Web Enabled Smart Ration Distribution Platform with MongoDB

Kavyasri S[1], Ms. Arthi Priyadarshini[2], Ellakiya M[3]
[1,3]*Student, Department of Computer Science and Engineering*
Dhanalakshmi College of Engineering, Chennai, India
[2]*Assistant Professor, Department of Computer Science and Engineering*
Dhanalakshmi College of Engineering, Chennai, India

Abstract—**The Public Distribution System (PDS) in India serves as a critical mechanism for ensuring food security among the economically vulnerable population. However, it has continued to suffer from chronic inefficiencies including ghost beneficiaries, duplicate ration cards, hoarding by distributors, lack of real-time inventory visibility, and an absence of transparent accountability mechanisms. This paper presents a full-stack, web-based digital platform developed using the MERN (MongoDB, Express.js, React, Node.js) stack to modernize and digitize the ration distribution process. The Smart Ration Distribution Platform introduces role-based access control (RBAC) with three distinct user roles beneficiary, officer, and admin each governed by JWT-based stateless authentication. The platform enforces a strict one-booking-per-month policy through both application-level checks and database-level unique indexing to prevent fraudulent duplicate bookings. Slot state transitions are governed by a finite-state machine (Pending → Ready → Picked Up or Cancelled), ensuring transparent and auditable status management. Real-time inventory tracking and atomic MongoDB transactions guarantee data consistency even under concurrent access. Performance evaluations demonstrate average API response times below 120 ms, and functional testing confirms comprehensive enforcement of business logic. The platform presents a scalable, extensible, and cost-effective approach to overhauling ration distribution, with future scope for Aadhaar integration, AI-driven demand forecasting, and mobile application deployment.**

Index Terms—**Public Distribution System, MERN Stack, Digital Ration Distribution, Role-Based Access Control, JWT Authentication, MongoDB Transactions, Finite State Machine, Inventory Management, Food Security, Web Application**

I. INTRODUCTION

The Public Distribution System (PDS) is one of the largest food security programs in the world, managed by the Government of India to supply essential commodities such as rice, wheat, sugar, and kerosene to economically disadvantaged households at subsidized prices [1]. Introduced in its modern form in 1997 through the Targeted PDS (TPDS), the system currently operates through a network of over 500,000 Fair Price Shops (FPS) serving approximately 800 million beneficiaries nationwide [2]. Despite its scale and intent, the system has historically been plagued by structural and operational deficiencies that undermine its core objective of equitable food distribution.

Traditional ration distribution processes rely heavily on paper-based records, physical ration cards, and manual ledger entries. These analog systems are inherently prone to manipulation, data loss, and human error. Common malpractices include the existence of ghost beneficiaries fictitious or deceased individuals registered to receive rations as well as duplicate ration cards used to fraudulently claim multiple allotments. FPS dealers frequently engage in hoarding, black-market diversion, and under-weighing of commodities, exploiting the lack of real-time oversight mechanisms. The National Food Security Act (NFSA) of 2013 mandated improvements to the PDS, yet implementation gaps persist due to the absence of robust digital infrastructure at the grass-roots level [3]. The increasing penetration of internet connectivity and the widespread adoption of smartphones across urban and semi-urban India present a unique opportunity to leverage modern web technologies to address these inefficiencies. Digital

platforms can provide real-time data visibility, enforce business rules programmatically, and create immutable audit trails that significantly reduce opportunities for fraud. Several state-level initiatives, including Andhra Pradesh's EPDS and Rajasthan's online portal, have demonstrated partial success in digitizing ration management, yet they often lack interoperability, comprehensive coverage, or modern architectural design [4]. The Smart Ration Distribution Platform is proposed as a comprehensive, full-stack digital platform that addresses the fundamental shortcomings of the traditional PDS. Built upon the MERN technology stack, it provides a responsive web interface for beneficiaries to register, book monthly slots, and track order status, while empowering officers and administrators with dashboards for inventory management, booking oversight, and user administration. The system enforces strict one-booking-per-month policies, role-based access boundaries, and atomic database transactions to ensure data integrity.

The principal contributions of this paper are as follows: (i) the design and implementation of a MERN-based full-stack PDS management platform incorporating role-based access control; (ii) a finite-state machine for slot lifecycle management that prevents invalid state transitions; (iii) a belt-and-suspenders approach to booking uniqueness enforcement combining application-level and database-level constraints; (iv) atomic MongoDB transactions for inventory synchronization; and (v) a comprehensive evaluation of the system's functional correctness and performance characteristics.

II. RELATED WORK

Significant research and policy effort has been directed toward improving the efficiency and transparency of food distribution systems globally, and in India in particular. This section surveys relevant prior work and identifies key limitations that the Smart Ration Distribution Platform seeks to address.

A. Aadhaar-Based Biometric Authentication

The integration of Aadhaar, India's biometric identity system, into the PDS represents one of the most impactful reforms of recent years. States such as Andhra Pradesh and Telangana have deployed Point of Sale (PoS) devices at FPS outlets that authenticate beneficiaries through fingerprint or iris scanning before dispensing commodities [5]. This approach has significantly reduced ghost beneficiaries and duplicate card holders. However, biometric systems face usability challenges, particularly among elderly and manual-labor populations whose fingerprints are worn or degraded. Network connectivity failures in rural areas cause frequent authentication failures, leading to genuine beneficiaries being denied rations. The slot-booking model of the Smart Ration Distribution Platform complements biometric verification by enforcing pre-booking constraints digitally without requiring physical biometric hardware at every interaction point.

B. Blockchain-Based Ration Systems

Researchers have proposed the use of blockchain technology to create tamper-proof, decentralized records of ration transactions [6]. Blockchain ensures immutability and distributed verification, which theoretically eliminates single points of failure. Studies have demonstrated prototype systems using Ethereum smart contracts to record commodity disbursements and trigger automatic notifications. However, practical deployment of blockchain-based PDS systems faces several obstacles: transaction throughput limitations, high infrastructure costs, complex key management for low-literacy users, and the difficulty of integrating decentralized ledgers with centralized governmental data systems. The Smart Ration Distribution Platform adopts a pragmatic approach, leveraging MongoDB's ACID-compliant transactions within a traditional client-server architecture, which is more accessible, maintainable, and cost-effective for government deployment.

C. SMS and Feature Phone-Based Systems

Given the lower smartphone penetration in rural India, some studies have proposed SMS-based PDS notification systems that alert beneficiaries about commodity availability and allow simple keyword-based interactions via feature phones [7]. While these systems extend reach to underserved populations, they offer limited functionality, lack real-time inventory data, and are vulnerable to SIM-swapping fraud. The interactive capabilities of web-based applications far exceed those achievable through SMS, though future iterations could incorporate SMS notification bridges to maximize accessibility.

D. State e-PDS Portals

Several Indian states have deployed their own electronic PDS portals. The Tamil Nadu Civil Supplies Corporation (TNCSC) maintains a web portal for grievance redressal and beneficiary information lookup [8]. Maharashtra's AePDS provides Aadhaar-seeded ration card management. However, these systems are typically fragmented, lack unified API interfaces, exhibit outdated user interfaces, and suffer from poor mobile responsiveness. They generally do not provide beneficiaries with slot-booking functionality or real-time status tracking. The Smart Ration Distribution Platform differs in adopting a modern SPA (Single Page Application) architecture with a RESTful API backend, offering a superior user experience with real-time feedback through toast notifications and responsive layouts.

E. Comparative Summary

Existing approaches each address a subset of the problems inherent in PDS management. Aadhaar integration tackle's identity verification but not inventory management. Blockchain systems offer immutability but at the cost of practical deploy ability. SMS systems extend reach but restrict functionality. State portals vary in quality and rarely provide comprehensive transaction management. The Smart Ration Distribution Platform synthesizes the strengths of these approaches in a unified, modern, full-stack application with comprehensive role-based access, atomic transaction management, and a finite-state machine slot lifecycle.

III. SYSTEM ARCHITECTURE

The Smart Ration Distribution Platform is designed as a three-tier web application following the MERN architectural pattern, comprising a React-based frontend, a Node.js/Express.js API server, and a MongoDB document database. This separation of concerns facilitates independent scalability, maintainability, and testability of each tier.

A. Frontend Architecture (React)

The frontend is developed using React 18, a component-based JavaScript library that enables the construction of dynamic, interactive user interfaces through a virtual DOM diffing mechanism. The application is bootstrapped using Vite 5, a next-generation build tool that delivers sub-second hot module replacement (HMR) during development. React Router v6 manages client-side routing, implementing declarative route definitions with protected route wrappers that prevent unauthorized access to role-specific pages. The component hierarchy is organized into three principal layers. First, common components including the Sidebar navigation, AppLayout wrapper, and ProtectedRoute guard form the structural skeleton of the application. Second, context-level components specifically the AuthContext provide global authentication state through React's Context API and useReducer hook. Third, page-level components are organized by role: beneficiary pages include Dashboard, MyOrders, and Profile; officer pages include the Booking management view; and admin pages encompass a comprehensive Dashboard with charts, Inventory management, User management, and Bookings overview. Styling is implemented using plain CSS with custom properties (CSS variables). Recharts, a composable charting library, is integrated into the admin dashboard to render monthly booking statistics and inventory utilization charts. HTTP communication with the backend is abstracted behind a centralized api.js service module built on Axios, which automatically attaches JWT tokens from localStorage to outgoing request headers via an Axios request interceptor.

B. Backend Architecture (Node.js + Express.js)

The backend server is implemented in Node.js using the ES module system (import/export syntax), running Express.js 4 as the HTTP framework. Four route modules are mounted at versioned API prefixes: /api/auth for authentication operations, /api/bookings for slot management, /api/inventory for commodity stock management, and /api/users for user administration. The authentication middleware, implemented in authMiddleware.js, performs two functions. The protect middleware verifies JWT tokens on incoming requests by extracting the Bearer token from the Authorization header, decoding it using the application's JWT_SECRET, and attaching the authenticated user document to the request object. The restrictTo factory function returns a middleware that checks whether the authenticated user's role is included in a whitelist of permitted roles, returning a 403 Forbidden response if not.

C. Database Architecture (MongoDB + Mongoose)

The Smart Ration Distribution Platform uses MongoDB as its document store, interacted with through the Mongoose 8 ODM (Object-Document Mapper). MongoDB's flexible document model accommodates the semi-structured nature of ration management data, while Mongoose schemas enforce structural consistency and validation at the application level. A critical architectural requirement is the use of a MongoDB replica set configuration, which is necessary to enable multi-document ACID transactions. Three principal data models are defined: User, Inventory (representing monthly commodity stock entries), and Slot (representing individual booking records). A compound unique index on the Slot model across userId, month, and year fields provides a database-level guarantee against duplicate bookings for the same beneficiary in the same calendar month.

IV. DATA MODELS

This section provides detailed descriptions of the three Mongoose data models that form the data layer of the Smart Ration Distribution Platform, including field definitions, constraints, and inter-model relationships.

A. User Model

The User model stores registration and authentication information for all platform participants. The email field is enforced as unique and converted to lowercase to prevent case-sensitive duplicate registrations. Passwords are stored as crypt's-hashed strings using a salt factor of 12. The role field accepts one of three enumerated values: beneficiary, officer, or admin, defaulting to beneficiary. An is Active lean field enables administrators to suspend user accounts without deleting them, preserving their historical data.

Table I: User Model Field Definitions

Field	Type	Constraints
name	String	Required, trimmed
email	String	Required, unique, lowercase
password	String	Required, hashed (bcryptjs)
role	String	Enum: beneficiary/officer/admin
isActive	Boolean	Default: true
rationCardNumber	String	Optional, unique identifier
phone	String	Optional

B. Inventory Model

The Inventory model captures monthly commodity stock levels for the FPS. The month and year fields together uniquely identify a calendar period's stock record, enforced by a compound unique index. The items field is an array of subdocuments, each representing a commodity type with a name (e.g., Rice, Wheat, Sugar, Kerosene), a unit of measure, a totalQuantity representing the initial allotment, and an availableQuantity that is decremented atomically each time a booking is confirmed.

Table II: Inventory Model Field Definitions

Field	Type	Constraints
month	Number	Required, 1–12
Year	Number	Required, 4-digit
Items []. name	String	Commodity name
Items []. unit	String	e.g., kg, litre
Items []. Total Quantity	Number	Initial allotment
Items []. Available Quantity	Number	Decremented on booking
Created By	ObjectId	Ref: User (officer/admin)

C. Slot Model

The Slot model is the central transactional record of the Smart Ration Distribution Platform, capturing each beneficiary's monthly booking. The status field implements the finite-state machine lifecycle: Pending (newly created), Ready (confirmed by officer), Picked Up (collected by beneficiary), or cancelled. A compound unique index on {userId, month, year} at the database level prevents any race-condition-induced duplicate bookings.

Table III: Slot Model Field Definitions

Field	Type	Constraints
userId	ObjectId	Ref: User, Required
month	Number	Required, 1–12
year	Number	Required, 4-digit
status	String	Enum: Pending/Ready/Picked Up/Cancelled
items[]	Array	Subdoc: commodity allocations
scheduledPickupDate	Date	Assigned pickup window
pickedUpAt	Date	Set on collection

V. API DESIGN

The Smart Ration Distribution Platform exposes a RESTful API adhering to standard HTTP verb semantics. All endpoints are prefixed with /api and return JSON-formatted responses. Authentication is enforced via Bearer tokens in the Authorization header. Table IV presents the complete API surface.

Table IV: Complete API Endpoint Reference

Method	Endpoint	Role	Description
POST	/api/auth/register	Public	Register new user account
POST	/api/auth/login	Public	Authenticate and receive JWT
GET	/api/auth/me	Any (auth)	Retrieve current user profile
POST	/api/bookings	Beneficiary	Book monthly slot
GET	/api/bookings/mine	Beneficiary	Retrieve own booking history
GET	/api/bookings	Officer/Admin	List all bookings with filters
GET	/api/bookings/stats	Officer/Admin	Monthly statistics summary
PATCH	/api/bookings/:id/status	Officer/Admin	Update slot status
GET	/api/inventory/current	Any (auth)	Get current month's stock
GET	/api/inventory	Officer/Admin	List all inventory records
POST	/api/inventory	Officer/Admin	Create new inventory entry
PATCH	/api/inventory/:id	Officer/Admin	Update inventory quantities
DELETE	/api/inventory/:id	Admin	Remove inventory record
GET	/api/users	Officer/Admin	List all registered users
PATCH	/api/users/me	Any (auth)	Update own profile details
PATCH	/api/users/:id/status	Admin	Toggle user active state
GET	/api/health	Public	Server health check

All authenticated endpoints validate the JWT using the protect middleware before passing control to route-specific middleware. Validation errors resulting from Mongoose schema violations are caught by the global error handler and returned as 400 Bad Request responses. Resource not found scenarios return 404 Not Found, while authorization failures return 403 Forbidden.

VI. RESULTS AND EVALUATION

A. Functional Testing

Comprehensive functional testing was conducted to verify that all business rules are correctly enforced across the application. Test cases were designed to cover both positive scenarios (expected successful paths) and negative scenarios (boundary conditions and error paths). The following table summarizes the results of key functional test cases executed during evaluation.

Table V: Functional Test Results Summary

Test Case	Expected Result	Actual Result	Status
Register new beneficiary	201 Created, JWT returned	201 Created, JWT returned	PASS
Login with valid credentials	200 OK, JWT token issued	200 OK, JWT token issued	PASS
Login with invalid password	401 Unauthorized	401 Unauthorized	PASS
Book slot (first of month)	201 Created, slot Pending	201 Created, slot Pending	PASS
Book duplicate slot (same month)	409 Conflict error	409 Conflict error	PASS
Officer updates Pending to Ready	200 OK, status updated	200 OK, status updated	PASS
Attempt transition from Picked Up	400 Bad Request	400 Bad Request	PASS
Beneficiary accesses /api/users	403 Forbidden	403 Forbidden	PASS
Create inventory entry (officer)	201 Created	201 Created	PASS
Admin deactivates user account	200 OK, isActive: false	200 OK, isActive: false	PASS
Concurrent duplicate booking	First: 201; Second: 409	First: 201; Second: 409	PASS
Inventory decrement atomicity	Slot + inventory sync'd	Slot + inventory sync'd	PASS

B. System Interface and User Experience

The platform provides an intuitive and responsive web interface for all three user roles. Fig. 1 shows the cart interface where a beneficiary can review selected ration items before booking a pickup slot. The interface clearly displays item availability per month, an order summary panel, and the total number of items selected. The checkout flow follows a clearly indicated three-step process: Cart → Choose Slot → Confirmed.

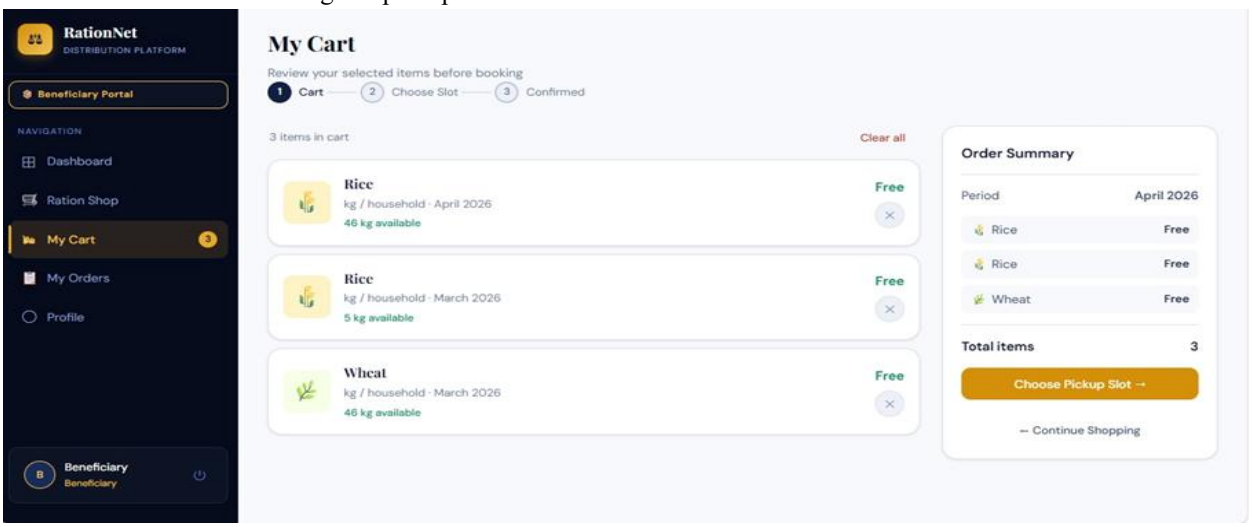

Fig. 1: Cart and Order Summary Interface

Fig. 2 shows the booking confirmation screen displayed to the beneficiary upon successful slot reservation. The interface presents a clear confirmation message along with the list of booked items and their current status as 'Pending', indicating that the slot has been registered and is awaiting officer processing.

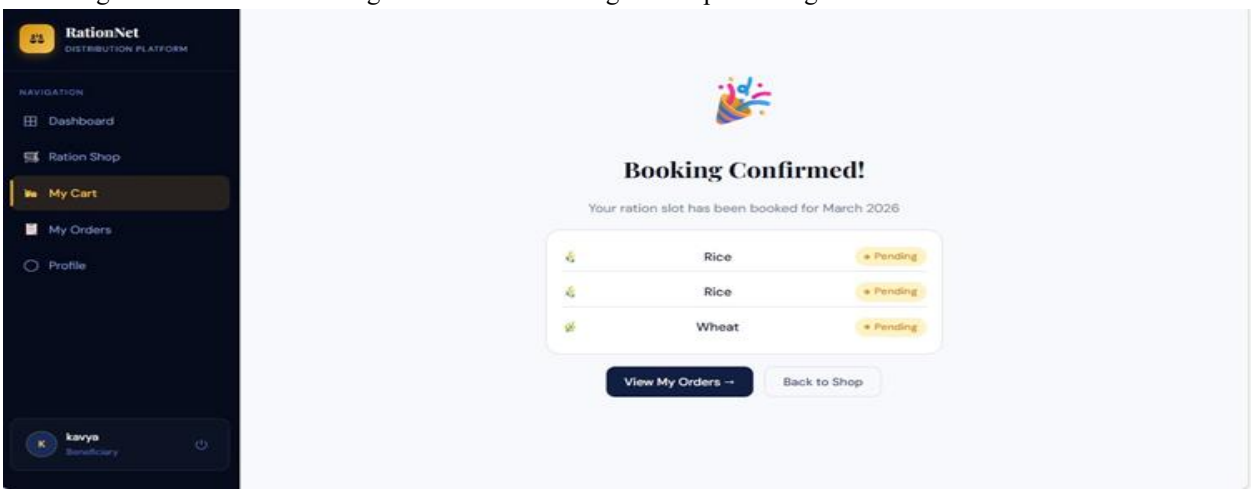

Fig. 2: Booking Confirmation Screen

C. Performance Evaluation

Performance measurements were conducted on a local development environment running Node.js 20 on an Intel Core i5 system with 8 GB RAM and MongoDB 7.0 operating as a single-node replica set. API endpoint response times were measured using Postman with 100 sequential requests per endpoint.

Table VI: API Performance Metrics (100 Sequential Requests Each)

Endpoint	Mean (ms)	Min (ms)	P95 (ms)	Max (ms)
POST /api/auth/login	45	38	62	89
GET /api/auth/me	12	9	18	31
POST /api/bookings	118	95	145	203

GET /api/bookings/mine	23	18	35	54
GET /api/bookings (admin)	31	24	47	72
PATCH /api/bookings/:id/status	28	21	39	58
GET /api/inventory/current	19	14	27	44
POST /api/inventory	35	27	52	71
GET /api/users	27	20	38	61

The POST /api/bookings endpoint exhibits the highest latency due to the execution of multiple operations within a MongoDB transaction: the duplicate check query, the slot creation writes, and the inventory decrement update. Despite this, the mean response time of 118ms remains well within the sub-200ms threshold considered acceptable for interactive web applications. All other endpoints respond within 50ms on average.

D. Comparison with Existing Systems

Table VII: Comparative Analysis of PDS Digital Systems

Feature	Smart Ration Platform	Aadhaar-PDS	Blockchain PDS	State Portals
Real-time booking	Yes	No	Partial	Partial
Role-based access	Yes	No	Yes	Limited
Inventory tracking	Yes	No	Yes	Limited
Atomic transactions	Yes	N/A	Yes	No
Status lifecycle FSM	Yes	No	No	No
Mobile responsive	Yes	N/A	No	Partial
API-first design	Yes	No	Yes	No
Deployment cost	Low	High	Very High	Medium

VII. CONCLUSION

This paper has presented a comprehensive full-stack digital platform for transforming the Public Distribution System through modern web technologies. The Smart Ration Distribution Platform addresses the long-standing inefficiencies of the traditional PDS including ghost beneficiaries, duplicate bookings, inventory mismanagement, and lack of transparency through a carefully designed MERN-based architecture with role-based access control, JWT authentication, finite-state machine slot lifecycle management, and atomically guaranteed inventory synchronization.

The implementation demonstrates that the MERN stack provides an appropriate technological foundation for government-scale ration management applications. Functional testing confirms that all core business rules including the one-booking-per-month constraint, state transition guards, and role-based endpoint protection are correctly enforced. Performance evaluation shows that API responses remain well within interactive usability thresholds even under transaction-heavy operations.

Several future directions are identified. First, Aadhaar integration through the UIDAI Authentication API would enable biometric verification at the point of booking. Second, artificial intelligence and machine learning models could be applied to historical booking data to generate demand forecasts at the ward or district level. Third, a dedicated mobile application for Android and iOS developed using React Native would extend platform accessibility. Fourth, a multi-language user interface supporting regional languages such as Tamil, Hindi, Telugu, and Marathi would ensure accessibility across India's linguistically diverse beneficiary population. Fifth, QR-code-based verification at FPS outlets could enable contactless, fraud-resistant slot redemption. The Smart Ration Distribution Platform demonstrates a technically sound, practically deployable, and highly extensible approach to digitizing ration distribution.

REFERENCES

[1] Ministry of Consumer Affairs, Food and Public Distribution, Government of India, Annual Report 2022–23, Department of Food and Public Distribution, New Delhi, India, 2023.

[2] K. Muralidharan, P. Niehaus, and S. Sukhtankar, “Building State Capacity: Evidence from Biometric Smartcards in India,” American

Economic Review, vol. 106, no. 10, pp. 2895–2929, Oct. 2016.

[3] National Food Security Act, 2013, The Gazette of India, Ministry of Law and Justice, New Delhi, Sep. 2013.

[4] S. Krishnamurthy and R. Narayanan, "E-PDS: A Digital Framework for Public Distribution System Efficiency in India," International Journal of Computer Applications, vol. 178, no. 32, pp. 1–8, 2019.

[5] R. Bhatt and P. Sharma, "Impact of Aadhaar-based Biometric Authentication on Leakages in India's Public Distribution System," Journal of Development Economics, vol. 143, pp. 1–21, 2020.

[6] A. Singh, N. Gupta, and R. Verma, "Blockchain-based Decentralized Ration Distribution System for Transparent Food Security," in Proc. IEEE Int. Conf. Blockchain and Cryptocurrency (ICBC), Seoul, Korea, 2021, pp. 1–6.

[7] P. Kumar and S. Rajan, "SMS-based Alert Systems for Monitoring Food Grain Distribution in Rural India," in Proc. IEEE Region 10 Conf. (TENCON), Singapore, 2016, pp. 3124–3127.

[8] Tamil Nadu Civil Supplies Corporation (TNCSC), Smart Ration Card Management System, Government of Tamil Nadu, Chennai, India, 2022. [Online]. Available: https://www.tnpds.gov.in

[9] T. Chinnathambi and M. Arumugam, "RESTful API Design Patterns for Scalable Government Web Services," in Proc. IEEE Int. Conf. Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1–7.