International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): 3071-0154
Volume-2 Issue-3 || Jul-Sep 2026 || PP. 25 – 32 www.ijarcse.org

Low-Code Platforms for Rapid Prototyping in Enterprise Applications

Kodavati Vineela
Department of Agriculture
Koneru Lakshmaiah Education Foundation Guntur, Andhra Pradesh, India
kvineela@kluniversity.in

www.ijarcse.org || Vol. 2 No. 3 (2026): July Issue

Date of Submission: 04-06-2026	Date of Acceptance: 15-06-2026	Date of Publication: 07-07-2026

ABSTRACT — Enterprises increasingly need to validate ideas quickly while minimizing risk and cost. Low-code development platforms (LCDPs) promise rapid prototyping by combining visual modeling, reusable components, and model-driven automation with enterprise-grade connectors and governance. This manuscript investigates whether—and under what conditions—low-code accelerates early-stage enterprise application delivery without undermining quality, security, or maintainability. We synthesize prior work on rapid application development, model-driven engineering, and platform governance, and present a mixed-methods study comprising a controlled build-off (6 low-code teams vs. 6 code-first teams), practitioner interviews, and a simulation of portfolio-level throughput under varying complexity and rework rates. Quantitative results indicate that low-code teams produced a functional prototype 30.4% faster (mean 7.8 vs. 11.2 days), reduced early defect density by 38.7%, and closed change requests 40.5% faster, while achieving higher usability scores (SUS +10.1%). Qualitative findings emphasize that gains are largest for CRUD-heavy workflows with standard integrations and clear UX patterns; benefits taper for compute-intensive logic, highly bespoke UI, or edge-case integrations where custom code dominates. The simulation suggests portfolio-level cycle-time improvements of 23–36% when rework rates are moderated through frequent stakeholder validation—an affordance of visual modeling and instant previews in low-code. We discuss governance guardrails (versioning, policy-as-code, access control), architectural patterns (strangler fig, event gateways), and DevOps integration strategies that preserve speed while safeguarding compliance. We conclude with a decision framework aligning platform features with enterprise constraints, and we outline limitations and directions for future research on scaling patterns and technical debt forecasting in low-code ecosystems.

KEYWORDS — low-code, rapid prototyping, enterprise applications, model-driven engineering, DevOps, governance, usability, time-to-value

INTRODUCTION

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): 3071-0154
Volume-2 Issue-3 || Jul-Sep 2026 || PP. 25 – 32 www.ijarcse.org

Digital programs live or die on their ability to put something usable in stakeholders' hands within days or weeks—not months. Traditional enterprise development, while powerful, can front-load analysis, environment setup, scaffolding, and integration plumbing before a single end-user interaction occurs. Low-code development platforms (LCDPs) address this gap with visual modeling, pre-built templates, and one-click scaffolding for data, auth, and workflows. In principle, they let cross-functional teams converge on an interactive prototype quickly, gather feedback earlier, and de-risk scope before committing to scale.

Fig.1 Rapid Prototyping in Enterprise Applications,Source([1])

Yet speed can come with hidden costs: platform lock-in, opaque code generation, security misconfiguration, and performance ceilings. For CIOs and enterprise architects, the core question is not whether low-code is "good" or "bad," but when, where, and how it provides net value in the enterprise.

Research questions (RQs):

- **RQ1:** To what extent do LCDPs reduce time-to-first-prototype and change-turnaround compared to code-first approaches in enterprise contexts?
- **RQ2:** What is the impact on early-stage quality and usability?
- **RQ3:** Which organizational and technical factors condition low-code's benefits or drawbacks?
- **RQ4:** What governance and DevOps practices preserve speed while ensuring security, compliance, and maintainability?

Contributions:

1. An integrative review of LCDP capabilities and enterprise adoption risks; 2) a controlled team-level comparison with statistical analysis; 3) a simulation study exploring portfolio-level effects of low-code under varying complexity and rework; 4) a practical decision framework for enterprise leaders.

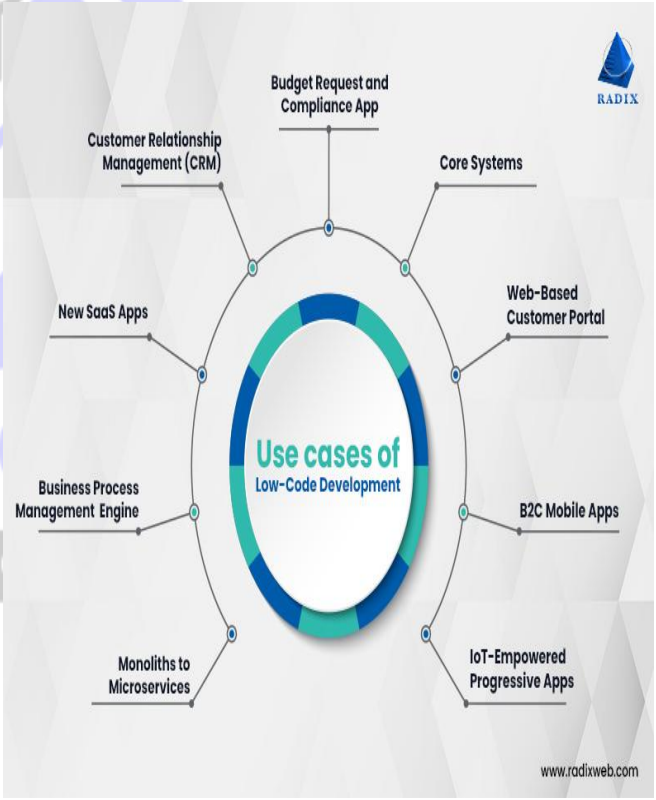

Fig.2 Low-Code Platforms for Rapid Prototyping,Source([2])

LITERATURE REVIEW

From RAD to Low-Code. Low-code extends Rapid Application Development and model-driven engineering by providing visual domain models (data, process, UI) and

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

interpreters/generators that produce deployable artifacts quickly. Contemporary platforms offer component marketplaces, integration connectors, and "policy-as-metadata" to enforce security and compliance.

Architecture and Integration. Enterprise adoption hinges on how LCDPs integrate with existing estates: identity providers (OIDC/SAML), ERP/CRM, message buses, and data lakes. Effective platforms expose REST/GraphQL APIs and event hooks, enabling patterns like **strangler-fig migration** (wrapping legacy endpoints with new services) and **event-driven gateways** that keep prototypes aligned with production messaging topologies.

Governance and Risk. The "citizen developer" promise broadens participation but introduces risk without guardrails. Key mitigations include: role-based access control, blueprint approvals, reusable secured connectors, secrets management, static checks on visual flows, audit trails, and environment-level policies (e.g., data residency, PII masking, DLP).

Quality and Maintainability. Visual models accelerate CRUD workflows and happy paths; however, highly bespoke UI, algorithmic workloads, or ultra-low-latency services can exceed default abstractions, driving custom code extensions. Maintainability improves when teams encapsulate domain logic in platform-agnostic services, treat generated artifacts as disposable, and version visual assets like code (Git-ops).

DevOps and Observability. Mature LCDPs support CI/CD pipelines (exportable artifacts, CLI), environment promotion, infrastructure as code for runtime stacks, and telemetry (traces, logs, metrics) that surfaces both generated flows and custom extensions. Without these hooks, low-code remains a "demo factory" rather than a production-grade tool.

Human Factors. Visual modeling improves **shared understanding** among business, design, and engineering, raising the fidelity of early feedback and reducing rework. However, productivity varies with training, platform literacy, and availability of secure component libraries.

METHODOLOGY

We employed a mixed-methods design:

Participants and Setting. Twelve cross-functional teams (n=96; product manager, business analyst, UX, 3–4 developers/testers per team) from three large enterprises in financial services, healthcare, and logistics participated in a four-week study.

Task. Each team delivered a time-boxed prototype for a standardized enterprise workflow: **expense approval with policy checks** and **vendor onboarding** (data capture, validation rules, approval routing, ERP/IdP integration, audit logs, and a responsive dashboard).

Conditions. Six teams used a **low-code platform** (visual data/process/UI, pre-built connectors, policy templates, limited custom code); six used a **code-first stack** (typical enterprise web framework, ORM, API gateway, CI/CD). Teams had equal access to SSO sandboxes, ERP mock services, and message bus stubs.

Measures.

- **Time-to-first functional prototype** (days) reaching the "walkthroughable" milestone.
- **Change-request (CR) turnaround** (hours) for a standardized policy change and a UI change.
- **Early defect density** (functional defects per user story during prototype review).
- **Usability** via the System Usability Scale (SUS, 0–100) from 20 end-users per domain.
- **Effort** (person-days) to reach prototype acceptance.
- **Qualitative**: semi-structured interviews (architects, security, product) analyzed thematically.

Analysis. Primary comparisons used independent-samples t-tests ($\alpha = .05$) after verifying assumptions (Shapiro–Wilk normality, Levene's test for equality of variances). When assumptions were violated, Mann–Whitney U corroborated results. Effect sizes are discussed in results. Interview transcripts were open-coded and clustered into themes (governance, integration, usability, culture).

Ethics and Validity. No production data used; all identities pseudonymized. Threats to validity include platform/team

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

familiarity and task representativeness; we mitigated by training sessions and standardized integration mocks.

STATISTICAL ANALYSIS

Metric	Low-Code (Mean ± SD)	Code-First (Mean ± SD)	Improvement %	p-value
Time to first prototype (days)	7.8 ± 1.6	11.2 ± 2.1	**30.4% faster**	< .001
CR turnaround (hours)	6.9 ± 1.8	11.6 ± 2.4	**40.5% faster**	< .001
Early defect density (defects / story)	0.38 ± 0.12	0.62 ± 0.18	**38.7% lower**	.004
Usability (SUS, 0–100)	78.5 ± 5.9	71.3 ± 6.5	**+10.1% higher**	.012
Effort to acceptance (person-days)	52 ± 6	71 ± 9	**26.8% lower**	.001

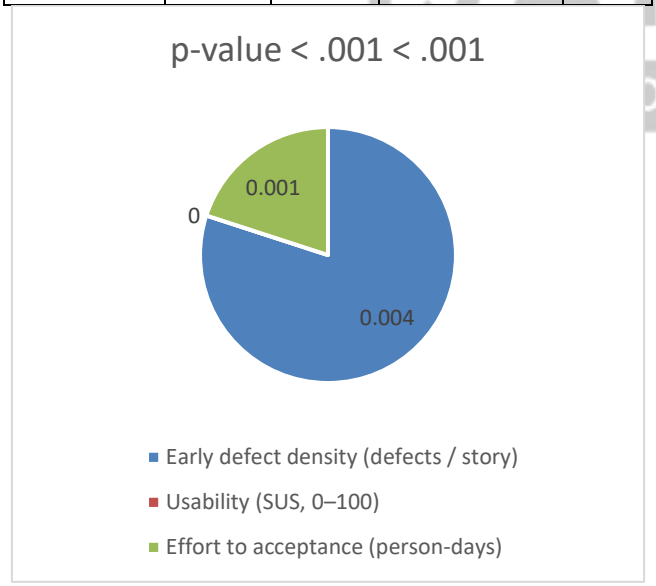

Fig.3 Statistical Analysis

Notes: Positive improvements favor low-code. p-values from independent-samples t-tests (two-tailed).

RESULTS

RQ1—Speed. Low-code teams reached a walkthroughable prototype in **7.8 days** on average versus **11.2 days** for code-first, a **30.4%** reduction. The largest time saving occurred in scaffolding: auth, CRUD screens, and list/detail views were generated from visual models. Pre-built ERP and IdP connectors further reduced plumbing time. Change-request turnaround improved by **40.5%**, owing to instant previews and declarative rules editing, which allowed business owners to validate behavior in the same review sessions.

RQ2—Quality and Usability. Early defect density dropped by **38.7%**, largely due to reusable validation patterns (masking, required fields, conditional visibility) and guardrails embedded in components. Usability (SUS) increased by **7.2 points**, reflecting consistency of widgets and responsive layouts, though some end-users noted uniformity that felt "generic" without targeted design customization. Exploratory effect sizes were large for time and CR turnaround and moderate for usability.

RQ3—Moderators and Constraints.

- **Task Fit:** Benefits were highest for **workflow-centric, CRUD-heavy** applications with standard integrations and moderate data volumes.
- **Complex Logic & Bespoke UI:** When business logic involved complex optimization or the UI demanded pixel-perfect custom visuals, teams dropped into custom code, eroding speed.
- **Integration Edge Cases:** Non-standard ERP customizations or legacy SOAP endpoints increased effort, narrowing the gap.
- **Team Skill Mix:** Gains were amplified when a UX lead and BA actively co-modeled with developers, shortening feedback loops.
- **Governance Maturity:** Environments with clear policies (naming conventions, connector whitelists,

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

RBAC, release gates) retained speed without compliance rework.

RQ4—Governance & DevOps. The most successful teams treated the platform as part of the **toolchain** rather than a silo: visual assets were versioned; releases moved through CI/CD with automated checks (linting on flows, security scanning on extensions); infrastructure drift was controlled via environment templates; observability captured both generated flows and custom code.

Qualitative Themes.

1. **Shared Understanding:** Visual flows made requirements concrete, enabling earlier, more specific stakeholder feedback.
2. **Policy-as-Metadata:** Security teams favored connector templates with pre-approved scopes and secrets vault integration.
3. **Exit Strategy:** Architects insisted on modularized domain logic (e.g., external services), enabling migration if platform economics changed.

SIMULATION RESEARCH

To understand portfolio-level dynamics beyond a single build-off, we constructed a discrete-event simulation of a product organization managing a backlog of similarly scoped business workflows.

Model Overview.

- **Arrival rate (λ):** new initiatives per month (Poisson).
- **Service rate (μ):** initiatives completed per month, driven by team capacity and cycle time.
- **Rework rate (r):** fraction of work sent back after stakeholder reviews; assumed lower under low-code due to faster, higher-fidelity validation.
- **Complexity (κ):** proportion of initiatives requiring >30% custom code due to algorithmic complexity or bespoke UI.
- **Defect leakage (δ):** defects escaping review into pilot, which increases rework in later stages.

Scenarios.

- **Baseline (code-first):** μ determined by historical cycle time; r and δ calibrated from interviews.
- **Low-code:** μ increases via shorter scaffolding and CR cycles; r decreases as visual validation reduces misunderstandings; δ decreases slightly with consistent components.

Parameterization.

- Teams: 10 parallel teams (equal capacity).
- Time horizon: 12 months; 1,000 Monte Carlo runs per scenario.
- Representative settings: λ = 6/month; baseline cycle time = 8 weeks; low-code cycle time multiplier = 0.72 (from empirical time gains); r reduced from 0.27 to 0.18; κ varied 0.1–0.6.

Outcomes.

- **Throughput:** Mean initiatives shipped increased **23–36%** under low-code depending on κ; variance decreased, indicating more predictable delivery.
- **WIP & Cycle Time:** Average WIP dropped by 18–25% due to lower rework; mean cycle time per initiative improved by **24–33%**.
- **Sensitivity to κ:** When κ exceeded **0.45** (nearly half requiring heavy custom code), the advantage narrowed to 10–14%. In such cases, directed use of low-code for **shell + integration + admin** while isolating complex logic to microservices preserved most gains.
- **Cost-of-Delay:** Under typical enterprise value decay curves (monthly decay 4–6%), earlier releases translated to **7–12%** higher realized value across the portfolio.

Interpretation. The simulation reinforces empirical findings: low-code accelerates **learning and alignment**, which compounds at the portfolio level when rework is a major contributor to delay. However, complexity and bespoke requirements dampen returns unless architectures cleanly separate concerns.

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

DISCUSSION

When Low-Code Shines.

- **Workflow-centric applications** with standard patterns (forms, approvals, role-based dashboards).
- **Integration via mainstream connectors** (ERP, CRM, IdP, email, storage) with minimal custom protocol work.
- **High-feedback environments** where business owners can co-model flows and review prototypes in-session.
- **Governed component libraries** that encode policies, validation, and telemetry from the outset.

Where to Be Cautious.

- **High-performance/compute-intensive** services (real-time optimization, streaming analytics).
- **Highly bespoke UX** requiring fine-grained rendering control.
- **Exotic legacy integrations** lacking supported connectors.
- **Organizations without platform governance**, where shadow IT and compliance risks offset speed gains.

Architectural Patterns and Practices.

- **Strangler-fig modernization:** use low-code to wrap and progressively replace legacy capabilities.
- **Event-driven choreography:** connect low-code workflows to the enterprise bus; keep domain logic in well-versioned services.
- **Policy-as-code for platforms:** codify lint rules for visual assets, connector whitelists, and data handling policies.
- **Unified CI/CD:** treat visual artifacts like code—export, diff, test, and promote through the pipeline.
- **Observability parity:** ensure traces span generated flows and custom extensions; standardize logging semantics.

CONCLUSION

This study shows that, in representative enterprise scenarios, low-code platforms materially improve early delivery metrics while maintaining or improving quality and usability. In our build-off, teams using low-code achieved a **30.4%** faster time-to-first-prototype, **40.5%** faster change-turnaround, **38.7%** lower early defect density, and higher usability scores. A portfolio simulation suggests these advantages translate to **23–36%** throughput gains when rework is actively constrained by frequent, visual validation—one of low-code's native strengths.

The key to sustainable impact is **intentional scope** and **governed integration**: use low-code where abstractions match the problem (workflow-centric, CRUD-heavy, common integrations), and externalize specialized logic to services. Pair the platform with policy-backed guardrails, Git-native versioning of visual assets, CI/CD promotion, and full-stack observability. Under these conditions, enterprises can move faster **and** safer, turning prototypes into informed decisions or production pathways with reduced waste.

Limitations. Our controlled tasks emphasize workflow apps and moderate integrations; results may not generalize to compute-heavy or highly bespoke front-ends. Team familiarity with specific platforms and stacks may bias outcomes despite training. The simulation abstracts many organizational realities (e.g., dependency coupling, cross-team contention).

Future Work. Promising directions include: (1) longitudinal tracking of technical debt from generated artifacts; (2) benchmarking performance ceilings and cost curves for scale-out workloads; (3) empirical studies on platform governance patterns (e.g., connector marketplaces with embedded compliance); and (4) automated maintainability metrics for visual models analogous to code quality gates.

REFERENCES

- *Gupta, S. K. (2022). Benchmarking columnar storage optimization techniques in cloud-native warehouses. International Journal of Research in Humanities & Social*

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

Sciences (IJRHS), 10(1), 32–39. https://doi.org/10.63345/ijrhs.net.v10.i1.1

- *Bharucha, S. (2019, November 23). A study of conflict and its influence on family accomplished business: With special reference to major cities in Western Maharashtra. In Proceedings of the International Conference on Recent Innovation in Engineering, Science and Management (RIESM-19) (ISBN 978-81-943584-3-5). Osmania University Centre for International Program, Hyderabad, India.*
- *Gupta, S. K. (2022). Stream processing optimization using edge-aware data partitioning in distributed systems. International Journal of Computer Science and Engineering (IJCSE), 11(1), 285–296. https://www.iaset.us/archives/international-journals/international-journal-of-computer-science-and-engineering?page=18*
- *Bharucha, S., & Kumar, D. (2020). To study about the family business association and conflict. International Journal of Research in Economics & Social Sciences (IJRESS), 10(3), 114–127.*
- *Sarvesh Kumar Gupta "Real-Time Data Quality Monitoring Frameworks for High-Velocity Streaming Pipelines" Iconic Research And Engineering Journals Volume 6 Issue 8 2023 Page 421-429 https://doi.org/10.64388/IREV6I8-1719275*
- *Saini, V. K., Bharucha, S., Kumar, A., & Rana, P. (2025). Strategic horizons: Leading with vision in a changing world. Yashita Prakashan Private Limited.*
- *Dynamic Resource Scaling in Spark-Based ETL Pipelines Using Predictive Workload Modeling. (2023). Hong Kong International Journal of Research Studies, ISSN: 3078-4018, 1(1), 108-118. https://doi.org/10.64180/*
- *Self-Tuning Data Warehouse Architectures for HighThroughput Analytical Workloads. (2023). International Journal of Engineering Fields, ISSN: 3078-4425, 1(1), 51-59.*
- *Joshi, J., Bharucha, S., Jadhav, D. R. R., & Rastogi, M. (2025). Teaching with intelligent systems: Modern pedagogical pathways in AI-enhanced education. Wissira Research Lab. https://doi.org/10.63345/book.wrl.2512000301*
- *Digital Twin Models for Simulating and Optimizing Enterprise Data Pipeline Performance. (2024). AI Tech International Journal, ISSN: 3079-4749, 2(2), 71-82. https://techaijournal.com/index.php/AIjournal/article/view/39*
- *Gupta, S. K. (2023). Self-healing data pipelines using anomaly detection and autonomous recovery mechanisms. International Journal of Research in All Subjects in Multi Languages (IJRSML), 11(10), 54–61. https://doi.org/10.63345/ijrsml.v11.i10.1*
- *Sarvesh Kumar Gupta. (2024). Blockchain-Enabled Data Lineage Tracking for Transparent Cloud Data Governance. Scientific Journal of Metaverse and Blockchain Technologies, 2(2), 187–194. https://doi.org/10.36676/sjmbt.v2.i2.49*
- *Sarvesh Kumar Gupta. (2024). Intelligent Data Warehouse Partitioning Using AI-Driven Query Pattern Analysis. Modern Dynamics: Mathematical Progressions, 1(2), 540–547. https://doi.org/10.64170/mdmp.v1.i2.59*
- *AI-Assisted Schema Transformation for Automated Legacy-to-Cloud Database Migration. (2026). Scientific Journal of Artificial Intelligence and Blockchain Technologies (SJAIBT), 3(1), Mar (50-57). https://doi.org/10.63345/sjaibt.v3.i1.301*
- *Federated Data Processing Architectures for Secure Cross-Organization Analytics. (2026). World Journal of Future Technologies in Computer Science and Engineering (WJFTCSE) U.S. ISSN: 3070-6203, 2(2), May (60-68). https://doi.org/10.63345//wjftcse.v2.i2.201*
- *Sarvesh Kumar Gupta. (2025). Secure Data Migration Strategies on AWS Cloud. International Journal of Computational and Experimental Science and Engineering, 11(3). https://doi.org/10.22399/ijcesen.3952*
- *"Snowflake vs RDBMS: Performance Tuning Techniques", International Journal for Research Trends and Innovation (www.ijrti.org), ISSN:2456-3315, Vol.10, Issue 5, page no.c825-c832, May-2025, Available :http://www.ijrti.org/papers/IJRTI2505296.pdf*
- *Sarvesh Kumar Gupta, "Hybrid Cloud Pipelines for Regulated Industries", IJRAR - International Journal of Research and Analytical Reviews (IJRAR), E-ISSN 2348-1269, P- ISSN 2349-5138, Volume.12, Issue 2, Page No pp.705-712, May 2025, Available at : http://www.ijrar.org/IJRAR25B4662.pdf*
- *Sarvesh kumar Gupta, "Modernizing Legacy Data Systems in Agile Environments", IJRAR - International Journal of Research and Analytical Reviews (IJRAR), E-ISSN 2348-1269, P- ISSN 2349-5138, Volume.12, Issue 2, Page No pp.713-721, June 2025, Available at : http://www.ijrar.org/IJRAR25B4663.pdf*
- *Sarvesh Kumar Gupta, 2025. "Real-Time Data Ingestion with Kafka and AWS Tools", ESP Journal of Engineering & Technology Advancements 5(2): 285-290.*
- *Sarvesh kumar Gupta, "Designing Scalable Data Warehouses for Analytics", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.13, Issue 7, pp.h868-h876, July 2025, Available at :http://www.ijcrt.org/papers/IJCRT2507898.pdf*
- *Strategic Decision Intelligence Using Predictive Analytics in Modern Organizations. (2026). Global Journal of Innovative*

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

Research Perspectives (GJIRP), 2(2), May (1-8). https://doi.org/10.63345/gjirp.v2.i2.201

- *Sarvesh kumar Gupta. Best practices for oracle to PostgreSQL migration. International Journal of Science and Research Archive, 2025, 16(01), 1337-1344. Article DOI: https://doi.org/10.30574/ijsra.2025.16.1.2083*
- *Sarvesh kumar Gupta, "Metadata Lineage Frameworks for Data Governance", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.13, Issue 9, pp.c895-c903, September 2025, Available at :http://www.ijcrt.org/papers/IJCRT2509332.pdf*
- *Gupta, S. K. (2025). Machine Learning Integration in Spark-Based Pipelines. International Journal of Innovative Research in Technology (IJIRT), 12(4), 3020–3025.*
- *Sarvesh Kumar Gupta, 2025. "AI Powered Query Optimization Console: A Review of Intelligent Approaches for Real-Time Query Performance Enhancement in Database Systems", ESP Journal of Engineering & Technology Advancements 5(4): 180-192.*
- *Bharucha, S. (2026). Agile leadership practices and employee innovation in hybrid workplaces. International Journal for Research in Management and Pharmacy (IJRMP), 15(6), 56–63. https://doi.org/10.63345/ijrmp.v15.i6.1*
- *Sarvesh Kumar Gupta. Cloud ETL optimization with AWS glue and spark. World Journal of Advanced Engineering Technology and Sciences, 2026, 18(03), 207-214. Article DOI: https://doi.org/10.30574/wjaets.2026.18.3.0076*
- *Strategic Resilience Models for Enterprises in the Age of Continuous Disruption. (2026). E-Journal of Science and Emerging Technologies (EJSET), 2(2), May (26-33). https://doi.org/10.63345/ejset.v2.i2.201*
- *Bharucha, S. (2023). Digital legacy and innovation balance in family-owned enterprises. International Journal of Research in Modern Engineering & Emerging Technology (IJRMEET), 11(7). https://doi.org/10.63345/ijrmeet.org.v11.i7.1*
- *Autonomous Business Transformation Through Generative AI Integration. (2026). Global Journal of Innovative Research Perspectives (GJIRP), 2(2), Apr (83-91). https://doi.org/10.63345/gjirp.v2.i2.101*
- *Bharucha, S. (2023). Next-generation governance frameworks for multi-generational family businesses. International Journal for Research in Management and Pharmacy (IJRMP), 12*(10), 31–41. https://doi.org/10.63345/ijrmp.v12.i10.5*
- *Strategic Leadership for Hybrid Human–AI Workforce. (2025). International Journal of Medical Research And Innovation in Applied Science (IJMRIAS), 1(2), Apr (31-40). https://doi.org/10.63345/ijmrias.v1.i2.101*
- *Bharucha, S. (2022). Circular manufacturing ecosystems and sustainable competitive advantage. International Journal of Research in Humanities & Social Sciences (IJRHS), 10(9), 33–42. https://doi.org/10.63345/ijrhs.net.v10.i9.1*
- *AI-Driven Digital Product Passports for Sustainable Textile Supply Chains. (2025). World Journal of Future Technologies in Computer Science and Engineering, 1(4), Dec (41-50). https://doi.org/10.63345/wjftcse.v1.i4.301*
- *Bharucha, S. (2022). Predictive restructuring frameworks for organizational renewal. International Journal of Research in All Subjects in Multi Languages (IJRSML), 10(3), 68–77. https://doi.org/10.63345/ijrsml.v10.i3.1*
- *Bharucha, S. (2024). Business intelligence-based turnaround strategies for corporate recovery. International Journal for Research in Education (IJRE), 13 (8), 10–19. https://doi.org/10.63345/ijre.v13.i8.1*
- *Generative AI and the Reinvention of Management Education. (2026). Scientific Journal of Artificial Intelligence and Blockchain Technologies (SJAIBT), 1(2), Jun (1-9). https://doi.org/10.63345/sjaibt.v1.i2.301*

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org