

ENGINEER



international scientific journal

ISSUE 1, 2026 Vol. 4

E-ISSN

3030-3893

ISSN

3060-5172



SLIB.UZ
Scientific library of Uzbekistan



A bridge between science and innovation



**TOSHKENT DAVLAT
TRANSPORT UNIVERSITETI**

Tashkent state
transport university



ENGINEER

A bridge between science and innovation

E-ISSN: 3030-3893

ISSN: 3060-5172

VOLUME 4, ISSUE 1

MARCH, 2026



engineer.tstu.uz

TASHKENT STATE TRANSPORT UNIVERSITY

ENGINEER

INTERNATIONAL SCIENTIFIC JOURNAL
VOLUME 4, ISSUE 1 MARCH, 2026

EDITOR-IN-CHIEF

SAID S. SHAUMAROV

Professor, Doctor of Sciences in Technics, Tashkent State Transport University

Deputy Chief Editor

Miraziz M. Talipov

Doctor of Philosophy in Technical Sciences, Tashkent State Transport University

Founder of the international scientific journal “Engineer” – Tashkent State Transport University, 100167, Republic of Uzbekistan, Tashkent, Temiryo‘lchilar str., 1, office: 465, e-mail: publication@tstu.uz.

The “Engineer” publishes the most significant results of scientific and applied research carried out in universities of transport profile, as well as other higher educational institutions, research institutes, and centers of the Republic of Uzbekistan and foreign countries.

The journal is published 4 times a year and contains publications in the following main areas:

- Engineering;
- General Engineering;
- Aerospace Engineering;
- Automotive Engineering;
- Civil and Structural Engineering;
- Computational Mechanics;
- Control and Systems Engineering;
- Electrical and Electronic Engineering;
- Industrial and Manufacturing Engineering;
- Mechanical Engineering;
- Mechanics of Materials;
- Safety, Risk, Reliability and Quality;
- Media Technology;
- Building and Construction;
- Architecture.

Tashkent State Transport University had the opportunity to publish the international scientific journal “Engineer” based on the **Certificate No. 1183** of the Information and Mass Communications Agency under the Administration of the President of the Republic of Uzbekistan. **E-ISSN: 3030-3893, ISSN: 3060-5172.** Articles in the journal are published in English language.

3	
engineer.tstu.uz	A bridge between science and innovation

AI-based university KPI system: React.js, ASP.NET Core, and PostgreSQL architecture

A.B. Tukhtakhodjaev¹^a, O.A. Turdiev¹^b

¹Tashkent state transport university, Tashkent, Uzbekistan

Abstract:

This article presents a layered architecture for a system designed to collect, analyze, and generate recommendations based on university employees' Key Performance Indicators (KPIs). The technology stack includes React.js, ASP.NET Core 8, PostgreSQL 16, and a Python-based AI service. The ETL process operates on a star-schema data model under the condition $Q_period \geq 0.90$. Anomaly detection is performed using Isolation Forest combined with forecasting residuals ($AUC = 0.94$), while recommendations are constrained by a weighted gap-based rule. The prototype achieved a p95 response time of 0.42 s and a System Usability Scale (SUS) score of 78, while reducing report preparation time from 36 hours to 9 hours. A human-in-the-loop approach is maintained: the AI system provides recommendations but does not issue commands.

Keywords:

system architecture, ETL, Isolation Forest, React, ASP.NET Core, PostgreSQL, explainable AI

1. Introduction

The proposed AI-based university KPI system is designed to centrally collect, process, and analyze data on employee performance and support management decisions. Its core functional requirements, F1–F10, cover ETL processes, calculation of the integrated indicator K^* , determination of weights using the Analytic Hierarchy Process (AHP) and monitoring of the consistency ratio CR, forecasting KPI values for the next period, anomaly detection, and generation of individual recommendations. Other essential functions include managing user permissions through role-based access control (RBAC), recording all significant actions in an audit log, and tracking the history of data changes.

The nonfunctional requirements focus on system reliability, responsiveness, and security. The response-time requirement for the API and user interface is $P(t_{resp} \leq 2\text{ s}) \geq 0.95$: at least 95% of requests must be processed within two seconds. Overall system availability must be at least 0.995, imposing a high requirement for uninterrupted service throughout the year. Personal data are handled according to the data minimization principle: only information necessary for KPI calculation and analysis is stored and processed.

The system is developed using a design science approach and is treated as a digital artifact that addresses a practical problem. Its architecture uses open-source technologies with relatively low licensing costs, selected to suit the budgets and existing technical infrastructure of national higher education institutions. The proposed solution is not intended to replace the university's existing ERP, HEMIS, LMS, or other enterprise information systems in full. Instead, it operates as an additional analytical platform that integrates data from existing systems and applies KPI analysis, forecasting, and intelligent recommendation mechanisms to those data.

2. Research methodology

ARCHITECTURE

The system architecture comprises five main layers. The first manages user roles and access rights. The second is a React.js visual interface that displays dashboards, KPI trends, rankings, and analytical charts. The third is an ASP.NET Core 8 backend that provides a REST API, JWT authentication, business logic, and background jobs through Hangfire. The fourth is an AI service based on FastAPI and PyTorch, which executes forecasting, anomaly detection, and recommendation algorithms. The fifth is a PostgreSQL 16 database, where an analytical data model is built using fact and dimension tables.

The central architectural principle is a modular monolith, with the AI module deployed in a separate container. For a university with 500–1,500 employees, a full microservice architecture could unjustifiably increase operational complexity and infrastructure costs; keeping the core business modules in a single application was therefore considered appropriate. Separating the computationally demanding AI component into an independent service allows it to be scaled and updated independently. Even if the AI service becomes temporarily unavailable, core KPI calculations, data display, and processing of actual K^* values continue.

Integration with external information systems uses the adapter pattern. HEMIS, LMS, Scopus, and other data sources connect through separate adapters implementing the common `ISourceAdapter` interface, allowing new sources to be added without changing existing business logic. The data warehouse uses Slowly Changing Dimension Type 2 (SCD2), preserving historical data when an employee's position, department, or other attributes change. Consequently, K^* values for earlier periods are not overwritten with new data, enabling reliable retrospective analysis over time.

^a <https://orcid.org/0009-0001-0640-559X>

^b <https://orcid.org/0000-0002-1651-5493>



Table 1

Technology stack and service requirements

Layer	Technology	SLA
UI	React 18, Recharts	2 s
API	ASP.NET Core 8	p95 < 0.5 s
AI	FastAPI, IF+LSTM	Overnight window
DB	PostgreSQL 16	$Q \geq 0.90$

ALGORITHMS

The algorithmic component of the AI-based KPI system combines data quality control, anomaly detection, and generation of suitable employee recommendations into a single computational pipeline. The completeness and consistency of incoming source data are checked first; only data that meet the specified quality criterion proceed to subsequent analytical stages. This prevents incomplete, duplicate, or conflicting records from directly affecting KPI results.

The algorithm sequence combines deterministic rules with AI models. Data quality and business rules are controlled through explicit formal conditions, while Isolation Forest and forecasting residuals identify unusual or unexpected behavior. This enables the system to detect both statistical deviations and significant departures from an employee's usual KPI trajectory.

The recommendation mechanism compares identified KPI gaps with their importance weights. The system prioritizes the areas with the greatest impact and presents users with a limited number of understandable recommendations. AI operates as an analytical decision support module rather than a final decision maker; conflicting cases and significant alerts are referred for human review.

$$Q = 1 - (|missing| + |conflict|) / |expected|, \text{ ETL permission: } Q \geq 0.90 \quad (1)$$

$$flag \Leftrightarrow (s_{IF} < s^*) \vee (|y - \hat{y}| > \kappa \cdot IQR(e)), \kappa = 2.2 \quad (2)$$

$$j^* = \underset{j}{\operatorname{argmax}} w_j (\bar{x}_j^{\{top\}} - \bar{x}_{ij}), |recommendations| \leq 3 \quad (3)$$

Conflicts are not resolved automatically; they enter a human review queue, reflecting a legal safeguard. Isolation Forest contamination is set to 0.03. Across 38 manually reviewed incidents, AUC was 0.79 for the z-score method, 0.88 for Isolation Forest, and 0.94 for Isolation Forest combined with residuals. SignalR delivers red alerts immediately and yellow alerts in a digest to reduce alert fatigue. Ethical visualization avoids traffic-light indicators on employee profiles; red is reserved for system risks.

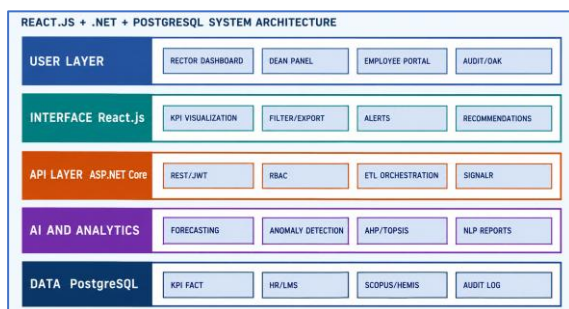


Fig. 1. Layered architecture: UI → API → AI/analytics → star schema; data flow is governed exclusively by DTO/JSON contracts

Security ETL discipline and ethical visualization

JWTs have a 15-minute lifetime. Salary and passport data are not stored, and the AI service receives only staff_id. Under RBAC, administrators cannot view employee K* values merely out of curiosity; access to detailed information by the rector requires break-glass access and logging. Idempotent upserts make overnight jobs safe to rerun; fact records remain unchanged if the raw-data hash does not change. Forecasting stops when $Q < 0.90$: an apparently reliable ranking is not displayed on the basis of incomplete data. SCD2 prevents the previous semester's data from being overwritten through normalization against a new faculty. SignalR delivers red alerts immediately and yellow alerts in a digest; otherwise, deans may experience alert fatigue.

Traffic-light coloring on employee profiles can stigmatize individuals. Personal scores therefore use a blue scale, with red reserved for system risks. Recommendations are advisory rather than mandatory: at most three are presented, and employees may accept or reject them without a penalty for rejection. For 500–1,500 employees, a modular monolith is more practical than microservices: AI runs separately, while the remaining components share one process. Dashboard p95 is 0.42 s, ETL takes 6.4 minutes, and LSTM processing on the CPU takes 11 minutes. A GPU is not required, meeting the resource constraints of a small higher education institution.

3. Conclusion

For 412 employees over 24 months, Q_{period} was 0.93, and the conflict rate fell from 11.4% to 2.1%. AUC was 0.94, compared with 0.79 for the z-score method. $U = 0.4 \cdot SUS/100 + 0.3 \cdot (1 - t/t_0) + 0.3 \cdot P_{ok} = 0.79 \geq 0.75$, with $SUS = 78$. Report preparation time fell from 36 hours to 9 hours, a 75% reduction. The conclusion is that the AI pipeline should not be activated on poor-quality data: first establish keys and Q , then the dashboard, and finally the model. Humans remain in the loop: AI does not issue orders for bonuses or dismissal. The limitation is a panel from a single university; transferability is assessed through Q_{period} and CR rather than MAE.

Anomalies are interpreted as four types: data entry errors, positive jumps, genuine declines, and system loopholes. Isolation Forest does not distinguish these types; humans assign the labels, which are retained for subsequent lightweight supervision. The policy dictionary is not hard-coded: a criterion_id and profile table allows the research department to edit its own entries. Export templates are independent of the fact layer, so changes to OAK/HEMIS forms do not require changes to the core. In user acceptance testing (UAT), the faculty K* task took 18 s (96%), and the anomaly explanation task took 31 s (88%). WCAG guidance and larger cards raised SUS from 71 to 78 in the age group of 50 years and older. For observability, /healthz and Hangfire status are shown in the UI; otherwise, users would be the first to report an overnight ETL failure the following morning.

In this article, the Hevner cycle reaches the artifact evaluation stage: the QoS requirement (3.1) is met, and AUC and SUS fall within the target ranges. The design science contribution is the working assembly of an explainable KPI pipeline under national constraints, rather than a new React component. Isolation Forest contamination is set to 0.03; at 0.08, false positives cause alert fatigue in the dean's office. Forecasts are hidden for new employees, and only actual K*



values are shown, avoiding the small-N trap. The next step is the before-and-after evaluation and the five-stage roadmap in Chapter IV. This article forms the Q1/Q2 core of Chapter III: the architecture is reproducible, algorithmic complexity is specified, and ethical rules are encoded.

References

- [1] Liu F.T., Ting K.M., Zhou Z.-H. Isolation Forest. ICDM, 2008. <https://doi.org/10.1109/ICDM.2008.17>
- [2] Hevner A. et al. MIS Quarterly, 2004.
- [3] Kimball R., Ross M. The Data Warehouse Toolkit. 2013.
- [4] PostgreSQL 16 docs. <https://www.postgresql.org/docs/16/>
- [5] <https://learn.microsoft.com/aspnet/core>
- [6] <https://react.dev/>
- [7] Brooke J. SUS. 1996. 8. Fielding R. REST. 2000. <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

Information about the author

**Adham
Tukhtakhodjaev**

Tashkent State Transport
University, Senior Lecturer in
the Department of Information
Systems and Technologies in
Transport

E-mail:

tuxtaxodjaev.a@tstu.uz

Tel.: +998903345333

<https://orcid.org/0009-0001-0640-559X>

**Odilzhan
Turdiev**

Tashkent State Transport
University, Associate
Professor in the Department of
Information Systems and
Technologies in Transport,
Candidate of Technical
Sciences (PhD)

E-mail:

odiljan.turdiev@mail.ru

Tel.: +998974607179

<https://orcid.org/0000-0002-1651-5493>



E. Tokhirov, R. Aliev, M. Aliev

Methods and solutions for reducing the amount of dust in order to ensure the sustainability of cities 84

E. Tokhirov, R. Aliev

Details and solutions to safety issues at railway LC 89

Z. Adilova, S. Asenova, M. Yokubjonov, A. Sadikova

Selection of a method for market segmentation in the field of transport and logistics services 96

S. Boltaev, Z. Toshboev, I. Yoldashev, B. Ganijonov, Sh. Kholboev

Enhancing the reliability of railway track circuit power supply systems using a microcontroller-based self-checking dual-channel architecture 101

M. Karimova, R. Bozorov, E. Asatov

Analysis of the freight transportation technology efficiency on the “Bukhara – Miskin” and “Angren – Pap” railway lines 106

I. Maturazov, D. Sarsenbaev

Smart maintenance scheduling via predictive logistics for post-flight engine maintenance in Central Asia 110

Sh. Tillaeva

Comparative analysis of neural network architectures for adaptive control of nonlinear thermal systems 113

E. Ruklinskaya, M. Ubaydullaev, S. Khudoyarov, N. Khavasova

Development of a non-autoclave technology for producing liquid glass from microsilica dust of ferroalloy production 118

R. Koldobskiy, M. Ubaydullaev, U. Kosimov, E. Ruklinskaya

Design and engineering of a laboratory-industrial setup for low-temperature liquid carbonitriding with an integrated ammonia neutralization system 122

J. Turgaev, V. Soy, N. Mukhammadiev

Non-autoclaved aerated concrete with a modified carbonate-sulfate additive based on soda ash production waste 126

S. Khudayberganov, Sh. Kayumov, J. Tukhtaev

A model for determining employee compliance with professional requirements in human-machine systems of railway transport 130

A. Tukhtakhodjaev, O. Turdiev

AI-based university KPI system: React.js, ASP.NET Core, and PostgreSQL architecture 134

A. Tukhtakhodjaev

Architecture of an AI-based university KPI system: React.js, ASP.NET Core, and PostgreSQL 137