ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

YOJAK: A Full-Stack Integrated Platform for Program, Beneficiary and Impact Management

Shraddha Sanjay Salunke
Department of Computer Science & Engineering, TPCT's College of Engineering Dharashiv (Osmanabad), Maharashtra, India
Dr. Babasaheb Ambedkar Technological University (DBATU)
Guide: Dr. S. N. Holambe

ABSTRACT

This paper presents YOJAK, a full-stack web platform designed to centralize program operations, beneficiary tracking, academic monitoring, attendance, financial management and reporting. The implementation uses React 18 and Vite for the presentation layer, Java 17 and Spring Boot 2.7.8 for REST services, and MariaDB for relational persistence. The backend integrates Spring Data JPA, Spring Security, JWT, file handling, Excel/PDF processing and dashboard services. The supplied database contains approximately 98 tables and the backend includes a large set of controllers, services and domain entities. The paper describes the architecture, module integration and domain-specific automation documented in the supplied source and knowledge-transfer material. The system demonstrates how a unified platform can reduce fragmented data management and support longitudinal program monitoring. Quantitative performance claims are intentionally excluded until controlled experiments are conducted.

Keywords: Program management; beneficiary management; full-stack application; React; Spring Boot; MariaDB; REST API; dashboard analytics; social impact management

1. INTRODUCTION

Organizations managing education and social-development programs must track beneficiaries, employees, projects, attendance, academic progress, activities and financial information. When these records are maintained through fragmented spreadsheets and disconnected workflows, timely reporting becomes difficult and duplicate data entry can reduce consistency. YOJAK addresses this problem through a unified web platform that connects operational, academic and program information through a common application and relational database model.

The supplied implementation uses a React/Vite frontend, a Java Spring Boot backend and MariaDB persistence. The backend exposes REST services and uses Spring Data JPA, Spring Security, JWT-oriented authentication, file handling, Excel/PDF processing and dashboard services. The supplied database contains approximately 98 tables covering beneficiaries, projects, employees, attendance, examinations, training, activities, budgets, expenditure, procurement, reports and master data.

The research contribution is implementation-oriented: it demonstrates how a domain-specific full-stack architecture can connect multiple program workflows while preserving relational integrity and providing reusable services for future reporting and analytics.

2. PROBLEM STATEMENT AND OBJECTIVES

Program-oriented organizations frequently maintain beneficiary, project, attendance, academic, financial and activity information in separate files or disconnected systems. This fragmentation can create duplicate entry, inconsistent records, weak traceability and additional effort during reporting. A centralized system is therefore required to maintain related entities, enforce business rules and provide a common interface for operational monitoring.

The objectives of YOJAK are to centralize program and beneficiary information; manage employees, users, roles and permissions; track attendance and academic progress; support project, training, parent-meeting and activity workflows; manage budgets, expenditure and procurement information; provide reports and dashboards;

support Excel/PDF-oriented workflows; and provide a maintainable full-stack architecture suitable for future analytical capabilities.

3. RELATED TECHNICAL BACKGROUND

YOJAK is based on a layered web-application approach in which the presentation, application and persistence concerns are separated. React provides component-based user interfaces, while Spring Boot provides REST services and the service layer. Spring Data JPA and Hibernate support repository-oriented persistence, and MariaDB provides relational storage, keys, indexes and transactional mechanisms.

The selected technologies are not presented as novel individually. The engineering contribution lies in their domain-oriented integration for program and impact management. The platform connects operational records that would otherwise be distributed across separate files and workflows. This integration provides the foundation for longitudinal monitoring of beneficiaries, projects, attendance, academic outcomes and engagement activities.

4. SYSTEM ARCHITECTURE

YOJAK follows a layered full-stack architecture. The presentation layer contains React components, forms, tables, filters and dashboards. The application layer exposes REST endpoints through Spring controllers and applies business rules through service classes. The persistence layer maps JPA entities to MariaDB tables through repository interfaces. Security and file/reporting integrations operate as cross-cutting capabilities.

The logical request path is:

React user interface → REST controller → service/business rule → repository/JPA → MariaDB → response → React user interface.

This separation supports modularity, maintainability and testability. It also permits individual functional domains to evolve without requiring the complete application to be redesigned.

5. FUNCTIONAL MODULES

The implementation covers a broad set of operational domains. Authentication and user-management functions provide access control. Project and employee modules maintain organizational records. Beneficiary functions support registration and project mapping. Attendance functions provide daily and summary information. Academic functions cover marks, board examinations and higher education. Additional domains include training, teacher training, parent meetings, counselling, activities, camps, budgets, expenditure, procurement, reports, surveys, success stories and dashboard analytics.

The value of these modules is not only their individual functionality but also their ability to share common entities and relationships. A beneficiary can therefore be associated with projects, attendance, academic records, engagement activities and higher education without manually merging separate spreadsheets.

6. DOMAIN-SPECIFIC AUTOMATION

The supplied project and knowledge-transfer material documents several business rules that go beyond simple data storage.

First, parent-meeting numbering is maintained on a project-wise basis, preventing meeting-number conflicts between projects. Second, attendance supports bulk persistence, reducing repetitive data-entry operations. Third, academic workflows include correct mapping of academic YearMaster identifiers and pivoted retrieval of examination details. Board-examination percentages can be calculated from the stored results. A qualifying 12th-standard result can trigger creation of a corresponding higher-education record.

These rules demonstrate a workflow-oriented approach in which the service layer interprets domain conditions rather than merely forwarding database operations.

7. DATABASE DESIGN AND DATA INTEGRITY

The supplied SQL dump contains approximately 98 CREATE TABLE definitions. The relational model connects projects, beneficiaries, employees, activities, attendance, academic records, budgets and master data. Primary keys identify records, while foreign-key relationships support referential integrity. Mapping tables support relationships between related domains.

Frequently searched and joined fields, such as beneficiary identifiers, project identifiers, dates and workflow keys, are appropriate candidates for indexing. Transaction boundaries are important for multi-record operations such as bulk attendance persistence and related workflow updates. Pagination and filtering are used for large collections to avoid unnecessarily loading complete datasets into the user interface.

8. SECURITY AND RELIABILITY

The supplied implementation includes Spring Security and JWT-oriented authentication capabilities. Role and permission management should follow least-privilege principles so that users receive only the access required for their responsibilities. Validation should reject malformed identifiers, invalid dates, impossible academic values and inconsistent relationships.

Transaction boundaries should protect multi-record updates, while centralized exception handling should avoid exposing internal implementation details to clients. Production deployment should additionally verify TLS, secret management, audit logging and backup/restore procedures. Because the system handles beneficiary and program information, institutional data-governance requirements should be applied before production use.

9. IMPLEMENTATION METHODOLOGY

The development process follows requirement analysis, domain decomposition, entity modelling, repository creation, service-layer business-rule implementation, REST controller development, validation and React component development. The source inventory identifies approximately 193 frontend JSX/JS component files and 515 backend Java source files, including approximately 87 entity files, 72 service files and 40 controller files. These counts are source-inventory observations from the supplied project package.

The development stack includes Java 17, Spring Boot 2.7.8, Spring Data JPA, Hibernate, Spring Security, MariaDB, React 18.3.1, Vite 5.3.4, Axios, React Router, Ant Design, Material Tailwind, Formik, Yup, XLSX/PDF-related libraries and visualization libraries. Maven, npm, Git, development IDEs, browser developer tools and MariaDB administration tools support development and maintenance.

10. EVALUATION METHODOLOGY

The supplied project material does not contain controlled benchmark measurements. Accordingly, this paper does not claim numerical performance improvement or statistical superiority over another system.

A reproducible evaluation should use controlled datasets and representative API workloads. Functional correctness can be tested using independently calculated reference values for attendance, academic percentages and workflow identifiers. Performance evaluation should document the hardware and software environment, dataset size, request mix, concurrency level, warm-up policy, repetitions, median latency, 95th-percentile latency, throughput and error rate. Scalability can be assessed by progressively increasing beneficiary, attendance and examination record counts.

Representative functional tests include beneficiary creation and retrieval, daily and bulk attendance persistence, monthly attendance calculation, paginated examination retrieval, baseline/midline/endline indicator calculation, board-examination progression, parent-meeting numbering, filtered meeting retrieval, unauthorized-access rejection, invalid-input validation and relational-integrity checks.

11. DISCUSSION

The principal strength of YOJAK is cross-domain integration. Beneficiary information, attendance, academic progress, parent engagement, projects and reporting can be connected through a common relational model rather than isolated spreadsheets. Modular services allow individual domains to evolve while sharing infrastructure.

The main engineering challenge is complexity as additional modules and business rules are introduced. Duplicated validation, inconsistent rules and inefficient database queries can reduce maintainability.

Automated API-contract testing, database indexing, centralized logging and benchmark-driven optimization are therefore important next steps. The current research position is implementation-oriented, and controlled experiments are required before quantitative claims are made.

12. LIMITATIONS

The supplied materials do not provide a controlled benchmark dataset, measured latency results, a comparative baseline, statistical analysis or a user study. Therefore, this manuscript does not report fabricated numerical performance results.

The platform is also domain-specific. Other organizations may require different roles, workflows, reporting definitions and governance policies. The architectural approach should therefore be treated as a reusable engineering foundation rather than a universal workflow specification.

13. FUTURE SCOPE

Future work can include predictive analytics for beneficiary outcomes, anomaly detection in attendance and expenditure, automated narrative generation for management reports, mobile or offline data capture, cloud-native deployment, data-warehouse integration and advanced business intelligence. Longitudinal analytics combining attendance, academic progress, training participation and intervention history can provide a stronger basis for program decision support.

AI-assisted decision support is a potential extension, but it should only be introduced after suitable data collection, validation, evaluation metrics, privacy controls and governance procedures are established.

14. CONCLUSION

YOJAK demonstrates how modern full-stack engineering can create a centralized platform for program operations, beneficiary management and impact monitoring. React, Spring Boot, REST services, Spring Data JPA and MariaDB are combined around domain-specific workflows covering academic, attendance, project, financial and engagement functions.

The architecture emphasizes integration, relational integrity, validation, transactions, pagination and reusable services. The platform provides a foundation for longitudinal monitoring and future analytics. The next research stage should add controlled quantitative evaluation so that functional correctness, response-time behavior, scalability and reliability can be reported using reproducible evidence.

REFERENCES

[1] Spring Boot Documentation, Spring Framework / VMware.

[2] Spring Security Documentation, Spring Framework / VMware.

[3] React Documentation, Meta Open Source.

[4] MariaDB Documentation.

[5] Apache POI Documentation, Apache Software Foundation.

[6] JSON Web Token (JWT/JWS) standards and documentation.

[7] YOJAK project source code, database dump and internal knowledge-transfer documents supplied for the project.

[8] YOJAK M.Tech CSE Project research-paper draft supplied for this manuscript.