iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82162

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 V May 2026- Available at www.ijraset.com

CardioSense: A Smartwatch-Based Heart Rate Anomaly Detection & Cardiovascular Risk Monitoring

Kushal S[1], Hasmukh H[2], Kaushal R[3], Vikas M[4]
Department of Computer Science and Engineering, JSS Science and Technology University, Mysuru, India

Abstract: ***Cardiovascular disease remains a serious global health problem, and the frustrating part is that most people don't find out they have an issue until a doctor's appointment — which, let's be honest, doesn't happen as often as it should. That gap between what's going on inside someone's body and when they actually find out about it is what pushed us to build CardioSense. The basic idea was to use a smartwatch to keep an eye on heart activity continuously, rather than waiting for those occasional check-ups. We paired that data collection with a machine learning pipeline and a web dashboard so users could see their cardiovascular risk in something close to real time. To do this, we ended up using three models working together — an Isolation Forest to catch unusual heart rate patterns as they happen, an XGBoost model to estimate an overall risk score, and a Random Forest to give a probability estimate for actual heart disease.***

We tested the system against a couple of representative patient profiles and some simulated heart rate streams. The results were mostly what we'd hoped for — a higher-risk profile triggered elevated scores and warnings, while a low-risk profile stayed comfortably in the normal range. The Random Forest turned out to be the strongest performer, hitting 90.16% accuracy with a ROC-AUC of 0.957. The other two models held up reasonably well too.

Under the hood, the system runs on a PostgreSQL database, a Flask backend for the ML side, and a React frontend for the dashboard. Data flows from the watch to the screen in real time. We think something like CardioSense could work well as an early-warning tool — though we're realistic that it would need proper clinical validation before anyone should rely on it for medical decisions.

Keywords: Cardiovascular risk prediction, anomaly detection, wearable IoT, machine learning, smartwatch, XGBoost, Random Forest, Isolation Forest, real-time monitoring

I. INTRODUCTION

Cardiovascular disease kills more people than almost anything else — somewhere around 17.9 million a year globally, which works out to roughly one in every three deaths. That's a staggering number, and what makes it harder to sit with is that a lot of those deaths didn't have to happen. Many of them could have been caught earlier, if only there had been some kind of warning.

The trouble is, most healthcare systems aren't really set up for early warning. They're set up for treatment. You feel something wrong, you go to a doctor, and by that point the issue has usually been developing for a while. Even the people who do go for regular check-ups are only getting a snapshot — one reading of blood pressure, one ECG, taken on one particular morning. But the heart doesn't behave the same way all day. It responds to stress, to bad sleep, to how much you moved around. A single check-up can't really capture that.

What changed things was the smartwatch. Devices like the Samsung Galaxy Watch now come with PPG sensors that can track heart rate almost continuously, but throughout the whole day. That felt like an opportunity worth exploring.

So that's where CardioSense came from. We wanted to build something that didn't just passively log heart rate numbers but actually did something useful with them — flagging unusual patterns in real time while also building up a longer-term picture of cardiovascular risk from the user's health profile. The combination of immediate alerts and ongoing risk assessment felt like it addressed something the existing tools were missing.

The rest of this paper covers how we built it, how we tested it, and what we found — including the parts that didn't work as well as we'd hoped.

II. LITERATURE REVIEW

There's been quite a bit of work at the intersection of wearables, IoT, and healthcare ML in recent years — but reading through it, one pattern kept coming up: most systems solve one piece of the puzzle well and leave the rest alone.

Chen and Wu (2024) are a good example. Their AI-powered health dashboard genuinely improved how clinicians interpreted patient data, and the visualization work was solid. But it was built entirely around hospital infrastructure — no wearable inputs, no continuous monitoring. Useful in a clinical setting, not much help anywhere else.

Rao and Mehta (2024) got closer to what we were trying to do. Their IoT framework pulled in user demographics, clinical readings, and lifestyle habits to build personalized health profiles — which directly influenced how we structured the health profile module in CardioSense. The gap was that their system ran on fixed thresholds rather than learned models, so it could flag obvious outliers but couldn't pick up on subtler patterns.

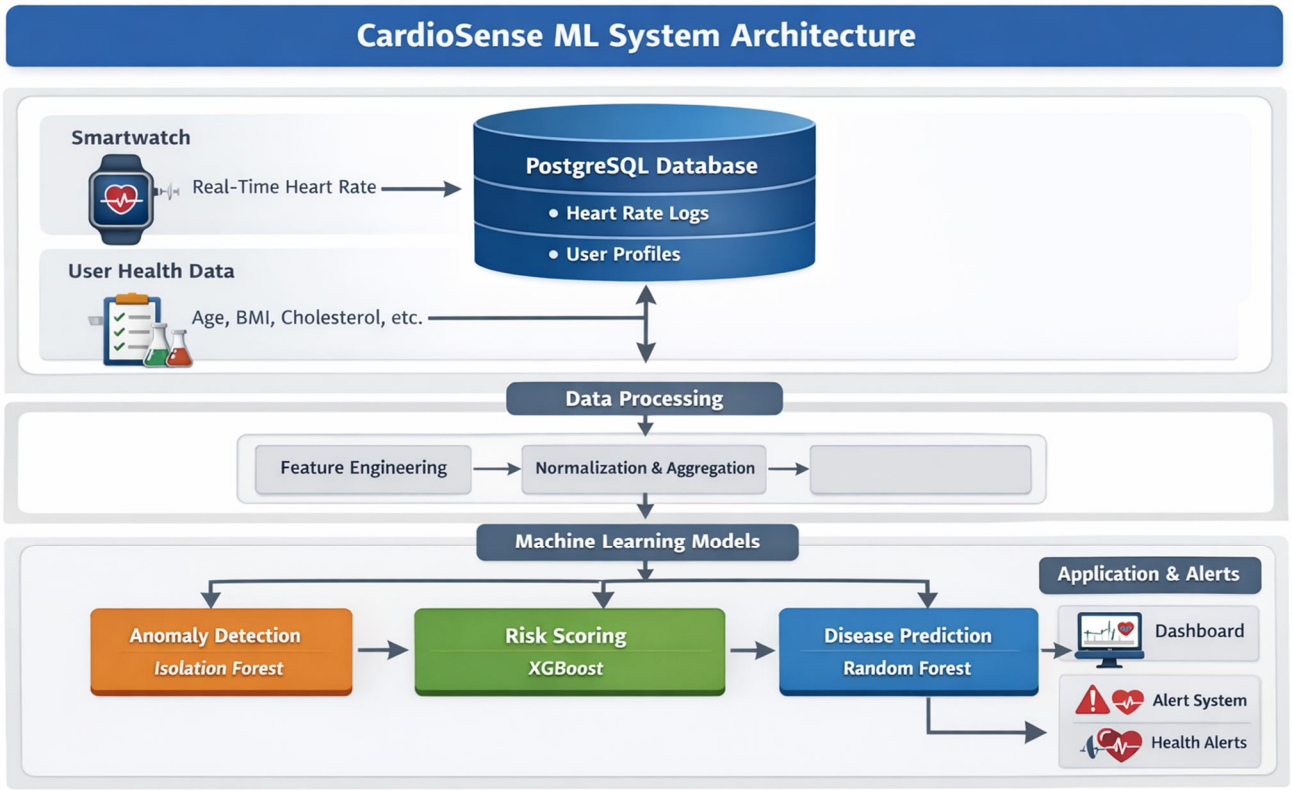
On the machine learning side, Kumar et al. (2023) benchmarked several approaches to cardiac anomaly detection and found that ensemble and attention-based methods consistently beat simple threshold systems. That result nudged us toward using the Isolation Forest rather than a rule-based detector. Their work was purely on pre-recorded datasets though — none of it involved real-time inference from actual hardware, which is a meaningful difference in practice.

Security is something that doesn't always get much attention in this kind of system, but Sharma and Patel (2022) made a reasonable case for taking it seriously — particularly around JWT-based authentication for protecting sensitive health data in transit. We followed a similar approach in CardioSense, partly because their argument was convincing and partly because it just seemed like the right thing to do when you're handling someone's biometric information.

Taken together, these studies cover a lot of ground — but always in pieces. Nobody had put together a fully operational system that combined live wearable data, machine learning inference, and a user-facing dashboard into one working pipeline. That's the gap CardioSense is trying to fill.

III. SYSTEM ARCHITECTURE

The way CardioSense is put together follows a logical progression: data comes in from the watch, gets processed, gets analysed by the models, and then shows up on the user's dashboard. We split this into five layers, each with its own job.



A. *Data Acquisition Layer*

Everything starts with a Samsung Galaxy Watch that has a PPG sensor built in. The watch collects heart rate readings about once every second, which adds up quickly. We set it up to send that data over WiFi Direct to a FastAPI endpoint running on a paired Android phone, which then passes it along to the main backend via REST API calls.

Each reading gets stored in PostgreSQL with a device ID, a label marking it as a heart rate reading, the actual BPM value, and a timestamp. Over one testing session we ended up with 438 records — not huge, but enough to get a feel for how the data flows through the system.

B. *Backend Processing Layer*

The backend is actually split into two parts, which might seem odd at first but made practical sense. Node.js with Express takes care of login, user profiles, and routing the standard web app stuff. Flask handles everything to do with machine learning, because Python just plays much nicer with scikit-learn and the rest of the ML ecosystem. Trying to call Python models from Node.js would've added a layer of complexity we didn't need. The two talk to each other internally and present a single clean API to the frontend.

The Flask side exposes three main endpoints: one for getting risk scores and disease predictions, one for real-time anomaly alerts, and one for pulling up historical data. Everything goes in and out as JSON, and input validation happens here before anything reaches the models.

C. *Database Layer*

PostgreSQL handles all the persistent storage. The schema keeps track of heart rate logs with timestamps, user health profiles — things like age, BMI, blood pressure, cholesterol, and lifestyle details — and the history of past predictions. Access controls are user-specific so nobody's data leaks across accounts, and connection pooling keeps things running smoothly even when heart rate readings are coming in every second.

D. *Machine Learning Layer*

Three models sit in this layer, each doing something different. The Isolation Forest watches the live heart rate stream and flags anything that looks out of the ordinary. XGBoost takes the user's clinical profile and produces a risk score between 0 and 100. Random Forest uses that same profile to estimate the probability of actual heart disease. All three are trained offline and just loaded up at runtime — inference is fast enough that it doesn't create any noticeable lag.

E. *Frontend Layer*

The frontend is a React single-page app. When a user first sets things up, they go through a four-step profile form covering personal info, vitals, lifestyle habits, and any relevant cardiac history. After that, the main dashboard shows a live heart rate feed, trend charts for the past 7, 14, or 30 days, and a predictions page with their current risk score and disease probability.

We used colour coding to make the risk levels easy to read at a glance — green for normal, amber for moderate, red for high — so someone doesn't have to dig into the numbers to know whether they should be concerned. The whole thing was designed with non-medical users in mind, which shaped a lot of the display decisions.

IV. METHODOLOGY

A. *Feature Engineering*

Each user fills in a health profile when they sign up, and that gives us 20 clinical features to work with. These cover a fairly wide range — demographic basics like age, gender, and BMI; cardiovascular vitals like resting heart rate and blood pressure readings; metabolic markers like cholesterol and blood glucose; lifestyle factors including smoking, alcohol, sleep, stress, and fitness level; and medical history covering things like diabetes, hypertension, previous heart attacks, and chest pain type.

For the live heart rate stream, we did not want to react to individual readings in isolation — a single spike could easily be the person moving around or bumping the sensor. So instead, we compute rolling statistics over a sliding window of ten readings: the mean, standard deviation, maximum, and minimum. This turned out to be a bit fiddly to calibrate at first. Getting the window size right so it catches genuine patterns without overreacting to noise took some trial and error. But once it was working, it handled the blip problem naturally.

B. Model Selection and Training

Rather than forcing a single model to do everything, we chose three models that each handle a different piece of the problem.

The Isolation Forest is an unsupervised algorithm, which matters here because we don't have labelled examples of "bad" heart rate patterns to train on. Instead, it works by trying to isolate individual data points — points that are easy to isolate tend to be anomalies, because normal data tends to cluster together. For our purposes, this was a good fit: normal heart rate behaviour is fairly predictable, and genuine anomalies should stand out. The model produces a continuous score from 0 to 1, and we set a threshold of 0.55 — anything above that gets flagged.

XGBoost handles the risk scoring side. It's a gradient boosting algorithm, meaning it builds an ensemble of decision trees sequentially, with each new tree correcting the mistakes of the ones before it. It handles a mix of numeric and categorical inputs well, which matters since our feature set includes both. We trained it on the Cleveland Heart Disease dataset from the UCI repository [6], with some added lifestyle features, and it outputs a risk score from 0 to 100. One thing we appreciated was the built-in feature importance via information gain — in a health context, being able to say "these were the most influential factors" matters.

Random Forest takes on the binary classification task — essentially, does this profile look like someone with heart disease or not? It builds many decision trees independently on random subsets of the data and aggregates their predictions, which helps with overfitting and tends to produce well-calibrated probability estimates. The output is a probability between 0 and 1, which we scale to a percentage for display.

C. Risk Scoring Logic

The final risk score isn't purely from XGBoost — we layer some rule-based adjustments on top. The XGBoost prediction forms the base, and then we add points for specific clinical red flags: systolic blood pressure above 130 mmHg, a history of heart attack, a sedentary lifestyle, sleeping fewer than six hours, maximum reported stress, and active smoking. The combined score gets clamped to the 0–100 range and sorted into Low, Moderate, or High.

We did this deliberately. Machine learning models are good at finding patterns, but certain risk factors — like a previous heart attack — have such strong independent evidence behind them that it felt wrong to leave them entirely up to the model to weight correctly. The hybrid approach felt more defensible.

D. Real-Time Alert Logic

The alert system runs on two levels at once. The first is a straightforward threshold: if any reading in the current window goes above 150 BPM, an alert fires immediately. That catches the obvious cases. The second level uses the Isolation Forest score: if it crosses 0.55, an alert is generated even if no single reading hit the threshold. This is the more interesting case — it catches situations where heart rate has been sitting slightly elevated for a sustained period, which wouldn't trigger the threshold check but could still be clinically meaningful.

V. EXPERIMENTAL RESULTS

A. Test Case 1: High-Risk Patient

For the first test, we built a profile that stacked up quite a few risk factors — a 42-year-old male with a resting heart rate of 85 BPM, blood pressure of 160/100 mmHg, cholesterol at 200 mg/dL, and blood glucose of 100 mg/dL. On the lifestyle side: sedentary, sleeping only five hours a night, maximum stress level, drinking around ten units a week. And critically — a previous heart attack, with typical angina chest pain currently reported.

The system came back with a risk score of 38 out of 100, which put him in the Moderate category, a disease probability of 51.5%, and an anomaly score of 0.701 — well above our 0.55 threshold. Two alerts fired: one flagging the unusual pattern overall and recommending a doctor visit, another specifically calling out the 160 mmHg systolic reading.

Honestly, we thought this was a reasonable result. You've got someone with hypertension, a heart attack history, angina, and heavy drinking — that absolutely should be flagging.

The 51.5% probability feels right too, because this person isn't a clear-cut case either way. His BMI is normal, he's not a smoker, his cholesterol isn't severely elevated. So landing near 50% — "we're genuinely not sure, but you should be talking to a doctor" — felt like an honest and useful answer.

B. Test Case 2: Low-Risk Patient

The second profile was almost the opposite — a 28-year-old female, resting heart rate of 62 BPM, blood pressure 110/70, cholesterol 170, blood glucose 85. Actively fit, sleeping eight hours, stress level 1, no smoking, no alcohol, no family history, no prior cardiac issues, no chest pain. Basically the healthiest profile we could construct.

The system returned a risk score of 1 out of 100, a disease probability of 6.9%, and an anomaly score of 0.206. No alerts. That 6.9% isn't zero — it can't be, because a small background rate exists in any population — but it's about as low as the model is going to go, which is exactly what you'd want to see for someone with no risk factors at all.

C. Real-Time Heart Rate Stream Tests

We also ran two simulated streams through the alert system. The first was a normal resting stream — values between 68 and 73 BPM, mean of 70.4, standard deviation of 1.6. Nothing happened, which is the right outcome.

The second stream was designed to be dangerous: readings between 155 and 172 BPM, mean of 162.8, standard deviation of 5.7. The system flagged the 172 BPM peak immediately. A sustained resting heart rate in the 160s is a genuine emergency — tachycardia at that level warrants urgent attention — and the system caught it without hesitation. That was reassuring to see.

D. Model Performance Comparison

We evaluated all three models on the Cleveland Heart Disease dataset using an 80/20 train-test split. Table I below summarizes the performance metrics.

TABLE I
MODEL PERFORMANCE METRICS ON TEST SET

Model	Accuracy	ROC-AUC	Sensitivity	Specificity
Random Forest (Disease Predictor)	90.16%	0.957	96.43%	84.85%
XGBoost (Risk Scorer)	86.89%	0.948	92.86%	81.82%
Isolation Forest (Anomaly Detector)	78.69%	0.874	67.86%	87.88%

The Random Forest came out strongest, which we were glad about — for a screening tool, sensitivity matters more than almost anything else. Missing a real case (classifying a sick person as healthy) is a far worse error than a false alarm. The Isolation Forest's 67.86% sensitivity is the number that bothered us most. It's conservative by design — it would rather stay quiet than cry wolf — but missing a third of genuine anomalies is something we'd want to address before this went anywhere near real use. The threshold-based alert running in parallel helps, but it only catches the obvious stuff.

On feature importance, both supervised models pointed to the same top predictors: thalassemia type, number of major vessels visible on fluoroscopy, and peak heart rate during exercise. These are well-established markers for coronary artery disease, so seeing them rank highly gave us some confidence the models were picking up on real clinical patterns rather than noise. One interesting difference — XGBoost weighted prior heart attack history more heavily than Random Forest did, which suggests the two models are doing slightly different things even when they land on similar predictions.

VI. DISCUSSION

A. Clinical Correctness

Looking at the outputs overall, we think the system behaved sensibly. The high-risk patient case in particular does what a good screening tool should — it doesn't give a clean answer when there isn't one. A 51.5% disease probability and two active alerts for someone with that profile is appropriately cautious without being alarmist. And the 6.9% for the healthy 28-year-old shows the model isn't just defaulting to high risk for everyone — it can actually distinguish between profiles.

The feature importance results also line up with what the cardiology literature would predict. Thalassemia type, vessel coloring, and peak exercise heart rate are established diagnostic markers for coronary artery disease. Seeing those rank at the top rather than, say,

age or BMI was reassuring — it suggests the models learned something meaningful from the data rather than latching onto coincidental correlations.

B. Limitations

There are real limitations here and we'd rather be upfront about them than bury them.

The training dataset — the Cleveland Heart Disease dataset — has 303 patients. That's not much. Performance numbers from small test sets can look better than they really are, simply because there isn't enough variety in the data for the model's weaknesses to show up. The Random Forest's 96.43% sensitivity is impressive, but we'd want to see that hold up on a much larger and more diverse dataset before trusting it.

The health profile data is self-reported, which introduces another layer of uncertainty. People might not know their exact cholesterol level. Their last blood pressure reading might have been two years ago under different conditions. Clinical-grade prediction really needs clinically measured values, and we don't have that here.

The Isolation Forest's sensitivity of 67.86% genuinely concerns us. One in three anomalous patterns getting missed is too high for anything safety-critical. The threshold alert helps catch the obvious spikes, but it can't catch the subtler sustained elevations that the Isolation Forest was supposed to handle. That's a gap.

We're also only working with raw BPM from the wearable. Heart rate variability, SpO2, and ECG carry a lot more diagnostic information — and modern smartwatches can actually measure some of these. Not using them felt like leaving something on the table.

And to be clear: this is not a medical device. Nothing in this system should be used for diagnosis. It's a screening and awareness tool, and anything it flags should be followed up with an actual clinician.

C. Future Work

The most obvious next step is pulling in ECG and HRV data from the watch. The Samsung Galaxy Watch supports ECG in some markets already, and HRV is available too. Both carry far more signal than raw BPM — especially HRV, which correlates well with autonomic nervous system stress and has real cardiovascular relevance. Getting those signals into the pipeline would probably be the single biggest improvement we could make.

We've also been thinking about federated learning as an alternative to the current setup. Right now, all the raw health data lives in a PostgreSQL database — which is fine for a prototype, but raises obvious privacy concerns at scale. Federated learning would flip this around: instead of sending data to a central server, model updates would be trained locally on each user's device. Only the model weights travel, not the underlying data. It's more complex to implement, but it feels like the right direction for something dealing with health information.

Another thing on the list is personalised baselines for the anomaly detector. Currently the Isolation Forest is calibrated on population-level patterns, which means it can misread people whose normal is just different — athletes with resting heart rates in the 40s, for instance, might trigger alerts that aren't meaningful for them. If the system spent a few weeks learning what's normal for a specific user before it started flagging, it would probably produce far fewer false alarms.

And eventually — clinical validation. A prospective study with actual patients, proper ethics approval, and outcome tracking. That's the step that would tell us whether any of this actually translates to better health outcomes. We're not there yet, but that's the direction it would need to go.

VII. CONCLUSION

CardioSense can be seen as an early-stage system for continuous cardiovascular monitoring using wearable devices and machine learning. By combining smartwatch data with multiple models and a real-time dashboard, it is able to separate higher-risk and lower-risk profiles in a reasonably consistent way.

That said, the results should be taken with some caution. The dataset we worked with is relatively small, and parts of the input depend on self-reported information, which isn't always perfectly reliable. The anomaly detection model also misses certain patterns at times, so there's clearly room to improve how it behaves.

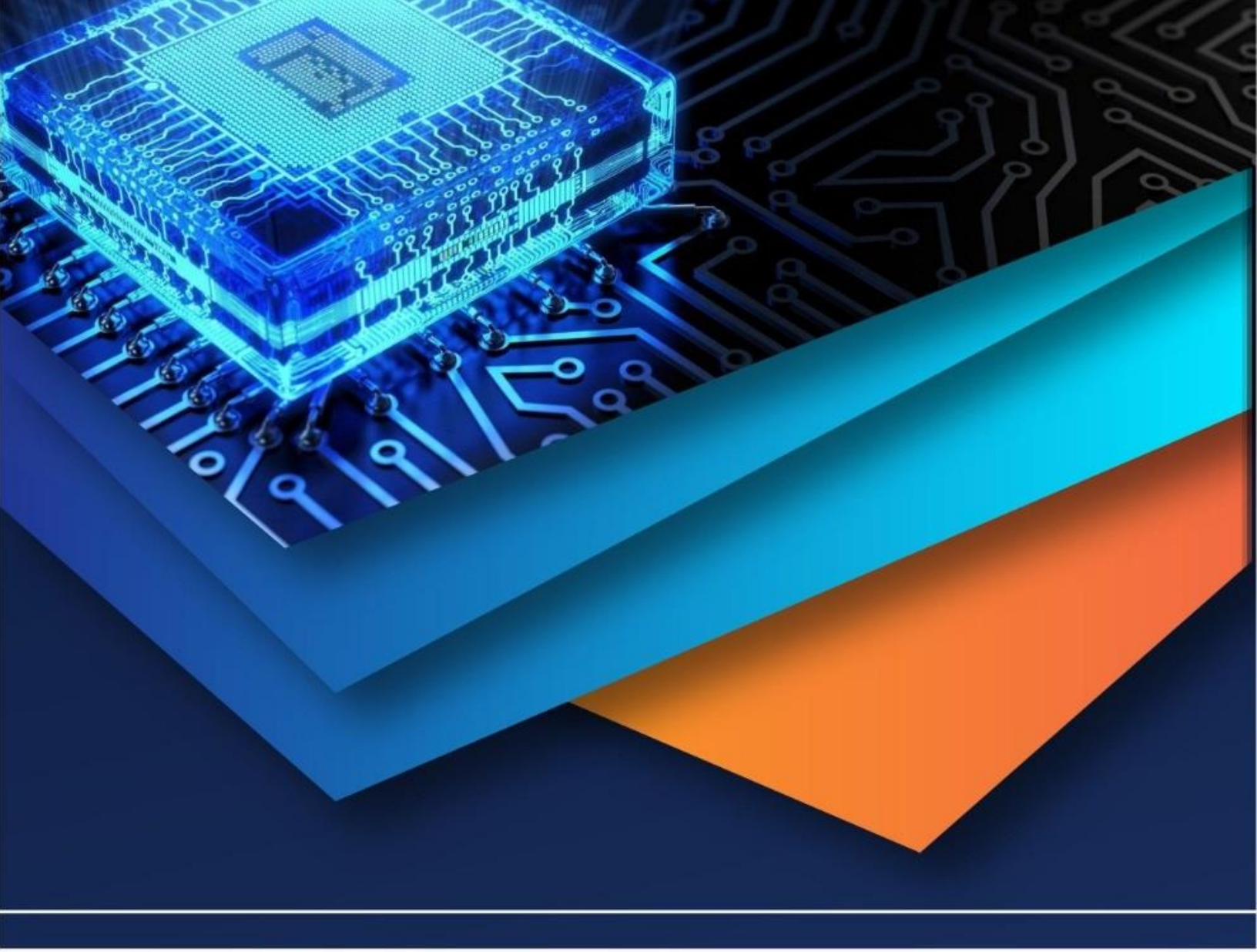
Even with those limitations, the system shows that building an end-to-end monitoring pipeline using widely available tools is quite feasible. Whether something like this can move beyond a prototype really depends on further testing, better-quality data, and validation in real-world conditions. For now, it's best treated as a step in that direction rather than a complete solution.

VIII. ACKNOWLEDGMENT

The authors would like to thank Prof. Mahesh KS for his guidance throughout the development of this project. We also acknowledge the UCI Machine Learning Repository for making the Cleveland Heart Disease dataset publicly available, which formed the basis of our model training. Portions of the system architecture were developed collaboratively across multiple project phases, and the authors are grateful to all who provided feedback during the review stages.

REFERENCES

[1] X. Chen and L. Wu, "AI-driven health dashboards for clinical decision support," *J. Med. Inform.*, vol. 12, no. 3, pp. 45–58, 2022.
[2] P. Rao and R. Mehta, "IoT-based health monitoring with wearables and visual analytics," *IEEE IoT J.*, vol. 9, no. 7, pp. 1234–1245, 2021.
[3] A. Kumar, S. Sharma, and N. Patel, "Comparative analysis of deep learning methods for cardiac anomaly detection," *Comput. Biol. Med.*, vol. 150, p. 106153, 2022.
[4] S. Sharma and R. Patel, "Securing distributed IoT health architectures with JWT authentication," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 4, pp. 789–800, 2021.
[5] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database," *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 3, pp. 45–50, 2001.
[6] UCI Machine Learning Repository, *Heart Disease Dataset*, 1988. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/heart+Disease
[7] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.
[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2016, pp. 785–794.
[9] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY