iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** XI **Month of publication:** November 2026

DOI: https://doi.org/10.22214/ijraset.2026.85536

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue XI Nov 2026- Available at www.ijraset.com

AI-Driven Insurance Claim Processing System

Ms. Kavitha K[1], Alina Afra A[2], Chandhana G[3], Kavya S[4]
Department Of CSE, Adhiyamaan College Of Engineering, India

Abstract: ***Insurance claims are still checked largely by hand: an adjuster reads scanned forms, compares them with the policy, looks for signs of fraud and then decides, which makes settlement slow and uneven. This paper presents the AI-Driven Insurance Claim Processing System, a role-based web application that automates the routine part of this workflow and keeps the adjuster in control of the rest. Policyholders submit a claim and upload supporting documents; optical character recognition (OCR) extracts the key fields; rule-based checks confirm that the policy is active and covers the claim; and a supervised machine-learning model estimates the probability of fraud and reports the factors behind it. A transparent weighted decision score built from document quality (30%), policy validity (30%), fraud safety (25%) and amount reasonableness (15%) routes each claim to an approval recommendation, adjuster review or investigation, and hard rules override the score. Every decision is stored with its component scores and reasons for audit. The system is built with React, FastAPI, scikit-learn and PostgreSQL. We describe the architecture, database design, REST interface and scoring algorithm, demonstrate the routing with a worked example, and discuss limitations and the planned evaluation.***

Keywords: insurance claims, claim processing, machine learning, fraud detection, OCR, decision support, explainable AI, FastAPI, React.

I. INTRODUCTION

Insurers receive a continuous stream of claims for motor accidents, medical treatment, property damage and other losses, and regulators expect them to be settled within prescribed time limits [1]. Most of this work is repetitive: reading a form, checking that the policy was in force, confirming that the right documents are attached and comparing the amount with what is normal. A small share of claims, however, is inflated or fraudulent, and these must be found without delaying the many honest claimants [2]. When both tasks are done by hand, settlement is slow, adjusters apply the checks differently, and the reasons for a decision are hard to audit.

This paper describes the AI-Driven Insurance Claim Processing System, a web application built to address two practical problems. The first is speed: routine, well-documented claims should move through with little manual effort. The second is trust: automated decisions must be explainable, and risky or unclear claims must reach a human adjuster. The main contributions are:

1) *End-to-end claim pipeline:* Claim entry, document reading, policy checks, fraud screening and decision routing are combined in one workflow, from submission to final decision.
2) *Transparent decision score:* A four-factor weighted score routes each claim to approval, review or investigation and returns a short human-readable reason; hard rules such as a lapsed policy override the score.
3) *Human-in-the-loop audit trail:* Automated outcomes are recommendations that an adjuster can confirm or override, and every decision is stored with its scores, model output and reviewer.

The rest of the paper is organised as follows. Section II reviews related work, Section III states the problem, Section IV presents the architecture, Section V explains the methodology and algorithm, Section VI describes the implementation with a worked example, Section VII discusses limitations and future work, and Section VIII concludes.

II. LITERATURE REVIEW

Fraud detection is the most studied use of machine learning in insurance. A review of data-mining techniques for financial fraud detection classifies the literature by technique and by type of fraud and identifies insurance as one of the main application areas [2]. For automobile insurance, a comparison of classification techniques on real claim data reported that several standard classifiers performed comparably well and that simple models were competitive [3]. This supports using a conventional supervised classifier, rather than a complex model, as the fraud component of our system.

Document handling is the other bottleneck. Open-source OCR engines such as Tesseract [4] convert scanned forms, bills and reports into text and give a confidence value, which can be used to decide whether a document is readable enough to trust. For tabular claim attributes, tree ensembles such as random forests [5] and gradient-boosted trees [6] are widely used because they handle mixed

©IJRASET: All Rights are Reserved

numerical and categorical features with little preprocessing. Because a claim decision affects a person's money, the reason for a score matters as much as the score; attribution methods such as SHAP [7] explain how each feature contributed to an individual prediction.

Many studies address only one of these steps, for example fraud classification alone. Few tools combine OCR, policy validation, fraud scoring and a visible routing rule in one lightweight system that a small insurer or a student team can host themselves. The AI-Driven Insurance Claim Processing System targets this gap.

III. PROBLEM STATEMENT AND OBJECTIVES

Given a submitted claim, its supporting documents and the issuing policy, the system must (a) extract the facts of the claim from the documents, (b) check them against the policy, (c) estimate how likely the claim is to be fraudulent, and (d) route the claim to the right level of human attention while recording the reasons. The objectives are to reduce the time spent on routine claims, to make the routing logic visible and adjustable, to keep an adjuster in control of uncertain cases, and to keep the system simple enough to deploy with open-source tools.

IV. SYSTEM ARCHITECTURE

A. Overview

The system follows a three-tier architecture (Fig. 1). A React single-page application provides the user interface; a FastAPI service implements the business rules and hosts the AI pipeline; and a PostgreSQL database stores users, policies, claims, documents and decisions. The client and server communicate through JSON over HTTP, following the REST style [8].

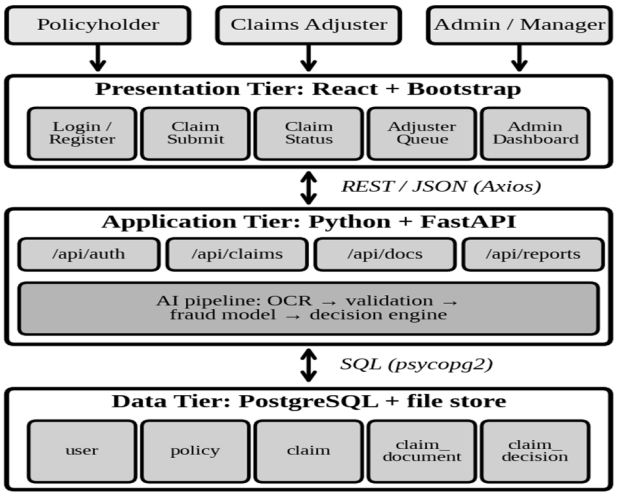


Fig. 1 Three-tier architecture of the AI-Driven Insurance Claim Processing System

B. Technology Stack

Table I lists the technologies used. All are open source and widely documented, which keeps the barrier to deployment low.

TABLE I
TECHNOLOGY STACK

Layer	Technology
Front end	React, Bootstrap, Axios [9]
Back end	Python, FastAPI [10]
AI components	Tesseract OCR [4], scikit-learn [11], SHAP [7]
Database	PostgreSQL [12]
Interface	REST over HTTP, JSON payloads [8]

C. Database Design

The schema has five tables. The user table holds login details and the role. The policy table stores the policy number, holder, type of cover, validity dates, sum insured and status. The claim table stores the policy reference, incident date, claim type, claimed amount, description and a status (Submitted, Processing, Approved, Held, Under Review or Under Investigation). The claim_document table links each uploaded file to a claim together with its OCR text and confidence. The claim_decision table stores the four component scores, the final score, the fraud probability, the routing outcome, the reason text, the reviewing user and a timestamp, which forms the audit trail. Policy numbers must be unique, and a claim with the same policy, incident date and amount cannot be submitted twice.

D. Role-Based Access

Three roles are supported. Public visitors can only view the home page and register. Access to the other pages is controlled by a protected-route component on the client, as summarised in Table II.

TABLE II
ROLES AND PERMITTED PAGES

Role	Permitted functions
Policyholder	Register, log in, submit claims, upload documents, track claim status
Claims adjuster	View review queue and score explanations, approve, reject or request documents
Admin	Manage users and policies, view all claims and reports, set thresholds

V. METHODOLOGY

The processing flow for every claim is shown in Fig. 2. Steps 1 to 4 gather evidence about the claim; step 5 combines it into one score that decides how much human attention the claim needs.

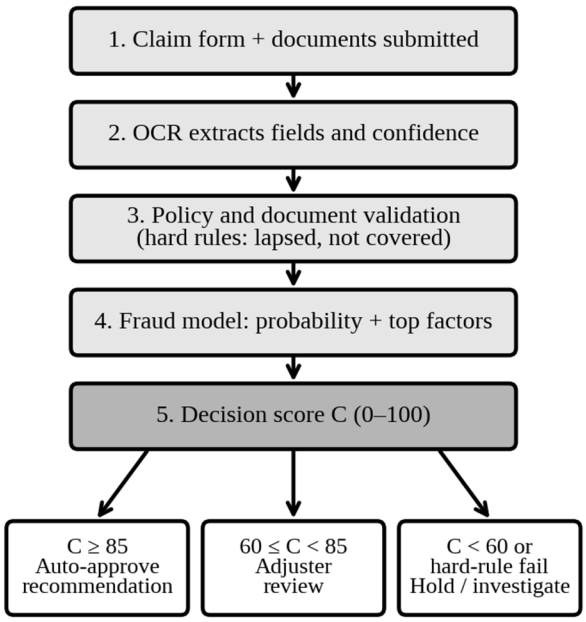


Fig. 2 Claim processing flow and routing outcomes

A. Claim Submission and Document Processing

A policyholder logs in, selects a policy and enters the incident date, claim type, amount and a short description. The required documents depend on the claim type, for example a claim form, identity proof and repair estimate for a motor claim, or a discharge summary and hospital bill for a health claim. Each uploaded file is passed to the OCR engine, which returns the text and a confidence value. Key fields such as name, policy number, dates and amounts are located with pattern rules and compared with the data the user entered; any mismatch is recorded as a flag.

B. Hard-Rule Validation

Before scoring, two hard rules are checked: the policy must be active on the incident date, and the claim type must be covered by the policy. A claim that fails either rule is held with the reason shown, and the adjuster confirms the rejection. These rules sit outside the weighted score so that a good score on other factors can never hide a policy failure.

C. Fraud Screening

A supervised classifier is trained on historical labelled claims. The inputs include the claimed amount, its ratio to the sum insured, the days between policy start and incident, the days between incident and report, the number of earlier claims by the same policyholder, the claim type and the number of document mismatch flags. A random forest [5] is used as the baseline and gradient boosting [6] can be substituted. The model outputs a fraud probability p between 0 and 1. The main factors behind each prediction are computed with SHAP [7] and shown to the adjuster. Class imbalance is handled by class weighting, and the investigation threshold is set by the insurer according to how many investigations it can handle.

D. Decision Score and Routing

For each claim that passes the hard rules the system computes a score from 0 to 100 using four components, combined as in (1):

$$C = 0.30D + 0.30P + 0.25F + 0.15A \qquad (1)$$

where D is the document score, P the policy score, F the fraud-safety score and A the amount score, each between 0 and 100. The final score is rounded to the nearest integer.

1) *Component scores:* The rules are given in Table III. D rewards complete and readable documents. P is 100 for an active policy that covers the claim within the remaining sum insured, 60 when the claimed amount exceeds the remaining sum insured, and 0 when a hard rule fails. F is 100 × (1 − p). A compares the claimed amount with the median amount for the same claim type, so unusually large claims lose points.

TABLE III
SCORING RULES FOR THE DECISION SCORE

Component	Weight	Rule (component value)
Document D	0.30	All present, OCR ≥ 90%: 100; present, OCR 70–90%: 70; one missing: 40; more missing: 0
Policy P	0.30	Active and covered: 100; above remaining sum insured: 60; hard-rule fail: 0
Fraud safety F	0.25	100 × (1 − p), p = fraud probability
Amount A	0.15	Up to 1.0× median: 100; to 1.5×: 70; to 2.0×: 40; above: 10

2) *Routing:* A claim with C ≥ 85 is recommended for approval; 60 ≤ C < 85 goes to adjuster review; C < 60 goes to investigation. Independently of the score, a claim with p ≥ 0.5 is sent to investigation, and a claim that fails a hard rule is held.
3) *Explanation:* With each decision the service returns the four component scores, the fraud probability, the top SHAP factors and a short reason such as “Documents complete, Policy active, Low fraud risk, Amount normal”, so an adjuster can see why a claim was routed and override the result if needed.

4) *Confirming a decision:* The adjuster confirms or overrides the recommendation, and the service writes a record to the claim_decision table with the outcome, reviewer and timestamp. A claim that has been finalised cannot be decided a second time.

E. REST Interface

Table IV summarises the main endpoints exposed by the FastAPI service.

TABLE IV
MAIN REST ENDPOINTS

Endpoint	Purpose
POST /api/auth/register, /login	Sign-up and sign-in
POST /api/claims	Submit a claim
POST /api/claims/:id/documents	Upload supporting documents
POST /api/claims/:id/process	Run OCR, validation, fraud model and scoring
GET /api/claims/:id	Claim status and decision details
GET /api/claims/queue	Claims awaiting review
PUT /api/claims/:id/decision	Confirm or override a decision
GET /api/reports/summary	Counts by status and outcome

VI. IMPLEMENTATION AND ILLUSTRATIVE EXAMPLE

A. User Interface

The front end consists of pages for the home screen, login, registration, policy details, claim submission, document upload, claim status, the adjuster review queue and the administrator dashboard. Policyholders see the status of each claim and any request for further documents. The adjuster's queue lists claims by score and shows the component scores, the fraud probability and the top factors next to the extracted document fields, with buttons to approve, reject or request documents. The dashboard shows counts of claims by status and outcome.

B. Worked Example

To illustrate the routing we apply the rules to four hypothetical claims. This example is constructed to explain the algorithm; it is not a field evaluation. The results are shown in Table V and Fig. 3.

TABLE V
EXAMPLE ROUTING OF FOUR HYPOTHETICAL CLAIMS

ID	Type	D / P / F / A	Score	Outcome
C1	Motor	100 / 100 / 95 / 100	99	Approve
C2	Health	70 / 100 / 72 / 40	75	Review
C3	Property	40 / 60 / 40 / 40	46	Investigate
C4	Motor	100 / 0 / 90 / 100	68	Hold

C1 is recommended for approval: the documents are complete and readable, the policy is active, the fraud probability is low (0.05) and the amount is typical. C2 has a poor-quality scan, a moderate fraud probability (0.28) and an amount about 1.8 times the median, so it goes to an adjuster. C3 has a missing document, an amount above the remaining sum insured and a fraud probability of 0.60, so it is sent for investigation. C4 would score 68 and be reviewed, but its policy had lapsed, so the hard rule holds it; this is why hard rules sit outside the weighted sum. Fig. 3 breaks each score into its weighted contributions.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

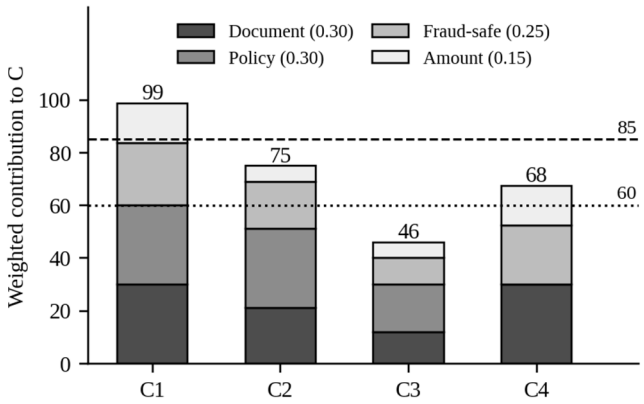


Fig. 3 Weighted contribution of each component to the decision score for the four example claims

VII. LIMITATIONS AND FUTURE WORK

The current system is a prototype and has limitations that we plan to address.

1) *Data:* The fraud model needs labelled historical claims, and real insurer data is confidential. The model must be trained and validated on representative data, and precision, recall and false-positive rate should be reported for each claim type before any real use. Past fraud labels may also reflect earlier investigator bias.
2) *OCR and extraction:* Pattern-based field extraction is fragile for handwritten or low-quality scans. Layout-aware document models and field-level confidence checks would improve it.
3) *Weights and thresholds:* The weights (0.30, 0.30, 0.25, 0.15) and thresholds (85, 60, 0.5) are set by design judgement. Future work should calibrate them with experienced adjusters, compare the routing against manual decisions on real or simulated claims, and measure the time saved.
4) *Security and fairness:* Claims contain personal and medical data. Passwords should be salted and hashed, sessions should use signed tokens with server-side role checks, files should be encrypted at rest and access should be logged. Model inputs should be reviewed so that protected attributes and their proxies are not used.

Further extensions include notifications by SMS or email, integration with policy-administration and payment systems, image-based damage assessment and a mobile client.

VIII. CONCLUSIONS

We presented the AI-Driven Insurance Claim Processing System, a role-based web system that combines OCR-based document reading, hard-rule policy validation, supervised fraud screening and an explainable weighted score that routes each claim to approval, review or investigation. The implementation uses common open-source technologies, which makes it easy to deploy and extend. The worked example shows how the routing behaves, and the discussion identifies the data, security and evaluation work needed to move from prototype to field use.

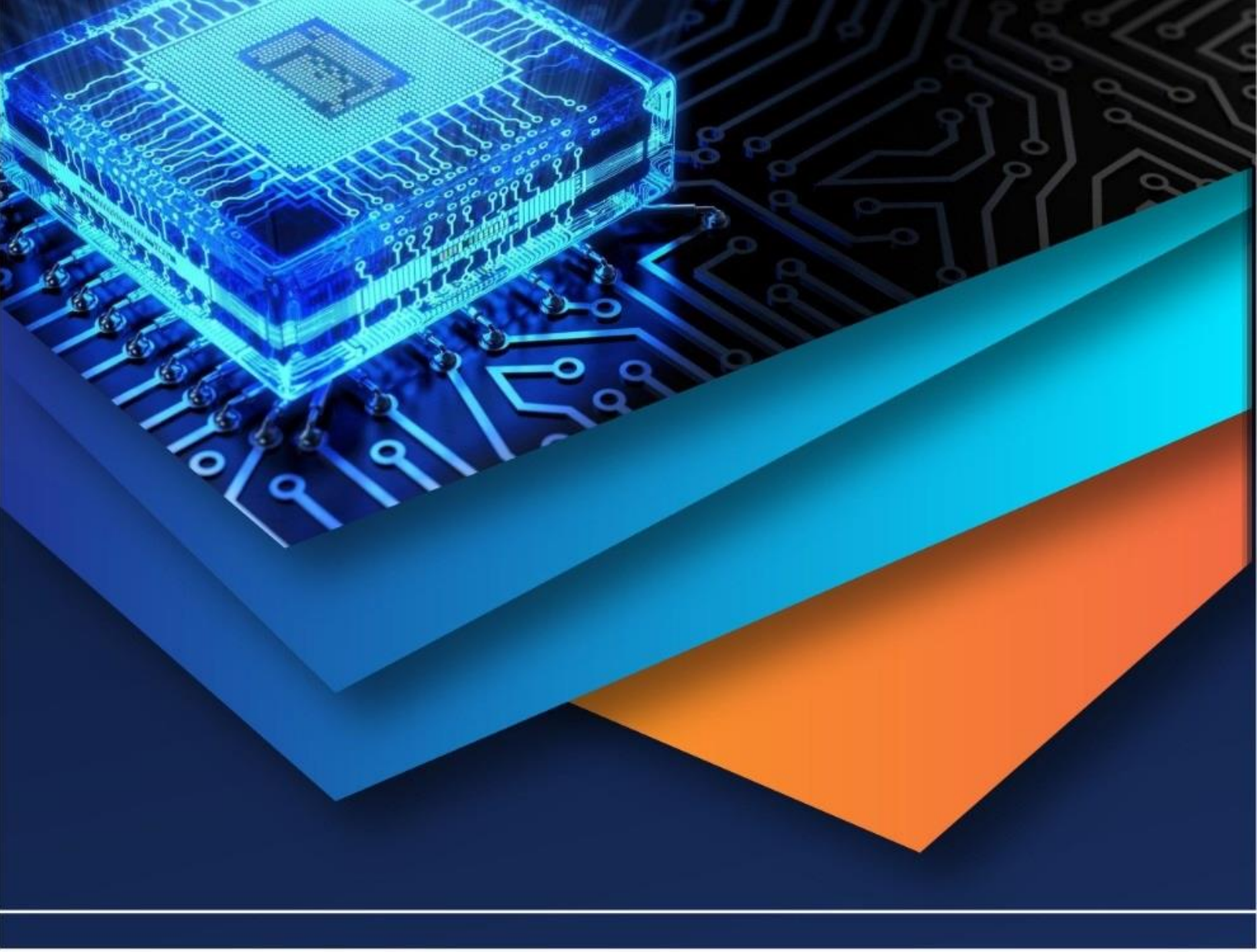
IX. ACKNOWLEDGMENT

The authors thank [GUIDE NAME] for guidance and support during this work.

REFERENCES

[1] Insurance Regulatory and Development Authority of India, *IRDAI (Protection of Policyholders' Interests) Regulations, 2017*. Hyderabad, India, 2017.
[2] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decis. Support Syst.*, vol. 50, no. 3, pp. 559–569, 2011.
[3] S. Viaene, R. A. Derrig, B. Baesens and G. Dedene, "A comparison of state-of-the-art classification techniques for expert automobile insurance claim fraud detection," *J. Risk Insur.*, vol. 69, no. 3, pp. 373–421, 2002.
[4] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, Curitiba, Brazil, 2007, pp. 629–633.
[5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4765–4774.
[8] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2000.
[9] Meta Open Source, "React." [Online]. Available: https://react.dev/
[10] S. Ramírez, "FastAPI." [Online]. Available: https://fastapi.tiangolo.com/
[11] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[12] PostgreSQL Global Development Group, *PostgreSQL Documentation*. [Online]. Available: https://www.postgresql.org/docs/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)