International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

A Document-Driven Welfare Scheme Eligibility Recommendation System for Karnataka: Design and Technique Survey

Saithreya N K,C Sritha,Sinchana A Patgar,Dr Chandrashekar D K(Associate Professor)
Department of Computer Science and Engineering (Data Science), SJB Institute of Technology, Bengaluru, India

***Abstract*—Residents of Karnataka have access to various welfare programs, but many are unaware of which ones they qualify for.The requirements and necessary documents for each scheme vary and are distributed across different departments and online portals.The platforms we examined depend on information provided directly by citizens.This paper introduces a recommendation system for Karnataka Government welfare schemes and outlines the methods it uses: image enhancement and OCR, field extraction, LLM-assisted drafting of scheme guidelines, rule-based eligibility verification, and privacy-focused management of personal information.The system is designed to analyze citizen documents locally, let them verify the extracted details, and compare them against rules approved by human experts.It provides only the schemes the citizen is eligible for, along with a list of any missing documents and the official application link.For each step, we evaluate common approaches and highlight their limitations.We also mention the challenges we identified: maintaining up-to-date rules, OCR accuracy for Kannada and handwritten text, legal considerations around Aadhaar data, and the fact that the system is still in development and has not undergone evaluation yet.**

***Keywords*—e-governance, welfare schemes, eligibility checking, OCR, information extraction, rule engine, Kannada, privacy**

I. Introduction

Karnataka operates numerous welfare schemes through various departments, in addition to Central initiatives that the state implements.These schemes cover areas like education, health, agriculture, housing, and social security.Each scheme has its own set of eligibility criteria, which may include factors such as age, income, category, gender, land holdings, type of ration card, and district.Each scheme also requires a specific set of documents.A person who meets the criteria may never become aware of the scheme.Someone who does learn about it might only discover at the application counter that a required document, such as an income certificate, caste certificate, or ration card, is missing.

For beneficiaries in Karnataka, four key challenges stand out.

Schemes are managed by different departments, such as Social Welfare, Women and Child Development, Food and Civil Supplies, Revenue, and Agriculture, so there is no single page that lists all eligibility conditions.The rules are outlined in government orders and notifications that can change over time and are sometimes written in Kannada.Applications can be submitted through various channels, including the Seva Sindhu portal, the State Scholarship Portal, and individual department websites.However, a citizen who satisfies all the conditions may still be denied due to the lack of a required document.

Portals like mySchemе [11] and Seva Sindhu [12] have made it easier for citizens to find schemes and services.

On these portals, users generally provide their own details, but automatic extraction of information from uploaded documents is not widely implemented in the sources reviewed.Most of what determines eligibility is already included in documents that people possess.

own: the Aadhaar card, income certificate, caste certificate, land record (RTC) and ration card.

That observation points to a simple idea: read the documents, build a structured profile, and test the profile against the scheme rules. Making this work reliably touches several research areas, and every choice has a cost. This paper makes three contributions.

- A stage-by-stage survey of the techniques needed for a document-driven eligibility recommender, with the main options compared.
- A proposed system for Karnataka Government welfare schemes, showing how these techniques fit together while keeping results explainable and citizen data private.
- A planned evaluation method and a discussion of what remains unsolved.

II. Problem Formulation

Let a citizen profile P contain attribute–value pairs such as age, gender, category, annual income, district, occupation, ration card type, land area, and disability percentage. Let H represent the set of documents provided by the citizen. Each welfare scheme s is represented by a set of eligibility rules $R(s)$, a set of required documents $D(s)$, and an official application link.

A scheme is classified as **eligible** when all its mandatory rules are satisfied by the confirmed profile. It is classified as **not eligible** when at least one mandatory rule is known to be false. If a required attribute is missing, conflicting, or awaiting confirmation, the system marks the assessment as **requiring verification** rather than assuming that the condition is satisfied.

For an eligible scheme, the missing-document set is calculated as $M(s) = D(s) - H$. The system presents eligible schemes, prioritizing those with fewer missing documents, and provides the missing-document list and official application link. The final eligibility decision remains with the relevant scheme authority.

The main technical challenges are therefore obtaining reliable profile attributes from citizen documents and converting official scheme information into accurate, verifiable rules.

III. Related Work and Positioning

A. Existing Government Service Portals

Government portals support scheme discovery and public-service delivery, but citizens may still need to identify relevant schemes, understand eligibility conditions, and prepare the required documents. The myScheme platform supports discovery of government schemes using citizen-provided information [11], while Seva Sindhu provides access to Karnataka government services [12]. Ramathilagam *et al.* [15] proposed a conversational agent for accessing government healthcare schemes. The proposed system focuses on integrating document-based profile extraction, citizen confirmation, eligibility-rule evaluation, and missing-document identification into a workflow for Karnataka welfare schemes.

TABLE I. COMPARISON WITH EXISTING PORTALS

Aspect	mySchеme [11]	Seva Sindhu [12]	Proposed system
Scope	Government scheme discovery across Central and State/UT schemes	Karnataka government services	Karnataka Government welfare schemes
Citizen information	Uses citizen-provided information for scheme discovery	Information is supplied through relevant service workflows	Intended to derive profile attributes from uploaded documents and obtain citizen confirmation
Primary purpose	Scheme discovery and related guidance	Access to government services	Eligibility assessment and recommendation
Eligibility handling	Provides scheme-related eligibility information	Depends on the individual service	Evaluates structured rules against the confirmed profile
Missing documents	Scheme information may list required documents	Depends on the service	Identifies documents still needed for each eligible scheme
Document processing	Automatic extraction from uploaded citizen documents is not established here	Automatic extraction from uploaded citizen documents is not established here	OCR and information extraction are part of the proposed design

B. Recommendation Systems for Government Services

Lin [14] examined government-service recommendation in smart cities, including semantic service discovery, proactive recommendations, and knowledge-based approaches. The paper also discussed challenges involving context awareness, explainability, privacy, and data governance. Ramathilagam *et al.* [15] described a profile-based approach to government healthcare scheme discovery using attributes such as age, gender, occupation, and location. The proposed system differs in its emphasis on deterministic eligibility assessment using explicit scheme rules. It presents schemes that satisfy the confirmed profile's mandatory conditions and prioritizes those requiring fewer additional documents. This separates eligibility determination from the subsequent ordering of eligible schemes.

C. AI-Enabled Chatbots, Generative AI and RAG

AI-enabled conversational systems can help citizens access government information using natural language. Morande *et al.* [16] examined the potential of Generative AI and Retrieval-Augmented Generation (RAG) for citizen-centric governance and discussed associated adoption, security, privacy, and transparency considerations. The proposed system uses a hosted language model to assist administrators in drafting structured scheme rules from public information. These drafts undergo validation and human approval before being used. Citizen eligibility decisions, however, are made by a deterministic rule engine rather than by the language model.

D. Research Gaps

The reviewed studies address government-service recommendation, scheme discovery through conversational agents, and AI-enabled citizen–government interaction [14]–[16]. The present work investigates how document-based profile extraction, citizen confirmation, explicit eligibility-rule evaluation, and missing-document identification can be integrated into a single workflow for Karnataka Government welfare schemes.

The proposed approach focuses on three aspects:

1. **Document-driven profile construction:** extracting relevant attributes from citizen documents and allowing users to review and correct the extracted information before eligibility assessment.
2. **Explainable eligibility checking:** evaluating approved scheme conditions through explicit rules so that decisions can be traced to the conditions used.
3. **Document-readiness support:** identifying missing documents for eligible schemes and prioritizing schemes that require fewer additional documents.

The extent to which this integrated workflow improves extraction quality, eligibility correctness, and citizen usability remains to be established through implementation and evaluation. These points describe the focus of the present work; they should not be interpreted as proof that every existing platform lacks these capabilities.

E. Positioning of the Proposed System

The proposed system combines citizen-document processing, profile confirmation, structured eligibility rules, and missing-document identification for Karnataka Government welfare schemes. Its intended contribution is an integrated workflow in which scheme recommendations can be traced to explicit conditions and accompanied by practical information about remaining documentation. Its effectiveness will be assessed using field-extraction quality, eligibility correctness, missing-document identification, and processing time.

IV. Survey of Enabling Techniques

A. Image Preprocessing and OCR

Citizen documents may be uploaded as PDFs, scans, or phone photographs with skew, noise, shadows, or uneven lighting. Image preprocessing can improve readability before optical character recognition (OCR). A typical pipeline uses OpenCV [13] to convert pages to grayscale, reduce noise, correct skew, and apply thresholding where appropriate. The effectiveness of each operation depends on the document's quality and layout.

Tesseract [1] provides a lightweight OCR option and supports Kannada through its language data. PaddleOCR [17] offers a multilingual OCR pipeline, while Surya [2] provides text recognition and layout-related capabilities. The proposed design uses Surya as the intended primary OCR engine and Tesseract as a fallback. Kannada recognition quality, processing time, and performance on low-quality documents must be measured on representative samples before the choice can be considered validated.

TABLE II. OCR ENGINES CONSIDERED

Engine	Strengths	Weaknesses	Role in the proposed system
Tesseract [1]	Light, CPU-friendly, Kannada pack (kan+eng)	Weaker layout understanding	Fallback, or main engine on CPU-only servers
PaddleOCR [17]	Mature multilingual pipeline	Heavier setup	Considered, not selected
Surya [2]	Layout, reading order, tables, Indic scripts	Slower on CPU, newer project	Primary engine

B. Information Extraction

The proposed system combines pattern-based extraction with named-entity recognition and zero-shot entity extraction.

Pattern-based extraction: Regular expressions and field-specific validators are suitable for structured values such as dates, PIN codes, and monetary amounts. Where applicable, checksum validation, such as the Verhoeff algorithm [5], can detect certain transcription errors in identifiers.

Named-entity recognition: IndicNER and related resources developed for Indian languages, including Naamapadam [18], may help identify entities such as person and location names. Their suitability must be verified for the document types and languages used by the system.

Zero-shot extraction: GLiNER [3] can identify entities using labels supplied at inference time, making it a candidate for extracting fields that do not follow fixed patterns, such as annual income, land area, or card type. Its output should be validated because entity extraction may be inaccurate when document text is noisy or field labels are ambiguous.

Normalization and verification: Extracted values should be converted to consistent formats, such as normalizing monetary amounts and dates. Category and gender values should be mapped only when supported by the source document. Each extracted field should retain its source document and confidence information. Missing, conflicting, or low-confidence values should be presented for citizen review instead of being silently accepted.

C. LLM-Assisted Drafting of Scheme Rules

Official scheme details are found in web pages, government orders, and notices, but the rule engine needs structured data.A hosted model like Llama 3 can create a structured record (including name, department, rules, and required documents) from the scheme text.However, there is a risk that the model might misread a number, miss a condition, or make up a rule or link.Our system reduces this risk in five ways:

- The prompt limits the model to a fixed set of fields, operators, and document names, and the validator checks for the same list again.
- Every rule must include a source quote, and a check confirms that the quote actually appears in the source text.
- Conditions that the profile can't test are listed separately and marked.
- Portal links and verification dates are entered by a person, not the model.
- A human must approve every draft, and approved records are versioned.
- Only public scheme text is sent to the hosted model.

Citizen **data stays** on the **local server**.

D. Eligibility Reasoning

A deterministic rule engine is suitable when eligibility conditions can be represented as explicit, testable rules. A learned classifier could predict eligibility from historical examples, but its predictions may be difficult to trace to individual scheme conditions and may not reflect recently amended rules.

In the proposed system, each scheme is stored as structured data containing its eligibility conditions and required documents. The engine evaluates these conditions against the citizen-confirmed profile and records the reasons for its decision. If all mandatory conditions are satisfied, the scheme is classified as eligible. If a mandatory condition is known to be false, it is classified as not eligible. If a required attribute is missing or unresolved, the assessment is marked as requiring verification rather than being treated as eligible. For eligible schemes, the system identifies missing documents and prioritizes schemes requiring fewer additional documents.

This design favors traceability and testability. Its correctness depends on the accuracy and currency of the stored rules and profile data, which must be verified during evaluation.

E. Identity, Security and Privacy

The proposed design aims to minimize the storage of sensitive identity information. It uses a keyed HMAC-SHA256 value [6] as a duplicate-check identifier instead of storing the full Aadhaar number. A keyed HMAC reduces the risk of offline guessing from a database leak when the secret key is protected separately. The full number should not be retained in application logs, temporary files, or database records. If the last four digits are retained for display or support purposes, the need for doing so and the associated access controls should be documented.

Passwords or PINs should be stored using a suitable password-hashing method, such as Argon2 [8] or bcrypt [7]. Profile data may be encrypted at rest, with encryption keys stored separately from the database. Session tokens (for example, JSON Web Tokens [9]) should have limited lifetimes and be transmitted only over HTTPS. Uploaded files and extracted OCR text should be deleted after processing unless there is a documented, justified retention requirement.

Aadhaar-related processing and personal-data protection requirements must be reviewed before deployment. The Digital Personal Data Protection Act, 2023 [10], its applicable rules, and relevant UIDAI requirements should be considered in the context of the actual operator, purpose, data flows, and legal basis. The described safeguards are design measures, not a claim of legal compliance or a substitute for a security assessment.

F. Summary of Technique Choices

TABLE III. SUMMARY OF TECHNIQUE CHOICES

Stage	Options considered	Chosen for the proposed system	Reason
Document type detection	Classifier model; per-slot upload	One upload slot per document	No classifier needed; slot picks the template
OCR	Tesseract, PaddleOCR, Surya	Surya, Tesseract fallback	Layout and Indic support; CPU fallback
Fixed fields	Model; regex	Regex with validators and Verhoeff	Rules are more reliable for fixed formats
Custom fields	Fine-tuned model; zero-shot	GLiNER	No training data required

Stage	Options considered	Chosen for the proposed system	Reason
Rule creation	Manual only; LLM only; hybrid	Hosted LLM draft plus human approval	Speed with accuracy control
Eligibility	ML classifier; rule engine	Python rule engine	Explainable and testable
Identity	Store Aadhaar; plain hash; HMAC	HMAC-SHA256	Duplicate check without storing the number

V. Proposed System for Karnataka Government Welfare Schemes

A. Scope and Design Decisions

The first version covers Karnataka Government schemes only, and Central schemes are left out. The main design choices are these: one upload slot per document instead of a document classifier; a quick validity check after OCR (keywords and checksums) to confirm that a file belongs in its slot; deterministic eligibility; eligible-only output; no free-text search layer; and local, open-source processing for all citizen data. Login uses a generated citizen ID with a password or PIN. The Aadhaar number is used once for the duplicate check and never stored.

B. Architecture and Implementation Status

The proposed system consists of two connected workflows. The administrator workflow collects official scheme information, drafts structured eligibility rules, validates them, and stores approved scheme records. The citizen workflow processes uploaded documents, extracts profile attributes, requests citizen confirmation, and evaluates the confirmed profile against approved scheme rules.

The documented backend design uses FastAPI, PostgreSQL, SQLAlchemy, Alembic, and Pydantic, with an HTML, CSS, and JavaScript frontend. PostgreSQL is intended to store citizen records, encrypted profile information, scheme rules, and eligibility logs. The rule engine reads the approved scheme records and the confirmed citizen profile to generate eligibility results and missing-document lists.

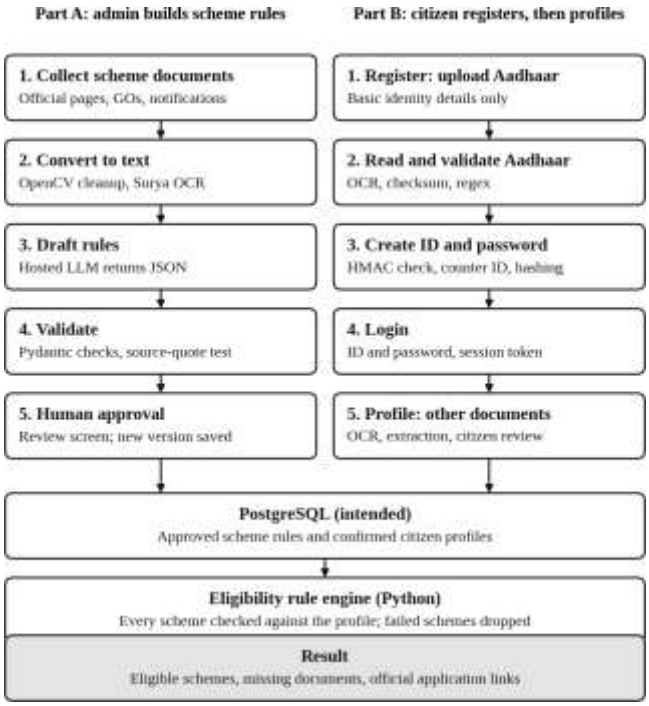


Fig. 1. System architecture: the admin flow fills the scheme tables, the citizen flow fills the profile tables, and the rule engine reads both.

Implementation status: The rules of the schemes are kept in JSON format and have been tested on a few government schemes from Karnataka.People can register using a hashed Aadhaar number and a special candidate ID.The website is built with HTML, CSS, and JavaScript.Right now, the team is working on using tools like Surya, Tesseract, GLiNER, and IndicNER to scan and pull information from documents.

C.

Part A: Building the Scheme Database

The admin pipeline has six steps.

All scheme pages, government orders, and notifications come only from official websites, not from other websites or blogs.Scanned documents go through the same process as citizen documents before being read by the software.A small group of 5 to 10 schemes is added by hand and checked again by someone else.This group becomes the standard for testing the system.The system then creates records and checks them automatically.A reviewer looks at each record on a screen where the source text is next to each rule and its quote.The reviewer can approve, edit, or reject the record.When approved, a new version of the scheme is created with the reviewer's name and date.Schemes should be checked again every 3 to 6 months and whenever there is a new government order.

The scheme rules are stored in JSON format and have been tested on a small set of Karnataka government schemes. Registration using a hashed Aadhaar identifier and a generated candidate ID has been implemented. The frontend is developed using HTML, CSS and JavaScript. Document OCR and field extraction using Surya, Tesseract, GLiNER and IndicNER are currently in progress.

C. Part A: Building the Scheme Database

The admin pipeline (Fig. 2) has six steps. Scheme pages, government orders and notifications are collected from official sites only, never from third-party blogs. Scanned sources go through the same preprocessing and OCR as citizen documents. A seed set of 5 to 10 schemes is entered by hand and checked by a second person, and it later serves as the answer key for judging the model. The hosted model then drafts records and automatic checks validate them. A reviewer approves, edits or rejects each draft on a screen that puts the source text beside every rule and its quote. Approval writes a new version of the scheme, stamped with the reviewer's name and date. Schemes should be rechecked every 3 to 6 months and after any new government order.

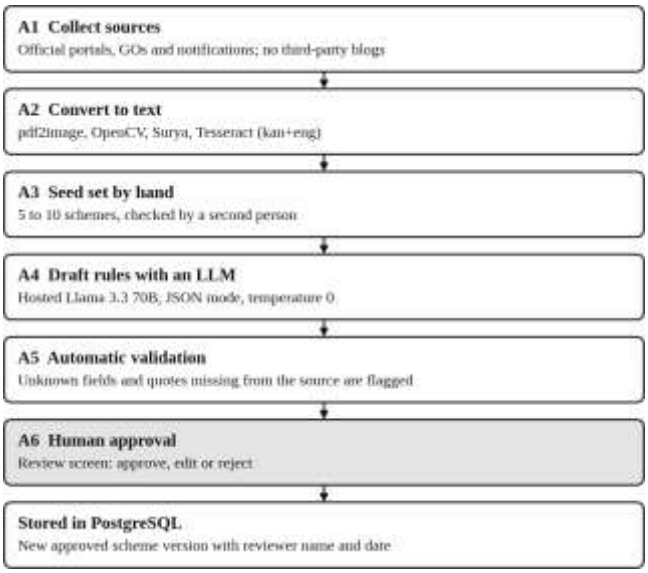


Fig. 2. Part A pipeline: every drafted rule passes automatic checks and human approval before it is stored.

D. Part B: Citizen Pipeline

The citizen workflow consists of four phases (Fig. 3).

1. **Registration:** The citizen supplies the information required by the registration process. If Aadhaar processing is used, the number is processed for the documented duplicate-check procedure, and a keyed HMAC is generated. The full Aadhaar number should not be stored. A checksum can help detect certain input or OCR errors, but it does not authenticate the identity of the person submitting the document. A generated, non-semantic citizen identifier is preferable to an identifier that exposes attributes such as gender or location.
2. **Login:** The citizen logs in using the registered identifier and password or PIN. Account lockout, rate limiting, secure password hashing, and consistent error messages are intended to reduce risks associated with repeated login attempts and identifier enumeration. These controls should be verified against the implemented application.
3. **Profile construction:** The citizen uploads documents through their designated document slots. Each document passes through preprocessing, OCR, document-validity checks, and field extraction. Extracted attributes are associated with their source documents and confidence information. The citizen reviews the resulting profile and resolves conflicting or uncertain values before confirming it. If a mandatory eligibility attribute remains unresolved, the engine must not treat that condition as satisfied.
4. **Eligibility results:** The rule engine evaluates the confirmed profile against approved scheme rules. Schemes that satisfy all mandatory conditions are displayed with their missing documents and official application links. Schemes requiring fewer additional documents are prioritized. Unresolved cases are not represented as confirmed eligibility, and the interface reminds citizens that the relevant scheme authority makes the final decision.

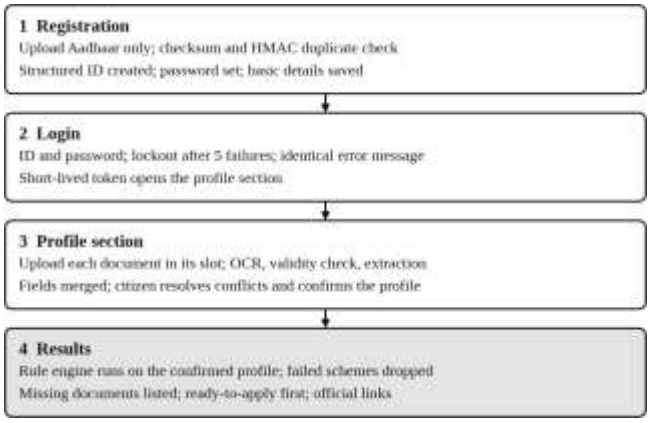


Fig. 3. Part B citizen journey in four phases. Preprocessing, OCR, the validity check and field extraction repeat for each uploaded document.

E. Data Model

The main tables are citizens (ID, Aadhaar HMAC, last four digits, PIN hash, lockout fields), id_counters, profiles (Fernet-encrypted JSON and a confirmed flag), profile_documents (which documents passed the validity check), schemes (versioned records with rules, required documents, links, last-verified date and status) and eligibility_logs (each decision and its reason, used for debugging). No table holds the full Aadhaar number, an uploaded image or OCR text.

F. Worked Example

Take a 22-year-old woman in category SC with an annual income of 120000 who has uploaded her Aadhaar, income certificate and caste certificate.

TABLE IV. WORKED EXAMPLE (ILLUSTRATIVE SCHEMES)

Scheme (illustrative)	Rules	Result
Scheme A	SC or ST, income up to 250000, age up to 25	Eligible; missing: marksheet
Scheme B	Income up to 100000	Not eligible; not shown
Scheme C	Women aged 18 to 40; needs only Aadhaar	Eligible; no missing documents, shown first as "Ready to apply"

VI. Security and Threat Analysis

TABLE V. THREATS AND CONTROLS

Threat	Control
Guessing IDs (predictable format)	Password required, lockout after 5 failures, identical error messages, rate limiting
Brute-forcing Aadhaar from stored data	Keyed HMAC with secret outside the database; number never stored
Database leak	Encrypted profile fields; hashed PINs; no images or Aadhaar number
Stolen token	Short expiry; HTTPS only
Malicious upload	5 MB limit, magic-byte type check, files never executed, deleted after use
Wrong rules reaching citizens	Validation, source-quote check, mandatory human approval, versioning
Hosted model seeing private data	Only public scheme text is sent

The application also refuses to start when a required secret is missing, rather than falling back to a default value.

VII. Evaluation Methodology

The proposed system will be evaluated at both component and end-to-end levels. The evaluation will use official scheme information verified against the relevant sources, manually reviewed scheme rules, and synthetic or appropriately anonymized test documents. Expected outputs will be prepared before testing so that system predictions can be compared with a reference.

Document-processing tests will include different image qualities, skewed pages, low-light images, Kannada and English text where supported, and documents uploaded to incorrect slots. Field-extraction results will be compared with manually labelled attributes. Eligibility tests will cover boundary conditions, missing attributes, conflicting values, and cases in which at least one mandatory rule is false. These tests will determine whether the system distinguishes eligible, not-eligible, and unresolved cases correctly.

Scheme-rule drafting will be assessed by comparing model-generated drafts with manually reviewed reference rules. The assessment will check whether mandatory conditions are omitted, incorrectly represented, or introduced without support from the source. End-to-end testing will compare the schemes displayed by the system with a manually prepared reference for each test profile. Processing time will also be measured.

No experimental results are reported at this stage. Quantitative findings will be added after the tests have been conducted.

TABLE VI. EVALUATION PLAN BY STAGE

Stage	Metric	Evaluation method
Preprocessing and OCR	Character error rate or word error rate	Compare OCR output with manually transcribed text, with and without preprocessing; compare selected OCR engines
Document-validity check	Accuracy, false acceptance, and false rejection rates	Test valid documents and documents uploaded to incorrect slots
Field extraction	Precision, recall, and F1-score where applicable	Compare extracted fields with manually labelled values, reporting results by field
Identifier validation	Correct acceptance and rejection of test inputs	Test valid and invalid checksum examples; do not interpret checksum validation as identity authentication
Identifier generation	Uniqueness and collision handling	Perform repeated and concurrent registrations using test data
Scheme-rule drafting	Rule-level precision and recall, omitted conditions, and unsupported conditions	Compare drafted rules with a manually reviewed reference set
Eligibility engine	Decision accuracy	Test eligible, not-eligible, boundary, missing-field, and conflicting-field cases against manually verified outcomes
Missing-document identification	Precision and recall	Compare identified missing documents with the expected set for each eligible scheme
End-to-end workflow	Agreement with the reference scheme list	Compare system output with manually prepared expected results for each test profile
Processing performance	Time per document and time from confirmed profile to results	Measure processing times across repeated test runs

The number and composition of test samples, the evaluation environment, and the measured results should be reported after testing. Results should not be generalized beyond the documents, schemes, and profiles covered by the evaluation.

VIII. Challenges and Limitations

- **Rules** are created based on official documents and become outdated when new orders or budgets are released.Each rule has a source link, version, and last-verified date.
- **OCR accuracy**: blurry images, handwritten notes, and some Indic scripts may have lower accuracy, which is why the citizen confirms the profile before any rule is applied.
- **Indicative results**: the final decision is made by the scheme authority.
- **Drafting errors**: the hosted model might misread an updated limit, so human approval is still required.
- **Hosted dependency**: drafting needs internet access, an API key, and free-tier limits that may change.Citizen processing does not rely on this.
- **Limited fields:** schemes that depend on information not shown in the uploaded document, like college enrollment, need a new document slot first.
- **Compute**: Surya and GLiNER run more slowly on machines without a GPU.
- **No exclusion reasons**: only eligible schemes are displayed, so a citizen cannot see why a scheme was not included.The reasons are stored in developer logs.
- **Legal:** Aadhaar handling and data protection rules must be checked before deployment.

IX. Future Scope

- Let admins upload scheme PDFs and images, with automatic OCR and drafting.
- Send alerts when a scheme page or government order changes.
- Allow a local model such as Qwen2.5 for drafting where hosted services are not permitted.
- Fine-tune IndicBERT or MuRIL on labelled Karnataka documents.
- Add a Kannada interface and voice guidance.
- Extend coverage to Central schemes implemented in Karnataka.
- Guide citizens who match no scheme, for example by suggesting the single document that would unlock the most schemes.

X. Conclusion

A Karnataka citizen often has to find the right department, read the conditions and gather the right certificates before any

benefit arrives. A document-driven recommender aims to remove much of that effort for Karnataka beneficiaries: it reads the documents they already hold, shows only the schemes they appear to qualify for, names the certificates still missing and links to the official application page. The survey showed that doing this reliably is mostly a data-quality problem, namely turning noisy documents into a correct profile and unstructured scheme text into correct rules. We compared the main options for OCR, extraction, rule drafting, eligibility reasoning and privacy, and described how the proposed system combines them: local open-source processing of citizen documents, citizen confirmation of the extracted profile, human-approved and versioned rules, deterministic and explainable evaluation, and minimal storage of identity data. The system is still under development, so the next step is to build it out and report measured results for every stage, using Karnataka scheme rules and test profiles.

REFERENCES

[1]R. Smith, "An overview of the Tesseract OCR engine," in Proc. 9th Int.
Conf. Document Analysis and Recognition (ICDAR), 2007, pp. 629–633.
[2]Surya OCR, open-source software for OCR, layout analysis and reading order, GitHub. https://github.com/datalab-to/surya, version 2.0.0 (accessed Oct. 17, 2026).
[3]U. Zaratiana et al., "GLiNER: Generalist model for named entity recognition using bidirectional transformer," in Proc. NAACL, 2024; arXiv:2311.08526.
[4]A. Dubey et al., "The Llama 3 herd of models," arXiv:2407.21783, 2024. [5]J. Verhoeff, Error Detecting Decimal Codes, Mathematical Centre Tract 29,
Mathematisch Centrum, Amsterdam, 1969.
[6]H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-hashing for message authentication," RFC 2104, 1997.
[7]N. Provos and D. Mazieres, "A future-adaptable password scheme," in Proc. USENIX Annual Technical Conf., 1999.
[8]A. Biryukov, D. Dinu, D. Khovratovich and S. Josefsson, "Argon2 memory-hard function for password hashing and proof-of-work applications," RFC 9106, 2021.
[9]M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, 2015.
[10]Government of India, The Digital Personal Data Protection Act, 2023. [11]myScheme, National Platform for Government Schemes, Government of
India. https://www.myscheme.gov.in/; https://negd.gov.in/our_projects/myscheme/; https://egovernance.vikaspedia.in/viewcontent/e-governance/digital-indi a/myscheme-national-platform?lgn=en (accessed Oct. 17, 2026).
[12]Seva Sindhu, Government of Karnataka. https://sevasindhu.karnataka.gov.in/ (accessed Oct. 16, 2026).
[13]G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
[14]R. Lin, "Recommender System for Government Service in Smart Cities," in Proc. IEEE/WIC/ACM WI-IAT, Niagara Falls, Canada, 2022, pp.
771–774, doi: 10.1109/WI-IAT55865.2022.00126.
[15]A. Ramathilagam et al., "AI-Driven Conversational Agent for Enhancing Government Schemes," in Proc. ICCIES, Coimbatore, India, 2025, doi: 10.1109/ICCIES63851.2025.11033042.
[16]S. Morande, G. Sogani, and S. Shah, "From Automation to Innovation: Harnessing the Synergy of Generative AI and RAG for Citizen-Centric Governance," in Proc. ITU Kaleidoscope, New Delhi, India, 2024, pp. 329–336, doi: 10.23919/ITUK62727.2024.10772932.
[17]Y. Du et al., "PP-OCR: A practical ultra lightweight OCR system," arXiv:2009.09941, 2020.
[18]A. Mhaske et al., "Naamapadam: A large-scale named entity annotated data for Indic languages," in Proc. 61st Annu. Meeting Assoc. Comput. Linguistics (ACL), Toronto, Canada, 2023.