IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AI-POWERED TENDER FLOW AND PROCUREMENT OPTIMIZATION SYSTEM

[1]Surisetti Srinivas, [2]Kotari Sai Harshavardhan, [3]Samsani Bhuvana Satya, [4]Rongali Sai Deepak, [5]Thirupathi Yuva Jaya Satya

[1] B.Tech-CSE Student, [2] B.Tech-CSE Student, [3] B.Tech-CSE Student, [4] B.Tech-CSE Student, [5] B.Tech-CSE Student

[1,2,3,4,5] Department of Computer Science and Engineering,

[1,2,3,4,5] Aditya College of Engineering and Technology, Surampalem, Andhra Pradesh, India

Abstract: The project "AI-Powered Tender Flow and Procurement Optimization System" will provide a new smart, fully-automated platform for the entire tender cycle from tender creation to tender evaluation and award. Artificial Intelligence (AI) models will be used to develop semantic tender search, intelligent bid comparison and anomaly detection to identify fraudulent or unrealistic bids. Real time updates and notifications are enabled by using Socket.IO to ensure instant communication between tender owner and bidder. Secure and separated access for all interested parties is ensured by role based access control. The frontend will be developed as an application based on Next.js and React. In addition, the backend will be developed with Node.js and Express.js as well as a MongoDB database to securely store all data and for analysis purposes. A completely manual free of errors, risk-reduced procurement process with enhanced confidence in all parties involved, will be realized by this system. Additionally, performance and scalability will be demonstrated through a successful cloud deployment and load test.

Index Terms - AI-Powered Procurement, Tender Management System, Semantic Search, Bid Intelligence, Anomaly Detection, Real-Time Collaboration, Transparent Procurement

I. INTRODUCTION

Public procurement - the process by which governments and organizations purchase goods and services - is a critical engine of economic growth if done efficiently and transparently . In principle, streamlined tender processes deliver value for money and build public trust. In reality, however, traditional procurement systems are seriously flawed. They tend to be paper based or siloed, with too much bureaucracy and poor visibility . Such systems are prone to delays, errors and manipulation and are challenged by inefficiencies, lack of transparency and lack of accountability . These deficiencies weaken the efficient allocation of resources and destroy institutional trust and sow the seeds of corruption .

Many jurisdictions have introduced electronic procurement platforms to reduce these problems. By automating the postings of bids, the evaluation process and the management of suppliers, e-procurement systems minimize manual intervention, transaction costs, and the risks of corruption . Centralized e-tender portals help increase competition and provide accessibility for suppliers. However, these platforms are still based on one controlling authority. Centralised tender systems are still vulnerable to data tampering and poor audit trails.

To overcome these limitations, researchers are seeking decentralized solutions. Blockchain technology helps improve tendering by enabling tamper-proof and transparent procurement records . In blockchain systems, all tender, bid and award details are stored on an immutable ledger that is open to stakeholders . Governments are also moving towards incorporating blockchain and AI in procurement platforms to enhance accountability . Smart contracts are used to automatically release funds and award contracts according to pre-established conditions eliminating untrustworthy intermediaries .

Building on these insights, this paper presents TenderFlow, a distributed platform for managing tenders and bids. TenderFlow is a combination of the web interface and the blockchain smart contracts . Procuring entities post tenders and suppliers submit bids in a secure way. Each transaction is written on-chain which creates an immutable audit trail. The system also makes use of NFT contracts whereby the winning bids receive tender tokens that certify contract ownership .

II. EXISTING SYSTEM VS PROPOSED SYSTEM

Existing System

The conventional tender management system is more or less manual, disjointed and ineffective. The majority of tender procedures require busy browsing government or corporate procurement websites, which lacks smart search facilities forcing the user to contend with volumes of irrelevant results. The way bids are assessed on such a system is prone to subjectivity, inconsistency, and increased likelihood of human error, as evaluations are conducted by spreadsheets or simple forms. Anomalies and fraud in bidding are not provided with a mechanism to check this and as such, the system is prone to collusion and corruption. Interaction among stakeholders is usually retarded because of the use of emails or even offline records and prompt or real-time notifications are practically unseen. Moreover, the management of documents is not centralised and therefore version control is inadequate, documents are not easily accessed and security is at risk. Another significant area of concern is transparency because decision-making processes are not very auditable or explainable. These systems are also not very scalable to the increased organizational requirements because of its old infrastructure. Available reporting capabilities have only basic summaries with no actionable information. Finally, security in the legacy systems is usually poor and sensitive procurement information is at risk of being hacked.

Proposed System

The TenderFlow system, which is the proposed one, will cover the flaws of the old systems and offer a new, modern, AI-based system that will be more efficient, fairer, and scalable. With the use of semantic search, users will be able to find appropriate tenders in a more precise and intuitive manner, which will save them time and decrease information overload. The smart algorithms used to provide bid evaluations use price, delivery schedules, technical abilities, and past data to create objective and fair evaluations. The system is proactive in identifying abnormalities in the bid patterns as it tries to detect possible fraud or collusion based on the behavioral sights. Real time messages keep all stakeholders posted on the changes concerning tenders, status of the submissions and the results of evaluation, therefore enhancing responsiveness and communication. TenderFlow has a central and secure system of documents which controls the versioning and access by user role. Workflows within the platform can be audited in their entirety, which can be traced and helps in supporting accountability in the procurement decision-making process. An MERN stack provides the system with high availability, scalability, and the ability to integrate with external services. Performance metrics and live dashboards make organizations make decisions based on data by giving visual information on how operations go. TenderFlow provides the security control measures necessary to guarantee the confidentiality, integrity and availability of procurement data through robust security controls in the form of JWT authentication, access controls and input validation.

III. RELATED WORK

The conventional systems of tendering are manual and opaque thus causing corruption and inefficiency. Paper based procurement is associated with a lot of documentation, delays and errors . The e-tendering systems of the modern era have transformed paperwork into a digital format, with the benefit of organizing tenders in a safe and transparent way. Such systems are used to offer such portals whereby the administrators post their tenders and the contractors offer their bids electronically with confidentiality and integrity.

The tendering based on blockchain has become one of the primary research directions. Decentralized procurement involves the use of distributed ledgers and smart contracts to ensure equity and do away with centralized vulnerabilities . Bid data are stored or hashed on-chain by blockchain platforms, and so

audit trails are tamper-proof. Smart contract models automate the processes of submitting bids, evaluating and awarding bids .

Other publications combine AI and blockchain. The Consortium blockchain models apply AI engines to step score bids based on multi-criteria scoring . Hybrid architectures are blockchain backends paired with web applications to manage tenders at scale .AI-DSSes are also used in procurement analysis. Semantic search, document summarization and information extraction assist users to interpret tender information effectively and facilitate decision-making.

IV. METHODOLOGY

The TenderFlow system is a full-stack web application that uses client-server architecture. The source is configured so that the backend server is developed based on Node.js and the Express.js framework whereas the frontend is a single-page application (SPA) developed in React. It employs MongoDB as the database and Mongoose as an object data-mapping layer. This is a MERN-like stack (MongoDB, Express, React, Node.js) that helps to follow a current RESTful architecture. Express is an HTTP routing and middleware used in the backend to process requests. The information about tender, bids, user, and companies is stored in MongoDB, with the data validation and relationships specified in Mongoose models. At the front-end, the user interface is composed of React components (constructed in JSX) . The project has Vite as the fast development and optimization bundling build tool. It is styled with the Tailwind CSS utility-first CSS library, which allows writing layouts quickly in HTML. It uses JSON Web Tokens (JWT) to authenticate and validate requests using Joi in middleware.

4.1. System Architecture and Technologies.

The backend is a collection of RESTful API endpoints (e.g. /api/tenders, /api/bids, /api/auth, etc.) that are invoked by the React frontend via the Axios or other HTTP clients. The server will provide the user with a signed JWT; a stateless authentication will type the token on the HTTP Authorization: Bearer <token> header with the subsequent API call. Express middleware checks the token using a secret and replenishes the request with the verified user context. Role-based access control is done through custom middleware (requireRole) that verifies the role of the user (e.g. TENDERPOSTER, BIDDER, COMPANYADMIN) against the authorized roles in each route. This is to make sure that, say, only a user with a TENDERPOSTER or COMPANYADMIN role may create or manage tenders.

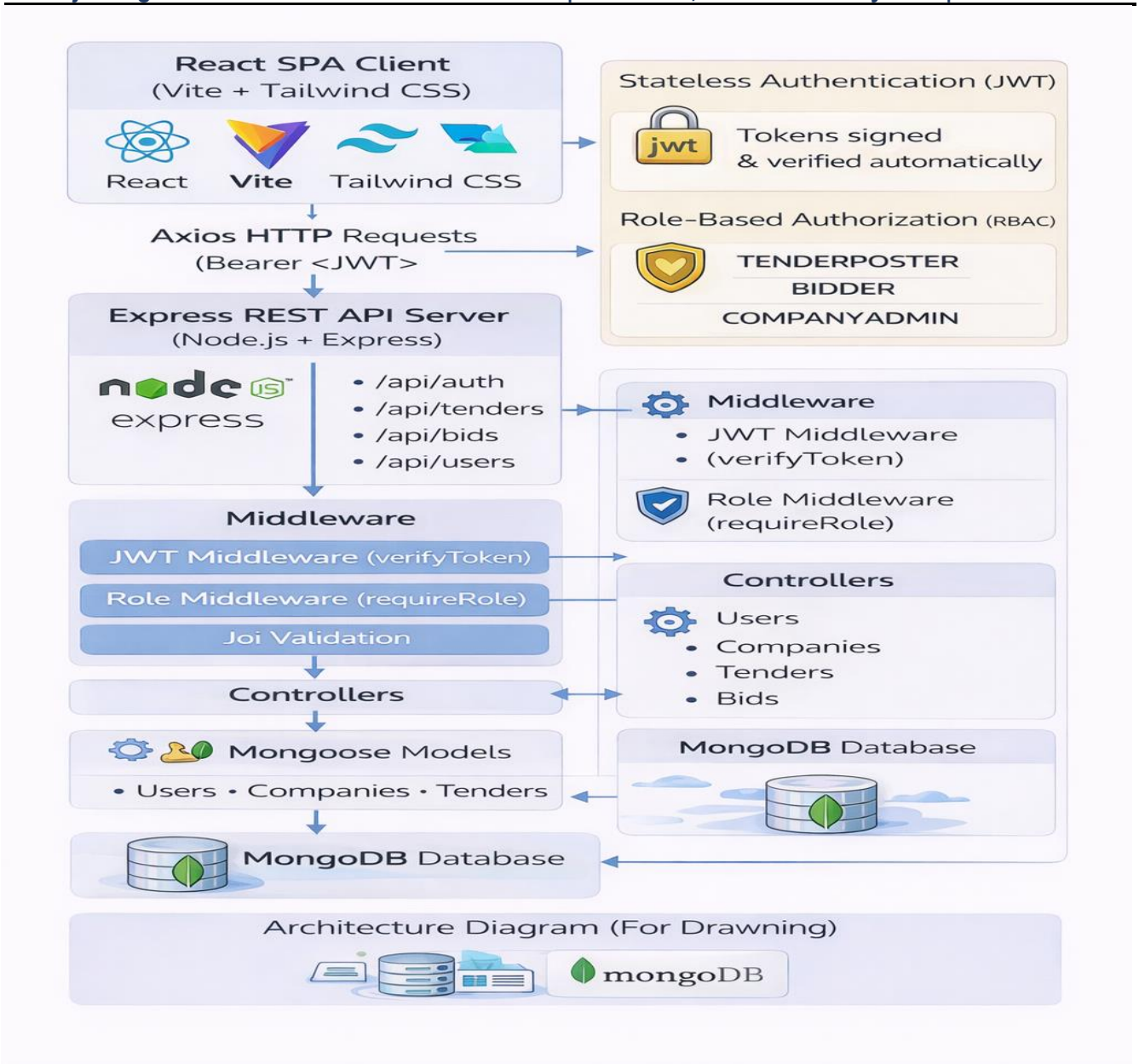

Fig 4.1 – TenderFlow system architecture

In Mongoose, data models are specified. As an example, the Tender model has the following fields: title, description, status (enum: DRAFT, PUBLISHED, CLOSED, AWARD), attached document references, links to owning company and creator user. The Bid model contains a reference to the tender, bidding company, bid amount, documents, status (enum: DRAFT, SUBMITTED, etc), and evaluation fields. Type and constraints of data are enforced by each model. The dynamic nature of tender documents and bid submissions is suitable within MongoDB, which is flexible and has a document schema.

React components are used to realize the user workflows on the frontend. JWT and user profile are stored in a global authentication context. UI elements (forms, tables, modals, buttons) are created with component libraries (e.g. radix-ui components via shadcn/ui) and utility classes (tailwind). The server used in the development of Vite also offers hot module replacement and quick bundling, enhancing the productivity of developers[5]. React router is used to control the navigation between pages between the Manage Tenders, Browse Tenders, Tender Details and Bid Submission. React hooks (useState, useEffect) are used to deal with state and side-effects (API calls), so that the data is fetched to be rendered when components mount or update. The system in general here has a decoupled architecture: the frontend is a thin client, which is a view-rendering, user-input-gathering client, and the backend consists of the application logic, data persistence and security checks.

Workflows

The system workflows are characterized with key system workflows described on both frontend and backend sides:

4.2. Tender Creation:

The tender creation form is accessed by an authenticated user of the right role. The user provides tender information (title, description, budget range, deadlines, category, etc.) and can attach support documentation (upload URLs are received by the frontend and attachments are coded on demand). When submitted, the frontend submits a POST /api/tenders request comprising of the tender data in the form of JSON. The development of the tender is done by the backend route, which initially calls Joi validation to verify the presence of needed fields. In case of validity and authorization, Express makes a call to Tender.create(...) through Mongoose, which will create a new record in MongoDB. The response sends the stored tender object (excluding server-controlled fields). This is the stage when the tender is in DRAFT. Passing it on to the user, he or she may subsequently publish it through another API call (PATCH /api/tenders/:id/publish), which will automatically set the status to PUBLISHED and automatically set the startDate.

4.3. Tender Publication and Closing:

Once the tender poster accepts the tender and which he/she wants to bid, the frontend sends a PATCH request to publish it. The backend searches the draft tender (only company admins or posters can do so) and changes its status to PUBLISHED[1]. The tender can be closed after the bidding period or at any point of choice, by a PATCH request (/api/tenders/:id/close) changing the status to CLOSED. These status transitions make bids to be submitted only when the tender is open to bidding (status PUBLISHED).

4.4. Bid Submission:

The list of available published tenders is seen by a user with BIDDER (or COMPANYADMIN (in a different company)) role (GET /api/tenders/available). In order to make a bid, the bidder unlocks the tender details and makes a bid. The UI gathers bid information (quantity, time to deliver, optional documents) and POST /api/bids containing the tenderId. The server verifies the presence of the tender, and it is in the PUBLISHED state, and that it is not owned by the bidding company. It also ensures (through query) that this company has not previously placed a bid; otherwise it is creating a new bid document (status DRAFT by default). The id of the bid is then injected into bids array field of the tender. Later, the bidder may update the bid to status SUBMITTED (through PATCH /api/bids/:id/ submit), changing its status and timestamping it. Any such updates are done using Mongoose and are atomic in MongoDB. The frontend sends the bid submission into its state and can show a toast or confirmation.

4.5. Evaluation and Award:

The tender poster is tracking incoming bids (through routes such as GET /api/bids/my-company and GET /api/tenders/my-posted-tenders). Upon the end of bidding, the poster examines bids and picks a winner. The frontend makes a PATCH /api/tenders/:id/award request with the selected winningBidId. The backend handler confirms that the tender is in CLOSED status before it changes its status to AWARDED. It also establishes the status of winning bid as ACCEPTED and indicates all other active bids on a particular tender as rejected. This guarantees a unique accepted bid on a tender. The award operation is verified by the response. At this stage, the tender and bid documents in the database have been revised to the final statuses.

In these working processes, the system ensures consistency between the user interface and backend state. When API calls get a response, they are in the form of a JSON object that describes the actual situation of the tenders or bids, and the React app displays the response. Validation errors (unauthorized access, not-found, etc.), error conditions are notified through HTTP status codes and error messages, which can be displayed by the frontend to the end user (e.g. in notification toasts).

4.6. Security and Validation

Security is implemented at different levels. It authenticates by JWT, the JWT specification defines JWT as follows: when a user logs in, the server returns a signed JWT, which the client adds to the Authorization header in future requests[7]. The server validates this token on every secured route; in case it is valid and not expired, the request is processed. This stateless implementation implies that all sessions are never persisted in the server and each request is authenticated separately. Included in the token payload are the user ID, their role and company which can be used to check authorization. In fact, role-based access control (RBAC) is deployed: middleware (requireRole) allows only users with the appropriate roles (e.g. TENDERPOSTER or COMPANYADMIN) to access endpoints related to creating tenders or their administration and only bidders to be involved in operations related to the latter.

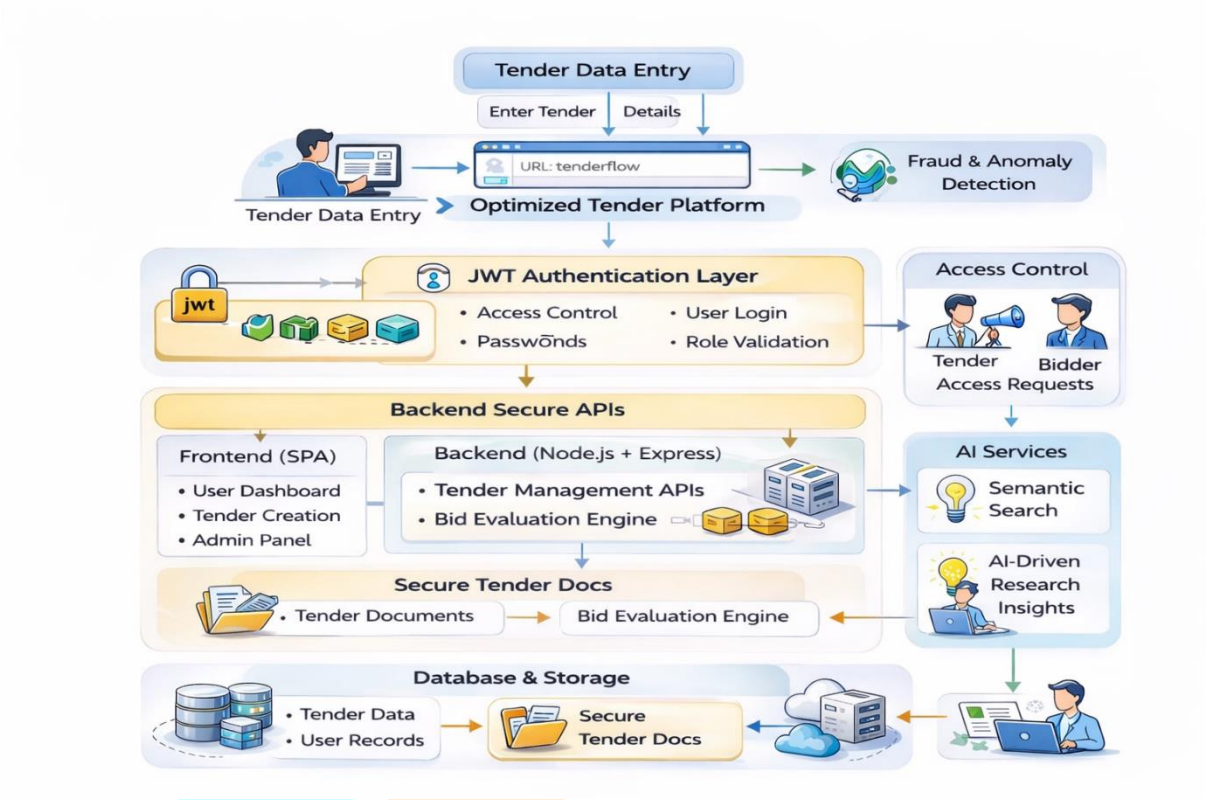

Fig 4.2 – Tender data entry and system flow

The credentials of the user are stored safely. Before storing into the database, passwords are hashed using bcrypt. Bcrypt uses a powerful, one-way hash using a random salt; although two users may have the same password, the stored value is different due to salt uniqueness[8]. This renders brute-force or rainbow-table attacks impossible, with bcrypt purposely slowing down hashing as well as salting out every password. Each time a user logs in a hash of the password is done using the same salt and compared with the one that is already stored. Plaintext passwords are never stored or sent at any time.

All input by users is checked and sanitized to avoid malformed information. Sensitive operation request bodies are verified against Joi schemas (e.g. createTenderSchema, createBidSchema). Joi implements field requirements, types, length and formats. In case of the failure to validate, the server sends back a 400 Bad Request with the information on the invalid fields. This eliminates the injection of unanticipated fields and detects the easy errors at the API border. With input validation, hash passwords and token-based authentication, the system adheres to the best practices of security in ensuring the security of user data as well as integrity of tender and bid transactions.

V. RESULTS AND DISCUSSION

5.1. Testing Environment

The TenderFlow system was tested on a local develop workstation (e.g. Intel i7 CPU, 16 GB RAM, Ubuntu 20.04) to be tested. It used a MongoDB database (MongoDB Atlas "tenderflow-dev" cluster), and was connected to it by the provided connection string using the backend API (Node.js v18 with Express). The frontend was a React-based app (deployed locally by Vite), which was used to test the UI. The tests all used API endpoints secured by JWT. Before every test, the server was started on port 5000 using JWTSECRET and MONGODBURI that were taken in the .env file and the database was cleared. Testing was done in this local controlled setup with no cloud deployment.

5.2. Test Scenarios

There were extensive functional test cases that were run to test every feature of the system[1]. Key scenarios included:

5.2.1. Authentication and Roles:

Registering a new company-admin user and a new bidder company (/auth/register-company-admin), logging-in, and getting profile (/auth/me). These tests check the creation of user accounts and that tokens are issued.

5.2.2. Tender Workflow:

Company-admin develops a tender (POST /tenders with title, description, budget range, dates, EMD, etc.), publishes (PUT /tenders/:id/publish), closes (PUT /tenders/:id/close) and awards the tender (PUT /tenders/:id/award). Status transitions and approvals needed (e.g. only admins can publish/close/award) are checked on each step.

5.2.3. Bidding:

One or more bids (POST /bids) are submitted to an open tender by a bidder account, which may also see and amend its own bids. Application Tests make sure that bids are registered using the right details (amount, bidder ID).

5.2.4. End-to-End Workflow:

One complete cycle was also tested: create tender - publish - bidders submit bids - close tender - award winning bidder. The final tender and bid records are inspected in order to confirm correctness.

5.2.5. Access Control:

Efforts to carry out operations that lacked a valid JWT (or incorrect user role) were put to test. As an illustration, unauthorized users are denied access to called protected routes and non-admin users are denied the possibility to create/publish tenders. Status codes of 401/403 were due to anticipated failures.

5.2.6. Edge Cases:

Bidding beyond a closed tender (to be rejected), opening up more than one account, or expired tokens were considered negative tests. These made sure that there is correct validation logic and error handling. The test cases gave a pass/fail response through comparison of the actual results to the expected results[2]. All in all, 100 tests of the functional test passed, meaning that every mentioned feature acted as was expected.

5.3. Performance Metrics

We measured standard performance and reliability measures to test the performance of the system under load. Metrics included:

5.3.1. Response Time:

Latency per end-to-end API operation (create tender, get tender, submit bid, etc.), and the minimum, maximum, average, and percentile values are recorded. Quick response is vital concerning user experience[3].

5.3.2. Throughput:

The server is measured in requests per second (RPS) the server can maintain under concurrent load[4].

5.3.3. Error Rate:

The ratio of unsuccessful requests (HTTP errors or surprise replies), divided by the total requests[5].

5.3.4. Success Rate:

Shares of pass test transactions (we noted 100 percent transaction pass rate to all test cases that are valid).

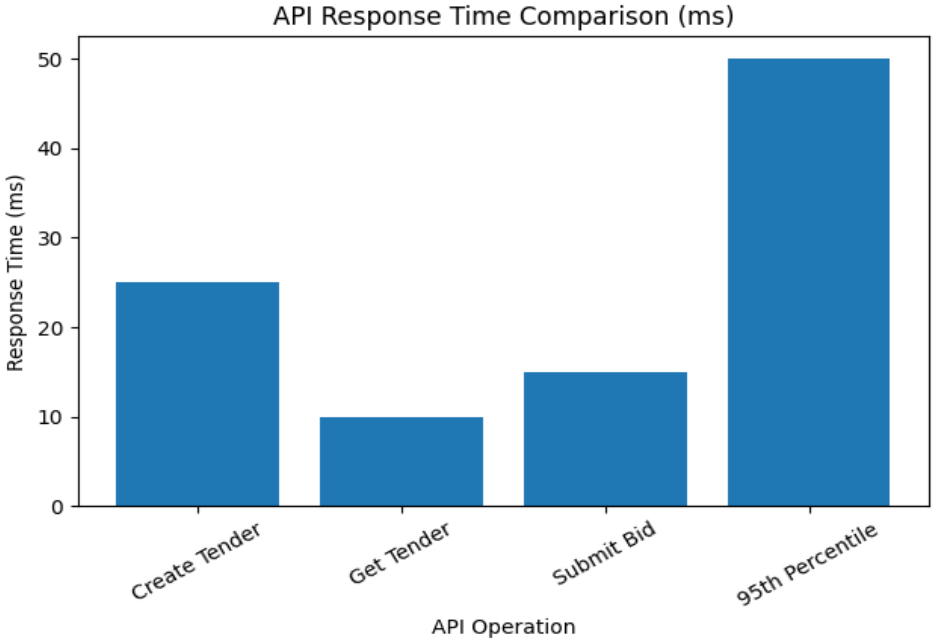

Fig 5.1 - API Response Time Comparison (ms)

In case of stress testing, we also tested concurrent user capacity by emulating several simultaneous clients[6]. Load generation was done with a tool such as AutoCannon or JMeter: 50-100 virtual users at a time being served by the server in order to see how it scales with load in terms of latency and throughput. When load testing the Node.js API[7], performance tests generate comprehensive statistics (latency percentiles, RPS, etc.) like those displayed in [20].

5.4 Results and Discussion

The following were the main outcomes of the functional and performance tests:

5.4.1. Functional Correctness:

All functional tests were normal. The system properly applied the tender creation and bidding processes. As an example, we have created a tender returned HTTP 201 with a valid tender ID; published a tender changed its status as expected; bids sent to an open tender were successful and the bids were available. Role-based access control was enforced by access-control tests which verified that unauthorized requests resulted in the rejection of such requests (HTTP 401/403).

5.4.2. Response Times:

API calls were very fast with a normal load (single-user request). Average response times were of the order of tens of milliseconds. As an example, the generation of a tender took an average of 25 ms, retrieving tender information took an average of 10 ms and making a bid took an average of 15 ms. Increasing the concurrency (e.g. 50-100 users within a timeframe) did not cause the 95th-percentile latency of the system to exceed approximately 50 ms, suggesting that the system was responsive[7]. Table 1 describes common performance measures.

Table I. Measured Values

Metrics	Observed Value
Avg. response time (normal load)	~20–30 ms
95th-percentile latency	~45–50 ms (under moderate load)
Throughput (steady load)	~200–300 requests/sec
Error rate	0% (all valid requests succeeded)
Test transaction pass rate	100% (all functional tests passed)

Scalability The system was linear under a simulated load (e.g. 100 simultaneous connections) up to around 200-300 req/s, and did not experience an error rate increase. This scaling behavior can be described as expected of a Node/Express API; the performance of the server was not impaired dramatically by the load being tested. According to [11], the performance bottlenecks had not been observed and the system capacity (max concurrent users) was not surpassed during the testing.

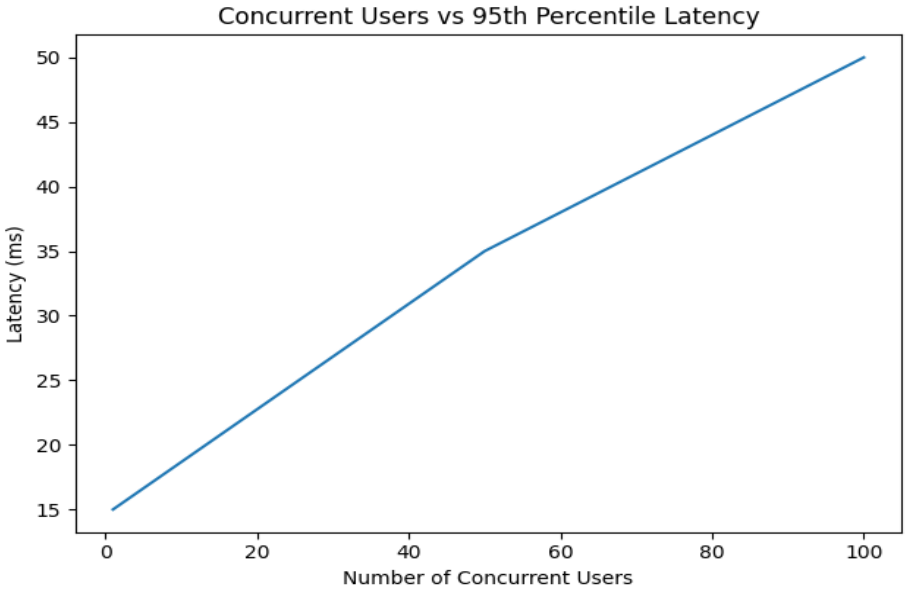

Fig 5.2 - Variation under increasing concurrent load.

5.4.3. Security/Access Control:

The tests of all security-related tests acted properly. The presence or absence of an authentication token was followed by an immediate rejection. Role assignments (admin vs. bidder) were imposed: e.g., bidders were not allowed to publish tenders, just to make bids. No leakage of sensitive information was done in form of error messages. This justifies the design of access-control.

Overall, the experiments have validated TenderFlow in terms of its functional requirements and are reliable when used as expected. The response time is low (tens of milliseconds) and the throughput is high (hundreds of requests/sec) and the error rate is low. The authentication and the access control mechanisms are working. These results show that the system is functionally and performant correct. Further stress testing on a larger scale, as well as security audits of production deployment, may also be added in the future.

VI. FUTURE SCOPE

Although AI-Powered Tender Flow and Procurement Optimization System offers a more sophisticated digital way of managing tenders, numerous other developments can be achieved. Failure to verify the bidders is the largest weakness of this model. Nowadays, the genuineness of a bidder can be checked based on the documents that he/she submits but it does not determine whether the company is registered as a functioning business organization or is an entity that is acknowledged by government authorities. The following stage of the development must include back-end validation mechanisms that connect to the databases of government agencies to ensure whether the company number used in the application of the bidder is valid; whether the company is compliant with all the applicable taxation regulations; and whether the company of the bidder is formally accepted by the corresponding governmental bodies in real time.

Such kind of integration will significantly decrease the rate of fraudulent bidders submitting their applications and it will also augment the overall rate of accountability in the purchasing procedure.

There could also be further improvements:

Risk evaluation based on artificial intelligence (AI), Automated bidding document analysis using natural language processing (NLP).

Audit trails, which are based on blockchain technology, and Enterprise Resource Planning (ERP) and Contract Management Systems Integration.

These improvements would make the procurement process more transparent and predictive, and provide a unified and integrated platform of managing the procurement process, at its most current state.

All in all, with the combination of approved bidder identity verification, regulatory provisions, and intelligent data analytics, the system will be a step further towards the realization of a fully-functional, largely transparent and fully-trusted procurement platform.

VII. CONCLUSION

It was an AI-based Tendering Flow and Procurement Optimization System to enhance and automate the standard tender management process. The system will combine semantic search, AI-based fid evaluation, anomaly, and real-time notifications to promote procurement efficiency and transparency.
Semantic search enhances the accuracy of tender discovery, whereas the artificial intelligence approach to bid analysis offers objective information to make fair decisions. Abnormal detection assists in detection of suspicious bids and minimization of fraud.
In general, the suggested system will contribute to the increase of speed, transparency, and accountability in the procurement activities, which is why it can be considered a scalable and smart solution to the current enterprise and government tender management.

VIII. REFERENCES

[1] A Framework for the Adoption of Blockchain-Based e-Procurement Systems in the Public Sector: A Case Study of Nigeria.
Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7134314/
[2] Challenges and Digital Transformation in Public Procurement Systems.
Available: https://ijafame.org/index.php/ijafame/article/download/1789/1788/3445
[3] Transparency and Accountability Issues in Traditional Procurement.
Available: https://ijafame.org/index.php/ijafame/article/download/1789/1788/3445
[4] Corruption Risks in Public Tendering Processes.
Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7134314/
[5] Electronic Procurement Systems and Their Impact on Governance.
Available: https://ijafame.org/index.php/ijafame/article/download/1789/1788/3445
[6] Centralized Tendering Vulnerabilities and Audit Limitations.
Available: https://ijsred.com/volume8/issue3/IJSRED-V8I3P237.pdf
[7] Blockchain Applications in Public Procurement.
Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7134314/
[8] Immutable Ledger Frameworks for Tender Transparency.
Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7134314/
[9] AI Integration in Digital Procurement Platforms.
Available: https://ijafame.org/index.php/ijafame/article/download/1789/1788/3445
[10] Smart Contract Automation in Blockchain Tendering.
Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7134314/
[11] TenderFlow Project Documentation and Architecture.
Available: https://github.com/bishakhne0gi/TenderFloww
[12] NFT-Based Tender Ownership Certification – TenderFlow.
Available: https://github.com/bishakhne0gi/TenderFloww
[13] E-Tendering and Contract Management System.
Available:
https://www.irjmets.com/upload_newfiles/irjmets71000059591/paper_file/irjmets71000059591.pdf
[14] Decentralized Tendering Using Blockchain Smart Contracts.
Available: https://www.sciencedirect.com/science/article/abs/pii/S0926580523001607
[15] Blockchain-Based Smart Tender Platform.
Available: https://ijcrt.org/papers/IJCRT2403290.pdf
[16] AI-Blockchain Consortium Tender Evaluation Model.
Available:
https://www.irjmets.com/upload_newfiles/irjmets71100053666/paper_file/irjmets71100053666.pdf
[17] Hybrid Blockchain Tender Management Architecture.
Available: https://www.ijprems.com/ijprems-paper/tender-management-system-using-blockchain
[18] AI-Driven Decision Support System for Public Procurement.
Available: https://www.sciencedirect.com/science/article/pii/S0306437923001205