International Journal of Advanced Engineering Application (IJAEA)
Volume 3 Issue 11 Nov 2026
ISSN :3048-6807

# GovAssist AI: An Agentic Approach for a Universal Government Schemes Navigator Using Quantum Machine Learning

**Gandikota Lavanya**[1], **Dhulipalla Sai Aditya**[2], **Gorugantula Manvita**[3], **Cherukubilli Geethanjali**[4], **Dunga Dineshbabu**[5], **Dr. V. Vasudha Rani**[6]
[1,2,3,4,5,6] Department of Information Technology, GMR Institute of Technology (GMRIT), Deemed to be University, Rajam, India.
Email :[1] lavanya2005g@gmail.com; [2] saiaditya0211@gmail.com; [3] manvita.gorugantula@gmail.com; [4] geethanjalicherukubilli@gmail.com; [5] adilaxmidinesh123@gmail.com; [6] vasudharani.v@gmrit.edu.in

***Abstract***

***Many people in India do not know about the Central and State Government welfare schemes they can claim. The information is spread over many different portals, the eligibility rules are hard to follow, and the steps to apply are confusing. This paper presents GovAssist AI, a web platform that lets citizens find, understand and apply for government schemes in one place. The system uses an agentic AI workflow built with LangGraph, a Large Language Model (LLM), and Retrieval-Augmented Generation (RAG) over a FAISS vector index of about 4,693 scheme records. It also has a Quantum Support Vector Machine (QSVM) module for eligibility, compared against a normal SVM baseline. A citizen enters a profile (age, income, state, occupation, education, social category, disability and household details), and the system uses it to suggest suitable schemes. The LLM then explains the difficult scheme text in simple words, based on the source, and gives step-by-step help for applying. The frontend is built with React, Vite and Tailwind CSS, and the backend with FastAPI and PostgreSQL. The platform works in English, Hindi and Telugu. It offers a scheme guide split into sections with voice read-aloud, document upload for eligibility checks, and a chat assistant that never claims to give official eligibility approval. The deployed prototype shows a working flow from creating a profile to finding schemes, getting recommendations, checking eligibility and receiving application guidance. GovAssist AI shows how agentic, retrieval-based and quantum-assisted machine learning can make government services easier to reach, clearer to follow and friendlier to citizens.***

***Keywords: Retrieval-Augmented Generation (RAG), Agentic Artificial Intelligence, Large Language Models (LLMs), Government Welfare Schemes, Quantum Support Vector Machine (QSVM), Eligibility Prediction.***

## I. INTRODUCTION

Government schemes give important support in areas like healthcare, education, farming, housing, jobs and social welfare. But many people, especially in villages, find it hard to learn which schemes are meant for them. The details are spread over many Central and State portals, the eligibility conditions are written in heavy official language, and the documents and steps needed change from one scheme to another. Because of this, people who deserve a benefit often never receive it.

Most existing welfare portals depend on manual or keyword search and give little personalised help. Recent studies have used machine learning, chatbots and RAG for this problem, but each one usually tackles only one part, such as recommendation, conversation or document checking. This paper presents GovAssist AI, a complete platform that brings together scheme discovery, eligibility analysis, document guidance and application help. It uses agentic AI to coordinate tasks, RAG to fetch answers from real scheme data, an LLM to explain things in plain language, and a QSVM classifier as a quantum machine learning add-on to the classical SVM baseline for eligibility analysis.

The main contributions of this work are: (i) one profile-based scheme navigator that covers Central and State/UT schemes; (ii) a LangGraph agent workflow that chooses between scheme search, eligibility checking, document guidance and application guidance; (iii) a RAG pipeline using sentence embeddings and a FAISS index, so that answers stay tied to the scheme data; (iv) a QSVM eligibility module with a classical SVM baseline and a safe fallback that never claims official approval; and (v) a deployed web application in several languages.

## II. LITERATURE SURVEY

One group of authors built a real-time government scheme navigator that gives personalised recommendations, checks eligibility automatically and sends notifications, using a mix of rules and machine learning over more than 150

schemes.[1] Another work proposes a framework that combines fuzzy matching, a multi-output Random Forest and Retrieval-Augmented Generation to recommend welfare schemes from a citizen's demographic profile and to cut down made-up answers.[2] Scheme Seva is an AI-based portal that joins rule-based eligibility filtering, Sentence-Transformer embeddings and an NLP chatbot to help with eligibility, documents and applications.[3] A study of Agentic AI in government found that autonomous agents and NLP-based workflows reduced meeting-scheduling time by 75%.[4] Another team built an AI digital assistant with K-Means clustering, Random Forest, TF-IDF and OCR, and reported 93.7% eligibility accuracy, a 93.3% F1-score and 88% user satisfaction on 300 annotated eligibility cases and 200 government documents.[5] A multi-intent government chatbot using BERT and spaCy that remembers context reached 92.3% intent accuracy with a 1.2-second response time.[6] A secure AI chatbot framework for a government website reported 90% response accuracy and a delay of under two seconds.[7] Other researchers proposed a modular and reproducible e-government design that combines LLMs, RAG, AI agents and conversational memory.[8] Finally, a government-specific LLM fine-tuned on 1,000 service question–answer pairs gave more accurate and more relevant answers to consultation questions.[9]

The basic ideas behind this work are well known. RAG ties LLM answers to retrieved documents,[10] ReAct-style agents mix reasoning with the use of tools,[11] Sentence-BERT gives meaningful text embeddings[12] and FAISS searches through them quickly.[13] Quantum kernel methods came from the quantum support vector machine[14] and from quantum-enhanced feature spaces.[15]

These studies leave three gaps. First, most are built on classical machine learning such as K-Means and Random Forest. Second, they handle recommendation, chatbots and document work as separate tools. Third, government chatbots find it hard to follow a conversation over several turns or to answer questions from one specific domain. GovAssist AI tries to close these gaps with a single design that combines QSVM, RAG, an LLM and agentic AI.

## III. METHODOLOGY

In the past, citizens had to visit several websites, read long eligibility text by themselves, and still could not tell which schemes were right for them. GovAssist AI replaces this with a pipeline driven by the user's profile. The frontend is a React single-page application. A FastAPI backend runs the AI components, and PostgreSQL stores user accounts and profiles. The overall architecture is shown in Fig. 1.

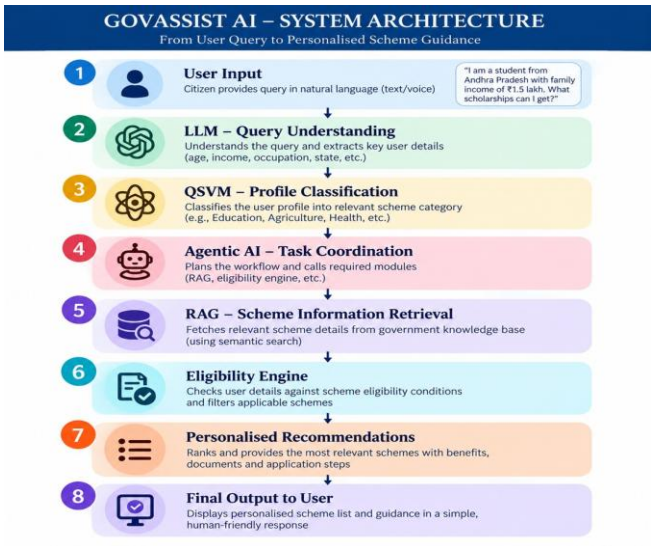


Fig 1: System Architecture of GovAssist AI

*A. Data Set:*

The scheme knowledge base is a carefully prepared collection of about 4,693 Central and State/UT scheme records, stored in Schemes.csv. Each record holds the scheme name, level (Central or State/UT), state, ministry, department, category, sector, description, benefits, eligibility conditions, required documents and the official application link. The

prototype offers five main sectors: Agriculture, Education, Healthcare, Welfare and Persons with Disabilities. User data is stored in a PostgreSQL database through SQLAlchemy. It keeps login details and an eligibility profile made up of age, gender, state, district, occupation, annual income, social category, education, household size and housing status.

*B. RAG: Embeddings and FAISS Retrieval:*

Scheme text is turned into numeric vectors using the Sentence-Transformers model all-MiniLM-L6-v2, which is loaded once per process and kept in memory. The vectors are stored in a FAISS index (schemes.faiss), along with a metadata file that links each vector back to its scheme record. The index is loaded from fixed paths in the source code and is never rebuilt when the app starts. For each query, the system converts the question into a vector, finds the closest scheme records and gives them to the LLM as context. Since the answer is produced only from official scheme data that was retrieved, the chance of unsupported statements becomes smaller.

*C. LLM Layer:*

The LLM takes the retrieved scheme information and turns it into simple explanations of the benefits, eligibility, required documents and application process, personalised with the user's profile. The backend can work with different LLM providers. During local development it uses Ollama with Llama 3.2, while the deployed version uses the hosted Gemini model (gemini-3.1-flash-lite), so no local model server is needed. Any OpenAI-compatible endpoint can also be set up. The system prompt tells the assistant to stay within the retrieved scheme data, to direct users to official portals, and never to claim official eligibility approval.

*D. Agentic AI Orchestration (LangGraph):*

Instead of a fixed chatbot pipeline, a LangGraph workflow keeps the user profile as shared state. An agent first understands the request and then picks the tools it needs: a Retriever over the scheme database, the QSVM eligibility predictor and the RAG scheme-knowledge module. The LLM then joins their outputs into one personalised reply (Fig. 2). The workflow handles scheme search, eligibility checking, document guidance and application guidance, and it can combine several of these in a single answer.

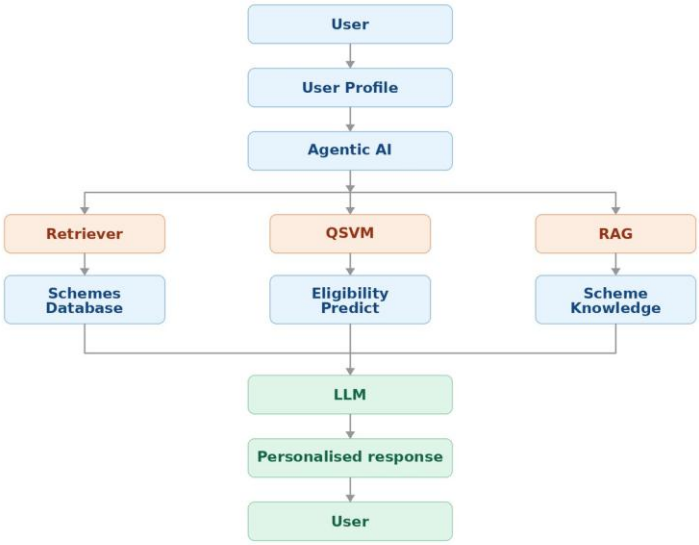


Fig 2: Agentic AI Workflow

*E. Eligibility Prediction: Classical SVM and QSVM:*

The citizen profile (age, income, state, occupation, education and social category) is encoded and scaled using saved preprocessing files (preprocessor.joblib, scaler.joblib, feature_names.json). A classical SVM (classical_svm.joblib) acts as the baseline. The QSVM turns each scaled profile x into a quantum state $|\varphi(x)\rangle$ using a quantum feature map, and the quantum kernel measures how similar two profiles are by the overlap of their states:

$$K(x, x') = |\langle\varphi(x)|\varphi(x')\rangle|^2 \quad (1)$$

The kernel matrix is then given to an SVM, and the decision function becomes

$$f(x) = sign(\Sigma_i \alpha_i y_i K(x_i, x) + b) \quad (2)$$

where $\alpha_i$ are the learned weights, $y_i$ are the training labels and b is the bias. The trained QSVM (qsvm_model.joblib) is loaded from saved artifacts and is not retrained during deployment. If the Qiskit runtime cannot load on the host machine, the application shows eligibility as unavailable or as "verification required" instead of making a claim it cannot support. All results are only indicative and must still be verified officially.

*F. Process Flow:*

The process has eight steps: (1) the user profile is collected; (2) the data is cleaned, checked and turned into structured features; (3) the QSVM sorts the profile into relevant scheme categories; (4) the agent studies the request and coordinates the tasks; (5) RAG fetches matching scheme information from the knowledge base; (6) the profile is compared with each scheme's eligibility criteria; (7) the schemes are ranked and shown with their benefits, conditions and documents; and (8) step-by-step application guidance with official links is given.

*G. Implementation and Deployment:*

The technology stack is listed in Table I. The frontend is deployed on Vercel, with a single-page-app fallback to index.html. The backend is packed as a Docker container that includes the application, the data, the FAISS files and the ML artifacts. Settings such as the database URL, JWT secret, LLM provider and CORS origins are passed through environment variables. The /health and /ready endpoints report the status of the application, database, FAISS, models and LLM.

TABLE I. TECHNOLOGY STACK OF GOVASSIST AI

| Layer | Technology |
|---|---|
| Frontend | React 18, Vite 5, Tailwind CSS 3, React Router 6, lucide-react |
| Backend | FastAPI (Python 3.12), JWT authentication |
| Database | PostgreSQL (SQLAlchemy) |
| Retrieval | Sentence-Transformers (all-MiniLM-L6-v2), FAISS |
| Agentic AI | LangGraph |
| LLM | Gemini (hosted) / Ollama Llama 3.2 (local) |
| Quantum ML | QSVM (Qiskit), classical SVM baseline |
| Hosting | Vercel (frontend), Render (backend) |

## IV. RESULTS

GovAssist AI was built and deployed as a working web application. The results below describe each module of the prototype along with its screens.

*A. Citizen Profile Intelligence:*

Users register, log in and then create an eligibility profile with age, income, state, occupation, education, disability and social category. This profile is used to personalise recommendations. After login, the user sees a navigation sidebar (Dashboard, Find Schemes, AI Assistant, Check Eligibility, My Recommendations, Saved Schemes, Documents, Profile and Settings), a language selector, a notification bell and their name with the role "Citizen". The dashboard (Fig. 3) shows profile completion, the number of recommended and saved schemes, the top recommendations, a shortcut to the AI Assistant and quick actions. For the demonstration user, the profile was 100% complete and 30 schemes were recommended.

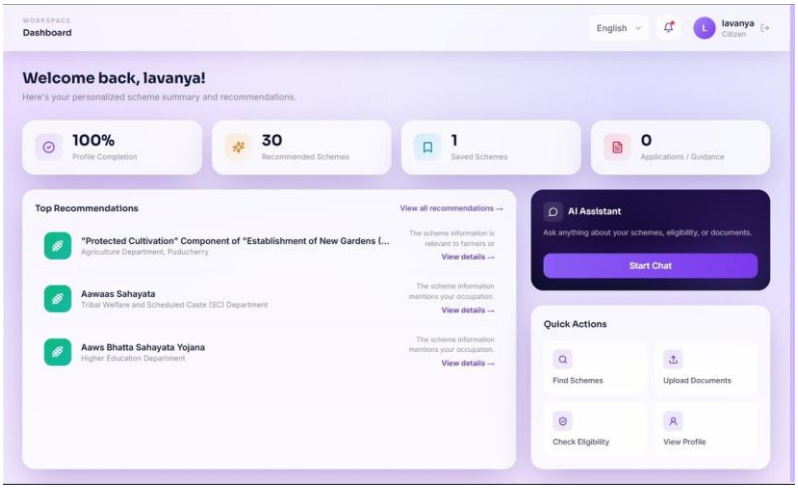


Fig 3: Citizen Dashboard

*B. AI-Powered Scheme Discovery:*

Users can search for schemes with everyday keywords and filter them by sector (five main sectors) and by level (Central or State/UT). The Find Schemes screen shows “10 of 2468 schemes” per page, sorted by name, and each card shows the state, sector, a short description and the key benefits, with a View Details button. FAISS-based retrieval finds the relevant scheme information. The detail page (Fig. 4) shows the ministry, department, level, category and sector, with Save and Apply buttons that link to the official source.

*C. Scheme Guide with Focus Mode and Voice Output:*

Below the scheme header, every scheme page has a Scheme Guide that turns long official text into six short sections: At a glance, What you get, Who can apply, Get ready, Next step and Official links (Fig. 4). A progress chip (for example “3 of 6”) shows where the reader is, and the section tabs let them jump between sections. Fullscreen opens a focus mode that shows one section at a time under a quick-read heading with a Next section button (Fig. 5), and a Listen control with pause and stop reads the section aloud.

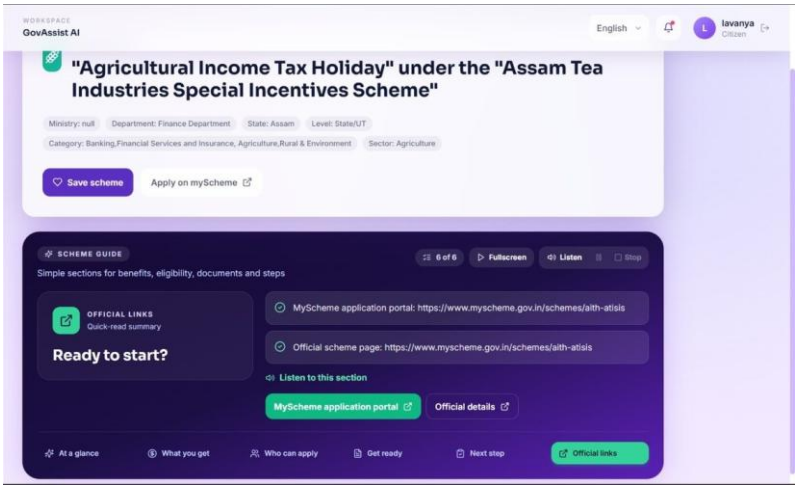


Fig 4: Scheme Guide Showing the Official Links Section

The example in Figs. 4 and 5 is the “Agricultural Income Tax Holiday” under the Assam Tea Industries Special Incentives Scheme (Finance Department, Assam; State/UT level; Agriculture). Its Who can apply section is a four-item checklist: a tea plantation of any size, a tea-manufacturing unit (crush, tear, curl, orthodox or speciality teas), located fully in Assam, with no part of the plantation or facility outside Assam. Official links gives the MyScheme application portal and the official scheme page, each with an open button. The guide is created only from the scheme record.

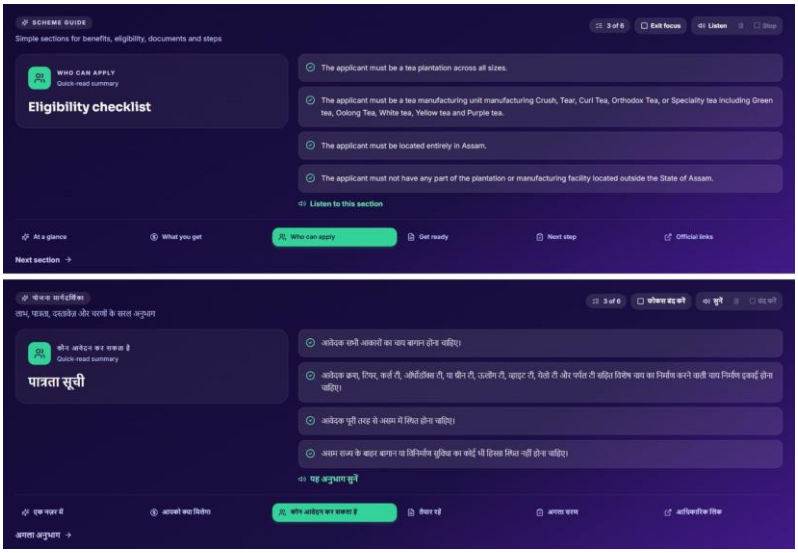


Fig 5: Focus Mode, Who Can Apply: (a) English, (b) Hindi

*D. Eligibility Prediction and Recommendations:*

Profile details are encoded and scaled before the quantum classification. The QSVM predicts eligibility status, with the classical SVM as the baseline. The My Recommendations screen (Fig. 6) lists schemes with a "Why it may be relevant" note, for example that the scheme mentions the user's occupation or type of residence, and it states that the results still need official verification. A numerical comparison of the QSVM and the classical SVM has not been done yet and is left for proper future experiments.

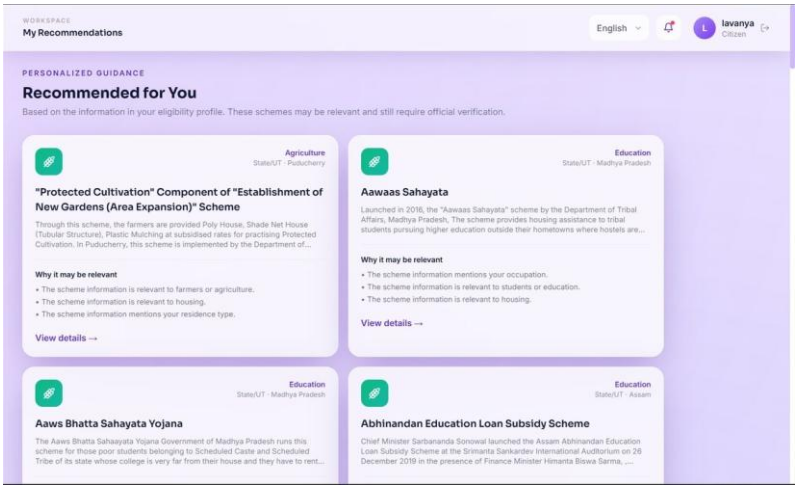


Fig 6: Personalised Recommendations

*E. Agentic Orchestration and LLM Assistance:*

The LangGraph workflow understands the citizen's request, chooses the needed action and combines several AI steps into one reply. In the AI Assistant (Fig. 7), a student from Andhra Pradesh asked, "What schemes are available for BTech students?". The assistant returned technical-education scholarships from the retrieved dataset: the State Technical Scholarship for ST Students (Nagaland), the Technical Education Scholarship Scheme for Diploma Level (Himachal Pradesh) and the BPL Student Welfare Scholarship (Chhattisgarh). It warned that these schemes may need residency in the particular state, pointed the user to the official MyScheme portal and did not claim official eligibility approval. Quick actions (Find Schemes, Check Eligibility, Docs Help) and a Listen button are available. At present, the reply shows some Markdown symbols (### and **) as plain text, which is a formatting issue still to be fixed.

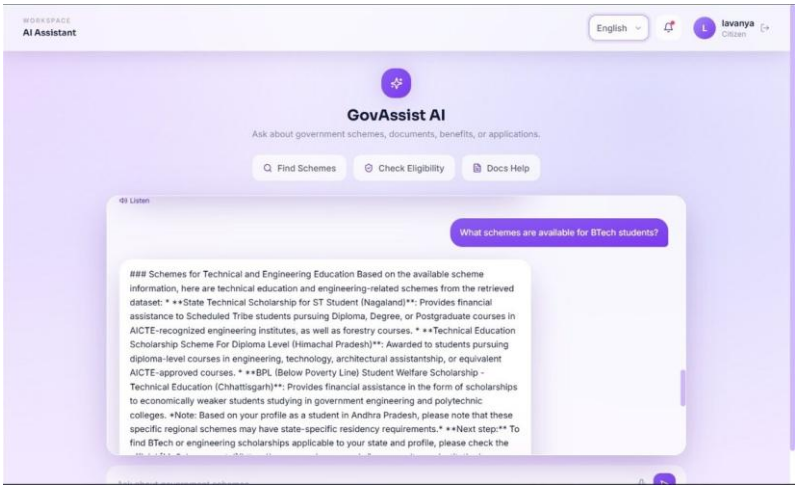


Fig 7: AI Assistant Response

*F. Multilingual Support and Document Intelligence:*

The interface language can be switched between English, Hindi and Telugu. The guide sections, tabs and buttons are translated (Fig. 5b and Fig. 8), and in Telugu the sidebar, metadata chips and the Save and Apply buttons are translated as well. AI-translated pages show a notice, displayed in Telugu, saying that the content was translated by AI and that the official source should be checked. Scheme names, links and some metadata values stay in English. In Fig. 8, the What you get section states a three-year agricultural income tax holiday for 2022-23 to 2024-25, as long as returns are filed on time, and the Next step section lists the application steps in order: portal registration, OTP verification, login, document upload, review before submission, a reference number for tracking, and direct bank transfer of the benefit. The Documents page accepts PDF, JPG or PNG files up to 10 MB; OCR is planned for later.

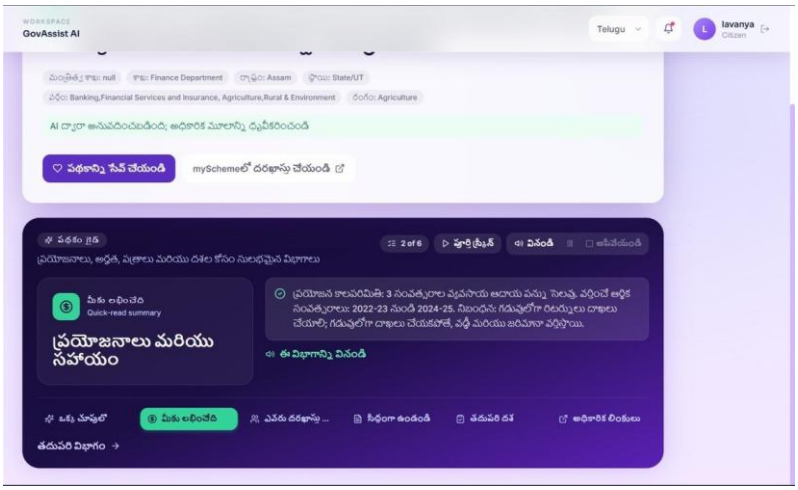


Fig 8: Telugu Scheme Page with Translated Guide

*G. End-to-End Assistance:*

All components work together as one platform (Figs. 1 and 2): citizen profile, RAG retrieval, agentic task selection, QSVM eligibility classification, a simple personalised LLM reply and action guidance covering documents, benefits and application steps. Among the systems we reviewed, none combines RAG, agentic orchestration and quantum-kernel classification in one citizen-facing workflow.

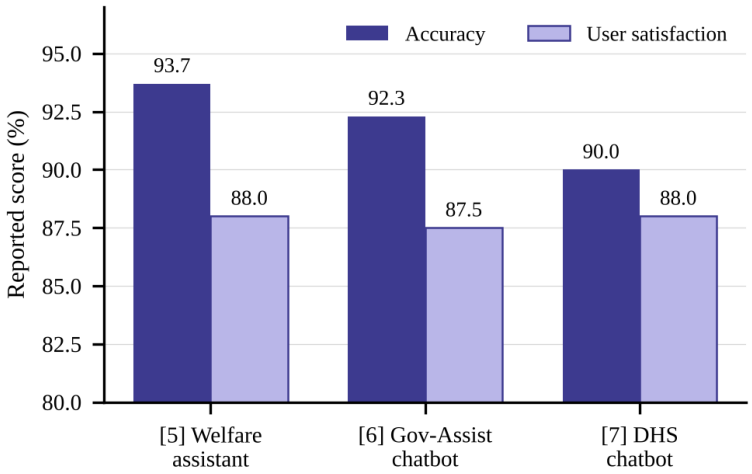


Fig 9: Performance Reported by Related Government-Service AI Systems

To give some context, Fig. 9 compares the accuracy and user satisfaction reported by three related systems.[5],[6],[7] These numbers come from the cited papers and not from GovAssist AI. They show the 90–94% accuracy range against which the QSVM module should be tested. The assessment of GovAssist AI itself is about functionality, not numbers: accuracy, F1-score and user-satisfaction figures for the QSVM module and for the assistant have not been measured in this version.

## V. DISCUSSION AND LIMITATIONS

The system also has some limitations. The quantum kernel is limited by the number of available qubits and by the cost of computing the kernel, so the QSVM is presented as a research extension, and no quantum advantage over the classical SVM is claimed. Answer quality, speed and cost depend on the LLM provider that is chosen. The scheme catalogue is a snapshot and can go out of date, and some records have missing details, such as an empty ministry field that shows as "null". Document OCR is not connected yet, and database tables are created by start-up DDL instead of a versioned migration tool. Rate limiting, central logging, backups and monitoring have to be provided by the hosting platform. Profiles contain sensitive data such as income and caste category, so they are kept private by default. Finally, all eligibility output is only indicative: only the concerned government department can confirm eligibility.

### *A. Planned Evaluation:*

A proper evaluation will be done in the next phase using these measures: (i) accuracy, precision, recall and F1-score of the QSVM against the classical SVM on labelled eligibility cases; (ii) retrieval quality of the FAISS index, measured by recall@k and mean reciprocal rank on a set of citizen queries; (iii) answer faithfulness, meaning the share of replies fully backed by the retrieved scheme text, to measure hallucination; (iv) end-to-end response time; and (v) user satisfaction and ease of use, collected through a questionnaire given to citizens and field workers.

## VI. FUTURE SCOPE

### *A. Use Without Internet:*

Many of the people who need schemes the most live in places with weak or no internet. The frontend could be made into an installable progressive web app that stores the scheme catalogue, the FAISS index and the user profile on the device. A small quantized LLM and the saved SVM/QSVM files could then run locally for search, eligibility checks and explanations. Requests made offline would wait in a queue and sync once the connection is back. SMS, IVR or USSD channels, and assisted use at village service centres, would reach people who do not have smartphones.

### *B. Cyber Security:*

Since the platform handles personal and income data, security has to be part of the design. The planned measures are end-to-end TLS and encryption of stored profiles and documents, multi-factor authentication, role-based access control, rate limiting and audit logs, similar to those used in secure government chatbots.[7] Further work includes protection against prompt injection and data leakage through the LLM, safe and consent-based handling of uploaded documents,

anonymised logs and regular vulnerability testing. Versioned database migrations and monitored backups would also be added.

*C. Support for All Indian Languages:*

At present the interface supports English, Hindi and Telugu. The goal is to cover all 22 scheduled languages of India, along with major regional dialects, in the menus, scheme explanations and chat. This needs multilingual embeddings for retrieval, Indic-language LLMs or translation services such as Bhashini, and translated scheme content that stays linked to the official source. Voice input and spoken output would help people who find reading difficult. Automatic catalogue refresh and deadline alerts are other possible additions.

## CONCLUSION

This work presented GovAssist AI, a web platform that helps citizens find and apply for Central and State government schemes. Existing portals depend on manual search, and earlier research treats recommendation, chatbots and document checking as separate problems. GovAssist AI brings them into one workflow, where a LangGraph agent coordinates a FAISS-based RAG retriever, an LLM and a QSVM eligibility module with a classical SVM baseline. The system was built with React, Vite, Tailwind CSS, FastAPI and PostgreSQL, and it was deployed with health and readiness checks. Its interface supports English, Hindi and Telugu, and its guidance stays based on scheme data and avoids claiming official approval. In the future, the QSVM module will be tested against the classical SVM on labelled eligibility data, and OCR-based document verification will be added. The platform will also be extended for offline use, stronger cyber security and support for all Indian languages, as described in Section VI.

## CONFLICT OF INTEREST

The authors declare that they have no conflict of interest related to this research and publication.

## USE OF AI TOOLS

GovAssist AI itself uses an LLM (Gemini or Llama 3.2) to explain retrieved scheme text. In preparing this manuscript, the authors used an AI assistant for language editing and for preparing the diagrams and layout; the authors reviewed all content and take full responsibility for it.

## REFERENCES

[1] M. Maheshwari, M. Roshini, and S. Sharu Dharshini, “A Real-Time Platform for Government Scheme Navigator,” Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE), vol. 14, no. 5, May 2025, doi: 10.17148/IJARCCE.2025.14518.

[2] A. Karunamurthy and S. Barath, “AI-Driven Welfare Scheme Recommendation Using Random Forest and RAG,” Int. J. Latest Technol. Eng., Manage. Appl. Sci. (IJLTEMAS), vol. XV, no. V, May 2026, doi: 10.51583/IJLTEMAS.2026.150500196.

[3] K. Y. Deore et al., “Scheme Seva: An AI-Powered Government Scheme Management Portal,” Int. J. Technol. Appl. Sci. (IJTAS), vol. 17, no. 5, May 2026.

[4] R. K. Pathak et al., “Agentic AI for Government: From NLP-Enhanced Workflows to Autonomous Public Service Agents,” Procedia Comput. Sci., vol. 283, 2026.

[5] S. Vinodh Kumar, P. Pravin, K. P. Sivathanu, and P. Vishal, “Enhancing Public Welfare Access Through an Intelligent AI-based Digital Assistant,” in Proc. ICICV, IEEE, 2026.

[6] L. Gupta et al., “Gov-Assist: Implementing an AI Bridge for User Support,” in Proc. 2nd Int. Conf. Adv. Comput. Emerg. Technol. (ACET), IEEE, 2025.

[7] B. S. Ingole et al., “AI Chatbot Implementation on Government Websites: A Framework for Development, User Engagement, and Security for DHS Website,” in Proc. Int. Conf. Intell. Cybern. Technol. Appl. (ICICyTA), IEEE, 2024.

[8] G. Papageorgiou, V. Sarlis, M. Maragoudakis, and C. Tjortjis, “Enhancing E-Government Services through State-of-the-Art, Modular, and Reproducible Architecture over Large Language Models,” Appl. Sci., vol. 14, no. 18, p. 8259, 2024.

[9] J. Han, J. Lu, Y. Xu, J. You, and B. Wu, “Intelligent Practices of Large Language Models in Digital Government Services,” IEEE Access, vol. 12, 2024.

[10] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 9459–9474.

[11] S. Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.

[12] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in Proc. EMNLP-IJCNLP, 2019.

[13] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” IEEE Trans. Big Data, vol. 7, no. 3, pp. 535–547, 2021.

[14] P. Rebentrost, M. Mohseni, and S. Lloyd, “Quantum support vector machine for big data classification,” Phys. Rev. Lett., vol. 113, no. 13, 130503, 2014.

[15] V. Havlíček et al., “Supervised learning with quantum-enhanced feature spaces,” Nature, vol. 567, pp. 209–212, 2019.