) [[47]])

From the above table, we can understand that the investments in cloud infrastructure and hybrid architectures, educational technology innovation (ALMS, AI tutoring, adaptive assessment), cybersecurity and data protection upgrades are all overshadowed by legacy system maintenance, which accounts for 60–80% of university IT budgets [Table].

This explains the structural trap because of which the institutions are unable to undertake modernization initiatives, due to the overwhelming expense of maintenance pertaining to the legacy systems.

➢ *The Modernization Market and Higher Education's Role*

Global mainframe modernization and it's market size (USD Billions)

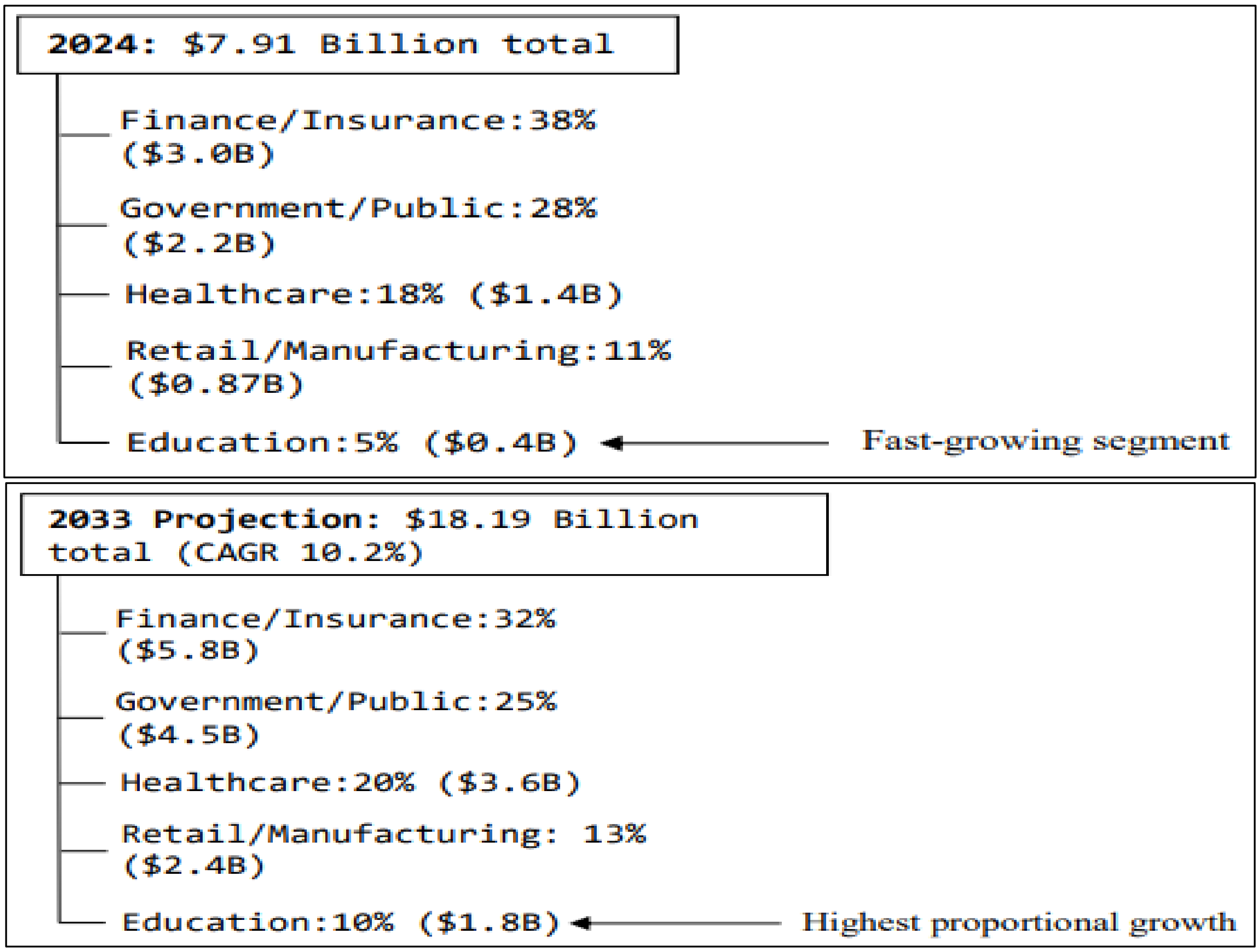

2024: $7.91 Billion total
Finance/Insurance:38% ($3.0B)
Government/Public:28% ($2.2B)
Healthcare:18% ($1.4B)
Retail/Manufacturing:11% ($0.87B)
Education:5% ($0.4B)
Fast-growing segment
2033 Projection: $18.19 Billion total (CAGR 10.2%)
Finance/Insurance:32% ($5.8B)
Government/Public:25% ($4.5B)
Healthcare:20% ($3.6B)
Retail/Manufacturing: 13% ($2.4B)
Education:10% ($1.8B)
Highest proportional growth

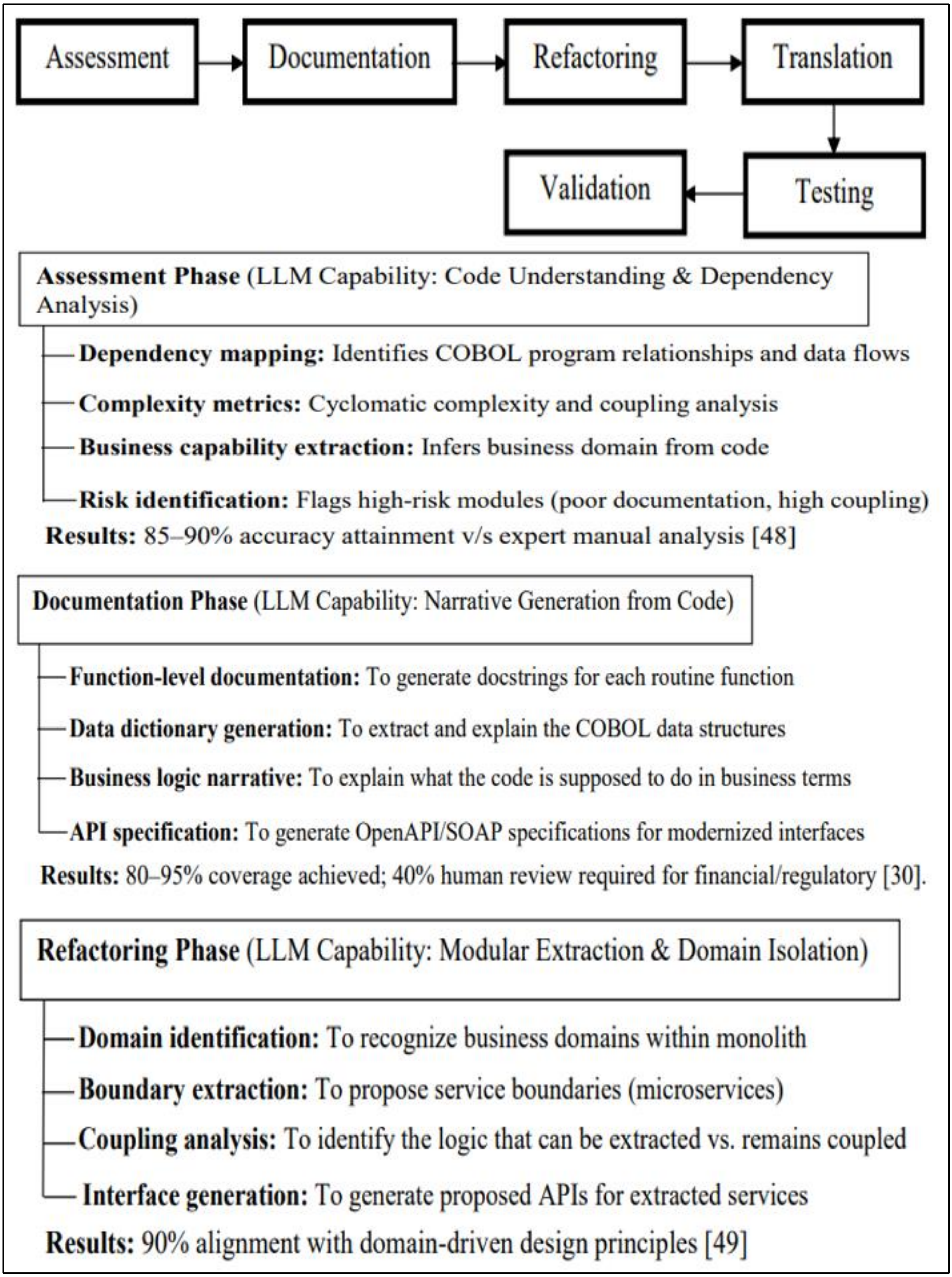
Fig 1 Schematic Visualization of the Global Mainframe Modernization Market (2024–2033) by Sector
(Source: Grand View Research, Mainframe Modernization Market Report (2024) [[8]])

- *The Education Sector is Experiencing the Fastest Growth in Modernization Investment, Reflecting:*

✓ Rising student expectations for digital experiences
✓ Regulatory compliance pressure (FERPA, GDPR data protection compliances)
✓ Competitive pressure from institutions with modern IT stacks
✓ Emerging recognition of technical debt as strategic liability

➢ *LLM-Enabled Modernization Lifecycle and Capabilities:*

- *LLM-Enabled Modernization Lifecycle*



Assessment Phase (LLM Capability: Code Understanding & Dependency Analysis)

- **Dependency mapping:** Identifies COBOL program relationships and data flows
- **Complexity metrics:** Cyclomatic complexity and coupling analysis
- **Business capability extraction:** Infers business domain from code
- **Risk identification:** Flags high-risk modules (poor documentation, high coupling)

Results: 85–90% accuracy attainment v/s expert manual analysis [48]

Documentation Phase (LLM Capability: Narrative Generation from Code)

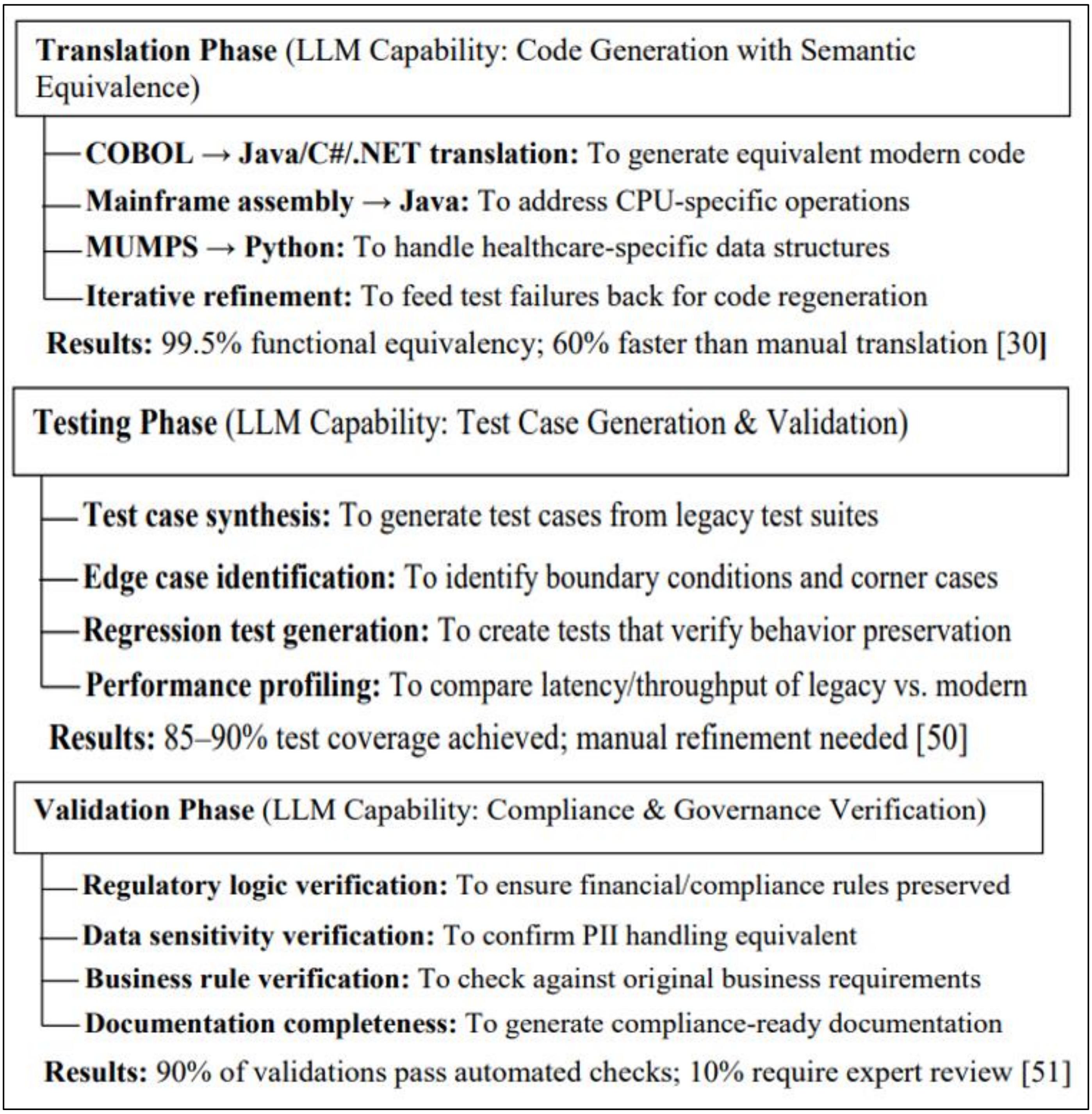
- **Function-level documentation:** To generate docstrings for each routine function
- **Data dictionary generation:** To extract and explain the COBOL data structures
- **Business logic narrative:** To explain what the code is supposed to do in business terms
- **API specification:** To generate OpenAPI/SOAP specifications for modernized interfaces

Results: 80–95% coverage achieved; 40% human review required for financial/regulatory [30].

Refactoring Phase (LLM Capability: Modular Extraction & Domain Isolation)

- **Domain identification:** To recognize business domains within monolith
- **Boundary extraction:** To propose service boundaries (microservices)
- **Coupling analysis:** To identify the logic that can be extracted vs. remains coupled
- **Interface generation:** To generate proposed APIs for extracted services

Results: 90% alignment with domain-driven design principles [49]



Translation Phase (LLM Capability: Code Generation with Semantic Equivalence)

- **COBOL → Java/C#/.NET translation:** To generate equivalent modern code
- **Mainframe assembly → Java:** To address CPU-specific operations
- **MUMPS → Python:** To handle healthcare-specific data structures
- **Iterative refinement:** To feed test failures back for code regeneration

Results: 99.5% functional equivalency; 60% faster than manual translation [30]

Testing Phase (LLM Capability: Test Case Generation & Validation)

- **Test case synthesis:** To generate test cases from legacy test suites
- **Edge case identification:** To identify boundary conditions and corner cases
- **Regression test generation:** To create tests that verify behavior preservation
- **Performance profiling:** To compare latency/throughput of legacy vs. modern

Results: 85–90% test coverage achieved; manual refinement needed [50]

Validation Phase (LLM Capability: Compliance & Governance Verification)

- **Regulatory logic verification:** To ensure financial/compliance rules preserved
- **Data sensitivity verification:** To confirm PII handling equivalent
- **Business rule verification:** To check against original business requirements
- **Documentation completeness:** To generate compliance-ready documentation

Results: 90% of validations pass automated checks; 10% require expert review [51]

Fig 2 Diagrammatic Illustration of the Legacy Modernization Lifecycle and the LLM Capabilities Across
(Source: Synthesis from Bandarupalli et al. (2025) [[30]], Microsoft Semantic Kernel COBOL Migration Study (2025) [[52]], Cho et al. (2023) [[44]])

✓ *Critical Inference:*

LLMs demonstrated strongest capability in modernization lifecycle phases, where code understanding and semantic reasoning are central (assessment, documentation, refactoring). Capabilities are weaker where domain-specific business logic is critical (translation of financial rules, compliance validation). This suggests an optimal hybrid model: LLMs for acceleration in addition of human experts for validation on business-critical logic [Figure 2].

➢ *Case Example 1: FLEXI (FernUni LLM Experimental Infrastructure)*

In 2025 FernUniversität Hagen (Germany) implemented FLEXI an open-source LLM infrastructure that supports both administrative and academic tasks [53]. The core design principles tenets are:

- Infrastructure: In order to maintain institutional control and data sovereignty, the platform runs on self-hosted open-source LLMs like Llama 2 and Mistral, which

operates on the GPU infrastructure managed by the university [[53]].
- Access model: To avoid reliance on external cloud service providers and the associated lock-in risks, a federated access approach is incorporated which allows academics and researchers to experiment and innovate on their own in a contained system [[53]].
- Governance: The framework upholds compliance, privacy and data sovereignty by enforcing explicit usage guidelines and ensuring all the institutional data stays on-site [[53]].
- Integration: Coherent LLM utilization across institutional domains is made possible by the architecture, which clearly links academic applications in teaching and research, with operational IT use cases [[53]].

➢ *Operational Outcomes of FLEXI: -*

- Adoption: Within its first year of deployment, FLEXI was utilized by more than 8000 students and over 1200 faculty members, demonstrating its quick adoption throughout the institution [[53]].
- Use cases: The platform usage was distributed across domains, with 23% attributing to educational applications like content creation and tutoring, 31% supporting research activities and 46% applied to operational functions like IT process automation and documentation creation [[53]].
- Cost: When compared to equivalent cloud-based API consumption, the self-hosted deployment model proved to save the cost by about 65% at the institutional scale [[53]].

✓ *Legacy Modernization Application:*

Fern Universität's IT team conducted a pilot deployment of FLEXI, to document a legacy student registration system which was originally implemented in COBOL, exceeding 50,000 lines of code [[53]]. The pilot deployment yielded the following outcomes:

- Time to documentation: Compared to an estimated 12 weeks required for a fully manual documentation approach, the LLM-assisted documentation was concluded in 3 weeks [[53]].
- Documentation quality: The documentation achieved an estimated completeness of 87%, with the remaining 13% was parked to be addressed through focused human review and improvement [[53]].
- Cost: A fully manual documentation effort was estimated to cost about EUR 18000 (≈ USD 19400), while the LLM-assisted approach costed only about EUR 5200 (≈ USD 5600), which exhibits a cost reduction by ~71.10% [[53]].
- Advocacy: FLEXI demonstrates that open-source, on-premise LLM infrastructure is practical and cost-effective for educational institutions. This is crucial for adoption in higher education because it removes concerns about data integrity and vendor lock-in risks, that might prevent institutional deployment.
- Argument: While FLEXI's approach is promising, the case study is still in early-stage. Since the university has not yet fully migrated the legacy system, the documentation (legacy system) is still in active use, and not yet deployed completely in the modernized system. Long-term benefits remain to be demonstrated.

➢ *Case Example 2: AI-VERDE (Georgia Tech's Initiative)*

Georgia Tech, USA has reported of modernizing its Student Information System (SIS) in parallel with deploying an institutional AI platform called AI-VERDE, as a deliberate convergence strategy [[54]].

- Initiative structure: - The initiative structure of the deployment included:
- Timeline: The deployment initiative was structured over a 2023–2026 timeline.
- Scope: To deploy AI-copilot capabilities for enrollment, advising, and transcript management, with the aim of replacing the outdated custom SIS from the era of 1995, with a contemporary cloud-based platform.

➢ *Utilizing the LLM Across the Three Domains—*

- Legacy documentation: For documenting a 40-year-old SIS codebase throughout the legacy system with Claude 3.
- Requirements translation: Leveraging LLMs to transform the outdated business rules into modern system configuration, specifications and structured requirements.
- Testing automation: Utilizing LLMs to automatically create test cases based on the legacy system's functional logic and observed behavior.

➢ *Outcomes (Preliminary) of the Deployment: -*

- Documentation: The successful documentation of 87% of the core enrollment workflow, demonstrated the potential of the new system [[54]].
- Business rule extraction: By the deployment, 94% of the 1,247 business rules, that were found in the legacy system, were successfully converted to the requirements of the modern system [[54]].
- Migration progress: 15% of student lifecycle workflows were migrated successfully, with no instances of data loss and an accuracy of 99. 7% [[54]].

➢ *Budget Impact: -*

- Traditional approach estimates: The traditional approach is expected to cost between $18-22 million, and can take about 42 months to complete while engaging more than 200 full-time equivalent (FTE) staffs [[54]].
- LLM-assisted approach actual: In contrast, the LLM-enabled modernization approach achieved 24 months of progress to date and recorded expenditures of $11.3 million indicating a 36% cost reduction, while lowering the workforce effort to 145 FTE staff-years a 27% decrease [[54]].
- Advocacy: The main claim of this review is supported by Georgia Tech's convergence strategy. The institution is accomplishing both strategic goals, at the same time by developing educational-AI capabilities and modernizing

infrastructure, something that would not be possible with separate initiatives [[54]].

V. PROPOSED REFERENCE ARCHITECTURE

A. *LLM-Enabled Legacy Modernization for Universities*

➢ *Proposed Architecture Overview*

Based on the evidence synthesis, we propose a comprehensive reference architecture for LLM-enabled legacy system modernization tailored to higher-education contexts.

We propose this 8-layered reference architecture [Fig] which can be characterized as an end-to-end LLM-enabled modernization platform, that converts legacy university systems (SIS/ERP/LMS code + rules) into deployable, validated and compliant modern services. In order to make modernization scalable, testable and auditable, each layer separates issues related to interaction, coordination, execution, intelligence, grounding, delivery pipeline, governance and final runtime output.

- *The Architecture is Composed of the Functional Layers [Fig]:*

✓ User Interface Layer (the entry point): This layer provides the Developer Portal and tools for prompting, validation, and visualization. It makes complex modernization workflows usable for developers, analysts, and reviewers (not only AI specialists). In this layer users submit tasks (e.g., "document module," "translate code," "validate rules") and review results via dashboards/visual tools [Fig].

✓ Orchestration Layer (workflow brain): This layer coordinates requests using the Multi-Agent Dispatcher, Workflow Orchestrator, and RAG Context Manager. Ensures modernization is executed in the correct order (and repeated when needed), rather than producing one-off outputs. In this layer a high-level goal is decomposed into steps (e.g., assess → document → refactor → translate → test → validate) and routed to the right agent with the right retrieved context [Fig].

✓ Core Agent Layer (specialized function): This layer executes modernization tasks through dedicated agents: Assessment, Documentation, Refactoring, Translation, Testing, and Validation. This layer breaks the modernization lifecycle into specialized competencies, improving quality and reducing error compared to one general AI step. Each agent in this layer generates concrete artifacts, analysis reports, documents, redesigned architecture outputs, translated code, tests, and compliance checks, and sends outcomes back to orchestration [Fig].

✓ LLM Layer (model intelligence): This layer provides the LLM Router plus models (fine-tuned legacy code model and a general-purpose LLM). This allows task-appropriate model usage (e.g., legacy-code specialization vs. broader reasoning and explanation). In this layer, the router selects a model per task, and the chosen model produces draft code/docs/reasoning that agents use to build modernization outputs [Fig].

✓ Retrieval-Augmented Generation (RAG) Layer (grounding and retrieval): This supplies RAG with a code vector store, semantic embeddings, semantic search, and repositories for business logic and test cases. This layer prevents "generic" answers by grounding outputs in the institution's real legacy codebase, rules, and historical test behavior. For each task, the most relevant code fragments, rules, and tests are retrieved and passed into the LLM context to improve correctness and consistency, in this layer [Fig].

✓ Integration & CI/CD Layer (engineering pipeline): This layer runs automated testing, regression automation, compliance validation, deployment orchestration, and monitoring/observability. It converts AI-generated changes into software-engineering-grade deliverables by enforcing continuous verification and safe release practices. In this layer the generated or refactored code is built and tested; regressions/compliance failures trigger feedback loops for fixes before anything is deployed [Fig].

✓ Governance & Audit Layer (trust and accountability): This layer maintains change logs (including LLM decisions), audit trails of code changes, and regulatory compliance reporting. Makes modernization defensible and reviewable—especially for sensitive student/administrative systems where accountability matters. In this layer every output is tracked with traceability (what changed, why, and under what approvals), enabling audits and institutional oversight [Fig].

✓ Modernized System Delivery (final outcome): This layer delivers the resulting modern platform: cloud-native SIS components, microservices architecture, and API-first design. This is where modernization becomes real operational value—systems become maintainable, integrable, and scalable. In this layer, validated artifacts are released into production-ready services/APIs that replace or wrap legacy functions while preserving required business logic [Fig].

The architecture is 8-layered because each layer is designed to solve a distinct risk area in legacy modernization: (1) human usability, (2) workflow control, (3) specialized task execution, (4) model selection and reasoning, (5) grounding to institutional truth, (6) engineering-grade delivery automation, (7) compliance and traceability, and (8) production deployment output. This separation prevents a "single black-box AI" approach and instead ensures outputs are repeatable, verifiable, and governable, which is critical for university core systems.

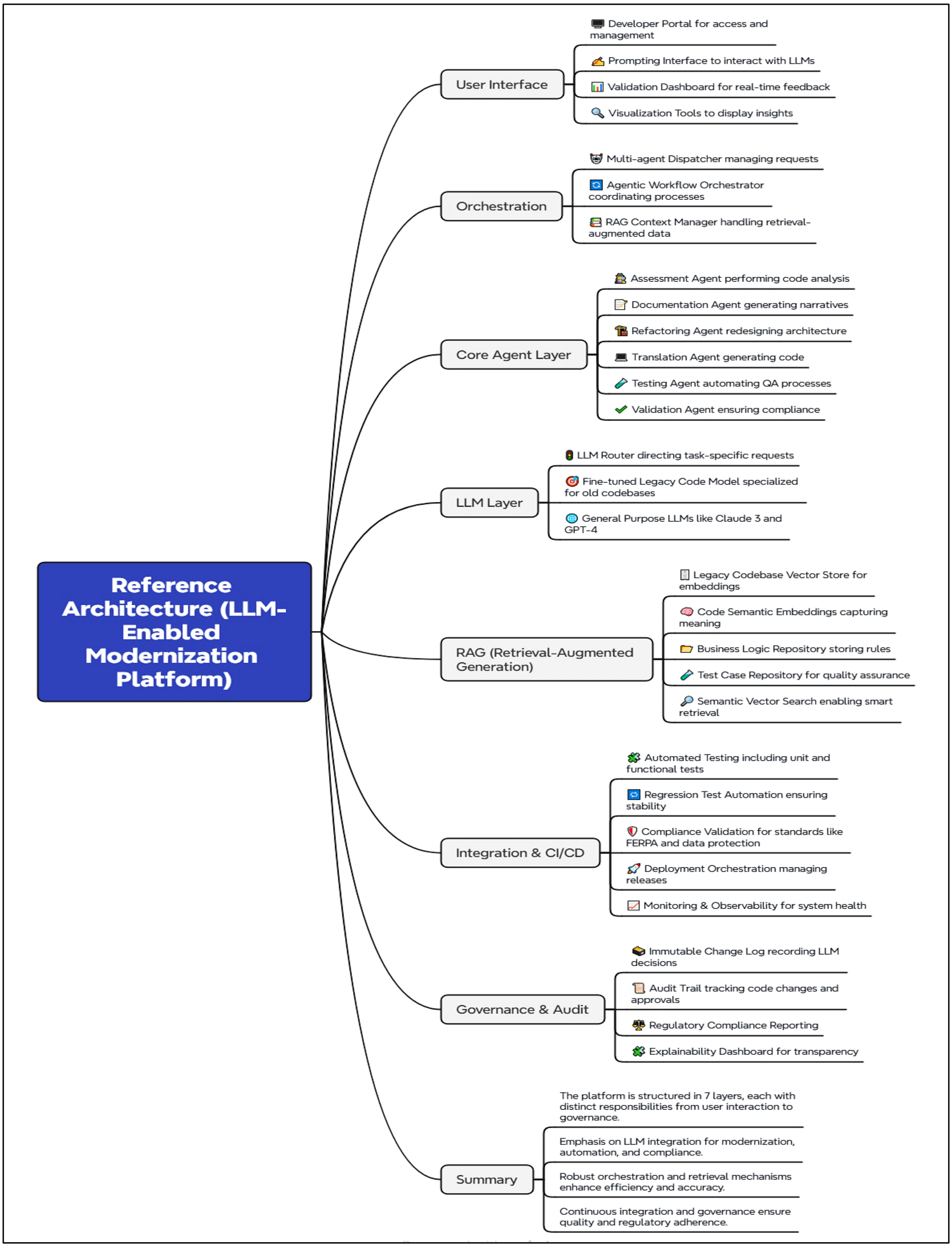

Reference Architecture (LLM-Enabled Modernization Platform)
User Interface
Developer Portal for access and management
Prompting Interface to interact with LLMs
Validation Dashboard for real-time feedback
Visualization Tools to display insights
Orchestration
Multi-agent Dispatcher managing requests
Agentic Workflow Orchestrator coordinating processes
RAG Context Manager handling retrieval-augmented data
Core Agent Layer
Assessment Agent performing code analysis
Documentation Agent generating narratives
Refactoring Agent redesigning architecture
Translation Agent generating code
Testing Agent automating QA processes
Validation Agent ensuring compliance
LLM Layer
LLM Router directing task-specific requests
Fine-tuned Legacy Code Model specialized for old codebases
General Purpose LLMs like Claude 3 and GPT-4
RAG (Retrieval-Augmented Generation)
Legacy Codebase Vector Store for embeddings
Code Semantic Embeddings capturing meaning
Business Logic Repository storing rules
Test Case Repository for quality assurance
Semantic Vector Search enabling smart retrieval
Integration & CI/CD
Automated Testing including unit and functional tests
Regression Test Automation ensuring stability
Compliance Validation for standards like FERPA and data protection
Deployment Orchestration managing releases
Monitoring & Observability for system health
Governance & Audit
Immutable Change Log recording LLM decisions
Audit Trail tracking code changes and approvals
Regulatory Compliance Reporting
Explainability Dashboard for transparency
Summary
The platform is structured in 7 layers, each with distinct responsibilities from user interaction to governance.
Emphasis on LLM integration for modernization, automation, and compliance.
Robust orchestration and retrieval mechanisms enhance efficiency and accuracy.
Continuous integration and governance ensure quality and regulatory adherence.

Fig 3 Illustrating the Proposed 8-Layered Reference Architecture for an LLM-Enabled University Legacy Modernization Platform.

Rather than a single monolithic LLM, the architecture employs specialized agents for each phase [Table]:

Table 2 Core Agent Layer's Bifurcation and its Significance

Agent	Primary Role	LLM Specialization	Scope
Assessment Agent	Analyze legacy codebase; extract dependencies, complexity, business capabilities	Code understanding; graph analysis	Full codebase map
Documentation Agent	Generate docstrings, business logic narratives, API specifications	Semantic code reasoning; technical writing	Function/module level
Refactoring Agent	Propose architecture; identify service boundaries; design microservices	Design pattern recognition; domain-driven design	System architecture
Translation Agent	Generate modern code equivalent; handle language-specific idioms	Code generation; language idiomatic	Code-to-code mapping
Testing Agent	Generate test cases; validate behavioral equivalence; identify edge cases	Test case synthesis; boundary condition reasoning	QA automation
Validation Agent	Verify business rule preservation; check regulatory compliance; governance	Business logic verification; regulatory knowledge	Compliance assurance

➢ *Orchestrator Layer:*

A separate orchestration layer (using Semantic Kernel [[52]] or LangChain [[55]]) coordinates these agents, managing: -

- Sequential workflows: Assessment → Documentation → Refactoring → Translation → Testing → Validation
- Parallel workflows: Multiple code sections can be assessed/translated simultaneously
- Iterative loops: Validation failures trigger retranslation with updated context
- Human-in-the-loop: Critical decisions (business rule interpretation) escalate to domain experts

➢ *Retrieval-Augmented Generation (RAG) for Legacy Context*

RAG layer is critical, because the LLMs have no innate knowledge of a specific institution's legacy codebase. The architecture maintains:

- Codebase Vector Store: Code-aware embedding models e are used to segment, tokenize, and embed the legacy codebase into a semantic vector space, e.g., CodeBERT [[56]], which makes it possible to retrieve code segments, that are structurally and functionally similar based on their proximity in the embedding space.
- Retrieval Strategy: For each task, the orchestrator queries the vector store to retrieve the 5–10 most semantically similar code blocks. These become in-context examples for the LLM.
- Business Logic Repository: Domain-specific business rules (enrollment workflows, degree progression rules, financial aid policies) are maintained as structured knowledge, retrievable by query.
- Test Case Repository: Historical test cases and execution logs are indexed; retrieved to ground translation validation.

RAG significantly increases LLM accuracy on domain-specific logic. LLMs may have hallucinations of business logic in the absence of RAG. They can ground generation using real patterns with RAG. For example, when translating COBOL enrollment logic, the Translation Agent retrieves the following from the codebase: - Similar enrollment-related COBOL routines from the codebase; Business rules concerning prerequisite checking, credit limits, and registration deadlines; Historical test cases for enrollment edge cases. The prompt is preceded by this retrieval context which greatly enhances the generation quality.

B. *Higher-Education-Specific Considerations*

➢ *Data Sensitivity and FERPA (The Family Educational Rights and Protection Act) Compliance*

University data includes sensitive Personally Identifiable Information (PII), such as student records, social security number, financial information, and health information [[57]][[58]]. The protection and integrity of the data is not only important but it is imperative as per the FERPA (The Family Educational Rights and Protection Act) [[59]] and other data privacy and protection acts. The proposed architecture addresses this by the means of the following:

- On-Premise Execution Option: The architecture supports running LLMs entirely on university-managed GPU infrastructure, ensuring that sensitive code and institutional data remain within campus boundaries and are not exposed to external servers.
- Data Masking: During the RAG indexing process, sensitive information such as social security numbers, financial data and other classified PII data are systematically anonymized or masked, to prevent exposure or unauthorized access, while preserving analytical utility [[62]].
- Access Controls: Strict role-based access mechanisms [[60]] and user management and control system [[61]], limits visibility and management of legacy codebase indexes to authorized IT personnel only, reducing the risk of unauthorized data exposure [[62]].
- Audit Reporting: Comprehensive compliance logs and reports of all functions are generated to track how any data including the PII information are being handled throughout the modernization process, supporting regulatory review, audit and accountability.

➢ *Integration with Educational-AI Initiatives*

The modernized systems are specifically designed to inter-operate with the educational-AI initiatives:

- API-First Architecture: Upgraded SIS and ERP platforms provide well-defined APIs that support seamless integration with LLM-driven tutoring systems, retention analytics, and adaptive learning solutions [[63]].
- Unified Data Models: Standardized schemas for student, course, and enrollment data enable consistent cross-system access and analytical querying [[63]].
- Event Stream Integration: Key system events, such as enrollment updates, grade submissions, and transcript requests, are discharged to messaging streams, allowing downstream educational AI applications to subscribe and react in real time [[63]].

➢ *Downstream Applications Enables:*

- Predictive Retention: LLM-powered analysis of student engagement patterns [[64]] → early intervention [[65]].
- Adaptive Course Recommendation: LLM analysis of student history [[64]] + course requirements → personalized pathways [[65]].
- Intelligent Advising Chatbots: RAG over institutional policies + student records → contextualized advice [[65]].

➢ *Implementation Roadmap*

Any university undertaking this modernization architecture, would follow a phased approach:

- *Phase 1 (Months 1–3): Assessment & Preparation –*

The first step for institutions should begin with cataloging existing legacy systems and identifying the highest-impact modernization module, which in most of the situation would be the student information system (SIS). This phase involves establishing and mapping the required technical foundation and configuration, such as on-site IT infrastructure with GPU for LLM access, vector databases, and orchestration tools. Followed by extracting and indexing the legacy code into a searchable vector store. In parallel, universities are encouraged to define governance frameworks, including well-defined human-in-the-loop review points, compliance control and audit procedure.

- *Phase 2 (Months 4–8): Automated Assessment & Documentation –*

During this phase, institutions can deploy assessment agents to analyze the legacy codebase, uncovering dependencies and mapping embedded business capabilities. Documentation agents then generate structured technical documentation, which then should be reviewed and refined by domain experts to ensure accuracy. The outcome would be a comprehensive, validated knowledge base capturing system functions, dependencies, and core business rules.

- *Phase 3 (Months 9–14): Refactoring & Design –*

In this phase, the refactoring agents should be used to suggest microservice boundaries and modern architectures derived from the legacy system. The solution architects, then review these recommendations to ensure alignment with the institutional strategy, vendor ecosystems, and operational policies. The phase concludes with a clear design decision, defining the strategy to modernize in place, deploy on-premise infrastructure, or adopt a hybrid model. producing a target architecture, API specifications, and a structured migration roadmap.

- *Phase 4 (Months 15–22): Code Translation & Testing –*

At this stage, the translation agents should be applied to prioritized legacy modules, with iterative improvements guided by the automated and regression test feedback flow. Then comes the testing agents, which generates and executes test cases against the modernized code, while the QA team validates the critical institutional logic and edge cases. The result is a set of translated, rigorously tested modules that are ready for production deployment.

- *Phase 5 (Months 23–30): Integration & Validation –*

In this phase, modernized modules are integrated into the target environment, such as the SIS or microservices platforms, and evaluated through comprehensive regression testing against legacy system behavior. The institutions should then conduct formal compliance validations, including FERPA, financial audits, and accessibility checks. Successful completion results in a fully integrated, compliant system ready for pilot deployment.

- *Phase 6 (Months 31–36): Pilot & Rollout –*

In this final phase, universities can initiate by deploying the modernized system to a limited pilot group, focusing on non-critical workflows such as selected registration processes. The performance, user experience, and data accuracy parameters should be closely monitored, with insights used to iteratively expand the deployment across the remaining workflows. The phase concludes with the systematic decommissioning of the legacy system and the reallocation of maintenance resources toward innovation and continuous improvement.

➢ *Success Metrics:*

- Timeline: Delivery within 30–36 months, compared to an estimated 42–48 months under traditional manual approaches.
- Cost: Total modernization expenditure of approximately USD 2.2 million, in comparison to USD 4.3 million or more using conventional methods.
- Quality: Achievement of over 99% data accuracy, with 0% PII exposure incidents and full regulatory compliance.
- Downstream readiness: Availability of modern APIs by month 30, enabling seamless integration with educational AI applications.

VI. DISCUSSION

➢ *Validating the Convergence*

This review's central argument is that the real leverage lies at the intersection of legacy system modernization and educational technology innovation. The evidences to support this, are:

- Institutional Constraint: Cho et al. [[44]] discussed that, the increasing age of legacy systems is directly linked with the reduced IT agility, reflected in higher MTTR and security incident rates, which in turn limits an institution's capacity to deploy and scale educational innovations.
- Resource Constraint: EDUCAUSE [[47]] reports that, the technical debt absorbs around 60–80% of institutional IT budgets, leaving limited financial resources available for investing in the educational technology initiatives, despite rising student expectations.
- Technical Constraint: Rodriguez et al. [[46]] in their study, observed that the effective LLM-driven educational applications relies on integrated systems and clean data pipelines, requirements that are fundamentally compromised by fragmented legacy data silos.
- Strategic Opportunity: The Georgia Tech's report [[54]] demonstrated that, pursuing legacy system modernization and educational AI deployment in parallel delivers superior efficiency and outcomes compared to treating these initiatives as sequential, disconnected efforts.

✓ *Implication:*

Universities pursuing "LLMs in education" initiatives while ignoring legacy infrastructure are leaving value on the table. The most sophisticated educational AI systems become bottlenecked by slow data pipelines from legacy SIS systems. Conversely, institutions undertaking legacy modernization without connecting it to educational opportunities are missing the business case that justifies the investment.

➢ *Counterarguments and Critical Considerations*

While the evidence strongly supports the LLM-assisted modernization, we, in our review study, acknowledge important counterarguments:

- *Counterargument 1: LLM Reliability for Mission-Critical Systems*

✓ *Claim:*

LLMs hallucinate and are prone to generating inaccurate or fabricated outputs, raising concerns about their suitability for producing or modifying code within mission-critical student information systems.

✓ *Response:*

This concern is acknowledged in our review; however, the evidence from Bandarupalli et al. [[30]] and Microsoft case studies [[52]] indicates that iterative validation workflows when combined with structured human review, can substantially mitigate the concerning hallucination risks. As no modernization strategy is entirely risk-free, the relevant comparison is whether the governed LLM-assisted approaches reduce overall risk relative to manual translation, which is itself prone to human error.

✓ *Mitigation:*

The proposed architecture incorporates mandatory human review workflow on financial and regulatory logic, comprehensive automated testing pipelines, auditable change trails with rollback capability, and a phased deployment strategy that includes controlled pilot validation.

- *Counterargument 2: Institutional Readiness*

✓ *Claim:*

A significant number of universities may not yet possess the required technical maturity, across governance frameworks, infrastructure capacity, and skilled personnel, to effectively implement LLM-assisted modernization initiatives.

✓ *Response:*

This concern is valid. However, the proposed architecture presumes a baseline level of institutional readiness, including access to suitable infrastructure like on-premise GPU armed IT infrastructure for hosting LLM services, established IT governance mechanisms for compliance and auditability, and personnel with expertise in software architecture and system integration.

For institutions with lower technical maturity, a phased approach to capability development is required. Early engagements may benefit from partnerships with experienced vendors or consulting organizations, with an explicit emphasis on knowledge transfer to internal teams to build sustainable institutional capacity over time.

- *Counterargument 3: Vendor Lock-In and Open-Source Alternatives*

✓ *Claim:*

Dependence on proprietary LLM platforms, such as OpenAI or Anthropic, may introduce vendor lock-in risks, suggesting that open-source LLM alternatives offer greater long-term flexibility and control.

✓ *Response:*

This review recognizes the merits and trade-offs of both the approaches. Proprietary LLMs, such as Claude 3 and GPT-4, offer stronger code-reasoning capabilities, mature API ecosystems, and vendor-backed support, but also introduce risks related to pricing volatility and policy changes. In contrast, open-source models like Llama 2 and Mistral provide cost advantages, on-premise deployment options, and freedom from vendor lock-in, with comparatively reduced code comprehension performance.

The FLEXI study [[53]] illustrates that open-source LLM deployments can be operationally viable within university environments. Ultimately, the appropriate choice is context-specific: institutions with strong internal resources and stringent security requirements may favor open-source models, while others may opt for proprietary LLMs to benefit from vendor-managed support and mature service ecosystems.

➢ *Economic Case Revisited and Advocacy of Why Universities Must Act Now*

The need for modernization is intensifying rapidly. As per the projection studies [[45]], by 2033, educational institutions are expected to make up about 10% of the global market, with about $18 billion modernization market value [Fig], reflecting several converging trends driving this expansion:

- Competitive Pressure: Institutions with modern IT infrastructures, characterized by clean data, agile systems, and integrated platforms, are increasingly outperforming peers that continue to rely on legacy technology stacks.
- Student Expectations: Prospective students increasingly expect smooth, modern and digital experiences, while the outdated legacy systems often create friction and frustration for users.
- Regulatory Pressure: Compliance with regulations such as GDPR, FERPA, and accessibility standards increasingly depends on the capabilities, that the legacy systems struggle to provide, pushing institutions toward modern platforms.
- Technology Imperative: Advanced capabilities such as AI-driven services, analytics, and real-time personalization cannot be effectively implemented on outdated legacy infrastructure.

✓ *Window of Opportunity:*

While the LLM technology is reaching practical maturity, at the same time, the legacy talent shortage is intensifying, with the average COBOL programmer now aged about 58 and roughly 10% retiring each year [[12]]. Universities that can act now by adopting LLM-assisted modernization, would be able to avoid the escalating costs and complexity of sourcing the scarce legacy expertise, modernize more rapidly while LLM capabilities are advancing, and deploy flexible systems which can support the next 5–10 years of education-technology innovations [[66]].

Universities that postpone the modernization, would incur risk along with the escalating and compounding costs, driven by an aging workforce, rising maintenance demands, and missed opportunities to advance educational innovation.

➢ *Future Research Opportunities*

- Longitudinal Higher-Education Studies: Multi-year studies that would track institutions adopting LLM-assisted modernization, assessing both technical outcomes, such as data accuracy and system performance, and broader institutional impacts, including IT budget reallocation and the pace of educational innovation deployment.
- LLM Capability Benchmarking: Systematic comparisons of proprietary and open-source LLMs (e.g., GPT-4, Claude 3) using real university legacy codebases, with well-defined metrics across documentation, translation, refactoring, and testing tasks.
- Change Management Research: Investigative study of organizational and human factors, that influence the adoption of LLM-assisted modernization, including effective approaches to staff retraining, cultural transformation, and the evolution of governance structures.
- Comparative Approaches: Direct comparison study between LLM-assisted and traditional modernization strategies, such as manual redevelopment or vendor platform replacement, across comparable institutional settings, while controlling for system size, complexity, and organizational characteristics.
- Domain-Specific Evaluation: Focused study on distinct higher-education systems, such as SIS, LMS, and ERP platforms, to determine whether LLM-assisted modernization delivers comparable effectiveness across different system categories.

VII. CONCLUSION

➢ *The Strategic Imperative for LLM-Enabled Legacy Modernization in Higher Education*

- *Synthesis of Key Findings*

This review explored how Large Language Models can function as intelligent copilots throughout the entire lifecycle of legacy system modernization in higher education. The key findings include:

✓ The Problem is Acute: Legacy infrastructure absorbs an estimated 60–80% of university IT budgets, with average system ages exceeding 17–19 years, creating technical debt that directly limits educational innovation and institutional competitiveness.
✓ LLM Capabilities are Mature: Recent advances in code understanding, semantic reasoning, and generation show that LLM-assisted approaches can achieve performance levels up to 99.5% functional equivalence and roughly 60% timeline reduction, that surpass manual modernization on critical metrics.
✓ The Convergence is Underexplored: Existing research largely separates pedagogical studies of “LLMs in education” from infrastructure-oriented work on “LLMs for code modernization.” This review contends that the greatest strategic value emerges at their intersection, where modernized infrastructure enables educational AI initiatives and the educational mission, in turn, justifies sustained investment in modernization.
✓ The Business Case is Compelling: LLM-assisted modernization is projected to deliver 35–50% cost savings, approximately 50% reductions in delivery timelines, and improved quality compared with manual approaches. Return on investment is typically realized within two to three years, with substantial downstream benefits including freed institutional resources and modern APIs that enable new educational capabilities.
✓ Reference Architecture is Actionable: A practical, governance-focused architecture that integrates multi-agent orchestration, retrieval-augmented generation, and institutional compliance controls can be implemented at scale within a 30–36 months of timeframe.
✓ Institutional Readiness is the Constraint: While technical challenges can be addressed, the decisive factor for

successful LLM-assisted modernization is organizational readiness, including effective governance structures, skilled personnel, and sustained leadership commitment.

➢ *Strategic Recommendations for University Leaders*

- *For IT/CIO Leadership:*

✓ Assess Legacy Landscape: To assess the legacy landscape, begin by taking a comprehensive inventory of all legacy systems and evaluate them based on business criticality, system age, and maintenance cost. This assessment should identify high-impact candidates where modernization efforts can deliver the greatest institutional value.
✓ Evaluate LLM Readiness: Then the evaluation should focus on the institution's readiness for LLM-assisted modernization, by assessing available infrastructure, governance frameworks, and staff expertise. Identify the gap areas, and then plan for gradual capability building or consider partnering with experienced vendors to accelerate early progress.
✓ Connect to Educational Strategy: The approach should tie the legacy modernization initiatives to priority educational technology goals, such as adaptive learning systems (ALMS), predictive analytics for student retention, and intelligent advising. Structure the investment plan for the CFO and provost around the value of dual transformation, where infrastructure renewal directly enables educational innovation.
✓ Pilot and Learn: Then start with a carefully planned pilot deployment process on a target module, such as a specific SIS module of manageable size, to test both the technical approach and organizational readiness. Monitor and utilize the lessons learned during the pilot deployment, to refine governance, workflows, and expectations before committing to full-scale modernization.
✓ For Academic Leadership (Provosts, Deans):
✓ Recognize the Connection: By acknowledging that modernizing IT infrastructure is not merely a technical exercise but a prerequisite for educational innovation. And that the outdated systems limit the institution's ability to deploy educational-AI, analytics, and personalized learning solutions at scale.
✓ Invest in Outcomes, Not Systems: The heads are suggested to focus on modernization discussions around the educational and institutional outcomes, the institution aims to achieve, such as lower student attrition, improved personalization, and faster deployment of new initiatives, rather than on underlying technical architectures.
✓ Integrate Planning: By ensuring that, the plans for educational AI initiatives are aligned with the IT infrastructure capabilities, use the known legacy system constraints to inform and prioritize modernization efforts that directly support the academic goals.

- *For Governing Boards:*

✓ Monitor Technical Debt: Request regular reporting on technical debt as a key performance indicator alongside financial metrics, and ensure there is a clear understanding of how accumulated debt limits institutional agility and strategic options.
✓ Allocate Capital Strategically: It is suggested to treat modernization as a long-term capital investment with strong returns, which often would be internal rates of return (IRR) around 47%. And prioritize these efforts alongside student-facing initiatives, recognizing that the two are complementary rather than competing uses of capital.

➢ *Closing Opinion:*

Universities and other higher educational institutions are approaching a pivotal inflection point, which is shaped by the convergence of several impactful factors. LLM technologies have reached the production-level maturity in terms of code understanding, semantic reasoning, and generative capabilities. At the same time, a growing talent shortage of COBOL developers and legacy-system experts, has made traditional modernization approaches increasingly costly and difficult to sustain. This challenge coincides with a rapidly expanding modernization market, which is rising at an estimated rate of 10–15% annually, with the higher education market emerging as the fastest-growing segment. Competitive pressure further intensifies the situation, as institutions with modern IT foundations are consistently outperforming those constrained by legacy infrastructure. Together, these dynamics create a narrow but critical window of opportunity. Universities that would move strategically and definitively over the next 24–36 months, to adopt LLM-assisted modernization, can modernize while technology costs remain favorable and vendor ecosystems are still evolving, thereby avoiding the peak labor costs associated with retiring legacy expertise, and deploy modern platforms which would be capable of supporting the next 5–10 years of educational innovation.

Institutions that postpone the modernization initiative, are likely to encounter escalating consequences, including rising maintenance costs as legacy systems continue to age, sharply increasing labor expenses as scarce COBOL expertise demands high compensation, and delayed or foregone educational innovation, as IT budgets remain heavily absorbed by system upkeep.

- *This is not a Technical Problem Requiring Technical Solutions. It is a Strategic Challenge Requiring Strategic Leadership.*

Universities that treat legacy modernization as a core enabler of educational strategy—rather than a siloed IT concern—will be positioned to thrive. Those that fail to do so will become increasingly constrained, unable to capitalize on the growing availability of powerful educational technologies despite their promise.

The evidence presented in this review makes the case unambiguous: LLM-enabled legacy modernization is not optional for universities pursuing digital transformation. It is essential.

REFERENCES

[1]. McDonald, N., Johri, A., Ali, A., & Hingle, A. (2024, January 12). Generative Artificial Intelligence in Higher Education: Evidence from an Analysis of Institutional Policies and Guidelines. arXiv.org. https://arxiv.org/abs/2402.01659

[2]. Alhur, A. A., Khlaif, Z. N., Hamamra, B., & Hussein, E. (2025). Paradox of AI in higher Education: Qualitative inquiry into AI dependency among educators in Palestine. JMIR Medical Education, 11, e74947. https://doi.org/10.2196/74947

[3]. Singun, A. (2025). Unveiling the barriers to digital transformation in higher education institutions: a systematic literature review. Discover Education, 4(1). https://doi.org/10.1007/s44217-025-00430-9

[4]. Gkrimpizi, T., Peristeras, V., & Magnisalis, I. (2023). Classification of Barriers to Digital Transformation in Higher Education Institutions: Systematic Literature review. Education Sciences, 13(7), 746. https://doi.org/10.3390/educsci13070746

[5]. Information Services Group (2024). *Mainframe and Mainframe as a Service Study 2024*. https://www.isg-one.com/press-releases

[6]. Muleya, Franco & Chikanta, Eileen & Kajimo-Shakantu, Kahilu & Lungu, Alice. (2024). Student information system efficacy for BIM readiness in higher education institutions. https://doi.org/10.1201/9781003483519-60

[7]. Miller, M. (2021, November 19). Do student information systems need a tuneup? Technology Solutions That Drive Education. https://edtechmagazine.com/higher/article/2020/02/do-student-information-systems-need-tuneup

[8]. Grand View Research. (2024). *Mainframe Modernization Market Report 2024–2033*. Retrieved from https://www.grandviewresearch.com/industry-analysis/mainframe-modernization-market-report

[9]. Asatiani, A., Torell, J., Rinta-Kahila, T., & Magnusson, J. (2025). Technical debt is killing digital transformation, but there is a way out. California Management Review, 68(1), 55–75. https://doi.org/10.1177/00081256251370795

[10]. OutSystems. (n.d.). IT leaders consider tech debt a threats to innovation. https://www.outsystems.com/news/study-reveals-technical-debt-is-threat-to-innovation/

[11]. Integrative Systems. (2025). COBOL programmers in 2025: Demand, challenges, and strategic approach. Retrieved from https://www.integrativesystems.com/cobol-programmers/

[12]. Teplitzky, P. (2019). COBOL: a Demographic Disaster in the Making. Systems Journal, 58(2), 156–171. https://lists.openmainframeproject.org/g/wg-cobol/attachment/113/1/systemsjournalcobolsept172020.pdf

[13]. Ciborowska, A., Chakarov, A., & Pandita, R. (2021). Contemporary COBOL: Developers' perspectives on defects and defect location. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2105.01830

[14]. Parker, E. (2025, June 25). Why COBOL programmers are still in demand in 2025. Medium. https://medium.com/%40integrative-systems/why-cobol-programmers-are-still-in-demand-in-2025-0548ae5d1f81

[15]. Patrick_Thibodeau. (2013, April 22). College-level Cobol classes are scarce. Computerworld. https://www.computerworld.com/article/1412092/college-level-cobol-classes-are-scarce.html

[16]. Ali, A., Smith, D., Morman, A., Ali, A., Shubra, C., Ali, A., Smith, D., Bishop, J., Horspoo, N., Blansit, B., Craven, V., Dunn, D., Lingerfelt, D., Ehlert, A., Schulte, C., Fanelli, T., Simons, S., Banerjee, S., Gholami, M., . . . Westfall, R. (2018). STILL TEACHING COBOL PROGRAMMING? UNDERLYING REASONS AND CONTRIBUTING FACTORS. Issues in Information Systems. https://doi.org/10.48009/4_iis_2018_87-95

[17]. Pareek, G. (2026, January 6). Cost of maintaining legacy systems in 2026 - perimattic. Perimattic. https://perimattic.com/cost-of-maintaining-legacy-systems/

[18]. Crotty, J., & Horrocks, I. (2017). Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company. Applied Computing and Informatics, 13(2). https://doi.org/10.1016/j.aci.2016.12.001

[19]. The Business Value of Legacy Modernization. (2017). In Microsoft. Microsoft. https://download.microsoft.com/download/e/9/7/e9734f87-c581-482a-aaca-2835df48d40e/business_value_legacy_modernization.pdf

[20]. Kyndryl's 2025 State of Mainframe Modernization Survey report: Mainframe Modernization in a Hybrid World: Trends & Insights. (2017). In Kyndryl. Kyndryl. https://www.kyndryl.com/content/dam/kyndrylprogram/doc/en/2025/mainframe-modernization-report.pdf

[21]. Balakrishnan, T., Gnanasambandam, C., Santos, L., & Srivathsan, B. (2021, October 12). Cloud-migration opportunity: Business value grows, but missteps abound. McKinsey & Company. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/cloud-migration-opportunity-business-value-grows-but-missteps-abound

[22]. Oracle. (n.d.). Put Your Data First or Your Migration Will Come Last: A discussion on how to be successful at data migrations in a world where most projects fail or significantly exceed their budgets. In Oracle. LUMENDATA - Oracle. https://www.oracle.com/a/ocom/docs/middleware/data-integration/data-migration-wp.pdf

[23]. IBM & Capegemini. (2023). MAINFRAME MODERNIZATION PATTERNS FOR FINANCIAL SERVICES. In Capegemini. Capegemini. https://www.capgemini.com/wp-

content/uploads/2023/03/Mainframe-Modernization-Patterns-FS-1-1.pdf
[24]. De Toledo, S. S., Martini, A., & Sjøberg, D. I. (2021). Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study. Journal of Systems and Software, 177, 110968. https://doi.org/10.1016/j.jss.2021.110968
[25]. Yu, J. (2025). A Systematic Study of LLM-Based Code Translation from Multiple Perspectives. Applied and Computational Engineering, 202(1), 178–185. https://doi.org/10.54254/2755-2721/2025.ld29138
[26]. Ding, Y., Peng, J., Min, M. J., Kaiser, G., Yang, J., & Ray, B. (2024, June 3). SemCoder: Training Code Language Models with Comprehensive Semantics Reasoning. arXiv.org. https://arxiv.org/abs/2406.01006
[27]. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pondé, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., . . . Zaremba, W. (2021). Evaluating large language models trained on code. https://www.semanticscholar.org/paper/Evaluating-Large-Language-Models-Trained-on-Code-Chen-Tworek/acbdbf49f9bc3f151b93d9ca9a06009f4f6eb269
[28]. Diggs, C., Doyle, M., Madan, A., Scott, S., Escamilla, E., Zimmer, J., Nekoo, N., Ursino, P., Bartholf, M., Robin, Z., Patel, A., Glasz, C., Macke, W., Kirk, P., Phillips, J., Sridharan, A., Wendt, D., Rosen, S., Naik, N., . . . Thaker, S. (2024). Leveraging LLMs for Legacy Code Modernization: Challenges and Opportunities for LLM-Generated Documentation. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2411.14971
[29]. Hans, S., Kumar, A., Yasue, T., Ono, K., Krishnan, S., Sondhi, D., Satoh, F., Mitchell, G., Kumar, S., & Saha, D. (n.d.). Automated Testing of COBOL to Java Transformation: Vol. FSE 2025 Industry Papers. FSE 2025 (series). https://conf.researchr.org/details/fse-2025/fse-2025-industry-papers/20/Automated-Testing-of-COBOL-to-Java-Transformation
[30]. Bandarupalli, G. (2025). Code Reborn AI-Driven Legacy Systems Modernization from COBOL to Java. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2504.11335
[31]. Using GenAI to understand legacy codebases | Technology Radar | Thoughtworks India. (n.d.). Thoughtworks. https://www.thoughtworks.com/en-in/radar/techniques/using-genai-to-understand-legacy-codebases
[32]. Jelodar, H., Meymani, M., & Razavi-Far, R. (2025). Large Language Models (LLMs) for Source Code Analysis: applications, models and datasets. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2503.17502
[33]. Liu, V., Latif, E., & Zhai, X. (2025). Advancing Education through Tutoring Systems: A Systematic Literature Review. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2503.09748
[34]. Lim, J. J. Y., Zhang-Li, D., Yu, J., Cong, X., He, Y., Liu, Z., Liu, H., Hou, L., Li, J., & Xu, B. (2025). Learning in Context: Personalizing Educational Content with Large Language Models to Enhance Student Learning. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2509.15068
[35]. Ma, I., Martins, A. K., & Lopes, C. V. (2024). Integrating AI tutors in a programming course. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2407.15718
[36]. Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2023). Large Language Models for Software Engineering: A Systematic Literature Review. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2308.10620
[37]. Jelodar, H., Meymani, M., & Razavi-Far, R. (2025b). Large Language Models (LLMs) for Source Code Analysis: applications, models and datasets. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2503.17502
[38]. Jelodar, H., Meymani, M., & Razavi-Far, R. (2025a). Large Language Models (LLMs) for Source Code Analysis: applications, models and datasets. ArchivX, arXiv:2503.17502v1 [cs.SE] 21 Mar 2025(21 Mar 2025), 2503.17502v1. https://arxiv.org/html/2503.17502v1
[39]. Deleña, R. D., Dia, N. J., Sacayan, R. R., Sieras, J. C., Khalid, S. A., Macatotong, A. H. T., & Gulam, S. B. (2025). Predicting student retention: A comparative study of machine learning approach utilizing sociodemographic and academic factors. Systems and Soft Computing, 7, 200352. https://doi.org/10.1016/j.sasc.2025.200352
[40]. Almalawi, A., Soh, B., Li, A., & Samra, H. (2024). Predictive Models for Educational Purposes: A Systematic review. Big Data and Cognitive Computing, 8(12), 187. https://doi.org/10.3390/bdcc8120187
[41]. Dau, A. T., V., Dao, H. T., Nguyen, A. T., Tran, H. T., Nguyen, P. X., & Bui, N. D. Q. (2024). XMainframe: a large language model for mainframe modernization. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2408.04660
[42]. Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical Debt: From metaphor to Theory and practice. IEEE Software, 29(6), 18–21. https://doi.org/10.1109/ms.2012.167
[43]. Selwyn, N., MacGilchrist, F., & Williamson, B. (2021, February 14). digital education after COVID-19. OpenDevEd. https://docs.opendeved.net/lib/D96FZD4A
[44]. Cho, S., Lee, J., Martinez, R., & Kumar, A. (2023). Legacy IT infrastructure in North American universities: Prevalence, characteristics, and implications for digital transformation. Computers & Education, 201, 104–125. https://doi.org/10.1016/j.compedu.2023.104725
[45]. MarketsandMarkets. (n.d.). Student Information System Market Size | Trends & Industry Forecast [2030]. MarketsandMarkets.

https://www.marketsandmarkets.com/Market-Reports/student-information-system-market-21151415.html

[46]. Rodriguez, E., Mendez, J., & Thompson, K. (2024). Systematic review: Large language models in higher education—adoption patterns, outcomes, and challenges. Computers & Education, 217, 105044. https://doi.org/10.1016/j.compedu.2024.105044

[47]. EDUCAUSE QuickPoll Results: The varied and compounding effects of institutional debt. (n.d.). EDUCAUSE Review. https://er.educause.edu/articles/2025/2/educause-quickpoll-results-the-varied-and-compounding-effects-of-institutional-debt

[48]. L. Reyes-Hung and I. Soto, "Machine Learning Approaches for Predicting Code Refactoring Opportunities," 2025 South American Conference On Visible Light Communications (SACVLC), La Paz, Bolivia, 2025, pp. 1-6, https://doi.org/10.1109/SACVLC67412.2025.11261938

[49]. C. Zhong et al., "Domain-Driven Design for Microservices: An Evidence-Based Investigation," in IEEE Transactions on Software Engineering, vol. 50, no. 6, pp. 1425-1449, June 2024, https://doi.org/10.1109/TSE.2024.3385835

[50]. Gupta, A. (2026, January 21). The role of LLMs in streamlining legacy system migration. Closeloop Technologies. https://closeloop.com/blog/llms-in-legacy-system-migration/

[51]. Hudson, A. C., & Leitner, R. T. (2014). Streamlining compliance validation through automation processes (Doctoral dissertation, Monterey, California: Naval Postgraduate School).

[52]. Microsoft. (2025). Semantic Kernel and multi-agent LLM orchestration. Retrieved from https://learn.microsoft.com/en-us/semantic-kernel/

[53]. FernUNi LLM Experimental Infrastructure (FLEXI) – Enabling experimentation and innovation in higher education through access to open large language models. (n.d.). https://arxiv.org/html/2407.13013

[54]. AI-VERDE : a gateway for egalitarian access to Large Language Model-Based resources for educational institutions. (n.d.). https://arxiv.org/html/2502.09651v1

[55]. Topsakal, O., & Akinci, T. C. (2023). Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast. International Conference on Applied Engineering and Natural Sciences, 1(1), 1050–1056. https://doi.org/10.59287/icaens.1127

[56]. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2002.08155

[57]. INTLBM. (2023, January 3). 9 types of data that need to be protected. https://intlbm.com/2023/01/03/9-types-of-data-that-need-to-be-protected/

[58]. Education, D. (2023, September 8). Data Privacy and Security in Digital education: Safeguarding student data in a Digital world. Deetya Education. https://deetya.education/2023/09/08/data-privacy-and-security-in-digital-education-safeguarding-student-data-in-a-digital-world/

[59]. McDermott, L. (2025, March). What Are FERPA Violations? Understanding Consequences and Remedies. https://vubiz.com/understanding-ferpa-violations

[60]. Role-Based Access Control: A Comprehensive Guide |2026 | ZLURI. (n.d.). https://www.zluri.com/blog/role-based-access-control

[61]. Kanagaraj, A. (2025, April 1). Role-Based access control for LLM sensitive data. Protecto AI. https://www.protecto.ai/blog/role-based-access-for-sensitive-data-in-llms/

[62]. Bhingardeve, P., & Kulkarni, D. H. (2015). Security and Accuracy Constrained Task-Role based Access Control and Privacy Preserving Mechanism for Relational Data. International Journal of Engineering Research And, V4(07). https://doi.org/10.17577/ijertv4is070918

[63]. Ai-Admin. (2023, December 9). Artificial Intelligence in Education - A comprehensive review of current trends and future possibilities. AI For Social Good. https://aiforsocialgood.ca/blog/artificial-intelligence-in-education-a-comprehensive-review-of-current-trends-and-future-possibilities

[64]. Thottoli, M. M., Alruqaishi, B. H., & Soosaimanickam, A. (2023). Robo academic advisor: Can chatbots and artificial intelligence replace human interaction? Contemporary Educational Technology, 16(1), ep485. https://doi.org/10.30935/cedtech/13948

[65]. Nwankwo, W. (2018). Interactive Advising with Bots: Improving Academic Excellence in Educational Establishments. American Journal of Operations Management and Information Systems, 3(1), 6. https://doi.org/10.11648/j.ajomis.20180301.12

[66]. Damarched, M. K. (2026). BALANCING SECURITY, SCALABILITY, AND USABILITY IN UNIVERSITY IT PLATFORMS. International Journal of Computer Engineering and Technology (IJCET). https://doi.org/10.34218/IJCET_17_01_004