IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, May 2026
Impact Factor: 8.2

Raffles CRM System: An Enterprise-Grade Backend for Automated Student Acquisition Lifecycle Management in Higher Education

Sanjeeb Ray[1] and Rajendra Singh[2]
[1] Department of Computer Science and Engineering
[2] Dean, Department of Computer Science and Engineering,
Raffles University, Neemrana, Rajasthan, India
sanjeeb.ray.official@gmail.com[1], rajendra.singh@rafflesuniversity.edu.in2

Abstract: *This paper presents the design, implementation, and deployment of the Raffles CRM System, an enterprise-grade backend Customer Relationship Management platform purpose-built for Raffles University, Neemrana. Developed using Java 21 and Spring Boot 4.0.2, the system automates the complete student acquisition lifecycle — from multi-channel lead intake through automated counselor assignment, offline session scheduling, mentor evaluation, and final enrollment. The architecture is structured as a seven-module Maven monorepo enforcing strict separation of concerns across 38 JPA entities, 47 service classes, and 170+ REST API endpoints. Key technical contributions include a multi-tier Round-Robin Lead Assignment Engine with thread-safe cyclic distribution, a 14-day automated lead demotion scheduler with paginated batch processing, a six-layer defense-in-depth security pipeline combining JWT HS512 authentication, Redis-backed token blacklisting, Bucket4j rate limiting, and role-based access control, and real-time WebSocket dashboard updates via STOMP/SockJS. The system is production-deployed on Neon Cloud PostgreSQL with Docker containerization and a GitHub Actions CI/CD pipeline. All 15 defined project objectives were successfully implemented and validated. The system demonstrates that institution-specific operational challenges in university admissions — traditionally managed through spreadsheets and informal communication — can be eliminated through systematic backend engineering.*

Keywords: Raffles CRM System: An Enterprise-Grade Backend for Automated Student Acquisition Lifecycle Management in Higher Education

I. INTRODUCTION

University admissions management represents one of the most operationally complex workflows in higher education administration. Prospective students submit inquiries through multiple digital channels simultaneously, counselors must respond within narrow time windows to remain competitive with peer institutions, workload distribution must be equitable across staff, and every interaction must be traceable for compliance and accountability. Traditional approaches to managing this process — spreadsheets, shared email inboxes, and informal phone-based coordination — are fundamentally incapable of meeting these requirements at scale.

The Raffles CRM System (https://rafunirp.com) was developed to address this operational gap. Unlike generic CRM platforms, which provide horizontal functionality across industries, this system is purpose-built for the specific workflows of higher education admissions. It models the full student acquisition lifecycle as a formal state machine, enforces data-driven assignment rules that eliminate human bias from lead distribution, and provides management with real-time operational visibility that previously required end-of-day manual reporting.

DOI: 10.48175/IJARSCT-35850

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 11, May 2026 Impact Factor: 8.2

The system is designed around three non-negotiable operational principles. First, zero lead leakage: every inquiry submitted through any channel is captured, deduplicated, and tracked from the moment of submission. Second, equitable workload distribution: no counselor is overloaded or underutilized, and assignment decisions are made algorithmically rather than manually. Third, complete audit traceability: every action on every entity is permanently recorded with actor identity and timestamp, providing a tamper-evident history for compliance and accountability.

This paper is organized as follows. Section II reviews related work in CRM systems and educational process automation. Section III describes the system architecture and methodology. Section IV presents the core implementation components. Section V discusses results and system validation. Section VI identifies limitations and future work. Section VII concludes the paper.

II. LITERATURE REVIEW

Customer Relationship Management systems have been extensively studied in commercial contexts, with a substantial body of literature addressing lead management, sales pipeline automation, and customer lifecycle tracking. However, the application of CRM principles to higher education admissions management remains comparatively underexplored, particularly at the backend engineering level.

Prior work in educational CRM has primarily focused on front-end user experience and adoption behavior rather than backend architecture. Studies examining CRM adoption in universities have identified lead leakage, counselor accountability, and response time as the three most significant operational failure modes in manual admissions processes. Research on round-robin scheduling algorithms in service assignment contexts has demonstrated that cyclic distribution with priority tiering produces statistically fairer workload distributions than manager-discretion-based assignment, particularly in teams with mixed seniority levels.

The application of JWT-based stateless authentication with Redis-backed token blacklisting for enterprise backends has been validated in production environments, with studies confirming that the combination of stateless tokens and stateful invalidation provides both horizontal scalability and instant session termination capability. Rate limiting through token bucket algorithms, as implemented by Bucket4j, has been shown to reduce brute-force attack success rates by eliminating timing-based credential enumeration.

Automated scheduler-based lead quality management — specifically the pattern of demoting stagnant leads based on inactivity thresholds — has been identified in sales operations literature as one of the highest-impact interventions for reducing lead neglect rates. The paginated batch processing pattern for large-dataset schedulers has been validated as the appropriate approach for avoiding JVM heap exhaustion in Spring Boot applications processing tens of thousands of records nightly.

Flyway-managed database migrations, used in this system for schema versioning, have been established as the industry standard for maintaining schema consistency across development, staging, and production environments in enterprise Java applications.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

A. Technology Stack

The system is built on Java 21 (LTS) with Spring Boot 4.0.2 as the application framework. Spring Data JPA with Hibernate manages the ORM layer across 38 entities. Apache Maven is used as the build tool with a multi-module POM configuration. API documentation is provided through SpringDoc OpenAPI 2.5.0 (Swagger UI).

The security layer uses Spring Security with JWT tokens (HS512 algorithm, jjwt 0.12.6), Bucket4j 8.10.0 for rate limiting, BCrypt for password hashing, and Spring Data Redis for token blacklisting and session state management. The data layer uses PostgreSQL 16 as the primary relational store, HikariCP for connection pooling, Flyway for versioned schema migrations, and Redis for high-speed in-memory caching. File storage is provided by AWS S3 (SDK v2.21.28, ap-south-1 region), Excel processing by Apache POI OOXML 5.2.3, and real-time communication by WebSocket over STOMP/SockJS.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, May 2026
Impact Factor: 8.2

B. Multi-Module Architecture

The system is structured as a seven-module Maven monorepo enforcing a strict unidirectional dependency hierarchy. The dependency chain flows as follows: CRM_ENTITIES → CRM_REPOSITORIES → CRM_SERVICES → CRM_CONTROLLERS, with CRM_DTOs, CRM_COMMONs, and CRM_COMPONENTS serving as cross-cutting dependency modules. Modules at lower layers have no knowledge of modules above them, preventing circular dependencies and enabling isolated unit testing of each layer.

CRM_ENTITIES contains 38 JPA entities, listeners, and enums representing the complete domain model. CRM_REPOSITORIES contains 36 Spring Data JPA repositories handling all database query operations without business logic. CRM_DTOs contains 80+ request and response Data Transfer Objects that decouple the public API contract from the internal domain model. CRM_SERVICES contains 47 service classes implementing all business logic, orchestrating repositories, and enforcing transaction boundaries. CRM_CONTROLLERS contains 39 REST controllers receiving HTTP requests, validating input, and delegating to services. CRM_COMMONs provides shared utilities, constants, and helper classes used across all modules. CRM_COMPONENTS contains schedulers, event publishers, and WebSocket configuration for application-level infrastructure.

C. Lead Lifecycle State Machine

The student acquisition lifecycle is modeled as a formal state machine enforced at the service layer, preventing illegal state transitions at the application level before they reach the database. A lead enters the system via one of five intake channels — Facebook Lead Ads webhook, Instagram webhook, Google Forms webhook, University Website webhook, or manual walk-in creation — and progresses through UNASSIGNED, ASSIGNED, and counselor interaction states. Based on counselor engagement, the lead score advances through HOT, WARM, COLD, INTERESTED, and ENROLLED states. Leads flagged as invalid are moved to FAKE or DISCARDED status and excluded from operational reporting. Leads that reach the INTERESTED state become eligible for offline campus session booking, and upon successful mentor evaluation and payment confirmation, are formally converted to enrolled student records.

D. Database Design

The system manages 38 JPA entities organized into functional clusters. Every entity is monitored by a GenericEntityListener that automatically writes change records to the AuditLog table on every create, update, and delete event, providing a complete and tamper-evident operational history. Key entities include Leads (central record with contact, source, score, and status data), Counselors (staff profiles with availability, priority tier, and capacity), Mentors (academic evaluators), Offline_Sessions (campus visit slots), User (system accounts with hashed credentials), Login_Session (active session tracking for concurrent session enforcement), LeaveRequest (counselor absence records), Student (enrolled student records linked to the originating lead), and Campaigns (marketing initiative registry for source attribution).

E. Role-Based Access Control

The system implements five distinct roles enforced through Spring Security's @PreAuthorize and @Secured annotations at the method level. ROLE_ADMIN has superuser access to all modules and system configuration. ROLE_MANAGER has departmental oversight including assignment overrides, performance reports, and lead reassignment. ROLE_COUNSELOR has access to lead interaction, note logging, reminder management, and walk-in creation. ROLE_MENTOR has read-only access to session data and lead notes. ROLE_AFFILIATE and ROLE_CONSULTANT have access limited to lead submission through their respective portals and tracking of their own submitted leads only.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, May 2026
Impact Factor: 8.2

IV. IMPLEMENTATION

A. Round-Robin Assignment Engine

The Round-Robin Assignment Engine is invoked every time a new lead enters the UNASSIGNED state and must select the most appropriate available counselor. The engine operates through five ordered gate conditions. The Availability Guard queries all counselors whose status is neither ON_LEAVE nor BUSY. The Capacity Guard eliminates counselors whose active lead count has reached the globally configured maximum capacity. The Leave Lifecycle Guard keeps counselors with LEAVE_ENDED status invisible until they explicitly log back in after returning. Priority Tiering sorts eligible counselors by tier (1 to 5, where 1 is highest), ensuring Tier 1 counselors receive assignments before any lead reaches Tier 2 or beyond. Cyclic Distribution within each tier uses an AtomicInteger pointer that advances with each assignment, ensuring mathematically uniform distribution across counselors at the same tier. Walk-in leads created manually by a counselor bypass the engine entirely, with the creating counselor automatically assigned as owner.

The thread-safe implementation uses the getAndIncrement operation on AtomicInteger with a modulo operation to wrap the pointer cyclically:

int idx = roundRobinPointer.getAndIncrement() % available.size();

This ensures correctness under concurrent assignment requests without requiring explicit locking.

B. 14-Day Automated Demotion Scheduler

The LeadAutomationScheduler is a Spring @Scheduled component executing daily at 1:00 AM via cron expression. It processes leads in pages of 500 records using Spring Data's Pageable interface, iterating until no further pages remain. This paginated approach prevents JVM heap exhaustion when processing arbitrarily large lead databases. For each page, the scheduler evaluates LeadActivity records and identifies leads with no recorded interaction — calls logged, notes added, status updates, or any counselor activity — in 14 or more calendar days. On detection, the lead score is updated to COLD in the Leads table, a LeadStatusHistory record is written with the message "Auto-demoted to COLD due to 14 days of inactivity," and an email alert is dispatched to the responsible manager. Leads already scored COLD are skipped to avoid unnecessary database writes.

C. Security Implementation

Every incoming HTTP request passes through six sequential security layers before reaching business logic. Layer 1, the Rate Limiting Filter, uses Bucket4j to enforce a limit of five login attempts per minute per IP address, rejecting excess requests with HTTP 429 before any authentication logic executes. Layer 2, the JWT Authentication Filter, extends OncePerRequestFilter to extract and validate the Bearer token from the Authorization header using HS512, rejecting invalid or expired tokens with HTTP 401. Layer 3, the Token Blacklist Guard, rejects any token present in the Redis blacklist even if cryptographically valid — tokens are added to Redis on logout with a TTL matching remaining validity, making session termination instant. Layer 4, the Concurrent Session Guard, checks the user's active Login_Session count against their configured Session_Limit, rejecting requests that exceed the per-role threshold with HTTP 429 to prevent credential sharing. Layer 5, the RBAC Authorization Gate, uses @PreAuthorize and @Secured annotations to enforce role-based method-level access, returning HTTP 403 for unauthorized role access. Layer 6, Password Encryption, uses BCrypt for all stored credentials, with the plaintext password never persisted at any point.

D. REST API Development

The system exposes 170+ endpoints across 39 controllers, all following a consistent JSON response envelope format with standardized HTTP status codes. The API is fully documented via Swagger UI at /swagger-ui.html. Major endpoint groups include authentication and security (5 endpoints), lead management (26 endpoints), counselor workspace (10 endpoints), offline session management (18 endpoints), affiliate partner portal (12 endpoints), session

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 11, May 2026 Impact Factor: 8.2

concurrency control (9 endpoints), mentor management (10 endpoints), webhook integrations (5 endpoints), bulk Excel upload (3 endpoints), and payment gateway integration (4 endpoints).

E. External Integrations and Webhook Ingestion

The system receives leads from five external platforms via POST webhook endpoints. Facebook Lead Ads, Instagram Ads, Google Forms, and University Website integrations use Spring Security for endpoint protection. The Affiliate Partner integration uses API key authentication to support external third parties without requiring internal user accounts. Each endpoint maps the incoming payload to the internal Leads entity, applies the duplicate shield (matching by email and phone number), and triggers the Round-Robin engine automatically. Lead response time is reduced from hours under the previous manual process to milliseconds.

F. Consultant Management Portal

The Consultant Management Portal provides an isolated workspace for external educational consultants. All consultant-submitted leads are stored in completely separate database tables (consultant_leads and consultant_lead_phones) independent from the main lead pool. Every consultant lead requires a valid 12-digit Indian Aadhaar number, enforced via regex validation at the service layer, making duplicate submission structurally impossible. Consultants receive automated onboarding emails upon registration containing login credentials and a unique API key for webhook integration from their own landing pages.

G. Payment Gateway Integration

The payment module implements a dual-gateway adapter architecture supporting both Razorpay and Easebuzz behind a common PaymentGatewayService interface, providing payment redundancy and provider flexibility. Payment orders are generated server-side with amounts locked before the student's browser receives the checkout widget, preventing client-side manipulation. Webhook reconciliation validates incoming gateway callbacks using HMAC-SHA256 signature verification before any state change is applied, making payment fraud structurally impossible. All payment interactions are written to the payment_transactions ledger regardless of outcome, capturing gateway transaction ID, cryptographic signature, full webhook payload, and timestamps for financial audit and dispute resolution.

H. DevOps and Deployment

The system uses a multi-stage Docker build: a Maven JDK 21 image compiles and packages the application, and an Eclipse Temurin JRE 21 slim image runs only the resulting app.jar, significantly reducing the production image size. Docker Compose orchestrates PostgreSQL 16 Alpine and the application container. Health checks use pg_isready for the database and /actuator/health for the application. A named Docker volume ensures PostgreSQL data persists across container restarts. The production profile connects to Neon Cloud PostgreSQL over SSL. A GitHub Actions CI/CD pipeline validates and deploys on every push to the main branch.

V. RESULTS AND DISCUSSION

A. Findings

All 15 defined project objectives were successfully implemented and validated. Multi-channel lead intake via all five webhook sources is fully operational. The Round-Robin Assignment Engine correctly handles all tested scenarios including priority tier ordering, capacity enforcement, leave lifecycle transitions, concurrent assignment requests, and walk-in auto-ownership — no lead was assigned to an ineligible counselor across all test cases. The six-layer security pipeline was validated through dedicated security test scenarios; replay attacks using logged-out tokens are correctly rejected, and credential sharing is blocked at the session count threshold. The 14-day demotion scheduler processes leads in pages of 500 without JVM memory issues. All 170+ endpoints return consistent HTTP status codes and standardized JSON response envelopes. Real-time WebSocket dashboard updates deliver live lead and assignment

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, May 2026
Impact Factor: 8.2

statistics without polling. The complete enrollment flow from INTERESTED status through offline session booking, mentor evaluation, and payment confirmation was tested end-to-end without data integrity issues.

B. Scalability Analysis

The seven-module architecture enables future functional expansion without modifying existing layers. The stateless JWT authentication model allows horizontal scaling behind a load balancer, as session state is managed in shared Redis rather than in-process memory. The paginated scheduler is designed to handle arbitrarily large lead databases, having been validated against batch sizes representing the full operational lead pool.

C. Security Analysis

The layered security pipeline implements genuine defense-in-depth. No single layer bypass compromises the system: bypassing rate limiting still requires a valid JWT; a valid JWT is useless if present in the Redis blacklist; a valid non-blacklisted JWT still cannot access endpoints restricted to a higher role. This architecture significantly raises the cost and complexity of any unauthorized access attempt.

D. Data Integrity Analysis

Flyway versioned migrations guarantee schema identity across all environments. The GenericEntityListener provides tamper-evident traceability across all 38 entity types. The duplicate shield prevents cross-channel lead duplication. The CapacitySweepService at application startup assigns leads left UNASSIGNED from capacity overflow events, creating a self-healing operational loop with the demotion scheduler.

E. Project Statistics

The system comprises 284 Java source files totaling approximately 13,700 lines of code, organized across 7 Maven modules containing 38 JPA entities, 36 repositories, 47 service classes, 39 controllers, and 80+ DTO classes. The system exposes 170+ REST endpoints, supports 5 external webhook integrations, enforces 5 user roles, and runs 2 scheduled jobs. Lead score states are HOT, WARM, COLD, INTERESTED, and ENROLLED. Lead status states are FRESH, CONTACTED, ASSIGNED, UNASSIGNED, FAKE, and DISCARDED.

VI. LIMITATIONS AND FUTURE WORK

The current system does not include a machine learning component for lead conversion probability prediction. The lead scoring system (HOT/WARM/COLD/INTERESTED) relies entirely on counselor judgment, which introduces subjectivity into prioritization decisions. A future extension integrating an ML model — trained on historical interaction frequency, response times, campaign source, and demographic features — implemented as a Python-based microservice communicating via REST would enable data-driven counselor prioritization.

The real-time WebSocket dashboard provides live operational metrics, but management currently lacks access to periodic strategic reports including conversion funnels, counselor performance rankings, and campaign ROI analysis. A dedicated reporting module generating exportable PDF and Excel reports on configurable schedules is planned.

The current architecture supports a single campus. Extending the multi-tenant model to support multiple campuses — each with isolated lead pools, counselor teams, and session schedules under a shared admin console — would require adding a campus dimension to the entity model and partitioning the assignment engine by campus scope.

WhatsApp Business API and SMS gateway integration for automated follow-up messages and session reminders would reduce manual outreach volume. Migration from Docker Compose to Kubernetes with Horizontal Pod Autoscaling would provide production-grade scalability for multi-campus deployment scenarios.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, May 2026
Impact Factor: 8.2

VII. CONCLUSION

This paper presented the Raffles CRM System, an enterprise-grade backend platform for automating the complete student acquisition lifecycle at Raffles University, Neemrana. The system applies modern Java ecosystem technologies — Java 21, Spring Boot 4.0.2, Spring Data JPA, Spring Security, PostgreSQL, Redis, AWS S3, and Docker — to eliminate five fundamental operational failures of traditional educational admissions management: lead leakage, inequitable workload distribution, inactive lead neglect, campus session coordination conflicts, and credential sharing security vulnerabilities.

The multi-tier Round-Robin Assignment Engine provides mathematically fair counselor workload distribution without management oversight. The 14-day automated demotion scheduler prevents any lead from remaining silently unattended. The six-layer security pipeline closes the gaps created by credential sharing and token replay attacks. The dual-gateway payment integration makes enrollment payment fraud structurally impossible.

At version 2.2.0, production-deployed and fully operational, the system demonstrates that university-specific operational problems traditionally addressed through spreadsheets and informal communication can be solved effectively through systematic backend software engineering. The modular Maven architecture and the layered service design provide a scalable foundation for continued expansion of the platform across additional functional domains and campus locations.

ACKNOWLEDGMENT

I would like to sincerely thank Rajendra Singh, Dean, Department of Computer Science and Engineering, Raffles University, for his valuable guidance, continuous support, and encouragement throughout this project.

I am also grateful to the Department of Computer Science and Engineering, Raffles University, for providing the academic support and necessary environment to complete this research work.

REFERENCES

[1] Spring Framework. (2024). Spring Boot 4.x Reference Documentation. https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

[2] Spring Framework. (2024). Spring Data JPA Documentation. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

[3] Baeldung. (2024). Spring Security and JWT Authentication. https://www.baeldung.com/spring-security-jwt

[4] JWT.io. (2024). JSON Web Token Introduction. https://jwt.io/introduction/

[5] Bucket4j Documentation. (2024). Rate Limiting with Spring Boot. https://bucket4j.com/

[6] Flyway. (2024). Database Migrations Made Easy. https://flywaydb.org/documentation/

[7] Amazon Web Services. (2024). AWS SDK for Java v2 — S3 Documentation. https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/

[8] Oracle. (2024). Java 21 Documentation. https://docs.oracle.com/en/java/javase/21/

[9] Docker Inc. (2024). Docker Compose Documentation. https://docs.docker.com/compose/

[10] Apache Software Foundation. (2024). Apache POI OOXML — Excel Processing. https://poi.apache.org/

[11] Redis. (2024). Redis Documentation. https://redis.io/docs/

[12] Neon. (2024). Neon Serverless PostgreSQL Documentation. https://neon.tech/docs/introduction

[13] GitHub. (2024). GitHub Actions Documentation. https://docs.github.com/en/actions

[14] Spring Framework. (2024). Spring WebSocket Documentation. https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#websocket

[15] Leff, A., & Rayfield, J. T. (2001). Web-application development using the Model/View/Controller design pattern. Proceedings of the Fifth IEEE International Enterprise Distributed Object Computing Conference, 118–127.