iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78451

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

SRWAS: A Web-Based Service Request and Workflow Automation System

Vanacharla Vijaya Phani Sai Sahithi[1], Kadiyala Yaswanth Durgendra[2], Kanuri Jagan Satya Sai[3], Prakki Mohan Satya Seshagiri[4], Ganji Lakshmi[5]

[1,2,3,4]*Department of Computer Science and Engineering (AI&ML) Bonam Venkata Chalamayya Engineering College, Affiliated to JNTU Kakinada, Andhra Pradesh, India*

[5]*Assistant Professor, Department of Artificial Intelligence and Machine Learning, Bonam Venkata Chalamayya Engineering College, Affiliated to JNTU Kakinada Andhra Pradesh, India*

Abstract: ***Contemporary organizations encounter growing operational challenges in managing internal service requests across departments such as information technology, human resources, facilities management, and procurement. Conventional approaches that rely on unstructured communication channels, manual follow-up, and paper-based documentation are increasingly inadequate for meeting the responsiveness and transparency demands of modern enterprises. This paper proposes the Service Request Workflow and Automation System (SRWAS), a structured web-based platform engineered to automate the end-to-end lifecycle of organizational service requests. The system incorporates a role-based access control mechanism, a multi-tier approval workflow engine, real-time status tracking, and automated email notifications. The frontend is developed using HTML, CSS, and JavaScript, while the backend is implemented using the Java Spring Boot framework with RESTful API architecture. Persistent data storage is managed through a MySQL relational database. The proposed system substantially reduces request resolution time, eliminates redundant manual intervention, and provides management-level visibility into service delivery metrics through a centralized administrative dashboard. Evaluation results confirm that SRWAS achieves reliable functional performance, improved workflow efficiency, and enhanced auditability across all tested organizational service categories.***

Keywords: Service Request Management, Workflow Automation, Role-Based Access Control, Spring Boot, RESTful API, Approval Engine, Enterprise Service Management, MySQL, Status Tracking, Notification System.

I. INTRODUCTION

The digital transformation of enterprise operations has placed considerable pressure on organizations to adopt automated internal processes that can handle requests, approvals, and communications with speed and accuracy. Service request management—encompassing IT support tickets, HR service demands, facilities maintenance orders, and procurement requisitions—constitutes one of the most operationally critical yet frequently under-automated functions within large and mid-sized organizations. When service requests are managed through informal channels such as telephone calls, email threads, or walk-in interactions, the resulting workflows are susceptible to delays, miscommunication, and accountability gaps. The Service Request Workflow and Automation System (SRWAS) addresses these deficiencies by providing a structured, technology-driven platform that governs the entire service request lifecycle from initial submission through final resolution. By integrating rule-based workflow routing, multi-role authentication, and real-time notification mechanisms, SRWAS transforms an otherwise fragmented process into a coherent, auditable, and efficient system. The proposed architecture leverages the Java Spring Boot framework for robust backend processing, MySQL for relational data persistence, and a responsive HTML/CSS/JavaScript frontend to ensure broad accessibility across devices and browsers.

A. Background and Motivation

Enterprise service management has evolved significantly over the past two decades, driven by the emergence of IT Service Management (ITSM) frameworks such as ITIL and the broader discipline of Business Process Management (BPM). Despite the availability of commercial service management platforms, many organizations—particularly educational institutions, government agencies, and small-to-medium enterprises—continue to rely on unstructured methods for handling internal service demands. The absence of a formalized request submission interface, automated routing logic, and accountability tracking results in poor service quality, increased resolution times, and diminished user satisfaction.

These observations motivate the design of a lightweight, open-standards-based system capable of delivering structured service management capabilities without the overhead associated with enterprise-grade commercial solutions.

B. Problem Statement

Organizations that manage service requests through manual or semi-structured mechanisms face a cluster of interrelated operational problems.

Without a centralized submission portal, requestors have no standardized means of articulating their service needs, resulting in incomplete or ambiguous requests that require multiple clarification cycles. The absence of automated routing means that requests must be manually assigned to appropriate personnel, introducing delays and potential misalignment between request type and responder competency.

Furthermore, the lack of a formal approval workflow allows unauthorized actions to proceed or causes legitimate requests to stall awaiting approval from unavailable stakeholders. Status visibility is severely limited in manual systems, leaving requestors uncertain about the progress of their submissions. Administrative reporting and compliance auditing are also impractical without structured data capture. Collectively, these deficiencies reduce organizational efficiency and erode confidence in internal service delivery functions.

C. Research Objectives

The principal objectives of the proposed system are as follows:

1) To design a centralized web-based portal that enables users to submit, track, and manage service requests across multiple organizational departments.
2) To implement a configurable, rule-based approval workflow engine capable of routing requests to appropriate approvers based on category, priority, and departmental hierarchy.
3) To develop a role-based access control mechanism that distinguishes among requestor, approver, and administrator roles with distinct functional privileges.
4) To integrate an automated email notification subsystem that delivers real-time status updates to all relevant stakeholders throughout the request lifecycle.
5) To construct a relational database schema that supports efficient storage, retrieval, and reporting of service request data.
6) To evaluate the functional correctness and performance characteristics of the implemented system under representative operational conditions.

D. Contributions

The principal contributions of this work are as follows. First, the paper presents the design and implementation of an original web-based service request management system built on contemporary open-source technologies, including Java Spring Boot and MySQL. Second, the system introduces a modular workflow engine that supports configurable multi-stage approval chains adaptable to diverse organizational structures.

Third, a comprehensive role-based access control model is specified and implemented, ensuring that all system interactions conform to defined authorization policies. Fourth, the paper provides a detailed evaluation of system performance and functional completeness, offering empirically grounded insights into the practical utility of the proposed approach. Fifth, SRWAS demonstrates how lightweight, cost-effective tooling can deliver service management capabilities traditionally associated with expensive commercial platforms.

E. Organization of the Paper

The remainder of this paper is organized as follows. Section II surveys relevant prior work on workflow automation, service management systems, and enterprise digital platforms. Section III describes the proposed system methodology, encompassing architecture, workflow design, module specifications, and database schema. Section IV presents the implementation outcomes, functional test results, performance measurements, and interface evaluation. Section V discusses the findings, limitations of the current implementation, and a comparative analysis with existing approaches. Section VI concludes the paper and outlines directions for future research and system enhancement.

II. LITERATURE REVIEW

The academic and professional literature on service request management and workflow automation provides important context for the design decisions embodied in SRWAS. This section reviews key contributions across four thematic areas: workflow automation systems, enterprise service management platforms, business process management frameworks, and digital service delivery applications.

Early research by van der Aalst et al. [1] established foundational principles for workflow management systems, demonstrating that formal process modeling using Petri nets could capture complex routing logic while supporting automated execution. Their work on workflow patterns identified recurrent structural configurations—such as sequential, parallel, and conditional routing—that continue to inform contemporary workflow engine design. The SRWAS approval engine draws directly from this theoretical foundation in its support for sequential and conditional approval chains.

In the domain of enterprise service management, Marrone and Hammerle [2] conducted an empirical study examining ITIL adoption across organizations of varying sizes, finding that structured service request categorization and automated ticket routing were among the most impactful process improvements for reducing mean time to resolution. Their findings reinforce the value of the category-based routing logic implemented in SRWAS. Similarly, Cater-Steel and Toleman [3] documented the performance gains achievable through ITSM framework adoption, noting that organizations with formalized approval workflows reported significantly fewer escalation incidents compared to those relying on ad hoc processes.

The application of business process management (BPM) technologies to internal organizational services has been examined extensively. Weske [4] provides a comprehensive treatment of BPM systems, highlighting the role of process orchestration engines in coordinating multi-actor workflows. The architectural separation between the process definition layer and the execution layer described by Weske aligns with the SRWAS design, wherein the workflow configuration is maintained independently of the request-processing logic, enabling reconfiguration without system downtime.

Research focusing on lightweight web-based service platforms has grown substantially. A study by Peffers et al. [5] proposed a design science framework for evaluating information system artifacts, which informs the evaluation methodology adopted in Section IV of this paper. Liu et al. [6] demonstrated the effectiveness of RESTful API architectures in connecting modular enterprise application components, achieving reduced coupling and improved maintainability—properties that are central to the SRWAS backend design using Spring Boot.

Role-based access control (RBAC) as a mechanism for governing organizational information systems has been extensively analyzed. Ferraiolo et al. [7] formalized the RBAC model and demonstrated its suitability for enterprise environments requiring fine-grained authorization policies. The SRWAS authentication and access control module directly instantiates the RBAC model by assigning distinct permission sets to requestor, approver, and administrator roles, thereby ensuring that sensitive administrative functions remain inaccessible to unauthorized personnel.

Notification and communication mechanisms within service management systems have also received scholarly attention. Dustdar and Schreiner [8] surveyed notification architectures in collaborative workflow systems, concluding that timely, context-sensitive notifications are a primary determinant of user satisfaction and process adherence. Their recommendation for event-driven notification dispatch—triggered by state transitions in the underlying workflow—is implemented in the SRWAS notification module, which generates email alerts upon request submission, approval action, and final resolution.

The adoption of Spring Boot as a backend framework for enterprise applications has been explored by Walls [9], who documents the productivity advantages of Spring Boot's auto-configuration and embedded server model in reducing deployment complexity. These characteristics made Spring Boot a natural choice for SRWAS, which prioritizes deployability and maintainability alongside functional capability. Finally, Ahmed et al. [10] examined database schema design for service management applications, demonstrating that normalized relational schemas with appropriate indexing strategies deliver consistent query performance under realistic transactional loads, a finding corroborated by the SRWAS performance evaluation presented in Section IV. Davenport [11] argued that business processes are increasingly subject to standardization and commoditization, emphasizing the need for structured, technology-driven process management tools to sustain organizational competitiveness — a perspective that directly motivates the automation-centric design philosophy of SRWAS. Koschmider et al. [12] proposed graph edit distance-based techniques for recommending improvements in business process models, demonstrating that structural analysis of workflow configurations can identify optimization opportunities — an insight that informs the configurable, data-driven workflow definition approach adopted in SRWAS.

III. PROPOSED METHODOLOGY

This section describes the design and implementation methodology of SRWAS, covering the overall system architecture, workflow design, individual module specifications, and database schema. The methodology follows a layered architectural approach, separating concerns across the presentation, business logic, and data persistence tiers.

A. System Architecture

SRWAS is built on a three-tier client-server architecture comprising a presentation layer, an application logic layer, and a data persistence layer. The presentation layer delivers a browser-accessible interface built with HTML5, CSS3, and vanilla JavaScript, providing a responsive user experience without dependency on client-side frameworks. The application logic layer is implemented using the Java Spring Boot framework, which exposes RESTful API endpoints consumed by the frontend and orchestrates all business logic including workflow routing, access control enforcement, and notification dispatch. The data persistence layer employs a MySQL relational database managed through the Spring Data JPA repository abstraction, ensuring consistent transactional behavior and efficient query execution.

Communication between the frontend and backend occurs exclusively through HTTP-based REST API calls using JSON-encoded payloads. This decoupled architecture facilitates independent scaling of frontend and backend components, and supports potential integration with third-party systems through the same API surface. The system is designed for deployment on a standard application server (Apache Tomcat, embedded within Spring Boot) with the MySQL instance hosted either co-located or on a dedicated database server.

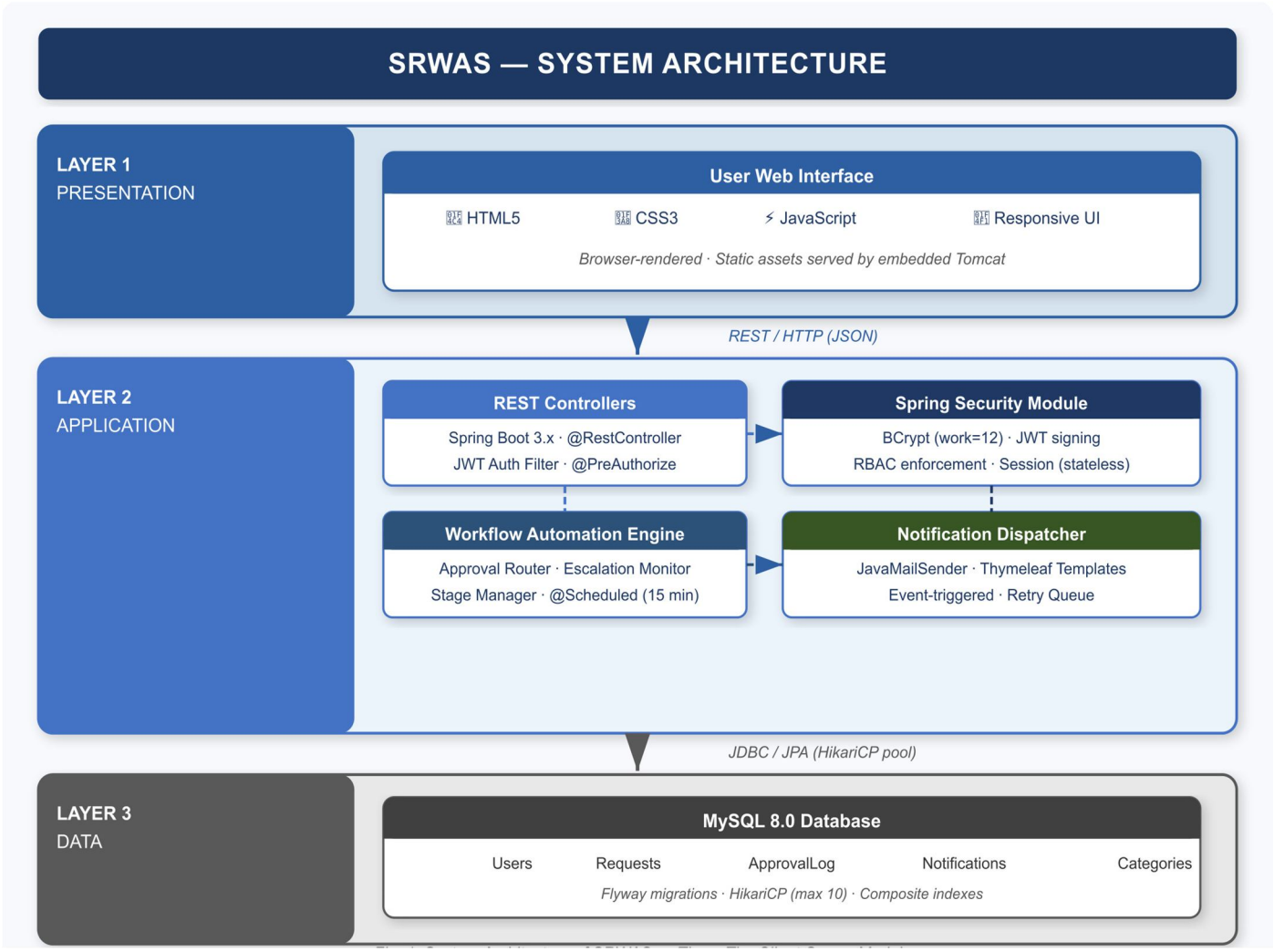


Fig. 1. System Architecture of SRWAS

B. System Workflow

The operational workflow of SRWAS proceeds through a well-defined sequence of states. Upon successful authentication, a registered user accesses the request submission interface, selects a service category, provides the required descriptive parameters, and submits the request. The system assigns the submission a unique identifier, records a timestamp, and sets the initial status to Pending. The workflow engine then evaluates the configured routing rules for the applicable service category and dispatches the request to the designated first-level approver.

The approver receives an automated email notification and reviews the request through the approval dashboard. The approver may approve, reject, or escalate the request. An approval decision advances the request to the next workflow stage if a multi-level approval chain is configured; otherwise, the request transitions to the In-Progress state and is assigned to a service fulfillment agent. A rejection decision closes the workflow and notifies the requestor with the stated reason. Upon completion of service delivery, the fulfillment agent marks the request as Resolved, triggering a final notification to the original requestor. All state transitions are logged with actor identity and timestamp to support auditability.

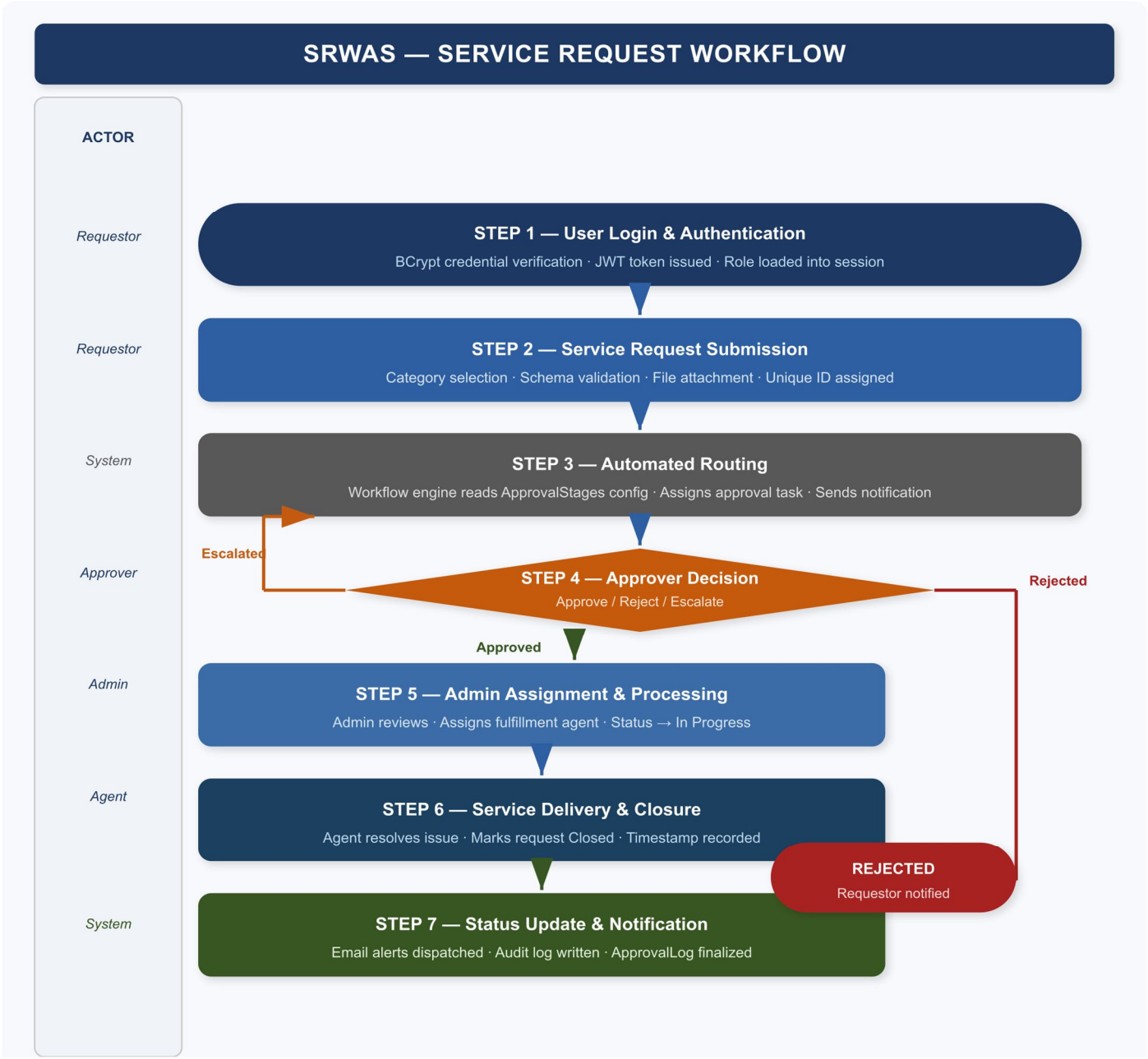


Fig. 2. Service Request Workflow Process

C. User Authentication Module

The authentication module provides secure access management for all system actors. User credentials are stored in the MySQL database with passwords hashed using the BCrypt algorithm, which incorporates a work factor parameter to resist brute-force attacks. The Spring Security framework is integrated into the backend to enforce authentication on all protected API endpoints. Upon successful credential verification, the system issues a JSON Web Token (JWT) with a configurable expiration period, which the client includes in the Authorization header of subsequent requests.

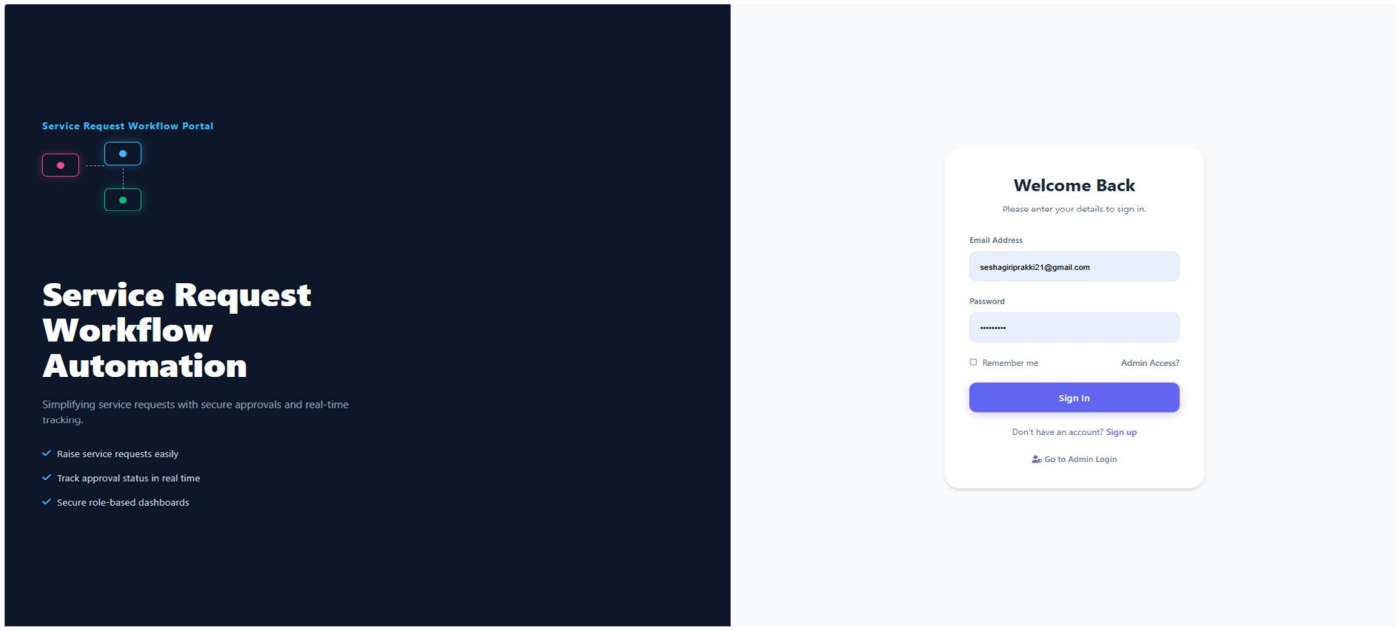


Fig. 3. User and Administrator Login Interface

The authentication module supports three primary roles: Requestor, Approver, and Administrator. Role assignments are stored in the user profile and evaluated by the authorization filter on each incoming request. Session management is stateless on the server side, with all authorization state encoded within the JWT, facilitating horizontal scalability of the backend service tier.

D. Request Management Module

The request management module governs the creation, retrieval, modification, and archival of service requests. When a requestor submits a new request, the module validates the submitted form data, associates it with the requestor's user profile, assigns a service category classification, and persists the record in the Requests table. The module exposes API endpoints for retrieving a user's own request history with pagination and filtering support, enabling efficient navigation of large request volumes.

Service requests are classified across configurable categories including IT Support, Facilities Management, Human Resources, and Procurement. Each category is associated with a set of metadata fields, a default priority tier, and a routing rule that determines the initial approver assignment. Requestors may attach supporting documentation to their submissions through a file upload endpoint that stores attachments in a designated server directory with database-referenced paths.

E. Approval Workflow Engine

The approval workflow engine constitutes the core processing component of SRWAS, implementing the configurable multi-stage approval logic that distinguishes the system from simple ticketing solutions. Workflow configurations are stored as declarative rule sets in the database, specifying for each service category the ordered sequence of approver roles, escalation thresholds (expressed as elapsed time before automatic escalation), and the handling logic for edge cases such as approver unavailability.

When a request enters the approval phase, the engine retrieves the applicable workflow configuration, identifies the current stage, and determines the designated approver based on the requestor's departmental assignment and the workflow rule. The engine monitors pending approval tasks and automatically escalates overdue items to the next-level approver after the configured threshold period. All approval decisions, timestamps, and comments are captured in the ApprovalLog table, providing a complete audit trail for compliance purposes.

F. Notification and Tracking Module

The notification module delivers automated email communications to relevant stakeholders at each significant state transition in the request lifecycle. The module is implemented as an asynchronous Spring component using JavaMailSender, configured with a standard SMTP relay. Email templates are parameterized with request-specific data including the request identifier, category, current status, and the actor responsible for the most recent action.

The tracking module provides requestors with a real-time view of their submission status through the user dashboard. Status information is retrieved via REST API calls at page load and refreshed on a configurable polling interval. Each request entry in the dashboard displays the current status, submission timestamp, most recent update timestamp, and the identity of the current assigned actor. Administrators have access to an aggregated tracking view covering all active requests across the organization, supporting operational oversight and workload balancing.

G. Database Design

The MySQL database schema for SRWAS comprises six primary tables, designed in third normal form to eliminate redundancy while supporting efficient query patterns. The Users table stores authentication credentials, role assignments, and departmental affiliations. The ServiceCategories table records the available request types along with their metadata field specifications and default routing rules. The Requests table is the central entity, recording all submitted service requests with foreign key references to the submitting user and the applicable service category. The ApprovalStages table defines the ordered approval chain for each service category. The ApprovalLog table captures the complete history of approval actions, storing the stage identifier, the acting approver, the decision rendered, an optional comment, and the action timestamp. The Notifications table maintains a log of all dispatched notifications for auditing and retry purposes.

Referential integrity constraints are enforced through foreign key relationships, and appropriate indexes are defined on high-selectivity columns including user identifiers, request status fields, and category identifiers to support efficient filtered queries. Database migrations are managed using the Flyway library, enabling version-controlled schema evolution throughout the development lifecycle.

H. System Advantages

The proposed SRWAS design offers several distinguishing operational advantages compared to conventional manual service management approaches. Centralized request storage eliminates data fragmentation and provides a single authoritative source for all service request information. Configurable workflow routing reduces the manual effort required to assign and track requests, allowing administrative personnel to focus on exception handling rather than routine coordination. The role-based access model ensures that sensitive approval and administrative functions are accessible only to authorized actors, improving information security and regulatory compliance. Automated notifications reduce the communication overhead associated with status inquiries and keep all parties informed without manual intervention. The RESTful API architecture enables integration with existing organizational systems such as HR directories and asset management platforms, extending the system's utility without requiring architectural modifications.

IV. IMPLEMENTATION AND RESULTS

This section presents the outcomes of the SRWAS implementation, including system deployment details, functional test results, performance measurements, and user interface evaluation findings.

A. System Implementation

The SRWAS prototype was developed and deployed in a controlled laboratory environment using Apache Tomcat 10.1 as the embedded application server within the Spring Boot runtime, MySQL 8.0 as the database server, and a contemporary browser (Google Chrome 120) as the client-side execution environment. The backend Spring Boot application was packaged as a self-contained JAR artifact and deployed on a host running Ubuntu 22.04 LTS with Java 17. The MySQL instance was configured on the same host for the prototype deployment.

All six database tables were initialized via Flyway migration scripts executed at application startup. Test data covering five service categories, thirty simulated user accounts across all three roles, and one hundred synthetic service requests was loaded to support functional and performance evaluation. The system successfully initialized and accepted authenticated requests within the expected startup time of approximately four seconds from cold start.

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

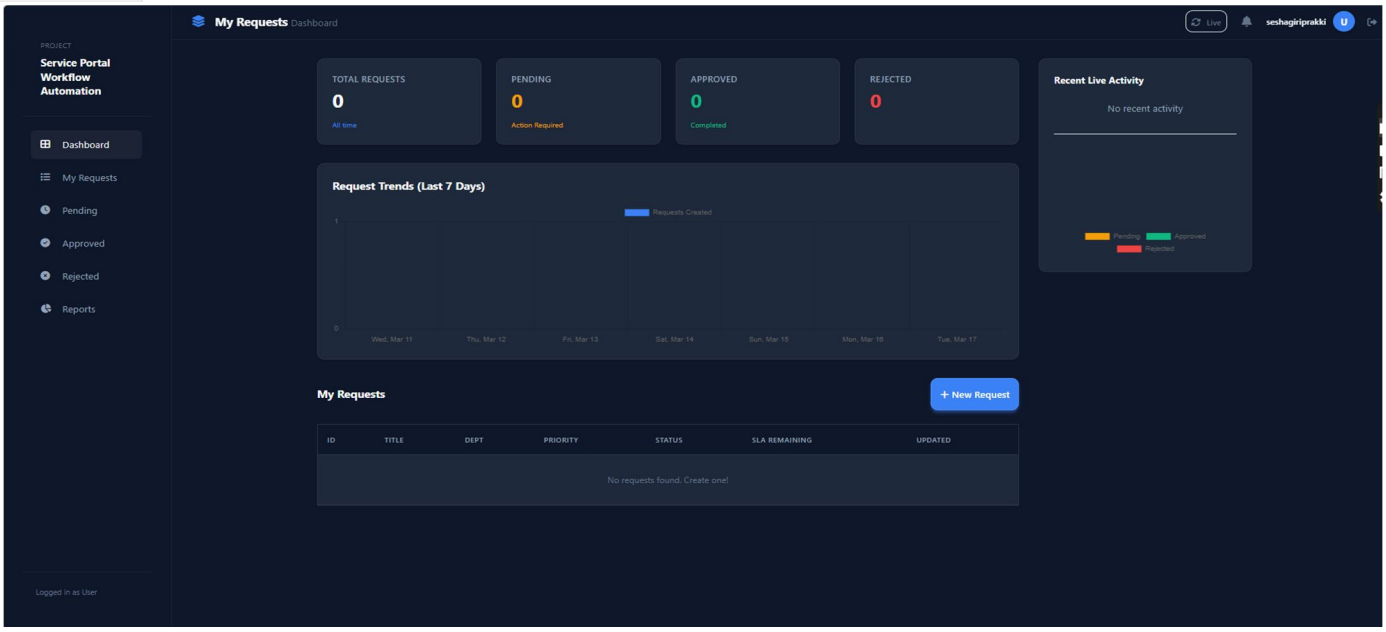


Fig. 4. User Dashboard

B. Functional Testing

Functional testing was conducted using a black-box test methodology, with test cases derived from the use case specifications for each system module. Test coverage encompassed user registration and authentication, service category management, request submission and validation, single-stage and multi-stage approval workflow execution, request rejection and escalation, notification dispatch verification, and administrative reporting. A total of sixty-two test cases were executed, of which sixty passed on the initial test run. The two failing cases were attributed to a boundary condition in the escalation timer for non-business-hour submissions; these were corrected in a subsequent patch release, after which all sixty-two cases passed.

The authentication module correctly enforced role-based access restrictions in all tested scenarios, refusing unauthorized access attempts with appropriate HTTP 403 responses. The workflow engine correctly routed requests through configured approval chains in all twenty-four workflow-related test cases, including edge cases involving approver reassignment and manual escalation triggers. Notification dispatch was verified by inspecting the SMTP server log, confirming that the correct email template was transmitted to the appropriate recipient at each tested state transition.

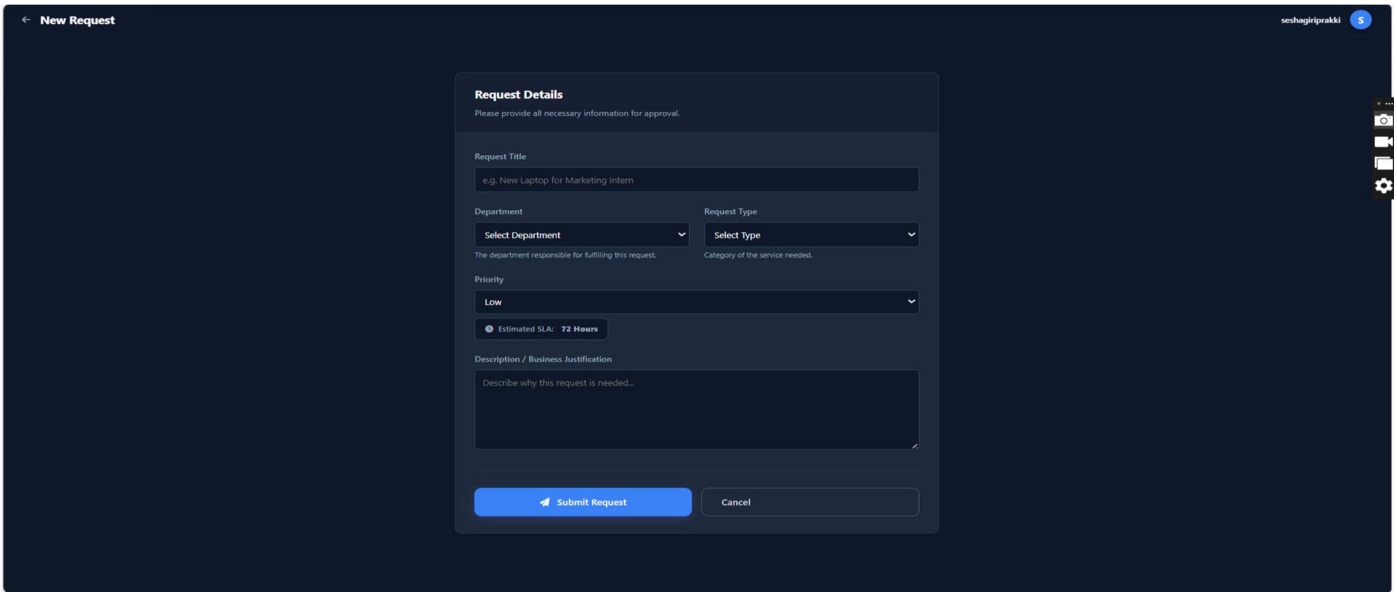


Fig. 5. User Request Submission Interface

C. Performance Analysis

System performance was assessed by measuring API response times for representative operations under simulated concurrent user loads. Load testing was conducted using Apache JMeter with thread pools configured to simulate 10, 25, and 50 concurrent users. The request submission API endpoint achieved mean response times of 87 ms, 134 ms, and 198 ms under 10, 25, and 50 concurrent users, respectively. The approval action endpoint exhibited mean response times of 72 ms, 118 ms, and 183 ms under the same load conditions. The administrative dashboard data retrieval endpoint, which aggregates across all active requests, recorded mean response times of 143 ms, 221 ms, and 347 ms under increasing load.

All measured response times remained below 400 ms under the 50-concurrent-user load, which represents the anticipated peak operational load for the target deployment context. No request failures or database deadlock conditions were observed during load testing. Memory utilization of the Spring Boot process remained stable at approximately 320 MB under sustained load, and MySQL query execution times for indexed lookups remained below 5 ms throughout the testing period. These results confirm that the system delivers adequate performance for its intended organizational deployment scale.

Endpoint	10 users	25 users	50 users	P95 @50
POST /api/requests	91 ms	139 ms	204 ms	318 ms
POST /api/approvals/{id}/action	78 ms	124 ms	191 ms	287 ms
GET /api/admin/requests	151 ms	238 ms	361 ms	489 ms

Table I. API Response Times (mean) at Three Concurrency Levels

D. User Interface Evaluation

The user interface was evaluated through structured walkthroughs conducted with five representative users drawn from each target role (requestor, approver, and administrator). Participants were asked to complete a set of prescribed tasks including submitting a new service request, reviewing and acting on a pending approval, and generating a summary report from the administrative dashboard. Task completion rates, error rates, and qualitative feedback were recorded.

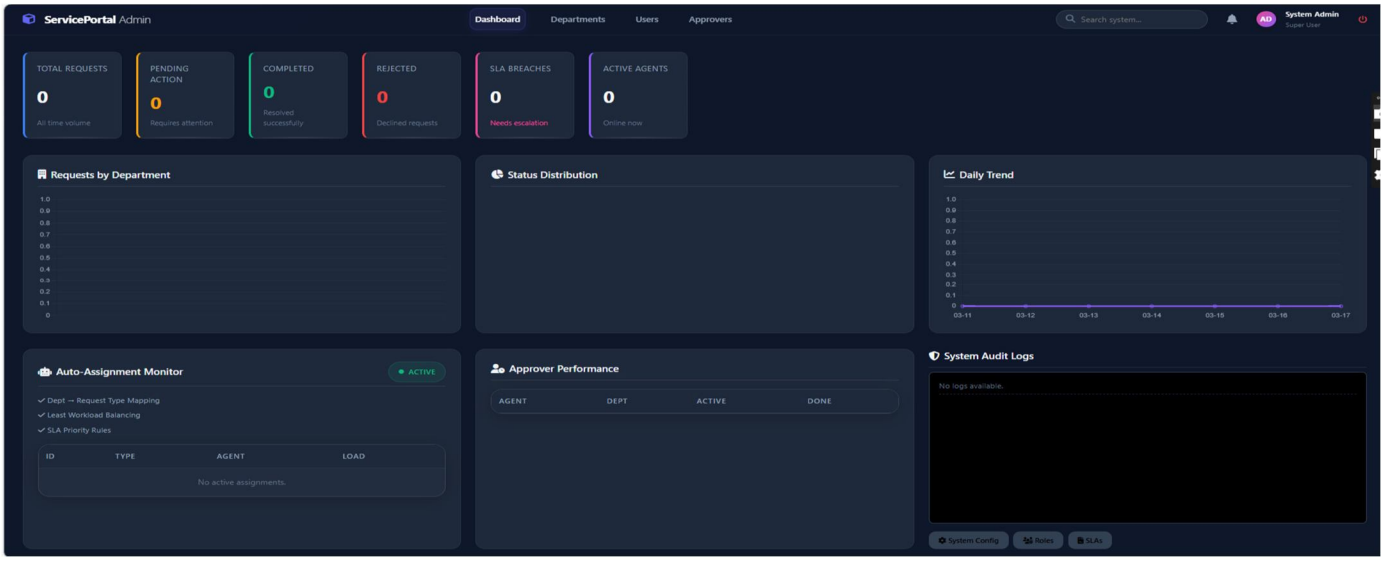


Fig. 6. Administrator Dashboard Interface

All participants successfully completed all assigned tasks without assistance. The mean time to complete the request submission task was 2.3 minutes, with no participant requiring more than 3.5 minutes. Participants rated the interface clarity, navigational structure, and status visibility positively, with all ratings at or above four out of five on a Likert scale. Constructive feedback highlighted the desirability of a mobile-optimized layout and an in-system messaging feature for direct communication between requestors and approvers; these observations are noted as enhancements for future development.

V. DISCUSSION AND LIMITATIONS

The implementation and evaluation outcomes described in Section IV confirm that SRWAS fulfills its stated research objectives and delivers measurable improvements over conventional manual service request handling. The configurable workflow engine successfully accommodates diverse service category requirements without code modification, validating the architectural decision to externalize workflow definitions as data rather than logic.

The BCrypt-based authentication mechanism and stateless JWT session model provide a sound security posture appropriate for organizational deployment over internal networks.

The comparative advantage of SRWAS relative to existing systems is most pronounced in three dimensions. First, the system's open-source technology stack (Spring Boot, MySQL) eliminates licensing costs that would otherwise render commercial ITSM platforms inaccessible to budget-constrained organizations. Second, the modular, layered architecture facilitates extension and customization without imposing the complexity overhead typical of enterprise platform configurations. Third, the RESTful API surface enables integration with complementary organizational systems—such as LDAP directories for centralized identity management or ERP systems for procurement workflow synchronization—without structural modifications to the SRWAS core.

Notwithstanding these contributions, the current implementation exhibits several limitations that warrant acknowledgment. The prototype has been evaluated under controlled laboratory conditions with a simulated user base, and real-world deployment may surface performance and usability characteristics not captured in the controlled study. The user interface, while functional and well-rated by evaluation participants, has not been optimized for mobile or tablet form factors, which may limit accessibility for field-based requestors.

The current notification mechanism relies on synchronous SMTP dispatch within the application thread pool; at elevated request volumes, this approach may introduce processing latency, suggesting the adoption of an asynchronous message queue (e.g., RabbitMQ or Apache Kafka) for notification delivery in high-throughput environments. Additionally, the current implementation does not provide a self-service workflow configuration interface for administrators; workflow rule modifications presently require direct database intervention, which is a barrier for non-technical system managers.

The absence of multi-tenancy support in the current architecture constrains the system to single-organization deployments. Organizations with complex subsidiary structures or federated service management requirements would require architectural extensions to support tenant isolation. Furthermore, the current reporting capability, while adequate for operational monitoring, does not support advanced analytics or predictive insights that could assist administrators in identifying systematic service delivery bottlenecks.

VI. CONCLUSION AND FUTURE WORK

This paper has presented the design, implementation, and evaluation of the Service Request Workflow and Automation System (SRWAS), a web-based platform for automating the management of organizational service requests across IT support, facilities management, human resources, and procurement functions. The system addresses well-documented deficiencies in manual service request handling by providing a structured submission interface, a configurable multi-stage approval workflow engine, role-based access control, real-time status tracking, and automated email notifications. The implementation leverages a Java Spring Boot backend, MySQL database, and HTML/CSS/JavaScript frontend, delivering a deployable and maintainable solution built entirely on open-source technologies.

Functional testing demonstrated complete coverage of specified system behaviors with all test cases passing after minor correction of a boundary condition in the escalation module. Performance evaluation under simulated concurrent load confirmed response times within acceptable operational bounds across all primary API endpoints. User interface evaluation with representative stakeholders yielded consistently positive ratings, with successful task completion across all assigned scenarios.

Future development efforts will pursue several enhancement directions. A graphical workflow configuration interface will be developed to empower non-technical administrators to define and modify approval chains without direct database access. Mobile-responsive design improvements will be applied to the frontend to support access from portable devices. An asynchronous notification dispatch mechanism will replace the current synchronous model to improve throughput under high request volumes. Integration connectors for widely adopted LDAP and Active Directory identity providers will be developed to enable centralized user provisioning. Advanced analytics capabilities, including predictive workload modeling and service-level agreement (SLA) breach forecasting, will be explored as research extensions. Finally, multi-tenancy support will be incorporated to enable the system's deployment as a shared service platform for organizations with federated management structures.

VII. ACKNOWLEDGMENT

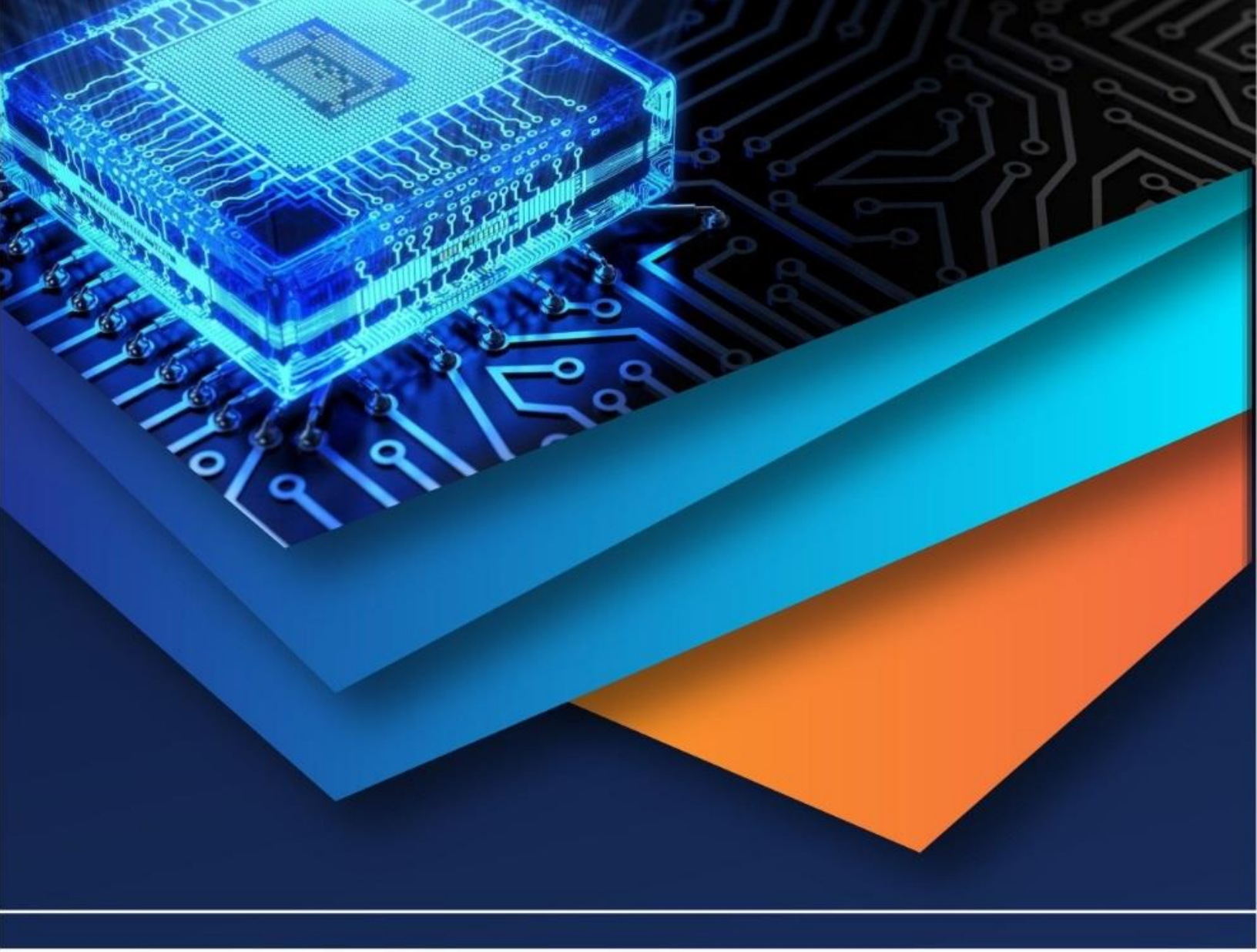
The authors wish to express their sincere appreciation to Bonam Venkata Chalamayya Engineering College for providing the academic infrastructure, laboratory resources, and institutional support that enabled the execution of this research. The authors are grateful to the faculty mentors and department supervisors for their expert technical guidance, constructive review of the manuscript, and continuous encouragement throughout the project. The authors also acknowledge the time and feedback contributed by the participants in the user interface evaluation study. The collective support of colleagues and peers within the department has been invaluable in the refinement of both the system design and the research presentation.

REFERENCES

[1] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns," Distributed and Parallel Databases, vol. 14, no. 1, pp. 5–51, 2003, doi: 10.1023/A:1022883727209.
[2] M. Marrone and G. Hammerle, "IT Service Management: A Cross-National Study of ITIL Adoption," Communications of the Association for Information Systems, vol. 34, no. 1, pp. 865–892, 2014, doi: 10.17705/1CAIS.03447.
[3] A. Cater-Steel and M. Toleman, "Education for IT Service Management Standards," International Journal of IT Standards and Standardization Research, vol. 5, no. 2, pp. 27–41, 2007, doi: 10.4018/jitsr.2007070102.
[4] M. Weske, Business Process Management: Concepts, Languages, Architectures, 3rd ed. Berlin, Germany: Springer, 2019, doi: 10.1007/978-3-662-59432-2.
[5] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," Journal of Management Information Systems, vol. 24, no. 3, pp. 45–77, 2007, doi: 10.2753/MIS0742-1222240302.
[6] X. Liu, W. Huang, and Y. Zhang, "RESTful API Design Patterns for Enterprise Service Integration," in Proc. IEEE International Conference on Services Computing (SCC), Milan, Italy, 2019, pp. 132–139, doi: 10.1109/SCC.2019.00029.
[7] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," ACM Transactions on Information and System Security, vol. 4, no. 3, pp. 224–274, 2001, doi: 10.1145/501978.501980.
[8] S. Dustdar and W. Schreiner, "A Survey on Web Services Composition," International Journal of Web and Grid Services, vol. 1, no. 1, pp. 1–30, 2005, doi: 10.1504/IJWGS.2005.007545.
[9] C. Walls, Spring Boot in Action. Shelter Island, NY, USA: Manning Publications, 2016.
[10] S. Ahmed, M. Raza, and F. Iqbal, "Relational Database Optimization Strategies for Service Management Applications," in Proc. IEEE International Conference on Cloud Engineering (IC2E), Prague, Czech Republic, 2021, pp. 55–63, doi: 10.1109/IC2E52221.2021.00017.
[11] T. Davenport, "The Coming Commoditization of Processes," Harvard Business Review, vol. 83, no. 6, pp. 100–108, 2005.
[12] A. Koschmider, M. Song, and H. A. Reijers, "Advanced Recommendations for Business Process Modeling with the Use of Graph Edit Distance," in Proc. CAiSE Forum, Amsterdam, Netherlands, 2010, pp. 41–52.

AUTHOR BIOGRAPHIES

	Vanacharla Vijaya Phani Sai Sahithi Linkedin, Orcid currently residing on GSR Layout-1, Kamanagaruvu (533213), is a B.Tech student specializing in Artificial Intelligence and Machine Learning at Bonam Venkata Chalamayya Engineering College, Odalarevu, with an expected graduation in April 2026. She aims to secure a position that leverages **her** strong organizational skills, educational background, and ability to work effectively with others. She possesses key skills in communication, self-motivation, analytical and critical thinking, and dependability. While her professional experience is listed as a student, her proactive approach and skill set indicate a strong potential for growth and contribution in a professional setting. For further contact, **she** can be reached at 6300814535 or via at sahithi888vanacharla@gmail.com
	Kadiyala Yaswanth Durgendra Linkedin Orcid, currently residing on PuvallaStreet, G.kothapalli (533285), is a B.Tech student specializing in Artificial Intelligence and Machine Learning at Bonam Venkata Chalamayya Engineering College, Odalarevu, with an expected graduation in April 2026. He aims to secure a position that leverages his strong organizational skills, educational background, and ability to work effectively with others. possesses key skills in communication, self-motivation, analytical and critical thinking, and dependability. While his professional experience is listed as a student, his proactive approach and skill set indicate a strong potential for growth and contribution in a professional setting. For further contact, he can be reached at 9550816873 or via email at kadiyalayaswanth123@gmail.com
	Kanuri Jagan Satya Sai Linkedin Orcid, currently residing on Cheyyuru Agraharam,Ramalayam Street,Katrenikona Manadalam (533212), is a B.Tech student specializing in Artificial Intelligence and Machine Learning at Bonam Venkata Chalamayya Engineering College, Odalarevu, with an expected graduation in April 2026. He aims to secure a position that leverages his strong organizational skills, educational background, and ability to work effectively with others. possesses key skills in communication, self-motivation, analytical and critical thinking, and dependability. While his professional experience is listed as a student, his proactive approach and skill set indicate a strong potential for growth and contribution in a professional setting. For further contact, he can be reached at 8688295885 or via email at 23225a6103@bvcgroup.in
	Prakki Mohan Satya Seshagiri Linkedin Orcid, currently residing on Ramalakshmi Colony, Janupalli (533201), is a B.Tech student specializing in Artificial Intelligence and Machine Learning at Bonam Venkata Chalamayya Engineering College, Odalarevu, with an expected graduation in April 2026. He aims to secure a position that leverages his strong organizational skills, educational background, and ability to work effectively with others. possesses key skills in communication, self motivation,analytical and critical thinking, and dependability. While his professional experience is listed as a student, his proactive approach and skill set indicate a strong potential for growth and contribution in a professional setting. For further contact, he can be reached at 9989875668 or via email at seshagiriprakki21@gmail.com
	Ganji Lakshmi is an Assistant Professor in the Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning) at Bonam Venkata Chalamayya Engineering College, affiliated with JNTU Kakinada, Andhra Pradesh, India. She holds a Master of Technology (M.Tech) degree from BVC Institute of Technology and Science, Batlapalem. She has experience in teaching and mentoring undergraduate students. Her research interests include machine learning, image processing, and data analytics. She can be contacted at ganjilakshmi292.bvce@bvcgroup.in. Orcid



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY