WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Supplier Risk Forecaster – A Procurement Risk Intelligence Platform

G. Hema Prabha[#1], Srihari K[#2]

[#1] Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

[#2] Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

ABSTRACT :

Procurement teams usually keep supplier details, purchase orders and delivery records in separate spreadsheets, so supplier risk is judged from memory or from the latest incident and a problem is noticed only after a delivery has failed. This paper presents Supplier Risk Forecaster, a web-based procurement risk intelligence platform that brings supplier management, purchase order tracking, risk scoring, forecasting, alerts and reporting into one application. A composite risk score from 0 to 100 is calculated for each supplier from delivery delay, quality compliance and lead-time stability. A forecasting module projects the score over the next 30 days and reports the trend, a confidence rating and the main risk drivers, and an alerting service notifies users when a supplier crosses a warning level. The system is built on the MERN stack (MongoDB, Express, React and Node.js) with JSON Web Token sessions, bcrypt password hashing and role-based access for Administrators and Procurement Managers. An executive dashboard and six reports exportable as CSV and PDF support decisions. We describe the architecture and modules, state the limitations of the prototype openly, and propose an evaluation protocol; no performance claims are made.

Keywords: Supplier Risk, Procurement, Risk Scoring, Forecasting, Alerts, Dashboard, MERN Stack, MongoDB, React, Node.js.

I. INTRODUCTION

Almost every organization depends on external suppliers, and disruptions such as late deliveries, quality failures and capacity shortfalls can halt production and damage customer commitments [1]. Reviews of supply chain risk research stress that visibility of supplier performance is the first step toward managing such risk [2]. In practice, however, supplier details, orders and performance records are scattered across spreadsheets and e-mail threads, and risk is judged by memory or by the most recent incident.

A second weakness is that risk assessment is mostly reactive. Criteria such as quality, delivery and performance history have been used to evaluate suppliers for decades [3], and weighted multi-criteria methods remain common [4], yet small and mid-sized procurement teams rarely have a tool that turns these criteria into a score, projects it forward and warns the buyer before a delivery fails. Digital tools and analytics can support such early warning [5].

This paper presents **Supplier Risk Forecaster**, a web-based procurement risk intelligence platform that combines supplier management, purchase order tracking, multi-factor risk scoring, 30-day forecasting, alerts and reporting. A composite score from 0 to 100 is calculated from delivery delay, quality compliance and lead-time stability, and a forecast reports the trend, a confidence rating and the key risk drivers. The main contributions are:

1) A supplier and purchase order module with search, filtering by risk level and rating, and an itemized order builder that computes tax and totals automatically.
2) A transparent composite risk score (0–100) built from delivery delay, quality compliance and lead-time stability and displayed together with its factor breakdown.
3) A 30-day risk forecast that returns the projected score, the trend direction, a confidence rating and the key risk drivers for each supplier.
4) An alerting and reporting layer: threshold-based alerts with an unread counter and resolution workflow, an executive dashboard, and six reports exportable as CSV and PDF.
5) A modular React, Express and MongoDB architecture with token-based sessions, hashed passwords and role-based access for Administrators and Procurement Managers, together with a transparent statement of the prototype's limitations and an evaluation protocol.

The remainder of the paper is organized as follows. Section II reviews related work. Section III describes the methodology and architecture. Section IV details the implementation of each module. Section V summarizes the advantages of the approach. Section VI reports the prototype status, limitations, and evaluation protocol. Sections VII and VIII present the conclusion and future work.

II. LITERATURE SURVEY

Supply Chain Risk Management. Chopra and Sodhi argue that supply chain disruptions differ in likelihood and impact and must be managed with different responses rather than one policy [1]. Tang's review of quantitative models for supply chain risk shows that visibility of supplier performance underlies every mitigation strategy [2]. Supplier Risk Forecaster follows this view by keeping delivery and quality information per supplier and presenting it in a form that supports a specific action, such as expediting an order or adding a second source.

Supplier Evaluation and Selection. Dickson's classic study listed the criteria buyers use when choosing suppliers, with quality, delivery and performance

history among the most important [3]. A later review of multi-criteria decision-making approaches found that weighted combinations of several criteria are still the most common practical method [4]. The present system adopts the same idea with three measurable criteria and keeps the combination simple enough to explain to a purchasing manager.

Risk Analytics and Early Warning. Studies of digital technology in supply chains show that analytics can reveal risk earlier and support faster response [5]. A useful early-warning rule has clear bands, a visible reason and a defined action. Supplier Risk Forecaster maps its score to risk levels, attaches the contributing factors to each result, and records an alert with a severity and a message when a warning level is crossed.

Time-Series Forecasting. Hyndman and Athanasopoulos describe simple and effective methods such as moving averages, exponential smoothing and trend extrapolation, and stress that a forecast should come with an indication of uncertainty [6]. Forecast accuracy should also be judged against a simple baseline using scale-aware measures [7]. For a small business application with limited history per supplier, such methods are more suitable than data-hungry models, and the confidence rating in this system reflects how much supporting data exists.

Dashboards and Decision Support. Few shows that dashboards should present the most important measures first and use charts that make comparison easy [8]. The dashboard in this work therefore places key performance indicators at the top, followed by risk distribution, order trend and status charts and a list of the highest-risk suppliers, with each severity also written as text so that meaning does not depend on color alone.

Web Application Frameworks. React builds the interface from components that are redrawn when their data changes [14], and Material UI supplies a consistent set of accessible components [20]. Node.js is an event-driven, non-blocking runtime suited to network programs that serve many requests [13], Express provides routing and middleware [15], and MongoDB stores records as flexible documents that suit nested data such as the items of a purchase order [16].

Web Architecture and Security. The back end follows the REST architectural style [9], authenticates requests with signed JSON Web Tokens [10], stores passwords with the adaptive bcrypt scheme [11], and addresses common risks such as broken access control, injection and security misconfiguration listed in the OWASP Top 10 [12] through role checks, request validation and secure headers.

Gaps Identified. Existing work addresses supplier evaluation, risk analytics, forecasting and dashboards separately. Little of it describes a single lightweight application in which supplier records, orders, a transparent score, a forecast with drivers, alerts and reports share one data store, or that states plainly which parts are validated. Supplier Risk Forecaster is built to join these parts and to report their current status honestly.

III. PROPOSED METHODOLOGY

Supplier Risk Forecaster follows a client–server design in which the React client handles presentation and the Express server owns identity, access rules, risk calculation and persistence. This separation keeps business rules and permissions out of the reach of a modified client.

Requirements. The requirements were derived from the product concept: (i) secure sign-in with two roles; (ii) a supplier directory with search and filtering by risk level and rating; (iii) purchase orders with line items, automatic tax and total, status and priority; (iv) a composite risk score from delivery delay, quality compliance and lead-time stability; (v) a 30-day forecast with trend, confidence and drivers; (vi) alerts with an unread count and a resolution workflow; (vii) a dashboard and six exportable reports; and (viii) profile, password and user administration with a safeguard against self-deactivation. Non-functional requirements are input validation, hashed credentials, role-based route protection and a modular code base.

Workflow. A user registers or logs in, and the server returns a signed token and the user profile. The dashboard then shows key indicators. The user adds suppliers and raises purchase orders against them. Delivery and quality results recorded for orders feed the risk service, which recalculates each supplier's score and level. The forecast module projects the score, and the alert service creates an alert when a warning level is crossed. The user reviews the alert in the Alerts Center, takes action, marks it resolved, and exports reports for review meetings.

Data Model. The core entities are stored in four collections: the user (name, e-mail, hashed password, role and active status), the supplier (profile, category, rating, status, performance values, risk score and level), the purchase order (supplier reference, line items, tax, total, status, priority and dates) and the risk alert (supplier reference, severity, message and status). References use object identifiers that are expanded with population when a screen needs names, and indexes support the most frequent queries.

Risk Scoring Logic. Each of the three factors is first converted to a common scale. Delivery delay reflects how late deliveries were compared with promised dates, quality compliance reflects the share of received goods that met requirements, and lead-time stability reflects how far actual lead times vary from expected ones. The factors are combined with weights into a composite score from 0 to 100, where a higher value means higher risk, and the score is mapped to risk levels. The contribution of each factor is shown on the supplier page so that a user can see why a supplier is rated as it is.

Forecasting Logic. The forecast module uses the recent history of a supplier's score and factors to project its value for the next 30 days. It returns the projected score, a trend label (improving, stable or worsening), a confidence rating that is lower when little history exists, and the key risk drivers that explain the movement. Suppliers with a worsening trend are listed on the dashboard so that attention goes first where it is most needed.

Alerts and Reporting. When a risk evaluation crosses a warning level, the alert service stores an alert with the supplier, a severity, a message and a status. The header bell shows the unread count, and the Alerts Center lists alerts with filters and a resolve action. Six report modules are prepared on the server and can be downloaded in the browser as CSV files or as PDF tables, so that exported figures match what is shown on screen.

Security by Design. Passwords are hashed with bcrypt before storage and are excluded from query results, and sessions use signed tokens. Protected routes pass through an authentication middleware, and administrator-only routes are guarded by a role check. Request bodies for authentication, suppliers and purchase orders are validated with express-validator, Helmet sets secure headers, and a central error handler returns a consistent response. Secrets are read from environment files that are excluded from source control, and an administrator cannot deactivate their own account.

IV. SYSTEM IMPLEMENTATION

The methodology is realized as a single-page web application and a REST service, organized into the following modules.

Technology Stack. The client is written with React 18 and Vite [14] using Material UI [20], React Router, Recharts, jsPDF and Axios service modules. The server runs on Node.js [13] with the Express framework [15] and exchanges JSON over HTTP. Data is stored in MongoDB [16] through the Mongoose object document mapper, and the bcrypt and jsonwebtoken libraries provide hashing and tokens, with Helmet, CORS, express-validator and

Morgan in the middleware pipeline.

Authentication Module. Registration, login and password recovery pages call separate endpoints, and the signed token is stored in the browser and added to every request. Protected and administrator-only route components redirect users who are not logged in or who lack the required role.

Supplier Module. The supplier list supports search and filters for risk level and rating and shows a colored risk badge for each supplier. Create and edit pages share one form component, deletion asks for confirmation, and the details page gathers the profile, risk analysis, forecast, alerts and orders of a supplier.

Purchase Order Module. The order builder adds line items dynamically and calculates line amounts, tax and the grand total automatically, so the saved total always agrees with the items. Orders carry a status and a priority shown as badges, and the list, details, edit and delete pages complete the workflow.

Risk and Forecast Module. Dedicated endpoints return the composite score with its factor breakdown and the 30-day forecast. The supplier page displays them as a risk analysis card and a forecast card, and the dashboard summarizes the distribution of risk levels and the suppliers whose risk is worsening.

Alert Module. The alert service creates alerts when a warning level is crossed. The notification bell and menu show unread alerts, and the Alerts Center provides filters, a detail view and a resolve action. Failures in alert creation are logged and do not stop the main action, such as saving an order.

Dashboard and Report Module. The analytics endpoint supplies the indicator cards and charts of the executive dashboard. The reports page offers six reports with filters, summary cards and a table, and the export buttons produce CSV files and PDF tables with jsPDF and jsPDF-AutoTable.

Client State and Error Handling. An authentication context shares the signed-in user across pages, and all server calls are placed in service modules so that pages call a named function instead of building requests. Each screen shows loading and error states, and server messages are surfaced to the user when a form is rejected.

Integration and Testing. The modules were integrated and exercised manually with the client and the server running together against a MongoDB database. The repository contains only a scratch authentication script, and no automated test suite or user study has been completed.

V. ADVANTAGES

1. Early Warning. Deteriorating suppliers are flagged by score, trend and alert before a delivery fails, which moves procurement from reaction to anticipation.

2. Transparent Scoring. Every score is built from three named factors and every forecast lists its drivers, so buyers can question and trust the result.

3. Consolidated Data. Suppliers, orders, alerts and reports share one database, which removes the need to combine several spreadsheets.

4. Role-Based Control. Administrator functions such as user management are restricted by server-side role checks, not only by hidden menus.

5. Automated Reporting. Six reports can be exported as CSV and PDF without manual preparation, and exported figures match the screen because the data are prepared once on the server.

6. Modularity and Extensibility. The client, API, services and database are separate, so new risk factors, reports or integrations can be added without changing existing screens.

7. Cost-Effective Stack. The system uses open-source components and runs in any modern browser without installation.

VI. RESULTS AND ANALYSIS

This section reports what the current prototype demonstrates, what it does not, and how it should be evaluated. No user study or performance benchmark has been run, so we make no quantitative claims about forecast accuracy, alert quality or user benefit.

A. Prototype Status. The status of each module is listed below. All modules are implemented and were exercised manually through the user interface; none has been covered by automated tests.

Authentication and access control: Implemented (JWT, bcrypt, two roles). Validation status: functionally exercised; no automated tests.

Supplier management: Implemented (search, filters, CRUD). Validation status: functionally exercised.

Purchase orders: Implemented (itemized builder, automatic totals). Validation status: functionally exercised; totals not yet checked against an independent reference.

Risk scoring: Implemented (three factors, 0–100 score). Validation status: weights not calibrated against real outcomes; no accuracy claim.

Forecasting: Implemented (30-day projection, trend, confidence, drivers). Validation status: not yet compared with a baseline; no accuracy claim.

Alerts: Implemented (bell, unread count, Alerts Center, resolution). Validation status: thresholds uncalibrated; alert delivery limited to in-app display.

Dashboard and reports: Implemented (charts, six reports, CSV and PDF export). Validation status: functionally exercised on sample data.

User administration: Implemented (edit, activate, deactivate, self-deactivation safeguard). Validation status: functionally exercised.

B. Limitations. (i) The scoring weights and the forecasting method are fixed rules; they do not learn from outcomes and have not been calibrated on real purchasing data. (ii) The forecast is only as good as the history behind it, and its confidence is low for suppliers with few orders. (iii) Data are entered by users; there is no automatic import from an enterprise resource planning system or from supplier portals. (iv) Alerts are shown inside the application only, with no e-mail or SMS delivery. (v) External signals such as financial health, weather or geopolitical events are not considered. (vi) There is no automated test suite and no user study, so correctness and usability are not formally established.

C. Proposed Evaluation Protocol. Before real deployment, each component should be evaluated against a defined reference. Precision, recall and F-measure [18] and ROC analysis [19] suit the alert thresholds, forecast error can be measured with scale-aware measures against a naive baseline [7], usability can be measured with the System Usability Scale [17], and the security review can follow the OWASP Top 10 [12]. The proposed protocol is listed below, component by component.

Risk score: Rank correlation between score and later delivery or quality failures. Reference standard: historical orders with known outcomes.

Forecast: Mean absolute error against a naive baseline. Reference standard: held-out later periods of supplier history.

Alerts: Precision, recall and lead time before failure. Reference standard: labeled incidents from pilot data.

Order totals and reports: Exact agreement of totals and counts. Reference standard: hand-calculated results on test data.

Security: Findings by severity. Reference standard: review against the OWASP Top 10.

Usability: SUS score, task time, error count. Reference standard: representative procurement users.

Performance: 95th-percentile API latency under load. Reference standard: load tests at target concurrency.

D. Discussion. The prototype shows that the proposed architecture can connect supplier records, purchase orders, a transparent risk score, a forecast with drivers, alerts and reports in one working application. It does not show that these features improve procurement decisions or reduce disruptions. Two design questions need particular attention. First, a fixed set of weights may not suit every industry, so the weights should be reviewed with procurement staff and adjusted. Second, a forecast built from few data points can be misleading, which is why the confidence rating is displayed next to every projection and why a baseline comparison is part of the proposed protocol.

VII. CONCLUSION

This paper presented Supplier Risk Forecaster, a web-based procurement risk intelligence platform that manages suppliers and purchase orders, calculates a composite risk score from delivery delay, quality compliance and lead-time stability, forecasts the score for the next 30 days, and raises alerts and reports. The system combines a React client, an Express REST API and a MongoDB database, with token-based sessions, hashed passwords, role-based access and server-side validation.

The current prototype is a functional proof of concept rather than a validated product. The scoring weights are uncalibrated, the forecast has not been compared with a baseline, alerts are shown only inside the application, and no user study has been conducted. We therefore described the limitations explicitly and proposed an evaluation protocol covering risk and forecast quality, alert quality, correctness of totals, security, usability and performance.

VIII. FUTURE WORK

Learned Risk Models. Future versions should replace the fixed weights with a model trained on historical orders and outcomes, compare it with the rule-based score, and add seasonal and machine-learning forecasting with proper accuracy measures.

External Signals and Integration. Financial ratings, news, weather and geopolitical indicators can be added as risk factors, and orders, receipts and quality results can be imported from an enterprise resource planning system or through electronic data interchange.

Notifications and Escalation. Alerts can be sent by e-mail, SMS or messaging applications, with escalation rules when an alert stays unresolved.

Authentication Hardening. One-time passwords, e-mail verification, refresh tokens and finer-grained permissions should be added before a public deployment.

Testing and Quality. Unit, API and end-to-end tests and a continuous integration pipeline should be introduced so that regressions in scoring, totals and permissions are detected early.

Platform and Analytics. Longer-term goals include a supplier portal, alternative-supplier recommendation, what-if simulation of losing a supplier, a mobile application with push alerts, and accessibility and language improvements.

IX. REFERENCES

[1]. Chopra, S., & Sodhi, M. S. (2004). Managing risk to avoid supply-chain breakdown. MIT Sloan Management Review, 46(1), 53–61.

[2]. Tang, C. S. (2006). Perspectives in supply chain risk management. International Journal of Production Economics, 103(2), 451–488. https://doi.org/10.1016/j.ijpe.2005.12.006

[3]. Dickson, G. W. (1966). An analysis of vendor selection systems and decisions. Journal of Purchasing, 2(1), 5–17.

[4]. Ho, W., Xu, X., & Dey, P. K. (2010). Multi-criteria decision making approaches for supplier evaluation and selection: A literature review. European Journal of Operational Research, 202(1), 16–24. https://doi.org/10.1016/j.ejor.2009.05.009

[5]. Ivanov, D., Dolgui, A., & Sokolov, B. (2019). The impact of digital technology and Industry 4.0 on the ripple effect and supply chain risk analytics. International Journal of Production Research, 57(3), 829–846. https://doi.org/10.1080/00207543.2018.1488086

[6]. Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice (3rd ed.). OTexts. https://otexts.com/fpp3/

[7]. Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

[8]. Few, S. (2013). Information Dashboard Design: Displaying Data for At-a-Glance Monitoring (2nd ed.). Analytics Press.

[9]. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation). University of California, Irvine.

[10]. Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519, Internet Engineering Task Force. https://doi.org/10.17487/RFC7519

[11]. Provos, N., & Mazières, D. (1999). A future-adaptable password scheme. Proceedings of the USENIX Annual Technical Conference, FREENIX Track (pp. 81–91).

[12]. OWASP Foundation. (2021). OWASP Top 10:2021. https://owasp.org/Top10/

[13]. Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, 14(6), 80–83. https://doi.org/10.1109/MIC.2010.145

[14]. Meta Open Source. (n.d.). React documentation. https://react.dev

[15]. OpenJS Foundation. (n.d.). Express – Node.js web application framework. https://expressjs.com

[16]. MongoDB, Inc. (n.d.). MongoDB manual. https://www.mongodb.com/docs/manual/

[17]. Brooke, J. (1996). SUS: a quick and dirty usability scale. In P. W. Jordan et al. (Eds.), Usability Evaluation in Industry (pp. 189–194). Taylor & Francis.

[18]. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437.

[19]. Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

[20]. MUI. (n.d.). Material UI documentation. https://mui.com/material-ui/