International Journal of Computer Science and Engineering Development
Volume 1, Issue 5 | November - December 2026 | www.ijcsed.com
ISSN: **3139-0862**

SMART FARMER QUEUE MANAGEMENT SYSTEM:

A MULTICHANNEL APPROACH TO ACCESSING THE SLOT FOR AGRICULTURAL AND PUBLIC SERVICES

Sanjay G[1], Hemanth S Gowda [2], Meghana M C[3], Vandana[4], Dr. Arjun B C[5]
Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning),
Malnad College of Engineering, Hassan, Karnataka, India
Dr. Arjun B C Professor and HOD; all other authors are students
Corresponding author: Sanjay G, sg9449070@gmail.com
hemanthsgowdru2006@gmail.com , meghanamc06@gmail.com , gudigarvandana@gmail.com , abc@mcehassan.ac.in .

Abstract

Farmers frequently need to visit multiple public-service offices to access services such as ration assistance, panchayat administration, agricultural-market support, crop insurance, agricultural inputs, certificates, government schemes, land records, and agricultural credit. When queue information is unavailable before or during a visit, users may have difficulty estimating their waiting time and planning their travel. This paper presents the design and implementation of the Smart Farmer Queue Management System, a software-based platform intended to coordinate service tokens across web and telephone-oriented access channels. The system uses a central FastAPI queue backend as the authoritative source for booking and token state, while an n8n workflow can orchestrate integration steps and Exotel provides the telephony/interactive voice response (IVR) layer where a provisioned account and number are available. The architecture is designed to prevent website and telephone bookings from creating independent queues. Its workflow covers language and service selection, taluk-related input, token creation, queue-status communication, and service-state progression. The paper describes the problem, architecture, core workflows, and an evaluation protocol. No quantitative improvement in waiting time, throughput, or user satisfaction is claimed because measured results were not supplied for this manuscript. A reproducible functional and performance evaluation is therefore specified as a necessary next step before submission to a peer-reviewed journal.

Keywords: farmer services, queue management, interactive voice response, FastAPI, workflow automation, multichannel systems, digital agriculture

1. Introduction

Public and agricultural services are often delivered through offices that handle several types of farmer requests. These requests may include agricultural inputs, crop insurance, certificates, scheme information, land records, and financial assistance. Digital agriculture initiatives seek to make information and services more accessible, but a digital application alone does not automatically coordinate the physical service queue at an office. India's Agriculture Information Management System, for example, describes a farmer-oriented platform for submitting applications and routing them through agricultural-department workflows [1]. Research on digital agricultural services likewise identifies information and communication technologies as an important means of supporting agricultural service delivery [2].

A queue-management system addresses a related but distinct operational problem: it organizes service requests, assigns tokens, communicates queue position, and records service progression. Web-only solutions may not serve users who prefer a telephone interaction or have limited access to a data connection. Conversely, a separate telephone queue can fragment the service process and make the displayed queue inconsistent with the queue used by office staff.

This work presents the Smart Farmer Queue Management System, which is designed around a shared queue engine. Website and telephone-originated bookings are intended to be processed by the same backend and database rather than maintaining independent token counters. The project combines a web interface, FastAPI service layer, workflow automation through n8n, and an Exotel IVR integration layer. The main contribution is the integration-oriented architecture and service workflow, rather than a new queueing-theory algorithm.

ISSN: **3139-0862**

2. Problem Statement and Objectives

The problem addressed is the lack of a unified, channel-independent way to register farmer service requests and maintain a consistent queue state across digital and telephone access. When channels are implemented separately, duplicate tokens, inconsistent queue positions, and incomplete administrative visibility can occur.

- Provide a central queue backend for bookings arriving from more than one access channel.
- Support service selection and a telephone-oriented IVR flow for users who cannot or do not wish to use the website.
- Generate tokens and maintain queue status in one authoritative data store.
- Support configurable service progression, including completion confirmation and a bounded extension/no-response workflow where implemented.
- Use workflow automation to connect external events and APIs without moving authoritative queue logic out of the backend.
- Define a measurable evaluation procedure for correctness, response time, concurrency, and usability.

3. Related Work

Digital agriculture platforms provide a foundation for organizing farmer-facing applications and administrative workflows. The National Informatics Centre's description of the Agriculture Information Management System notes farmer registration, service applications, departmental processing, and configurable workflows [1]. Bhaskar discusses the role of information and communication technologies in the development of digital agricultural services in India [2]. These initiatives establish the relevance of accessible digital service delivery, although they do not by themselves establish that a particular office queue is synchronized across web and voice channels.

Queue-management research has examined the replacement of manual ticketing and physical queues with digital registration, token assignment, monitoring, and reporting. Recent system-design studies illustrate common functional patterns such as centralized records and queue-status monitoring [3], while website-based queue research has also evaluated waiting-time and service-capacity measures [4]. These studies suggest that functional implementation and empirical evaluation should be treated separately: a working prototype demonstrates feasibility, but claims about reduced waiting time require observations or controlled tests.

The proposed system focuses on channel convergence: all supported booking channels submit requests to the same queue service. Its distinguishing design goal is not simply to provide an online token, but to avoid treating the website and telephone IVR as separate queue owners. The system should be evaluated against this consistency requirement as well as ordinary functional and performance requirements.

4. System Architecture

The system follows a layered architecture. The access layer contains the farmer-facing website and the telephone IVR configured through Exotel when telephony provisioning is available. The integration layer uses n8n to receive events, route selected values, and invoke backend APIs. The application layer is implemented using FastAPI and contains the queue-related business rules. A shared database stores booking and service state. Administrative and farmer-facing dashboards read queue information from the same backend.

Figure 1 should be prepared from the actual deployed implementation before journal submission. The logical data flow is: Farmer → Website or Exotel IVR → n8n/integration endpoint (where used) → FastAPI Queue API → Shared database → dashboards and notifications. The IVR provider supplies telephony and voice-menu capabilities; n8n does not itself provide a telephone number. The queue backend remains responsible for token assignment and authoritative state transitions.

4.1 Shared Queue Principle

Each booking channel submits a booking request to the same backend endpoint or service operation. The backend validates the request, assigns a token, stores the booking, and returns a result that the originating channel can communicate to the farmer. The design should enforce token uniqueness and queue ordering at the database or transaction level. An integration workflow must not independently generate a second token or maintain a duplicate queue.

4.2 Main Functional Components

- Farmer access interface: service selection and display of token/queue information.
- Telephone IVR interface: language and service menu choices, collection of required keypad inputs, and spoken confirmation where configured.

ISSN: **3139-0862**

- Integration workflow: maps channel input to backend requests, handles expected responses, and reports errors.
- Queue API: validates bookings, creates tokens, updates status, and exposes the current queue to authorized clients.
- Database: persists booking, service, token, timestamps, and status information.
- Administrative view: enables authorized staff to monitor and manage service progression.
- Notification/automation component: supports alerts or scheduled follow-up only when configured and tested.

5. Workflow and Queue-State Management

5.1 Booking Workflow

A farmer selects a service through the website or telephone menu. The request is normalized into a common backend format, including the selected service and the identifying or location information required by the implementation. FastAPI validates the request and creates a queue record. The assigned token is returned to the requesting channel. The same record should then be visible to the administrative interface and any other channel that reads the shared queue.

5.2 Service Progression

The project workflow is intended to support explicit queue states such as booked, waiting, called, in service, confirmation pending, extended, completed, and cancelled, subject to the states actually present in the final code. Where the completion-confirmation feature is enabled, the system requests confirmation when the configured service period expires. A reported completion advances the queue; a negative response can extend the service interval; and a missing response may trigger a bounded retry policy before the record is advanced according to the configured rules. These transitions should be performed by the backend or its reliable background worker, not by a browser timer alone.

5.3 Consistency and Failure Handling

The main consistency requirement is that a booking accepted through one channel must not disappear from the other channel's queue view. Tests should cover duplicate submissions, simultaneous booking requests, malformed keypad input, integration timeouts, and retries. Requests should use validation and idempotency protection where possible so that a repeated webhook does not unintentionally create multiple bookings.

6. Implementation Technology

FastAPI serves as the backend API framework, n8n is used for workflow orchestration, and Exotel is the intended telephony/IVR provider. The web interface and database are part of the project implementation. The exact frontend framework, database engine, hosting configuration, and deployment versions should be inserted from the actual repository and environment before publication; they are not assumed in this draft.

A critical architectural decision is to keep queue ownership in FastAPI and the shared database. Workflow automation is useful for integration and notification tasks, but duplicating token-generation logic in n8n would weaken consistency. Likewise, telephony provisioning and production call handling depend on the provider account, number status, IVR configuration, and applicable onboarding requirements.

7. Evaluation Methodology

A publishable evaluation should report reproducible measurements rather than rely only on screenshots or statements that the application works. The following test plan is proposed for execution against the final system. Values must be filled from observed runs; they must not be estimated.

Test	Procedure	Report
Functional booking	Submit valid website and IVR-originated bookings; verify token returned and record stored.	Success rate; token uniqueness.
Cross-channel visibility	Create bookings from each channel and inspect both farmer/admin queue views.	Number of consistency failures.
Concurrent booking	Submit a controlled batch of simultaneous booking requests.	Duplicate tokens; failed requests; response-time distribution.
IVR input validation	Test valid and invalid language, service, and location/keypad inputs.	Correct acceptance/rejection rate.
Queue-state transitions	Exercise call, in-service, completion, extension, timeout, and cancellation paths that are implemented.	Expected versus observed transitions.
API performance	Run repeated requests under a documented load and hardware/network setup.	Median and 95th-percentile response time; throughput.
Recovery and retries	Simulate timeout, repeated webhook, and temporary backend failure.	Duplicate records and recovery outcome.
Usability review	Ask representative users or evaluators to complete defined tasks, with appropriate consent	Task completion, time, and structured feedback.

ISSN: **3139-0862**

8. Results and Discussion

The present manuscript documents the system design and its intended end-to-end workflow. Quantitative test logs, load-test outputs, user-study results, and measured before/after waiting times were not provided for this draft. Accordingly, no numerical claims are made about percentage reductions in waiting time, improvements in throughput, accuracy, scalability, or farmer satisfaction. This is an important evidence limitation, not a negative test result.

After testing, this section should report (i) the number of tests run and passed, (ii) the test environment and software versions, (iii) latency and throughput measurements, (iv) cross-channel consistency results, and (v) any failures or constraints. If the goal is to claim improved service efficiency, collect baseline and system-assisted observations under comparable conditions and explain the sample size and measurement procedure. If no field trial has taken place, describe the work as a prototype evaluation rather than a field-validated deployment.

9. Limitations and Threats to Validity

- The manuscript does not yet include measured performance results or a controlled comparison with a manual queue.
- A configured IVR flow does not by itself prove that a public toll-free number is provisioned or that production calls succeed; this should be verified and documented.
- User accessibility, language quality, network availability, and the reliability of voice recognition or keypad input require testing with intended users.
- Concurrency safety depends on the actual database transactions, token-generation implementation, and deployment configuration.
- The system coordinates bookings and queue state; it does not itself guarantee shorter real-world waiting times, faster government service, or official approval of any farmer application.
- Any user study should follow institutional requirements for consent, privacy, and handling of personal information.

10. Conclusion and Future Work

This paper presented the design of a Smart Farmer Queue Management System intended to coordinate farmer service bookings across a website and a telephone-oriented IVR channel. Its central design principle is a single FastAPI queue service and shared data store, with n8n used for integration workflows and Exotel used for telephony where provisioned. The architecture is intended to provide consistent token assignment, queue visibility, and controlled service-state progression without maintaining separate queues for different channels.

The next research step is empirical validation. Functional tests should establish that bookings from each channel produce consistent records and that state transitions behave as specified. Performance tests should measure latency and concurrent request handling, while usability evaluation should investigate whether the interface and IVR flow are understandable to intended users. Field deployment and measured waiting-time comparisons would be required before claiming real-world efficiency gains.

References

1. National Informatics Centre, Government of India, “Agriculture Information Management System.” Available: https://www.nic.gov.in/project/aims/ (accessed 24 October 2026).
2. G. Bhaskar, “Digital Agricultural Services: The Future of Indian Farming,” MANAGE Journal of Agricultural Extension Management, July–December 2022. Available: https://www.manage.gov.in/publications/journal/jul-dec2022.pdf.
3. F. S. Tofa, N. M. Baba, M. S. Rabiu, and S. Y. Muhammad, “Design and Implementation of an Automated Queue Management System,” Scholarly Journal of Science and Technology Research and Development, vol. 5, no. 3, pp. 159–169, 2026, doi: 10.46654/b6a8mf51.
4. A. M. Iqbal and D. P. Sari, “Optimization of Services Through the Development of a Website-Based Online Queueing System,” INOVTEK Polbeng – Seri Informatika, 2025, doi: 10.35314/pdx5hg42.
5. IEEE Author Center, “Ethical Requirements.” Available: https://journals.ieeeauthorcenter.ieee.org/become-an-ieee-journal-author/publishing-ethics/ethical-requirements/ (accessed 24 October 2026).

AUTHOR AND RESEARCH-INTEGRITY STATEMENT

Dr. Arjun B C is listed as Professor and HOD, and Sanjay G, Hemanth S Gowda, Vandana, and Meghana M C are listed as students of Malnad College of Engineering, Hassan, Karnataka, India. The corresponding author for this manuscript is Sanjay G (sg9449070@gmail.com; 9449070373).

This manuscript presents the **Smart Farmer Queue Management System**, a multichannel digital platform designed to improve queue management for farmers accessing public and agricultural services at the taluk level. The proposed system integrates a React-

ISSN: **3139-0862**

based web application with an Exotel-powered toll-free telephone booking facility, enabling farmers to select their preferred language, choose a service, enter the relevant six-digit taluk code, and receive a queue token. The system is designed to support a shared queue-management workflow, with an administrative dashboard for monitoring and managing service queues.

AI-assisted coding and writing tools may be used during system implementation and manuscript preparation. The authors remain responsible for the system architecture, source verification, implementation, experimental design, numerical results, interpretation, citations, and final submission. Any performance measurements, usability findings, or efficiency improvements reported for the Smart Farmer Queue Management System must be independently tested and reproduced before being presented as validated scientific results. Claims about reduced waiting time, improved accessibility, or increased operational efficiency must be supported by appropriate experimental evidence.