E-GOVERNANCE

SmartGram: An Integrated Digital Governance Framework for Gram Panchayat Services

Sakshi Sargar, Samiksha Mali, Janhavi Gavali, Ganesh Kande, N. J. Bhalkikar

Department of Computer Science and Engineering, Sanjay Ghodawat Institute, Atigre, Kolhapur, Maharashtra, India

Published November 2026

Abstract

Gram Panchayats keep citizen records, issue certificates, collect local taxes and handle complaints, but these services are usually supported by separate registers, spreadsheets and applications. This paper presents SmartGram, an integrated digital governance framework that organises Gram Panchayat services as modules of one system with a shared backend and database. A review of work published in 2025 and 2026 identifies six gaps: thin outcome evidence, unverifiable complaints, weak payment-to-bill linkage, rural access and equity, security and access control, and office-level workflows. The proposed design uses a React and TypeScript front end, a Node.js and Express backend, Prisma ORM and a MySQL database. It comprises a four-layer architecture, a logical data model with ten core entities, four algorithms for authentication, complaint registration with live photograph and permission-based location capture, tax calculation and payment verification, a status workflow with permitted transitions, and role-based dashboards. The framework is partially implemented; a requirement-to-test traceability table, representative test cases and evaluation metrics are defined so that outcomes can be measured during a pilot deployment.

Keywords: E-governance; Gram Panchayat; digital governance; rural development; complaint management; role-based access control; decision support

1 Introduction

Gram Panchayats are the lowest tier of local government in rural India. They keep citizen records, issue certificates, collect local taxes, take in complaints about water, roads and sanitation, and carry out government programmes at village level. For most residents the Panchayat office is the place where public services actually reach them.

Digital tools have made some of these services easier to reach, but they have rarely been joined up. One office may hold property data in a register, complaint details in a spreadsheet and certificate requests in a separate portal. A citizen asking about a pending application may have to visit more than one desk, and a clerk finishing a single task may have to search several places. When records are not linked, mistakes creep in and the citizen has no clear view of progress.

SmartGram is proposed as a way around this. Instead of building each service as a separate application, it organises them as modules of one system that share a backend and a database. The rest of the paper is arranged as follows. Section 2 reviews recent literature, Section 3 states the research gap, Section 4 the objectives and Section 5 the contributions. Section 6 presents the architecture, Section 7 the algorithms, Section 8 the decision logic, Section 9 the outputs and Section 10 the graphs. Section 11 reports results and the evaluation plan and Section 12 gives limitations and future work. Sections 13–20 discuss requirements engineering, data governance, privacy and evidence handling, transac-

tion integrity, accessibility, system operations, evaluation design and the implementation roadmap, and Section 21 concludes.

2 Literature review

This review concentrates on work published in 2025 and 2026, together with two official publications of the Ministry of Panchayati Raj. It is arranged by theme so that each group of studies can be tied to a design decision later in the paper. Sources were included if they appeared in 2025 or 2026, dealt with rural or local e-governance in India or with smart-village services, and described a system, a survey or an analysis that could inform the design. Eight journal papers and two official releases met this test and are listed in Table 1. Journal quality varies, so the findings of any single paper are treated as indications and not as settled facts.

Table 1. Reviewed Sources from 2025 and 2026.

Ref.	Year	Type	Contribution to this paper
[1]	2025	Journal	Panchayat platform (SARK): records, monitoring, information
[2]	2026	Journal	Panchayat platform (DigiGram) with chatbot and complaint ranking
[3]	2025	Journal	Panchayat portal with certificates, grievances and QR-based tax payment
[4]	2025	Journal	Survey of rural citizens' attitudes to e-government
[5]	2025	Journal	Study of rural access and service equity in Himachal Pradesh
[6]	2025	Journal	ICT for public services in smart villages
[7]	2026	Journal	Mapping of ICT applications in public-sector services
[8]	2026	Journal	Analytics for governance at the grassroots
[9]	2025	Official release	National platforms for Panchayat planning, audit and meetings
[10]	2026	Official release	Payments through eGramSwaraj and language reach of SabhaSaar

2.1 Rural e-governance platforms

Kamble et al. [1] present SARK, a web platform meant to streamline Gram Panchayat operations. It combines citizen-record management, administrative monitoring and the circulation of government information, and the authors discuss its architecture, the difficulties met in implementation and its potential to narrow the digital divide at village level. Chougule et al. [2] describe DigiGram, a platform built for a Gram Panchayat in Sangli district, Maharashtra, that serves more than fifteen thousand residents. It brings together complaints, certificates, property and water taxes, scheme information and meeting management, and adds a chatbot and an ensemble machine-learning model that ranks complaints by priority. Its summary reports measurable gains in throughput, complaint resolution time and transparency.

Salokhe et al. [3] propose a Digital Gram-Panchayat portal built with Python and Django. Citizens apply for certificates and follow their status, register grievances, read about welfare schemes and pay taxes by scanning a QR code with their own UPI application, while role-based authentication protects the records. The results section lists benefits in general terms, such as faster processing and clearer tracking, but gives no measured figures.

2.2 Citizen adoption and service equity

A digital service is only useful if people keep using it. Gangadhara and Senthil Kumar [4] surveyed 497 rural residents in Karnataka and found that service quality, information quality, trust and supporting facilities shaped attitudes towards e-government. Sharma [5] studied 300 citizens and 50 service providers in three districts of Himachal Pradesh. Digital platforms had widened access, yet rural residents still faced low digital literacy, weak connectivity and uneven grievance handling, and the study argues for decentralised, human-centred approaches with timely feedback. Kazlauskienė and Atkočiūnienė [6] approach the question from the side of infrastructure, examining how information and communication technology supports public service management in smart villages and what slows it down.

2.3 Public-sector integration and analytics

Fridayani et al. [7] map ICT applications across public-sector service provision and point to digital inclusion, institutional capacity and cybersecurity as recurring concerns. Jagadeesan et al. [8] study how analytics can strengthen governance at the grassroots in India, stressing coordination and decisions based on data. Together they suggest that moving paper forms online is not enough; the information behind the forms has to be linked, kept accurate and placed in front of the people who decide.

2.4 National initiatives

The Ministry of Panchayati Raj already runs several national platforms, including eGramSwaraj for planning and accounts, AuditOnline for audits, Meri Panchayat for citizen information and SabhaSaar for recording meetings [9]. More recent figures report that over Rs 3 lakh crore in cumulative payments has passed through eGramSwaraj and that SabhaSaar now works in 23 languages [10]. These platforms mostly serve planning, finance and oversight. Table 2 summarises how the works above relate to SmartGram.

Table 2. Summary of Reviewed Work.

Work	Main focus	Relation to SmartGram
SARK [1]	Records, monitoring and information for a Panchayat	Same goal of linked services; SmartGram adds verified transactions and role checks
DigiGram [2]	Complaints, certificates, taxes, schemes, chatbot, complaint ranking	Closest in scope; SmartGram stresses evidence capture and honest reporting of test status
Portal [3]	Certificates, grievances, QR-based tax payment	Overlaps in services; payment is settled outside the portal
Studies [4]–[6]	Adoption, equity and infrastructure	Inform interface, trust and access requirements
Studies [7, 8]	Digital inclusion, security and analytics	Inform security design and dashboards
Platforms [9, 10]	Planning, accounts, audit, meetings	System level; SmartGram targets one office's daily workflows

2.5 Synthesis

Three observations follow from the review. First, the Panchayat platforms in [1–3] converge on a similar service set (records, certificates, complaints, taxes), so the contribution of a new system has to lie

in how those services are connected and verified, not in the list of services. Second, the adoption and equity studies [4]–[6] show that technical completeness does not guarantee use, which makes clear statuses, assisted access and trust part of the design problem. Third, the integration and analytics work [7, 8] and the national platforms [9, 10] argue for linked data and decisions based on it, which SmartGram pursues at the level of one office.

3 Research gap

Reading the reviewed work against the situation at village offices points to six gaps:

- **Thin outcome evidence.** Of the Panchayat platforms reviewed, one reports results only in general terms [3], one concentrates on architecture and challenges [1], and the summary of another reports gains without the underlying figures [2].
- **Complaint evidence.** The descriptions we could read do not mention a live photograph and a permission-based location tied to a complaint record, so complaints can be hard to verify and to prioritise.
- **Payment and bill link.** Where payment is completed through an external UPI application [3], the description leaves open how the transaction is matched to a bill, a property and a taxpayer.
- **Rural access and equity.** Low digital literacy, weak connectivity and uneven grievance handling persist [5], so a platform must work for assisted as well as self-service users.
- **Security and access control.** Personal data, photographs and financial records call for server-side permission checks, and inclusion and cybersecurity are recurring concerns [7].
- **Office-level workflows.** National platforms emphasise planning, finance and audit [9, 10]; the daily citizen-facing workflows of a single office, and decisions drawn from them [8], receive less attention.

Table 3 shows where each gap is taken up in the design.

Table 3. Research Gaps and Design Response.

Gap	Design response	Section
Thin outcome evidence	Defined metrics, test cases and an untested-until-run rule	11
Complaint evidence	Live camera capture and permission-based GPS in Algorithm 2	7
Payment and bill link	Server-side verification and one transaction (Algorithm 4)	7
Rural access and equity	Light pages, assisted entry, clear statuses and notifications	6, 8
Security and access control	Server-side role checks, audit history, minimal data	6
Office-level workflows	Shared database, status workflow and indicators	6, 8

4 Objectives

SmartGram aims to:

- offer one platform for Gram Panchayat services and keep citizen and application records in a central database;
- simplify complaint registration, assignment and tracking, and support approval-based certificate issue;

- manage property, assessments, bills, payments and receipts as linked records computed on the server;
- provide role-based dashboards, graphs and a history of important actions and decisions;
- collect camera evidence and GPS location only through genuine, permission-based capture; and
- establish a testing approach for reliability, security and data consistency, with results reported only when they are measured.

These objectives translate into the requirements in Table 4, which also act as the acceptance criteria used later.

Table 4. Summary of System Requirements.

ID	Requirement
FR1	Register and authenticate users and apply role-based permissions
FR2	Accept service applications and let citizens track their status
FR3	Register complaints with a live photograph and optional GPS location
FR4	Issue certificates only after approval, with QR-code verification
FR5	Compute tax, bills, payments and receipts on the server
FR6	Show dashboards and graphs built from stored records
NFR1	Payment and status changes must complete fully or not at all
NFR2	Server-side authorisation and protection of personal data and evidence
NFR3	Each status change records who made it and when
NFR4	Readable statuses and simple forms, workable on low-cost devices

5 Contributions

The paper makes the following contributions:

- a layered architecture and logical data model that connect citizen services, complaints, certificates, taxation and payments through one shared data store;
- four algorithms covering authentication, complaint registration, tax calculation and payment verification, with rules for consistency and failure handling;
- evidence-aware complaint handling, in which a live photograph and a permission-based location are attached to a single complaint record;
- a status-based decision workflow with permitted transitions and decision rules that turn stored records into management indicators;
- a comparison with related systems and a requirement-to-test traceability table, so that untested features are not reported as finished.

The scope is deliberately limited. The framework works with the user roles already present in the project, and it is described at the level of design, algorithms and test planning.

6 System architecture

6.1 Layered design

SmartGram is split into four layers, shown in Figure 1. The presentation layer is a React and TypeScript front end that holds registration and login, dashboards, application forms, complaint and certificate pages, property and tax screens, notifications and reports. The application layer, written with Node.js and Express, receives requests from the front end and performs authentication, authorisation, input validation, complaint and application processing, tax calculation, payment verification and report generation. Prisma ORM forms the data access layer and maps structured models and queries onto the database. MySQL, the database layer, stores records for citizens, applications, complaints, certificates, properties, assessments, bills, payments, receipts and system activity.

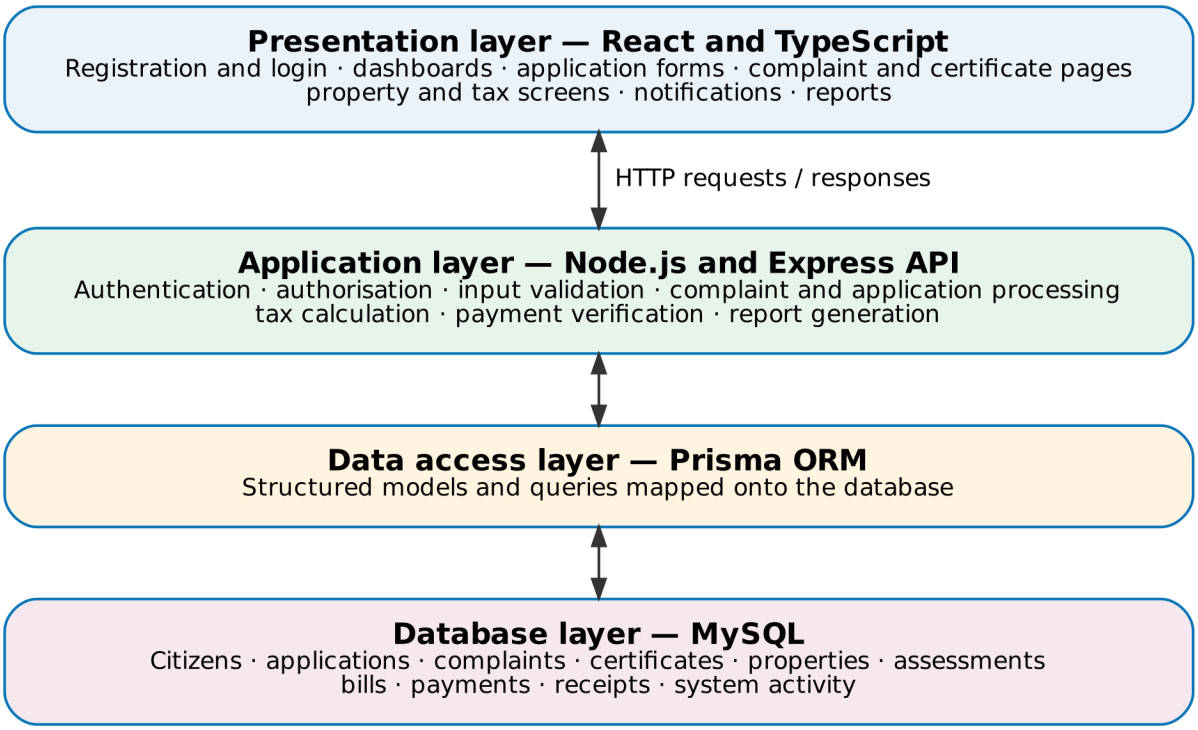


Figure 1. Layered architecture of the proposed SmartGram system.

A typical request travels as follows. The user opens the interface, and the front end sends a request to the API. The backend validates the request and checks the caller's permissions, then applies the business rules for that operation. Prisma carries out the database work, the backend returns the result, and the interface refreshes with the stored data. Keeping interface, business logic and storage apart makes each part easier to maintain and to test on its own.

6.2 Functional modules

Citizen management. Keeps names, contact details and addresses, lets authorised users register, search and update citizens, and lets each citizen view their own applications, complaints, certificates, bills and payments.

Citizen services and certificates. A submitted application receives a unique reference number and is reviewed according to a defined approval process. A certificate, with a QR code for verification, is produced only after the required approval has been recorded.

Complaints. A citizen chooses a category and describes the problem. The system stores a complaint number, submission date, location details and status history, and staff can assign the complaint and log the action taken.

Camera and GPS capture. The complaint form opens the browser camera so that the citizen can take a photograph, preview it and retake it. Location comes from the browser Geolocation API after the citizen grants permission. The system never substitutes made-up coordinates, and it does not treat image metadata as proof that a photograph is authentic.

Property and tax. Links property records to taxpayers. Assessments use configured rates and verified rules on the server, bills come from saved assessments, and receipts are issued only from verified payments.

Notifications, history and dashboards. Citizens are told when an application, complaint or payment changes state. Important actions are logged with the user and time. Each role sees its own dashboard: citizens see their own records, staff see permitted applications and complaints, the Sarpanch sees summaries, and the administrator sees configuration and audit information.

6.3 Data model

The logical data model, drawn in Figure 2, is built around the citizen. A citizen can own many applications, complaints and properties. An approved application may lead to one certificate, and each complaint can hold several pieces of evidence. On the financial side, a property has many assessments, each assessment produces one bill, a bill can receive several payment attempts, and a verified payment yields one receipt. Users, roles and an audit log sit alongside these entities. The model is logical, and the final Prisma schema may split or merge tables.

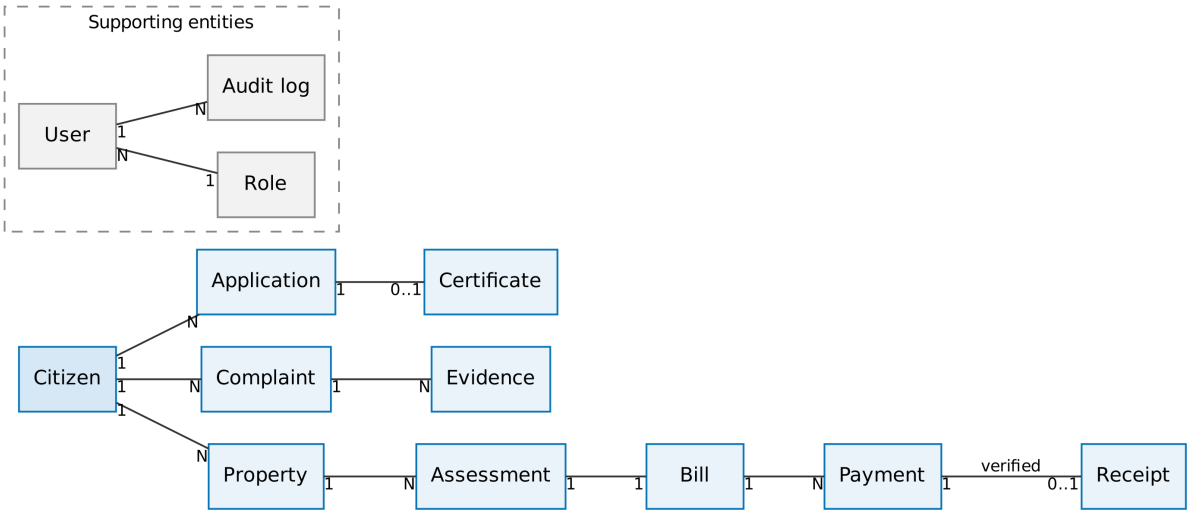


Figure 2. Logical data model showing the main entities and their cardinalities.

Unique constraints are intended on bill number, complaint number and payment transaction reference, so that duplicates are rejected by the database itself. Foreign keys tie every payment to a bill, every bill to a property and every property to a taxpayer, so orphaned financial records cannot be created.

6.4 Security and privacy design

Because SmartGram handles personal details, financial records, photographs and location data, security is part of the design. Passwords are checked through a secure verification mechanism, and every protected operation is checked on the server against the caller's role and, where relevant, ownership of the record; hiding a button is never treated as access control. The backend validates all incoming data and ignores client-supplied amounts, rates and statuses. Photographs and coordinates are attached to a single complaint and are readable only by the complainant and authorised staff, and location is collected only after permission is granted. Status changes, approvals and payment events are written to a history that records the responsible user and time. Only the data a service needs is requested, and roles that do not need individual records see aggregates.

Table 5 pairs the main threats considered with the intended response.

Table 5. Main Threats and Intended Mitigation.

Threat	Intended mitigation
Reading another citizen's records	Ownership and role checks on the server
Altered amounts or rates in a request	Server-side calculation; client-supplied values ignored
Replayed payment reference	Unique transaction reference and duplicate check
Forged certificate	Issue only after approval; QR code checked against the stored record
Misleading evidence	Live capture only; metadata not treated as proof; staff review
Exposure through dashboards	Role-scoped aggregates; individual records restricted

7 Algorithms and methodology

Four procedures carry most of the system's logic. All of them rely on server-side validation, permission checks and database transactions to keep records consistent.

7.1 Core algorithms

Algorithm 1: User Authentication

Input: username or email, password

Output: authentication result and dashboard redirect

1. Accept the credentials and send them to the backend.
2. Validate the input fields and fetch the matching user record.
3. Compare the password using the secure verification mechanism.
4. If valid, create the authenticated session or token and load the user's role and permissions.
5. Redirect the user to the dashboard for that role.
6. If invalid, show an error message and do not create a session.

Algorithm 2: Complaint Registration

Input: category, title, description, photograph, optional location

Output: stored complaint and unique complaint number

1. Open the complaint form and collect the details.

2. Request camera permission, show the live preview and capture a photograph; allow preview and retake.
3. If location is needed, request GPS permission and read the position.
4. Validate the image and the location data, then send the complaint to the backend.
5. Generate a unique complaint reference number.
6. Save the complaint and its evidence, with submission date and initial status.
7. Show confirmation only after the database confirms the save.

Algorithm 3: Tax Calculation

Input: property details, assessment data, tax rate, financial year

Output: tax amount and bill record

1. Retrieve the property and identify its tax category.
2. Retrieve the configured rate and validate the assessment data.
3. Calculate the tax amount on the backend.
4. Check that no bill already exists for the same property and period.
5. Generate a unique bill number and save the assessment and bill.
6. Show the bill to the authorised user.

The rates used must come from rules issued by the competent authority. Example rates are never assumed to be official.

Algorithm 4: Payment Verification

Input: bill number, payment transaction details

Output: updated bill status and receipt

1. Retrieve the selected bill and validate its details.
2. Start the configured payment process and receive the response.
3. Verify the transaction on the backend.
4. Check that the transaction has not already been processed.
5. Save the verified payment and update the outstanding balance.
6. Generate the receipt from the saved transaction and show the new status.

The message flow of Algorithm 4 is drawn in Figure 3. The duplicate check and the balance update are the sensitive steps, so they are grouped into a single database transaction. If any step fails, nothing is saved and the bill keeps its earlier status.

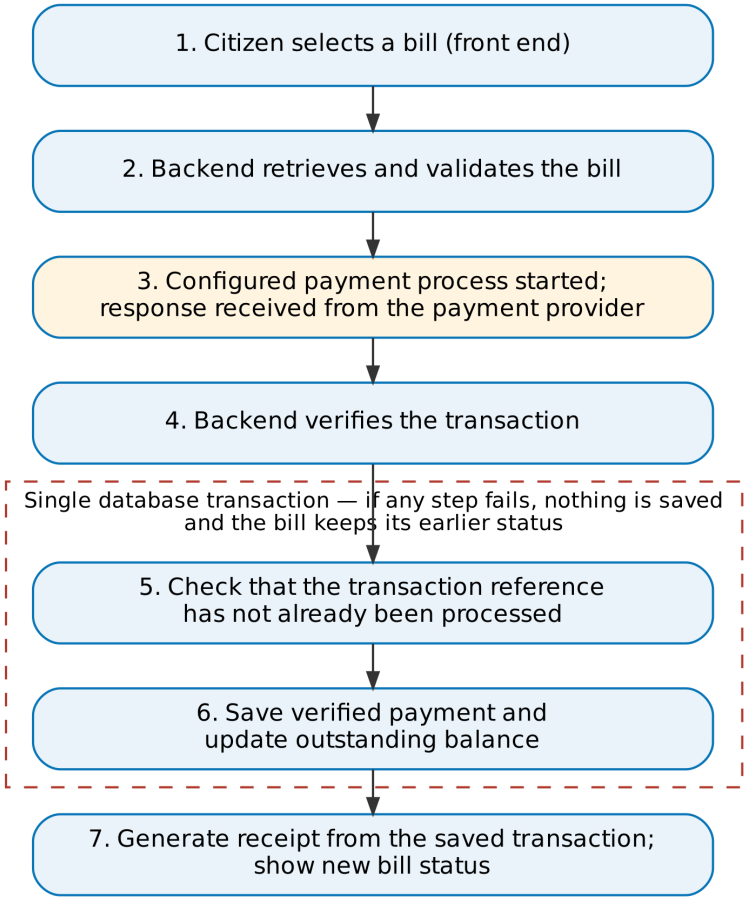


Figure 3. Message flow for payment verification.

7.2 Consistency and failure handling

Financial and status data must stay correct even when users repeat actions, connections drop or two staff members work on one record. Operations that change several tables, such as saving a payment and reducing the balance, run inside one database transaction. Repeated submissions are made harmless, since a bill is not created twice for the same property and period and a transaction reference is processed once. A record can change status only along the permitted transitions of Table 7, and confirmations are shown only after the database has acknowledged the write. Table 6 lists the intended response to the failures most likely in rural use.

Table 6. Common Failure Cases and Intended Handling.

Situation	Intended handling
Camera permission refused	Explain why access is needed and allow a retry; never store a placeholder image
Location denied or unavailable	Save the complaint without coordinates when location is optional
Connection lost during submission	Show an error; confirm a record only if the server acknowledged the save
Payment response late or missing	Leave the bill status unchanged until the transaction is verified
Form submitted twice	Detect through unique references and return the existing record

7.3 Development method

Development follows an incremental plan in seven phases: requirement analysis; system design (architecture, database relations, API structure and interface flow); backend development; frontend integration; module integration across services, complaints, certificates, taxation and notifications; testing (functional, security, database and end to end); and evaluation against the stated requirements.

8 Decision logic and decision support

Decisions in SmartGram follow from three things: the permissions of the person acting, the data submitted, and the current status of the record. A predefined condition decides whether an operation may go ahead.

8.1 Application approval

On submission the backend checks that all mandatory fields are present. A valid application is stored and given a reference number. It then moves through the states in Figure 4: submitted, under review, additional information required, approved or rejected. Every change of state is saved together with the user responsible and the time.

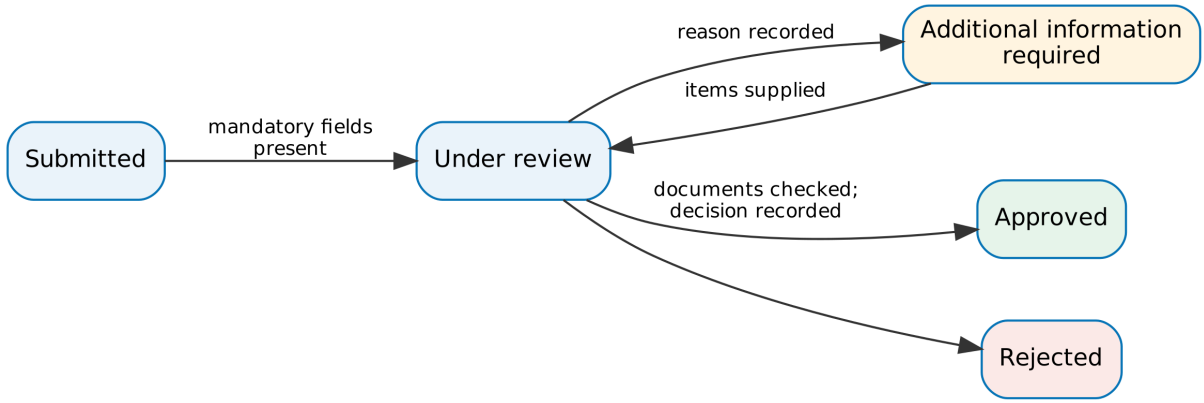


Figure 4. Status lifecycle of a service application.

Table 7 lists the permitted transitions for applications and complaints and the roles allowed to trigger them. Any request outside this table is refused by the backend, whatever the interface allows. Each transition can trigger a notification, which is created only after the change has been saved.

Table 7. Permitted Status Transitions.

Transition	Allowed role	Condition
Submitted → Under review	Panchayat staff	Mandatory fields present
Under review → Additional information required	Reviewing staff	Reason recorded for the citizen
Additional information → Under review	Citizen	Requested items supplied
Under review → Approved or Rejected	Authorised reviewer	Documents checked; decision recorded
Complaint registered → Assigned	Panchayat staff	Assignee holds an authorised role
Assigned → In progress → Resolved	Assignee	Action taken is recorded

Transition	Allowed role	Condition
Resolved → Closed	Authorised staff	Closure recorded with user and time

8.2 Certificates, bills and payments

For certificates, the system checks that the required documents and approvals are complete; if they are not, the request stays in its current status. A bill is created only after the assessment data has been validated, and a payment changes status only after the backend has verified it.

8.3 Decision support

Indicators drawn from stored records help staff and the Sarpanch decide where to act. They support the decision but do not make it; final administrative decisions rest with authorised personnel. Table 8 gives examples of how an indicator may be read and the kind of action it may prompt.

Table 8. Indicators and Possible Decisions.

Indicator	Possible reading	Possible decision
Pending complaints rising	Workload or slow assignment	Review assignment and staffing
Bills issued far above verified payments	Collection is lagging	Send reminders; check unpaid cases
Many applications awaiting information	Instructions unclear	Revise form guidance
Long resolution time in one category	Category lacks resources	Prioritise that category

9 Output

The outputs of the system are the records and documents that users actually receive or rely on. Table 9 lists them with the module that produces each and the people who see it. Every output is generated from stored records and only after the relevant operation has been confirmed.

Table 9. Outputs of the System.

Output	Produced by	Seen by
Application reference and status history	Service module	Citizen, staff
Certificate with QR code	Certificate module	Citizen
Complaint number, evidence and history	Complaint module	Citizen, staff
Tax assessment and bill	Tax module	Citizen, staff
Receipt from a verified payment	Payment module	Citizen
Notification messages	Notification service	Citizen
Dashboards and graphs	Reporting module	Staff, Sarpanch, administrator
Audit history	Activity log	Administrator

Two outputs deserve a closer note. A certificate carries a QR code that points back to the stored record, so anyone checking it can confirm that it was issued and has not been altered. A receipt is generated only from a payment that the backend has verified and saved, and it quotes the bill number and the transaction reference, so it can be matched to the record later.

10 Graph and analysis

Graphs give citizens and staff a quick view of patterns in applications, complaints and finances. Four views are planned: applications by status, complaints by category, the property tax position (demand, verified collection and outstanding amount), and the monthly mean complaint resolution time.

In the live system every value must come from a database query over a chosen date range, and the three tax figures must be calculated independently from the saved financial records. Three simple indicators are used: the collection ratio is verified collections divided by tax demand; the pending share is open complaints divided by all complaints in the period; and the mean resolution time is the average of closure time minus registration time over closed complaints. A rise in pending complaints suggests a growing workload, and a gap between bills issued and payments verified points to money still outstanding, but how far such readings can be trusted depends on the quality, completeness and date range of the stored data. Where data is missing for a period, the chart should say so instead of showing a zero that could be mistaken for a real value. Reports are scoped by role, so that a citizen sees only personal summaries.

11 Results and evaluation plan

SmartGram is partially implemented and end-to-end testing has not been carried out. The results below are therefore design results and a comparison with published work. Measured results will be reported only after testing.

11.1 Design results

The work so far has produced a four-layer architecture, a logical data model with ten core entities, four algorithms, a status model with seven permitted transitions, eight kinds of system output (Table 9) and four planned dashboard views. Each requirement in Table 4 is traced to a module and to a test in Table 11.

Once the build is complete, the design is expected to deliver a central database of citizen and administrative records, online submission and tracking of applications, complaints with live evidence, server-side tax and payment records, role-based dashboards and searchable activity histories. These are expected outputs, not observed ones.

11.2 Comparison with related systems

Table 10 compares SmartGram's design with the three Panchayat platforms reviewed in Section 2. An entry of n.r. means the feature is not reported in the sources we could read; it does not prove that the feature is absent.

Table 10. Feature Comparison with Related Platforms.

Feature	[1]	[2]	[3]	SmartGram
Certificate services	n.r.	Yes	Yes	Yes
Complaint handling	n.r.	Yes	Yes	Yes
Tax and payments	n.r.	Yes	Yes (QR)	Yes
Live photo and GPS evidence	n.r.	n.r.	n.r.	Yes
Role-based access	n.r.	n.r.	Yes	Yes

Feature	[1]	[2]	[3]	SmartGram
AI or chatbot features	n.r.	Yes	n.r.	No
Measured outcomes reported	n.r.	Yes	No	Not yet

SmartGram's design covers the same core services as [2] and [3] and adds live evidence capture and a server-verified link between payment and bill. It leaves out AI features on purpose, to keep the first version small and explainable. Its main weakness against [2] is that it has no measured outcomes yet.

11.3 Traceability and evaluation

Table 11 links each requirement to the module or rule that meets it and to the test that will check it. A requirement whose tests have not been run is reported as untested, not as satisfied.

Table 11. Requirement-to-Test Traceability.

ID	Module or rule	Verified by
FR1	Algorithm 1; role checks	T1, T2
FR2	Service module; Table VII	T10; module checks
FR3	Algorithm 2; camera and GPS	T3, T4
FR4	Certificate approval gate	T5
FR5	Algorithms 3 and 4	T6, T7, T8
FR6	Reporting queries	T9
NFR1	Database transactions	T7, T8
NFR2	Server-side authorisation	T2; security tests
NFR3	Status history and audit log	History review

The cases themselves are listed in Table 12, and include negative cases in which the system must refuse an action. Table 13 defines the measures to be taken once a working build exists.

Table 12. Representative Test Cases.

ID	Scenario	Expected result
T1	Login with a wrong password	Rejected; no session created
T2	Citizen opens another citizen's application	Access denied
T3	Camera permission refused	Clear message; no placeholder image stored
T4	Optional location denied	Complaint saved without coordinates
T5	Certificate requested before approval	Request refused; status unchanged
T6	Second bill for the same property and period	Rejected as a duplicate
T7	Same payment reference submitted twice	Processed once only
T8	Failure during payment saving	No partial update; balance unchanged
T9	Dashboard totals against direct counts	Figures match
T10	Refresh after submitting a form	Saved record is still shown

Table 13. Proposed Evaluation Metrics.

Metric	Definition
Processing time	Time from application submission to final decision
Resolution time	Time from complaint registration to authorised closure
Dashboard accuracy	Share of dashboard figures equal to a direct database count
Payment reconciliation	Share of verified payments matching the recorded bill balance
Access-control success	Share of unauthorised requests rejected by the backend
Task success rate	Share of test tasks users complete without help

Processing time and resolution time should be taken before and after introduction of the system, for the same office and a comparable period. Usability will be examined with a small group of citizens and staff who attempt realistic tasks, with success, time and difficulties recorded and a short questionnaire completed voluntarily. Security tests will try to break the rules the design depends on, such as opening another citizen's record, altering amounts or replaying a payment reference. A pilot in a single Panchayat limits how far findings can be generalised, and staff who know they are observed may work differently; these limits should accompany any result.

11.4 Discussion

Bringing related activities into one application can reduce the number of separate registers and give users one consistent place to look. Server-side validation and transactions matter most for financial operations, and role-based permissions limit who can see sensitive citizen data. How well the system works in practice will depend on a successful build, data quality, network availability, user acceptance and staff participation. Tax rates, certificate types and complaint categories are held as configuration, which should let another Panchayat set its own rules; a pilot must show whether that holds. SmartGram is not meant to replace the national platforms [9, 10]. Where authorised, bills, payments and certificates could later be reconciled with them, so that the office does not keep the same record twice.

12 Limitations and future work

No end-to-end test results exist yet, so the reliability and performance of the modules are unknown. Internet access and suitable devices may be limited in rural areas, and citizens with little digital experience may need help, which is why staff-assisted entry through the same system is planned. Tax results are only as accurate as the assessment rules and property data behind them, and the payment workflow depends on a third-party provider outside the project's control. Finally, personal details, financial records, photographs and location data call for strong security and privacy controls, including sensible retention periods.

Photographs may show people or private property, so the form should warn the citizen before capture, staff should open evidence only for the complaint concerned, and the retention period for evidence and location data should follow the rules of the competent authority.

A staged deployment is planned so that problems appear on a small scale first. Existing registers are reviewed and the data worth keeping is loaded into the database, roles are created with no more access than each task needs, staff are trained on their own daily tasks, and citizens who are not

comfortable with digital forms are helped by a staff member entering the application with them. Regular backups, access logs and an occasional test restore keep the service dependable.

Future work will concentrate on:

- completing the remaining modules and running the tests and metrics defined in Section 11;
- usability testing with citizens and Panchayat staff, staff training and better English and Marathi language support;
- vulnerability and performance testing under different loads; and
- a pilot deployment in a Gram Panchayat, with user feedback collected throughout and unfavourable results reported as well.

13 Requirements engineering and stakeholder workflow

A Panchayat platform serves people with different levels of digital experience and different responsibilities. Requirements should therefore be gathered from citizens, clerical staff, reviewers, the Sarpanch and system administrators. A citizen is primarily concerned with submitting a request, understanding what information is needed and receiving a clear status. Staff need queues, assignment information and a reliable record of decisions. The Sarpanch needs aggregate indicators without unrestricted access to every personal record. Administrators need configuration and audit facilities.

A useful requirements process begins by mapping the current paper or digital process for each service. The team records who receives a request, which documents are checked, who approves it, what fee applies and what evidence is retained. Exceptions are equally important: missing documents, duplicate applications, incorrect property details, payment timeouts and requests submitted by a person who needs assistance. These cases should be translated into explicit acceptance criteria before implementation.

The requirements in Table 4 can be extended with operational acceptance conditions. For example, a citizen should be able to retrieve a submitted application using its reference number; a staff member should be able to see only the queue permitted by their role; a certificate should not be issued before approval; and a payment receipt should not be generated merely because the browser returned a success screen. These conditions make the requirements testable and help prevent the interface from becoming the only place where business rules are enforced.

14 Data governance and record lifecycle

The shared database is central to SmartGram's proposed integration, but centralisation alone does not guarantee trustworthy records. Each data item should have a clear source, an accountable owner, a permitted purpose and a retention rule. Citizen details may be entered by the citizen or by an authorised staff member assisting them. Property assessments should be based on approved records and configured rules. Payment status should come from verified transaction information rather than a value supplied by the browser.

The data model should preserve the relationships described in the paper: a citizen can have multiple applications, complaints and properties; a property can have multiple assessments; an assessment can produce a bill; a bill can receive multiple payment attempts; and a verified payment can produce a receipt. Foreign-key constraints and unique identifiers help prevent orphaned records. A change

history is also necessary because the latest value alone may not explain who corrected a record or why.

Data quality controls should include required-field checks, duplicate detection, valid date ranges, standardised category values and periodic review of records that remain incomplete. Where data is imported from an existing register, the import should preserve the source reference and record the date of migration. Uncertain or conflicting entries should be flagged for human review rather than silently resolved by an automated rule.

15 Privacy, consent and evidence handling

Complaint photographs and location information can contain details about people, homes and public spaces. The capture screen should explain why the information is requested, whether location is optional and which staff roles may review the evidence. Permission should be requested at the point of use, and refusal should be handled without fabricating an image or substituting coordinates. If location is optional, the complaint should remain submittable without it.

The system should collect only information needed to assess and resolve the complaint. Access to evidence should be limited to the complainant and staff authorised to handle the case, subject to the final policy adopted by the Panchayat. Evidence access should be logged, and retention periods should be set by the competent authority. A deletion or archival process should account for legal, audit and recordkeeping obligations rather than retaining all evidence indefinitely.

A live camera capture provides a more direct capture path than an upload of an old image, but it does not prove that the scene is genuine or that the photograph accurately represents the reported issue. Image metadata is not proof of authenticity. Staff review, cross-checking and, where appropriate, a site visit remain important. The paper therefore treats the photograph and optional GPS coordinates as supporting evidence rather than an automatic basis for a decision.

16 Transaction integrity and financial reconciliation

The link between a property, assessment, bill, payment and receipt should be maintained through stable identifiers. The backend should calculate the amount due using the configured rate and approved assessment data. It should not trust an amount submitted by the client. A bill number should be unique for the relevant property and financial period, and a payment transaction reference should be processed only once.

Payment handling must account for uncertain network outcomes. A user may complete a payment with an external provider while the browser loses connectivity before receiving the response. In that situation, the application should not assume either success or failure based only on the browser. It should query or verify the transaction through the configured provider mechanism and update the bill only after confirmation. Duplicate callbacks or repeated user submissions should be handled idempotently.

The bill status, verified payment record, outstanding balance and receipt should be updated consistently. If several database changes form one logical operation, they should be committed in a transaction. If verification is pending, the bill should remain in its prior confirmed state and the interface should communicate that the result is being checked. Reconciliation reports should

compare provider-confirmed transactions with internal payment records and flag mismatches for authorised staff.

17 Accessibility and rural service delivery

A digital service should not assume that every resident owns a recent smartphone, has reliable mobile data or is comfortable reading long forms. Pages should load efficiently, use familiar labels, avoid unnecessary steps and present status information in plain language. Forms should identify mandatory fields and explain why a document is needed. Error messages should describe the next action rather than displaying technical codes.

Staff-assisted entry can support residents who cannot or do not wish to complete an online form independently. The system should distinguish the person requesting the service from the staff member who entered the information. The citizen should be able to review the captured details where practical, and the audit history should record the assisting user's identity. Assistance should expand access without creating a route for staff to impersonate citizens.

Language support is a functional requirement rather than a cosmetic feature. English and Marathi labels, help text and notifications should be reviewed by fluent speakers and tested with users. A translation should preserve the meaning of official service terms and required documents. Where a translated label is ambiguous, the interface can show a short explanation instead of relying on a literal translation.

18 System operations, backup and continuity

The proposed application depends on a backend service, database, network connectivity and, for payments, an external provider. Each dependency creates a possible interruption. The deployment plan should define how errors are logged, how administrators are notified, how backups are protected and how service is restored after a failure. A backup is useful only if it can be restored, so restoration exercises should be scheduled and documented.

Operational monitoring should distinguish application errors from expected user errors. Repeated failed logins, unexpected access-denied responses, payment verification delays and database connection failures may require different responses. Logs should include enough information to investigate an event while avoiding passwords, full payment credentials or unnecessary personal data. Access to logs should itself be restricted.

A continuity plan should explain how urgent citizen requests are handled during a prolonged outage and how records created through an approved offline process are reconciled later. Offline work introduces risks of duplicate entry and inconsistent status. Any such process should use temporary references and a controlled reconciliation step rather than allowing parallel untracked registers to become permanent.

19 Evaluation design and validity

The evaluation plan should separate functional correctness, security, usability and service outcomes. Functional testing checks whether each requirement behaves as specified. Security testing checks whether unauthorised users can access records, alter amounts, replay payment references or bypass approval states. Usability testing observes whether citizens and staff can complete representative

tasks and records task success, time and points of confusion. Service outcome evaluation compares measures before and after introduction under comparable conditions.

The proposed metrics require precise operational definitions. Processing time should identify whether it starts at initial submission or after the application is marked complete. Complaint resolution time should distinguish the first staff response from authorised closure. Dashboard accuracy should compare displayed totals with direct database queries over the same filters and date range. Payment reconciliation should count only transactions for which the provider's verification result is available.

A pilot in one Gram Panchayat can establish whether the system is workable in that setting, but it cannot automatically represent all Panchayats. Local service rules, staffing, connectivity, language use and citizen expectations differ. A before-and-after study can also be affected by seasonal workloads, policy changes or staff learning. The report should state these limitations and include negative or inconclusive findings rather than selecting only favourable outcomes.

20 Implementation roadmap and governance

A staged implementation reduces the risk of introducing every service at once. The first stage should establish identity, role permissions, citizen records and audit history. The next stage can introduce service applications and status tracking. Complaint handling and evidence capture can follow after privacy and access rules are approved. Property assessment, billing and payment verification should be introduced only after financial identifiers, provider integration and reconciliation procedures are tested.

Before each stage is released, the project team should demonstrate the workflow to the staff responsible for it, complete the associated test cases and document known limitations. Training should use realistic tasks rather than a feature-by-feature presentation. A named person should be responsible for reviewing configuration such as tax rates, service categories, certificate templates and notification wording. Changes to these settings should be versioned and logged.

Governance arrangements should define who may approve a certificate, who may close a complaint, who may correct property information and who may view financial summaries. These responsibilities should be reflected in server-side permissions and reviewed when staff roles change. The system should support administrative accountability without replacing the legal authority or judgement of Panchayat officials.

21 Conclusion

This paper presented SmartGram, an integrated framework that connects citizen management, service applications, certificate processing, complaint handling, property taxation, payments, notifications and reporting for Gram Panchayats. The design uses a React and TypeScript front end, a Node.js and Express backend, Prisma ORM and a MySQL database, with emphasis on consistent data, role-based permissions and traceable administrative actions.

The review of 2025 and 2026 work shows that outcome evidence, complaint evidence, payment-to-bill linkage, rural equity and security all need attention. SmartGram addresses these in its design, but its effect on service delivery, administrative efficiency and citizen satisfaction can only be established through full implementation, systematic testing and evaluation in a real Panchayat. The additional

discussion in this paper makes explicit the operational controls needed to turn those design elements into a dependable public service.

The framework remains partially implemented, and end-to-end testing has not been completed. The architecture, algorithms, outputs and evaluation measures are therefore design contributions rather than proof of improved service outcomes. A careful pilot should validate the rules with Panchayat staff and citizens, measure usability and processing outcomes, test security and privacy controls, and report limitations. SmartGram's success should ultimately be judged by whether residents can access services fairly and whether officials can make accountable decisions from accurate records.

References

[1] Kamble RR, Suriya AY, Kishor SB. Improving rural e-governance in India: Challenges, strategies, and a proposed framework. *Int J Sci Technol (IJSAT)*. 2025;16(2). https://doi.org/10.71097/IJSAT.v16.i2.5430

[2] Chougule S, Hulyalkar HS, Nayak D, Mulik PY, Khade S. DigiGram: An AI-powered digital governance platform for rural Panchayats in India. *Multidiscip J Res Eng Technol*. 2026;13(1):133–142. https://doi.org/10.65521/mjret.v13i1.3107

[3] Salokhe AA, Kshirsagar K, Makanikar D, Gore N, Kale G, Sangrampurkar P. *Digital Gram-Panchayat*. Int J Creative Res Thoughts (IJCRT). 2025;13(4):a37–a42. Paper IJCRT2504006.

[4] Gangadhara H, Senthil Kumar JP. How does the unified model of electronic government acceptance and service quality frame rural citizens' continuous behavioural intention and attitudes towards e-government services in the Indian context. *Int J Rural Manage*. 2025;21(3). https://doi.org/10.1177/09730052251352673

[5] Sharma A. Citizen-centric e-governance in Himachal Pradesh: Assessing the role of digital platforms in enhancing public service delivery and administrative responsiveness. *J Res Develop*. 2025;17(3):85–93. https://doi.org/10.5281/zenodo.15287164

[6] Kazlauskienė I, Atkočiūnienė V. Application of information and communication technologies for public services management in smart villages. *Businesses*. 2025;5(3):31. https://doi.org/10.3390/businesses5030031

[7] Fridayani HD, Younus M, Isnadi Y, Chiang LC, et al. Digital governance for better services with mapping of ICT applications in public sector service provision. *Discover Global Soc*. 2026;4:107. https://doi.org/10.1007/s44282-026-00458-8

[8] Jagadeesan V, Sivanandham S, Srinivas SS. Convergence through analytics: Enhancing governance at the grassroots level in India. *Local Econ*. 2026. https://doi.org/10.1177/02690942261483631

[9] Ministry of Panchayati Raj, Government of India. *Driving digital transformation in Gram Panchayats* [press release]. New Delhi: Press Information Bureau; 2025 Sep 25.

[10] Ministry of Panchayati Raj, Government of India. *Digital push deepens in rural India: eGramSwaraj records over ₹3 lakh crores in online payments; SabhaSaar expands to 23 languages* [press release]. New Delhi: Press Information Bureau; 2026 Mar 31.