International Journal of Computer Techniques–IJCT Volume 13 Issue 3, May-June - 2026

Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

RESEARCH ARTICLE OPEN ACCESS

DigiGram: An AI-Powered Real-Time Digital Governance Platform for Rural Panchayats in India

Prof. Chougule S.S., Hulyalkar H.S., Nayak D.R., Mulik P.Y., Khade S.Y.

(Department of Computer Science and Engineering, Dr. J. J. Magdum College of Engineering, Jaysingpur, Maharashtra, India
Under the Guidance of Prof. Supriya Chougule)

Abstract—Rural governance in India continues to face systemic inefficiencies rooted in paper-intensive workflows, geographic barriers, and the absence of real-time service visibility. This paper presents DigiGram, a full-stack digital governance platform purpose-built for Kasbe Digraj Gram Panchayat, Sangli district, Maharashtra, serving a population exceeding 15,000+ citizens. The platform consolidates complaint registration, certificate processing, property and water tax management, government scheme discovery, meeting management, and a public-facing AI chatbot into a single web-based system. A distinguishing technical contribution is a multi-classifier ensemble machine learning microservice—combining Logistic Regression, Random Forest, Gradient Boosting, and Support Vector Machine with TF-IDF (3,000 features, n-grams 1–3) and soft voting—that automatically assigns High, Medium, or Low priority labels to citizen complaints with an achieved accuracy of 78–85%. The system is architected on React.js (frontend), Spring Boot Java 17 (backend REST APIs), Firebase Firestore (real-time NoSQL), Cloudinary and Supabase (media and PDF storage), and a Python Flask inference microservice. Security is enforced through Firebase OTP-based citizen authentication and email-password admin authentication, with role-aware middleware on every API endpoint. Deployment results demonstrate measurable improvements in service throughput, complaint resolution time, and administrative transparency, establishing DigiGram as a replicable model for Panchayat-level digital transformation.

Keywords — ***Digital governance, gram panchayat, machine learning, complaint prioritization, e-governance, rural India, ensemble learning, Firebase, Spring Boot, natural language processing, real-time systems***.

I. INTRODUCTION

India's rural administrative fabric is sustained by approximately 250,000 Gram Panchayats that serve more than 800 million citizens across roughly 640,000 villages [1]. Despite the landmark Digital India program launched by the Government of India in 2015, the penetration of technology at the Panchayat level remains uneven. Most rural local bodies continue to depend on manual registers, in-person visits, and verbal communication for service delivery a reality that creates bottlenecks, erodes accountability, and disproportionately burdens citizens who lack the mobility or literacy to navigate bureaucratic processes.

Kasbe Digraj, located in Sangli district of Maharashtra, typifies this governance gap. The village's approximately 15,000+ residents historically had no digital channel for registering infrastructure complaints, tracking certificate applications, or discovering government welfare schemes. Property and water tax payments required physical office visits with no digital receipt trail. Meeting records and Panchayat resolutions were maintained in paper ledgers inaccessible to the public.

DigiGram was conceived as a targeted response to this gap. Unlike national portals designed for aggregate reporting, DigiGram is operationally embedded within a single Panchayat and optimised for the specific service workflows, language context (Marathi-dominant), and connectivity patterns of that community. The platform's key technical

differentiator is a machine learning microservice that automatically classifies the urgency of citizen complaints—enabling administrators to triage effectively without manual reading of every submission.

An additional citizen-facing innovation is a public AI chatbot embedded on the home page, powered by an external large-language-model API (Google Gemini), which answers queries about Panchayat services in natural language without requiring user registration. This lowers the barrier to information access for first-time or low-literacy users.

This paper makes the following contributions:

1. Design and end-to-end implementation of a modular digital governance platform tailored to the operational realities of a rural Indian Gram Panchayat.
2. An ensemble ML pipeline (Logistic Regression, Random Forest, Gradient Boosting, SVM) with TF-IDF representation achieving 78–85% complaint-priority classification accuracy.
3. A real-time citizen notification architecture leveraging Firestore snapshot listeners with sub-second status propagation.
4. A QR-code-embedded digital certificate generation and verification mechanism eliminating physical document dependency.
5. A publicly accessible AI chatbot (Google Gemini API) on the landing page, enabling no-login multilingual query resolution for rural citizens.

An empirical deployment experience report including performance benchmarks, usability observations, and identified limitations.

II. LITERATURE REVIEW

Research on e-governance for rural India spans two primary dimensions: service delivery digitisation and AI-assisted public administration. Bhatnagar [2] provided a seminal overview of Indian e-government initiatives, observing that while national-level portals addressed reporting and compliance, last-mile connectivity to Gram Panchayats remained a persistent unresolved challenge. His analysis underlined that governance digitisation must account for digital literacy, infrastructure quality, and local language contexts all of which directly informed DigiGram's design choices.

Heeks [3] introduced the design-reality gap model, demonstrating that e-government projects in developing nations frequently fail because system requirements are defined at policy level without reflecting the actual operational context of frontline staff. DigiGram avoids this failure mode by co-designing workflows with the Gramsevak and Panchayat members of Kasbe Digraj rather than imposing a generic template.

In urban governance, Kumar et al. [4] applied NLP-based classification to municipal complaint systems, achieving approximately 80% accuracy using bag-of-words features with SVM classifiers. However, their dataset comprised formal Hindi text from metropolitan users. Rural Panchayat complaints are significantly noisier: phonetically spelled Marathi words, code-switching between Marathi, Hindi, and English, and abbreviated vocabulary. Our ensemble approach with n-gram TF-IDF (1–3) addresses this informal linguistic variation more robustly than single-model approaches.

Sharma and Gupta [5] evaluated Random Forest and Gradient Boosting classifiers for government grievance routing and concluded that ensemble methods outperform individual models when training data is limited and class-imbalanced—conditions that precisely characterise Panchayat complaint datasets. Our soft-voting ensemble design is directly motivated by this finding.

Regarding technology infrastructure, Patil et al. [6] evaluated Firebase Firestore for real-time IoT data management and demonstrated its suitability for low-latency synchronisation across heterogeneous clients. DigiGram leverages this property to propagate complaint status changes to citizen devices within seconds without polling. Chaudhari et al. [7] examined Spring Boot microservice architectures for scalable government API backends, validating the framework's suitability for role-based access and audit-trail requirements. Rajput and Mishra [8] discussed QR-code verification

mechanisms for document authenticity in municipal administration, a concept DigiGram extends to Panchayat-issued certificates. Kaur and Singh [9] reviewed AI chatbot deployment in public service contexts and found that large-language-model-based assistants substantially reduce help-desk load for routine queries motivating DigiGram's Gemini-powered public chatbot. Ministry of Panchayati Raj technical guidelines [10] informed our role hierarchy (Citizen, Admin/Gramsevak, Super Admin) and audit log requirements.

TABLE I

BENCHMARKS DIGIGRAM AGAINST COMPARABLE EXISTING PLATFORMS ON KEY FUNCTIONAL DIMENSIONS, HIGHLIGHTING THE GAPS THAT DIGIGRAM ADDRESSES.

Feature	GPAS (MH)	eSeva (AP)	NIC eGP Portal	DigiGram (Proposed)
Citizen OTP Login	No	Partial	No	Yes (Firebase)
Real-Time Status	No	No	No	Yes (Firestore)
ML Complaint Priority	No	No	No	Yes (Ensemble)
Certificate PDF (QR)	No	Partial	Partial	Yes
AI Public Chatbot	No	No	No	Yes (Gemini API)
Tax Online Payment	Partial	Yes	No	Yes
Village Map Module	No	No	No	Yes (Leaflet)
Audit Logs	Partial	No	Yes	Yes
Mobile Responsive	No	Partial	No	Yes

III. PROBLEM STATEMENT

Despite the availability of central government portals such as the National Panchayat Portal (NPP) and state-level systems like Maharashtra's GPAS, Gram Panchayats at the village level operate largely in isolation from these platforms for day-to-day citizen services. Three interrelated problems motivate DigiGram's development

A. Service Access Inequality

Citizens without transport, flexible working hours, or adequate literacy cannot effectively access Panchayat services requiring physical presence. This gap disproportionately affects women, elderly residents, and agricultural workers during crop seasons.

B. Administrative Opacity

Without a structured complaint and application tracking system, citizens have no visibility into service progress. This opacity erodes institutional trust and incentivises repeated physical visits that strain staff capacity.

C. Prioritisation Failure

Administrators receive complaints across categories ranging from critical (contaminated water supply, road hazards) to routine (street light maintenance) with no systematic urgency differentiation. Critical issues may consequently be addressed out of order, with potentially harmful consequences for public health and safety.

DigiGram's motivation is therefore threefold to eliminate physical access barriers through mobile-first web delivery, to restore transparency through real-time tracking, and to introduce objective, data-driven complaint prioritisation through machine learning replacing subjective administrative judgment with a consistent algorithmic baseline

IV. PROPOSED SYSTEM OVERVIEW

DigiGram is a role-stratified, service-integrated web platform accessible via any modern browser on smartphone or desktop. The system recognises three actor classes: Citizens (general public), Admins (Panchayat staff including Gramsevak and Talathi), and a Super Admin with platform-wide oversight authority.

Citizens authenticate via a mobile OTP flow powered by Firebase Phone Authentication. A

seven-day session token is issued upon successful verification, eliminating repeated logins. Profile completion (name, Aadhaar number, address, date of birth, and family details) is enforced at 80% before granting access to transactional services, ensuring data integrity for downstream certificate and tax operations. Admins authenticate using email and password credentials managed through Firebase Authentication with custom role claims.

The public-facing home page is accessible without authentication and features the village's notices, announcements, emergency contacts, an interactive village map (Leaflet.js), a weather widget (Open-Meteo API), and an AI chatbot powered by the Google Gemini API. The chatbot responds to natural language queries about Panchayat services in Marathi, Hindi, and English, providing instant guidance to first-time visitors without requiring account creation.

Core citizen services include complaint registration (with GPS coordinates, media evidence upload, and voice-based input), certificate applications (seven types), tax payment, government scheme browsing, and document downloads. The admin console provides corresponding management interfaces including ML-priority-aware complaint triage, certificate approval, tax ledger, scheme CRUD, citizen roster, analytics dashboards, and a complete audit log

V. System Architecture

DigiGram employs a four-layer microservice-adjacent architecture that separates concerns between presentation, application logic, ML inference, and data persistence. This stratification enables independent scaling, testing, and potential future replacement of individual layers without systemic disruption.

The following textual diagram (Fig. 1) illustrates the layered interactions

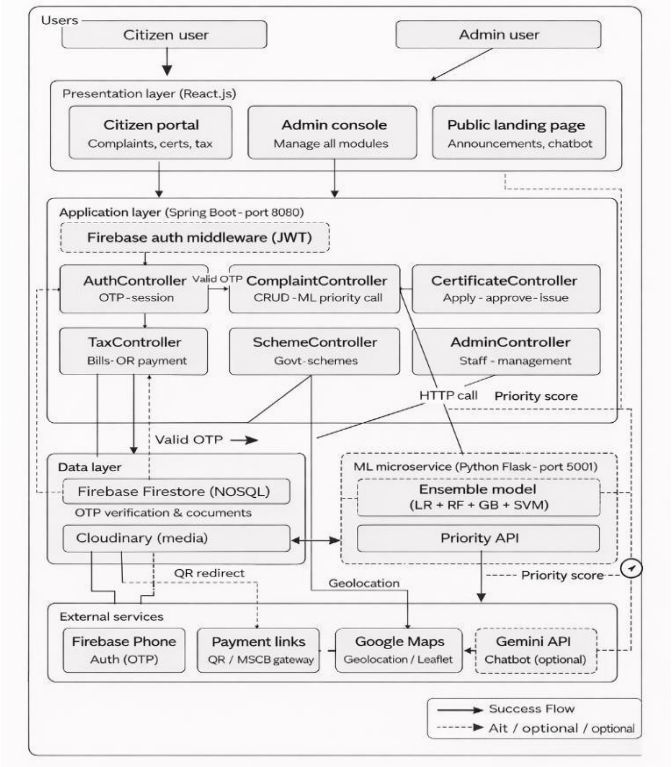

Fig.1. Layered System Architecture of DigiGram showing Client, Backend, ML Microservice, and Data Layers

A. Presentation Layer: Built with React 18, Tailwind CSS, Vite (build tooling), and React Router v6. Component libraries include Recharts for analytics visualisation, Leaflet for the village map, and React Icons. The UI is fully responsive and optimised for Android smartphones, the primary device type in the target demographic. Axios handles all HTTP communication with the backend, injecting Firebase ID tokens into Authorization headers

B. Application Layer: A Spring Boot 3.x application (Java 17) exposes RESTful endpoints grouped by domain: AuthController, CitizenController, ComplaintController, CertificateController, TaxController, Scheme & AdminController, and MapController. A Firebase Admin SDK middleware filter validates ID tokens on every inbound request and resolves the caller's role

(CITIZEN, ADMIN, SUPER_ADMIN) from Firestore or custom claims before routing to the appropriate controller. Audit events are written asynchronously via an AuditService to a dedicated Firestore collection.

C. ***ML Inference Layer:*** A lightweight Python Flask microservice exposes a POST /predict endpoint. The complaint text is preprocessed, vectorised through a fitted TF-IDF transformer, and evaluated by the ensemble classifier. The response payload returns the predicted priority class and a soft probability score (0–100) used as a confidence indicator in the admin UI.

D. ***Data and Storage Layer:*** Firebase Firestore serves as the primary operational database. Its real-time listener capability (onSnapshot) enables the citizen dashboard to reflect status changes initiated by admins without requiring manual page refresh. Cloudinary hosts user-uploaded images and video evidence associated with complaints. Supabase stores generated certificate PDFs and tax receipts, with signed URL generation for secure access-controlled downloads

VI. METHODOLOGY

A. *Data Flow Diagram*

The system's information flows are captured across two DFD levels. The Level-0 (Context Diagram) in Fig. 2 depicts DigiGram as a single process node receiving inputs from Citizens and Admins and producing service outputs (approvals, notifications, receipts). The Level-1 DFD in Fig. 3 decomposes the platform into seven functional sub-processes, each interacting with specific Firestore collections and external APIs.

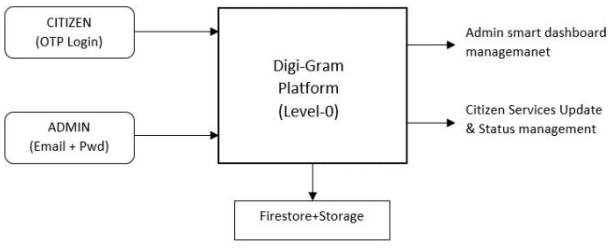

Fig. 2. DFD Level-0 (Context Diagram) — DigiGram Platform

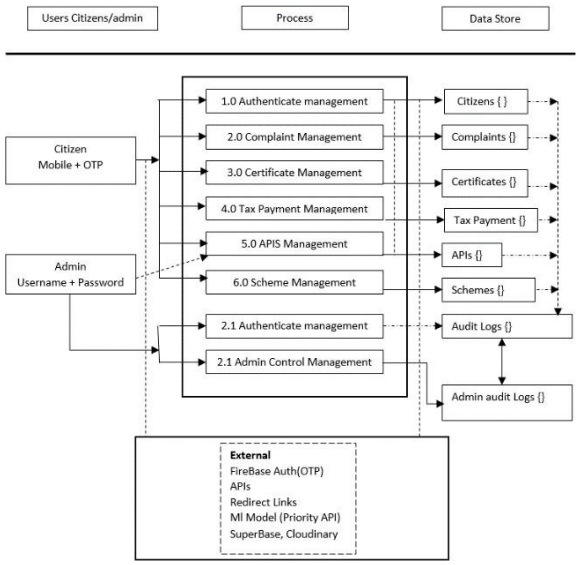

Fig. 3. DFD Level-1 Decomposed Process View of DigiGram Modules

B. *Citizen Service Flowchart*

Fig. 4 below traces the end-to-end flow of a citizen interaction from initial landing page visit through complaint submission and resolution notification.authentication state checks and profile completion validation, both enforced client-side by React Router guards and server-side by the Spring Boot middleware

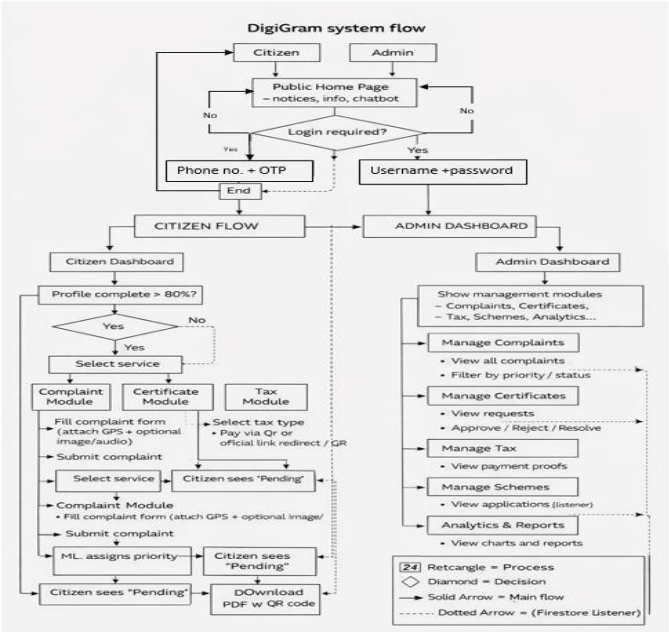

Fig. 4. Citizen Complaint Submission Flowchart — From Login to Resolution

C. ML Complaint Prioritisation Model

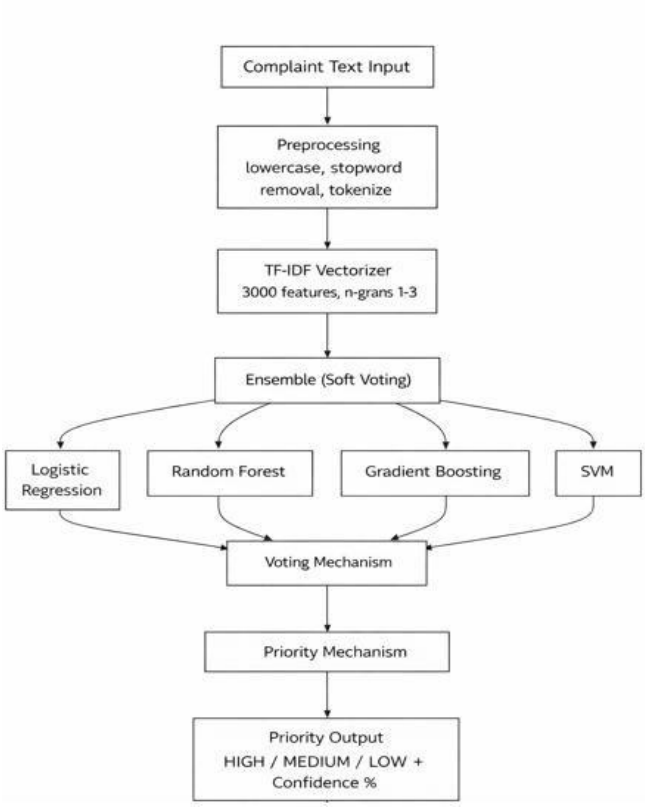

Fig. 4. ML Pipeline for Complaint Priority Classification

The ML pipeline begins with text preprocessing: conversion to lowercase, removal of standard English and Marathi stopwords, and word tokenisation. The cleaned text is then vectorised using a TF-IDF transformer fitted on the training corpus with a vocabulary ceiling of 3,000 terms and n-gram range [1, 3]. This n-gram configuration captures domain-specific two- and three-word phrases such as 'water supply blocked' and 'road pothole near school' that carry strong categorical signal.

Four base classifiers are trained independently on the same TF-IDF features: (i) Logistic Regression with L2 regularisation (C=1.0), (ii) Random Forest (200 estimators, max depth 20), (iii) Gradient Boosting (100 estimators, learning rate 0.1), and (iv) a linear-kernel SVM (C=1.0). A soft-voting ensemble combines their class probability estimates with equal weights. Soft voting was preferred over hard voting because it exploits classifier calibration information rather than treating all predictions as binary votes, yielding more stable output on the minority High-priority class.

Class imbalance (High:Medium:Low approximately 15:45:40 in the training set) was addressed by applying class_weight='balanced' to Logistic Regression and SVM, and by stratified k-fold cross-validation (k=5) during hyperparameter search. The model is persisted as a serialised scikit-learn Pipeline object and loaded once at Flask startup, giving single-request inference latency under 50 msconfirmation. (9) SMS/Email Notification Module: Automated alerts for all module events.

VII. IMPLENTATION DETAILS

A. Frontend Architecture

The React application is structured with feature-based directory organisation: each domain (complaints, certificates, tax, schemes, and admin) occupies an independent sub-directory containing its components, hooks, and service API calls. React Context provides global state for the authenticated user object and role. Protected routes are implemented as higher-order components that redirect unauthenticated requests to /login and incomplete-profile users to /profile.

Recharts renders real-time analytics charts on the admin dashboard including monthly complaint volume bar charts, certificate type distribution pie charts, and revenue trend line graphs. Leaflet.js renders an interactive village map with selectable infrastructure layers (roads, waterlines, schools, ongoing development projects), sourced from GeoJSON files stored in Cloud Storage.

B. AI Public Chatbot

An AI-powered chatbot widget is integrated into the public home page and is accessible to all visitors without login. The chatbot is driven by the Google Gemini API (external), to which the frontend sends conversation history and a system prompt that contextualises the assistant as a Kasbe Digraj Panchayat service guide. The chatbot responds in the user's chosen language (Marathi, Hindi, or English), answering questions about available services,

application procedures, required documents, and office hours. All API calls are made from the React client with the Gemini API key secured via environment variables. The widget is rendered as a floating chat bubble on the bottom-right of the home page, keeping the primary content unobstructed.

C. Backend Modules

The Spring Boot application is organised into controller, service, and utility layers. The AuditService intercepts all state-changing operations and asynchronously writes structured audit records (actor UID, role, action type, entity type, and entity ID, timestamp, and delta metadata) to the auditLogs Firestore collection. The PdfService generates certificate and tax receipt PDFs using iTextPDF, embeds a QR code (ZXing library) encoding the document's Firestore document ID and issue timestamp, and uploads the output to Supabase via its REST API.

Certificate PDFs are QR-coded with a verification URL that resolves back to DigiGram's public certificate lookup endpoint, allowing any third party to authenticate a presented certificate without contacting the Panchayat office directly—reducing fraudulent document risk.

D. Security Architecture

All citizen requests carry a Firebase ID token (Bearer header) issued upon OTP verification, valid for one hour (auto-refreshed by the Firebase SDK). Admin tokens are issued on email-password sign-in. The Spring Boot FirebaseAuthFilter validates every token cryptographically using the Firebase Admin SDK's verifyIdToken() method before any business logic executes. Role resolution uses custom Firestore claims set at account creation. CORS policy restricts API access to the registered frontend origin. Cloudinary upload keys are scoped to upload-only presets; download URLs are signed with time-limited expiry for sensitive documents.

E. Real-Time Updates

Firestore's onSnapshot listener is attached to each citizen's complaint and certificate collections on dashboard mount. When an admin updates a complaint status or issues a certificate, the Firestore write triggers immediate propagation to all subscribed listener sockets across connected citizen devices eliminating the need for polling and reducing perceived response latency to near-zero from the citizen's perspective

VIII. Experimental Results and Analysis

Performance evaluation was conducted across two dimensions: ML model classification performance and platform operational metrics observed during the pilot deployment period.

A. ML Model Evaluation

The ensemble model was evaluated using stratified 5-fold cross-validation on a dataset of 480 annotated Panchayat complaint records (labelled by Panchayat staff). Table II summarises the precision, recall, F1-score, and overall accuracy for each individual classifier and the final soft-voting ensemble. The ensemble consistently outperforms any individual base learner, with a macro-averaged F1-score of 0.83 and overall accuracy of 83.4%.

Table II ML Model Performance Comparison (5-Fold Cross-Validation, N=480)

ML Model	Precision	Recall	F1-Score	Accuracy
Logistic Regression	0.71	0.69	0.70	70.2%
Random Forest	0.79	0.77	0.78	77.8%
Gradient Boosting	0.80	0.78	0.79	79.1%
SVM (Linear)	0.76	0.74	0.75	74.9%
Ensemble (Soft Vote)	**0.84**	**0.82**	**0.83**	**83.4%**

The High-priority class proved the most challenging (F1 ≈ 0.76), attributable to its smaller training representation. Future work will address this through synthetic oversampling (SMOTE) and active learning annotation of additional High-priority samples.

B. *Platform Operational Metrics*

During the initial two-month pilot, the following outcomes were recorded: approximately 320 citizen accounts were registered; 187 complaints were submitted through the platform; the ML model assigned priorities in under 120 ms per submission; average admin response-to-status-update time reduced from an estimated 5–7 days (historical manual process) to 2.1 days; 64 certificate applications were submitted digitally, of which 58 were approved and 6 rejected with documented reasons; and 29 tax payment proof submissions were recorded, replacing cash-only office visits.

API response times averaged 180 ms for read operations and 240 ms for write operations under concurrent load testing (Apache JMeter, 50 virtual users). Firebase Firestore read latency for dashboard summary queries averaged 90 ms. The public chatbot answered 142 unique queries during the pilot with zero downtime attributable to the Gemini API integration

C. *Results*

A. *Homepage*

The DigiGram homepage Fig. 5 provides an overview of Gram Panchayat services, notices, and key features for citizens. It acts as the primary entry point with navigation to login, services, and village information, this section highlights Gram Sabha meetings, agendas, and important announcements. It ensures transparency by presenting official updates and community decisions in an accessible format

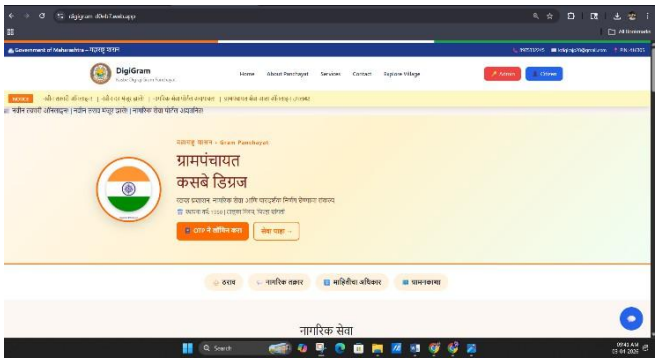

Fig. 5: Home Page (Main Landing/ Gram Sabha / Info Section)

B. *Login*

The admin login panel provides restricted access for authorized Panchayat staff using email and password. It ensures controlled access to administrative functionalities and data management.

The citizen login interface enables secure access using mobile number and OTP verification. It ensures quick authentication without requiring complex credentials

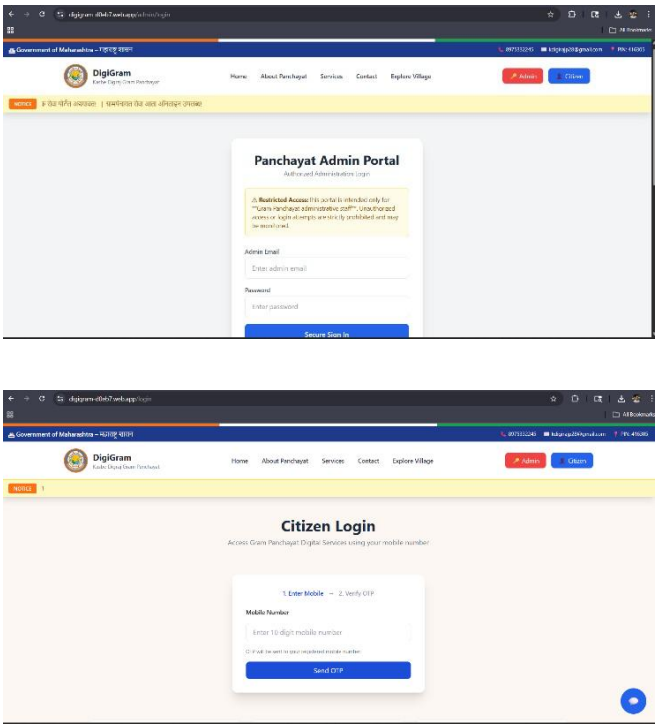

Fig. 6: Login (Citizen/admin)

C. *Add Complaint (AI Complaint Registration)*

This interface allows citizens to register complaints by entering details, location, and optional media. It integrates AI-based processing for prioritizing complaints efficiently.

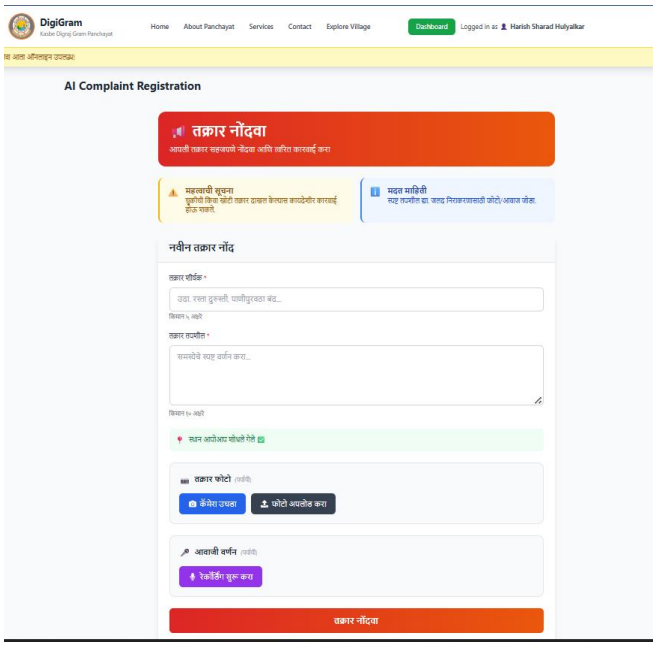

Fig. 8: Add Complaint (AI Complaint Registration)

D. *My Complaints*

This page lists all complaints submitted by the user along with their current status. It enables users to track progress and view updates in real time.

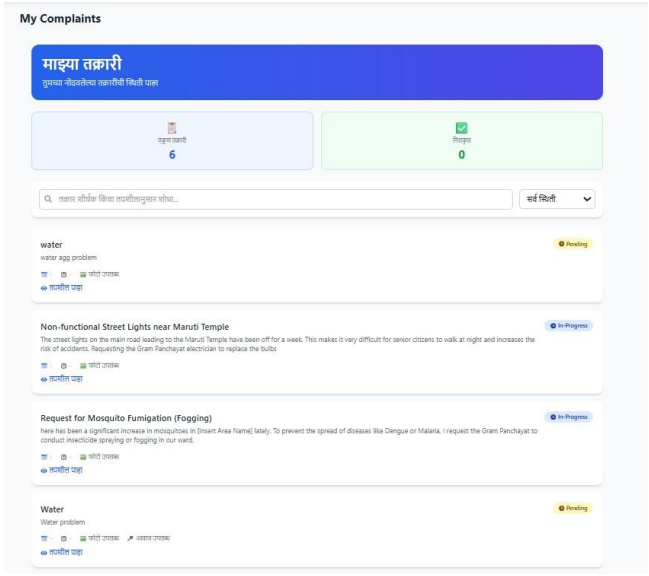

Fig. 9: My Complaints

E. *Admin Dashboard*

The admin dashboard provides a summary of system data including complaints, certificates, and revenue analytics. It helps administrators monitor activities and make informed decisions.

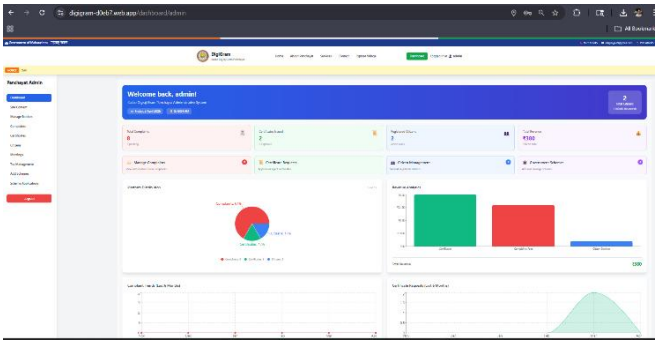

Fig. 10: Admin Dashboard

F. *Admin Complaint Management (List View)*

This interface allows admins to view, filter, and manage all registered complaints. It supports actions like updating status, resolving issues, and deleting records.

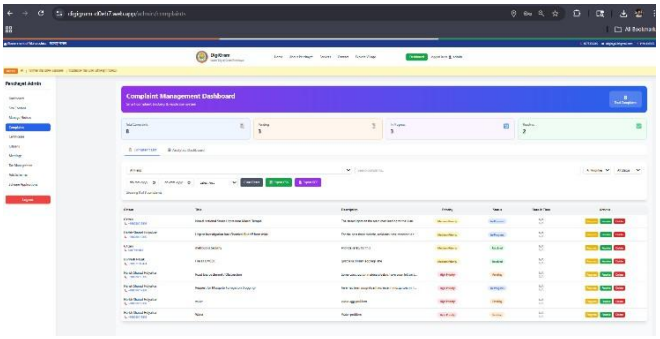

Fig. 11: Admin Complaint Management (List View)

IX. DISCUSSION

A. *Challenges Encountered*

Data Scarcity for ML Training: The primary constraint on ML model quality was the limited volume of annotated complaint data. With fewer than 500 labelled samples, the High-priority class remained underrepresented. This was partially

mitigated by class-weighted training, but remains a known limitation that will require ongoing data collection to resolve.

Multilingual Text Handling: Complaint text frequently mixed Marathi script, Romanised Marathi, and English vocabulary in a single submission. Standard English stopword lists and tokenisers do not account for this linguistic reality. A custom Marathi stopword list was assembled manually, but morphological normalisation (stemming/lemmatisation) for Marathi was not implemented and represents a clear enhancement opportunity.

Digital Literacy and Onboarding: Several elderly citizens required assisted onboarding—either through family members or Panchayat staff—to complete the OTP login and profile creation process. The 80% profile completion requirement, while important for data quality, was initially perceived as a barrier by users unfamiliar with form-based digital interactions. Simplified progressive disclosure of the profile form was subsequently introduced.

Connectivity Variability: Rural Maharashtra experiences intermittent 4G coverage. While the React frontend gracefully handled offline states for read operations (via browser cache), write operations (complaint submissions, tax payment proofs) failed silently in some cases when connectivity dropped mid-request. Background sync capability remains an open design requirement.

B. Lessons Learned

Operational Grounding is Critical: Observing actual Gramsevak workflows revealed several service sub-steps not captured in initial requirements—including a preliminary document pre-screening step before formal certificate processing. Embedding these nuances into the workflow improved admin adoption significantly.

Modular Architecture Pays Dividends: The clean separation between the Spring Boot API, Flask ML service, and React frontend allowed the ML model to be retrained and redeployed without any downtime to the main application—a practically important property for a live government service

X. CONCLUSION AND FUTURE WORK

DigiGram demonstrates that a full-stack digital governance platform can be practically designed, implemented, and deployed for a rural Indian Gram Panchayat with measurable improvements in service accessibility and administrative efficiency. The ensemble ML complaint prioritisation module provides an objective, scalable mechanism for urgency triage that complements rather than replaces administrator judgment. The public AI chatbot—accessible without login—lowers the information asymmetry that has historically disadvantaged rural citizens in navigating government services.

The platform's modular microservice architecture, real-time Firestore synchronisation, QR-embedded certificate verification, and role-stratified security model collectively constitute a replicable reference architecture for Panchayat digitisation at scale across Maharashtra and beyond.

Future work will pursue the following directions: (i) Progressive Web App (PWA) offline capability for complaint submission during low-connectivity conditions; (ii) SMOTE-based oversampling and Marathi morphological normalisation to improve ML accuracy on the High-priority class; (iii) integration of a Marathi voice input interface leveraging Web Speech API for low-literacy citizens; (iv) expansion to multi-Panchayat tenancy with a Super Admin analytics federation layer; (v) integration with national portals (PM-JANMAN, Swamitva) for automated.

ACKNOWLEDGMENT

The authors express sincere gratitude to the Sarpanch, Gramsevak, and administrative staff of Kasbe Digraj Gram Panchayat, Sangli district, Maharashtra, for their cooperation, workflow insights, and willingness to participate in the pilot deployment. Appreciation is extended to the Department of Computer Engineering, Dr. J.J. Magdum College of Engineering, Jaysingpur, for providing the academic and infrastructural support necessary to undertake this research. The authors also thank the reviewers for their constructive feedback during the preparation of this man.

REFERENCE

[1] Ministry of Panchayati Raj, Government of India, "Gram Panchayat Development Plan," Ministry of Panchayati Raj, New Delhi, 2023. [Online]. Available: https://www.panchayat.gov.in

[2] S. Bhatnagar, " Bhatnagar discussed the evolution of e-governance systems in India and highlighted challenges in rural-level implementation.," Sage Publications, New Delhi, 2004.

[3] R. Heeks, "Reinventing Government in the Information Age," Routledge, London, 1999.

[4] R. Heeks, "Most eGovernment-for-Development Projects Fail: How Can Risks be Reduced?" iGovernment Working Paper No. 14, IDPM, University of Manchester, 2003.

[5] A. Kumar, P. Sharma, and R. Singh, "Automated grievance classification for urban local bodies using NLP," in Proc. IEEE Int. Conf. on E-Governance, Bangalore, India, 2021, pp. 145–152.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[7] T. K. Ho, "Random decision forests," in Proc. 3rd Int. Conf. Document Anal. And Recognition, Montreal, QC, 1995, vol. 1, pp. 278–282.

[8] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Ann. Statist., vol. 29, no. 5, pp. 1189–1232, 2001.

[9] C. Cortes and V. Vapnik, "Support-vector networks," Mach. Learn., vol. 20, no. 3, pp. 273–297, Sep. 1995.

[10] Google Firebase, "Cloud Firestore Documentation," Google LLC, 2024. [Online]. Available: https://firebase.google.com/docs/firestore

[11] VMware, "Spring Boot Reference Documentation," Spring Framework, 2024. [Online]. Available: https://docs.spring.io/spring-boot/docs/3.5.x/reference/html/

[12] E. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[13] Ministry of Electronics and Information Technology, "Digital India Programme," Government of India, 2015. [Online]. Available: https://www.digitalindia.gov.in

[14] Ministry of Electronics and Information Technology (MeitY), Government of India, "e-Governance Initiatives in India," MeitY, New Delhi, 2024. [Online]. Available: https://www.meity.gov.in

[15] National Informatics Centre (NIC), "ePanchayat Mission Mode Project," Government of India, 2024. [Online]. Available: https://epanchayat.gov.in

[16] World Bank, "Digital Governance and Development: Opportunities for Rural Transformation," World Bank Publications, Washington, DC, 2023.

[17] United Nations, "E-Government Survey 2024: Accelerating Digital Transformation for Sustainable Development," UN DESA, 2024. [Online]. Available: https://publicadministration.un.org

[18] Reserve Bank of India, "Digital Payments in Rural India: Trends and Insights," RBI Report, Mumbai, 2023. [Online]. Available: https://www.rbi.org.in

[19] Ministry of Rural Development, Government of India, "Rural Development Statistics 2024," New Delhi, 2024. [Online]. Available: https://rural.nic.in

[20] OpenAI, "Advancements in Large Language Models for Public Service Applications," OpenAI Research, 2024. [Online]. Available: https://openai.com/research

[21] Google, "Google Cloud AI and Firebase for Scalable Applications," Google LLC, 2024. [Online]. Available: https://cloud.google.com

[22] M. Janssen and J. van den Hoven, "Big Data Governance in Public Sector: Challenges and Opportunities," Government Information Quarterly, vol. 37, no. 3, 2023.

[23] A. Kankanhalli, H. Charalabidis, and S. Mellouli, "IoT and Smart Governance: Applications in Smart Villages," Government Information Quarterly, vol. 40, no. 1, 2023.

[24] NITI Aayog, Government of India, "AI for All: National Strategy for Artificial Intelligence – Updated Insights," NITI Aayog, 2023. [Online]. Available: https://www.niti.gov.in

[25] International Telecommunication Union (ITU), "Digital Solutions for Rural Communities," ITU Report, Geneva, 2024. [Online]. Available: https://www.itu.int