Infortech, Vol. 8, No. 1, June 2026, E-ISSN: 2715-8160, Page. 93-102
Published: LPPM Universitas Bina Sarana Informatika
JURNAL INFORTECH
website: http://jurnal.bsi.ac.id/index.php/infortech/

Development of a Web-Based Smart Ecosystem Platform for Sustainable Public Service Automation

Shoffan Saifullah[1], Muhammad Iqbal[2], Lisnawanty[3], Weiskhy Steven Dharmawan[4], Fahmi Raditya[5]

[1]Faculty of Computer Science, AGH University of Krakow, Krakow, Poland
[1]Department of Informatics, Veteran National Development University, Yogyakarta, Indonesia
[2,3,5]Computer Science Program, Pontianak Campus, Faculty of Engineering and Computer Science, Bina Sarana Informatika University, Pontianak, Indonesia
[4]Accounting Information Systems Program, Pontianak Campus, Faculty of Engineering and Computer Science, Bina Sarana Informatika University, Pontianak, Indonesia

ARTICLE INFORMATION

Artikel History:
Received: 20-05-2026
Revised: 03-06-2026
Accepted: 22-06-2026
Available Online: 22-06-2026

Keyword:
Digital Public Service
Service Automation
Smart Platform
E-Government
Automatic Document Validation

ABSTRACT

Public service delivery in Indonesia continues to face fundamental challenges including inefficient manual administrative processes, error-prone document validation, and the absence of real-time tracking systems. This research aims to develop the PANDU (Pelayanan Publik Digital Terpadu) platform as a web-based smart ecosystem that automates public services sustainably. The platform is built using the Waterfall development method with Model-View-Controller architecture based on Laravel 12 framework, Filament 4.0 administration panel, and Tailwind CSS 4.0 responsive interface. Four main smart features are integrated: automatic document validation, duplicate request detection within a 30-day window, category-based related service recommendations, and automatic priority calculation using multi-criteria scoring algorithm. The platform produces three separate panels for citizens, officers, and administrators, equipped with real-time tracking system through public API and configurable multi-step approval workflows. Black box testing results using equivalence partitioning technique demonstrate one hundred percent functional success rate, while usability evaluation using System Usability Scale yields an average score in the Excellent category with Acceptable acceptability level. The PANDU platform successfully bridges the gap between smart government theoretical frameworks and operational implementation, providing significant contribution to accelerating sustainable digital transformation of public services in Indonesia.

Corresponding Author:

Muhammad Iqbal,
Computer Science Program, Pontianak Campus, Faculty of Engineering and Computer Science,
Bina Sarana Informatika University,
Abdul Rahman Saleh Street, Number 18, Bangka Belitung Laut, Pontianak, Indonesia, 78124,
Email: iqbal.mdq@bsi.ac.id

INTRODUCTION

Digital transformation in the public sector has become a global strategic priority in efforts to improve the efficiency, transparency, and accessibility of public services for citizens. Haug et al. (2023), through a systematic review of 164 studies, demonstrate that digitally driven changes in the public sector have significant transformative impacts, ranging from incremental changes in administrative processes to cumulative effects on society as a whole. This phenomenon is driving governments in various countries to shift from conventional e-government models toward the concept of smart government, which leverages smart technologies such as artificial intelligence (AI), the Internet of Things (IoT), cloud computing, and process automation to provide public services that are more personalized, efficient, and sustainable (Hujran et al., 2023).

DOI: https://doi.org/10.31294/infortech.v8i1.

In the Indonesian context, the progress of digital government transformation shows a positive trend. The national Electronic-Based Government System (SPBE) Index increased from 2.34 in 2022 to 2.79 in 2023, while Indonesia's ranking in the 2024 UN E-Government Survey rose by 13 positions from 77th to 64th out of 193 countries (Gusman, 2024). Nevertheless, the implementation of e-government at the regional level still faces various fundamental challenges. Nyeleker et al (2025) identified that the transition from e-government to digital government in Indonesia requires continuous digital innovation, where platforms such as SmartASN are developed to address the increasing need for digital services, particularly in the wake of the COVID-19 pandemic. However, there remains a significant gap between the availability of digital platforms and the effectiveness of the services provided to the public.

The main challenges in the delivery of public services in Indonesia today encompass several critical aspects. First, administrative processes that remain manual and bureaucratic result in lengthy and inefficient service delivery times. Latupeirissa et al (2024), in a comprehensive review of public service digitization initiatives, found that digital transformation can improve efficiency through the implementation of digital services and workflow reorganization; however, many local government agencies have not yet optimally adopted this approach. Second, manual document validation is prone to errors and is time-consuming. Third, the absence of a real-time tracking system prevents citizens from monitoring the status of their service requests, thereby reducing trust and satisfaction with government services. Ye et al (2023) emphasize that the quality of digital government services, as measured by citizen feedback, is a crucial factor in determining the success of digital service adoption, where a citizen-feedback-based analytical framework can provide more targeted managerial insights to improve public satisfaction.

The integration of smart technologies into public services offers a promising solution to these challenges. Al-Ansi et al. (2024), through a systematic review of scientific articles published between 2014 and 2024, found that the integration of AI and IoT in the context of e-government significantly improves service efficiency, citizen engagement, and government accountability. In line with these findings, Bojović et al (2023) proposed an interconnected government services approach using a weighted graph model to design interconnected government services, thereby enabling integrated virtual procedures for the collection and processing of data needed by citizens. This approach represents a paradigm shift from fragmented services toward an integrated and smart public service ecosystem.

At the local level, smart city initiatives have made a significant contribution to driving digital-based public service innovation. Sofyan (2024), in a study on the implementation of smart city initiatives in the city of Bandung, shows that digital innovation is a fundamental driver in transforming the delivery of public services, particularly within the framework of urban governance. However, challenges such as resistance to change, the digital literacy gap, infrastructure disparities between urban and rural areas, and concerns regarding data security remain major obstacles (Gusman, 2024). Gräfe et al (2024) add that the digitization and automation of public administrative work not only changes tasks and workflows but also requires the adaptation of organizational structures and the competencies of civil servants to ensure that digital transformation proceeds effectively.

Based on an analysis of various previous studies, there is a significant research gap. Most existing studies focus on policy aspects, maturity models, or the evaluation of e-government service quality in general; however, few have developed integrated smart ecosystem platforms that specifically automate public service processes end-to-end. Hujran et al (2023) proposed the SMARTGOV maturity model for the transformation of e-government into smart government, while Haug et al (2023) mapped the digitally induced changes in the public sector theoretically. However, practical implementations in the form of web-based platforms that integrate smart features such as automatic document validation, duplicate request detection, service recommendations, and automatic priority calculation into a single integrated ecosystem remain very limited.

This study proposes the development of a web-based smart ecosystem platform named PANDU (Integrated Digital Public Services), designed to automate public services in a sustainable manner. The platform is built using the Laravel 12 framework with a Filament 4.0 administration panel and a responsive interface based on Tailwind CSS 4.0, supporting three main user roles: administrator, staff, and citizens. The novelty of this research lies in the integration of four key smart features into a single integrated platform: (1) automatic document validation that verifies file format, size, and completeness in real-time; (2) duplication detection that prevents duplicate submissions for the same service within a 30-day period; (3) recommendations for related services based on the selected category; and (4) automatic priority calculation based on service type, Service Level Agreement (SLA) duration, and service category. This platform supports more than 100 types of public services across 8 main categories, equipped with a real-time tracking system, multi-stage approval workflow management, and a citizen feedback mechanism, thereby creating a comprehensive, transparent, and sustainable digital public service ecosystem.

RESEARCH METHOD

1. Research Design

This study employs a Research and Development (R&D) approach using the Waterfall

software development model. The Waterfall model was chosen due to its sequential and structured nature, in which each phase must be fully completed before proceeding to the next (Yas et al., 2023). Yas et al. (2023), in a comprehensive review of Software Development Life Cycle (SDLC) methodologies, assert that the Waterfall model is highly suitable for software development projects with clearly defined requirements from the outset, structured documentation, and high standards of accountability—characteristics that align with the development of government public service platforms. This model consists of five main phases: (1) requirements analysis, (2) system design, (3) implementation, (4) testing, and (5) maintenance, as illustrated in Figure 1.

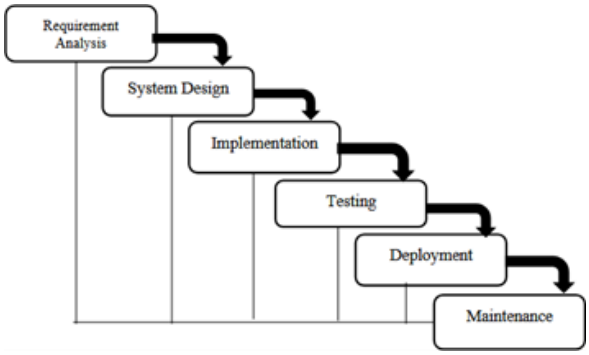

Figure 1. Stages of the Waterfall Model

2. Research Procedures

2.1. Needs Analysis

The requirements analysis was conducted through a literature review and observation of existing public service processes. Functional and non-functional requirements were identified and documented as the basis for system design. The primary functional requirements identified include: (a) a role-based access control system with three user roles: administrator, staff, and citizen; (b) public service management covering 8 main categories; (c) a mechanism for submitting service requests with the upload of required documents; (d) a multi-step approval workflow; (e) real-time tracking of application status; and (f) four intelligent features that automate the processes of validation, detection, recommendation, and prioritization. Non-functional requirements include data security, interface responsiveness, scalability, and compliance with government standards.

2.2. System Design

The system design adopts a Model-View-Controller (MVC) architecture that modularly separates the data components (Model), user interface (View), and business logic (Controller). Huynh et al. (2022) demonstrate that the MVC architecture within the Laravel framework enables standardized development with high scalability, thereby providing greater efficiency in web application implementation. The database design employs a relational model with 13 main tables interconnected via foreign key constraints. Table 1 illustrates the core database structure of the PANDU platform.

Table 1. Core Database Structure of the PANDU Platform

No	Table	Key Relationships
1	users	-
2	services	belongsTo→ service_categories
3	service_requests	belongsTo → users, services
4	request_documents	belongsTo → service_requests
5	approval_steps	belongsTo → services
6	audit_logs	Polymorphic

Note: Other supporting tables include `service_categories`, `service_requirements`, `approval_actions`, `request_trackings`, and `feedbacks` to complete the data relationships and history.

The system workflow design follows the service request status diagram shown in Figure 2.

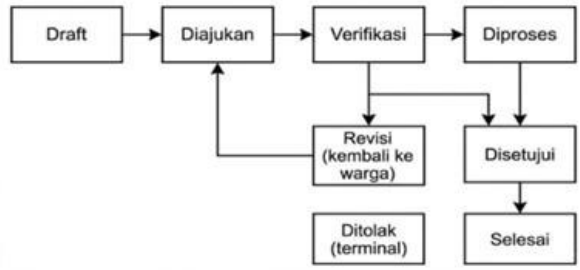

Figure 2. Service Request Status Flowchart

2.3. Implementation

The implementation uses the Laravel 12 framework as the backend, the Filament 4.0 admin panel for the management interface, and Tailwind CSS 4.0 for the responsive interface. The system architecture consists of three separate panels: (1) the Administrator Panel (`/admin`) for service management, user management, and auditing; (2) the Officer Panel (`/petugas`) for request processing and approval workflow management; and (3) the Citizen Panel (`/warga`) for submitting requests, uploading documents, and monitoring status. The implementation of the smart features is encapsulated in the `SmartService` service class, which consists of four main algorithms:

1. Automatic Document Completeness Validation: The algorithm checks the completeness of required documents by comparing the list of requirements (`service_requirements` WHERE `is_wajib = TRUE`) against the uploaded documents (`request_documents`). Validation occurs at two levels: (a) on the client side, which checks file formats (PDF, JPEG, PNG) and maximum size (2 MB); and (b) on the server side, which verifies the completeness of all required

documents before the application is submitted. If any documents are missing, the system rejects the application and displays a notification listing the documents that have not been uploaded.

2. Duplicate Request Detection: The algorithm checks whether the same user has an active request for the same service within the last 30 days using the query `WHERE user_id = userId AND service_id = serviceId AND status NOT IN [‘completed’, ‘rejected’] AND created_at >= (NOW - 30 days)`. Requests with the status `completed` or `rejected` are excluded because they have already been processed. If a duplicate is detected, the system stops the submission process and displays a warning to the user.
3. Related Service Recommendations: The algorithm uses a content-based filtering approach based on category similarity with the query `WHERE service_category_id = categoryId AND id ≠ current_service.id AND is_active = TRUE LIMIT 5`. The system automatically displays up to 5 related services along with estimated completion times to help residents find other services relevant to their needs.

Algorithm 4. Automatic Priority Calculation

```
ALGORITHM CalculatePriority
INPUT  : service (layanan yang diajukan)
OUTPUT : priorityLevel ∈ {rendah, sedang, tinggi, urgent}

BEGIN
  score ← 0

  // FAKTOR 1: Prioritas default layanan (0-4 poin)
  SWITCH service.prioritas_default
    CASE 'urgent' : score ←
    score + 4  CASE 'tinggi' :
    score ← score + 3  CASE
    'sedang' : score ← score + 2
    CASE 'rendah' : score ←
    score + 1  DEFAULT
                : score ←
    score + 2
  END SWITCH

  // FAKTOR 2: Durasi SLA (0-3 poin, SLA pendek = prioritas tinggi)
  IF service.sla_hari ≤ 2 THEN
    score ← score + 3          // SLA mendesak
  ELSE IF service.sla_hari ≤ 5 THEN
    score ← score + 2          // SLA pendek
  ELSE IF service.sla_hari ≤ 14 THEN
    score ← score + 1          // SLA
  menengah  END IF
  // SLA > 14 hari: tidak ada bonus

  // FAKTOR 3: Bobot kategori layanan (0-3 poin)
  categoryWeights ← {
    'kesehatan'   : 3,   // Kesehatan:
    tertinggi  'kependudukan' : 2,   //
    Kependudukan: tinggi  'perizinan'   : 1
                    // Perizinan: normal
  }
  categorySlug ← service.category.slug
  score ← score + categoryWeights[categorySlug] OR 0

  // TENTUKAN LEVEL PRIORITAS BERDASARKAN SKOR (rentang 0-10)
  IF score ≥ 8 THEN
    RETURN 'urgent'
  ELSE IF score ≥ 6 THEN
    RETURN 'tinggi'
  ELSE IF score ≥ 4 THEN
    RETURN
  'sedang'  ELSE
    RETURN
  'rendah'  END IF
END
```

The algorithm uses a multi-criteria scoring approach with three factors: (1) default service priority (score 1–4); (2) SLA duration, where shorter SLAs receive higher scores (maximum 3 points for SLAs ≤ 2 days); and (3) service category weight, with the healthcare category receiving the highest weight (3 points). The total score (range 0–10) is mapped to four levels: `low` (0–3), `medium` (4–5), `high` (6–7), and `urgent` (≥8). Table 2 presents an example of priority calculation.

Table 2. Automatic Priority Calculation

Service	SLA (days)	Total	Priorities
New ID Card	7	5	medium
IUMK	7	4	medium
Certificate of Good Health	2	7	high
Emergency Social Assistance	3	6	high

4. Agent Task Distribution (Load Balancing): The algorithm distributes new requests to the agent with the lowest workload using the query `WHERE role = ‘agent’ AND is_active = TRUE`, then calculates the number of active tasks per agent (`COUNT service_requests WHERE status IN [‘submitted’, ‘verified’, ‘in_progress’]`), and sorts by task count in ascending order. The agent with the lowest workload is selected to receive the new assignment, ensuring an even distribution of tasks and preventing a backlog of requests on specific agents.

2.4. Testing

System testing was conducted using two main methods: black-box testing and usability testing using the System Usability Scale (SUS).

1. Black-Box Testing
   Black-box testing uses the equivalence partitioning technique, which divides the input domain into equivalence classes, where each class represents a set of input data that is treated identically by the system (Azizah et al., 2024). This technique focuses on verifying functionality without considering the internal structure of the code, with the primary focus on inputs and outputs. Table 3 shows the test scenario design for all functional modules of the PANDU platform.

Table 3. Black-Box Testing Scenario Design

Modules Tested	Test Class	Test Focus
Authentication	Valid (3), Invalid (1)	Role-based login, invalid credentials
Service Submission	Valid (1), Invalid (2)	Upload all required documents

& Document Validation		, including their size and format
Duplicate Detection	Duplicate (1), Non-duplicate (1)	Applications filed within <30 days vs. >30 days SLA
Prioritization Automated	Multi-class	Variations and Service Categories
Approval Workflow	Valid (1), Revision (1)	Approve, reject, request revisions
Tracking & Recommendations	Valid (3)	Tracking numbers, recommendations, feedback

Note: A total of 14 test scenarios cover valid and invalid equivalence classes for each functional module.

2. Usability Testing

Usability was evaluated using the System Usability Scale (SUS), which consists of 10 statements on a 5-point Likert scale (Suria, 2024). The SUS score is calculated using the following formula:

$$SUS\ Score = ((\Sigma\ scores\ for\ odd - numbered\ items - 5) + (25 - \Sigma\ scores\ for\ even - numbered\ items)) \times 2.5 \quad (1)$$

For odd-numbered statements (1, 3, 5, 7, 9), 1 is subtracted from the respondent's answer, while for even-numbered statements (2, 4, 6, 8, 10), 5 is subtracted from the respondent's answer. The final score ranges from 0 to 100, with interpretations as shown in Table 4.

Table 4. Interpretation of SUS Scores

Score Range	Grade	Adjective Rating	Acceptability
≥ 80,3	A	Excellent	Acceptable
68 – 80,3	B	Good	Acceptable
68	Okay	Marginal	Marginal
51 – 68	D	Poor	Marginal
≤ 51	F	Awful	Not Acceptable

3. Data Acquisition and Testing Environment

Test data was acquired through two approaches: (1) synthetic data (seeders) generated using Laravel's database seeding feature, including 4 user accounts, 5 service categories, and 6 service types along with document requirements and approval stages; and (2) usability evaluation data collected from respondents via a SUS questionnaire after they tried the PANDU platform. The testing environment was configured using an in-memory SQLite database to ensure isolation between scenarios without shared state. Testing was performed using PHPUnit with two test suites: unit tests (`tests/Unit/`) for individual components and feature tests (`tests/Feature/`) for integration between components. All smart feature algorithms and approval workflows were implemented in two service classes: `SmartService` and `WorkflowService`, in line with the separation of concerns principle in MVC architecture (Saravanos & Curinga, 2023).

RESULTS AND DISCUSSION

1. Results of the PANDU Platform Implementation

The PANDU (Integrated Digital Public Services) platform was successfully implemented as a web application based on MVC architecture using the Laravel 12 framework, the Filament 4.0 admin panel, and the responsive Tailwind CSS 4.0 interface. The platform features three separate panels tailored to user roles, as well as a public page that can be accessed without authentication. Table 5 summarizes the technical specifications of the platform implementation.

Table 5. Technical Specifications of the PANDU Platform

Components	Specifications
Backend Stack	PHP 8.2+, Laravel 12, Filament 4.0
Frontend Stack	Tailwind CSS 4.0, Vite 7.x
Database	MySQL 8.0+ / PostgreSQL 13+
Architecture	Model-View-Controller (MVC)
Database Scaling	13 tables, 11 Eloquent models
Dashboard Structure	3 panels (Residents, Staff, Admin)
Implementation Services	6 services across 5 categories (scalable)
Public API	2 endpoints (tracking, estimation)

1.1. Public Pages and Tracking System

The public page (landing page) serves as the main entry point for all visitors without requiring authentication. This page consists of six main sections: (1) a hero section with brief statistics on users and services; (2) a form for tracking application status using a tracking number in the format `REQ-YYYYMMDD-XXXXX`; (3) a table estimating the processing time for all active services; (4) a presentation of four smart features; (5) a catalog of 8 service categories; and (6) a 4-step application workflow guide. The status tracking system is implemented via the `POST /api/tracking` API endpoint, which accepts a tracking number and returns a JSON response containing complete request

information, including current status, color-coded status labels, service name, priority, submission date, estimated completion time, and status change history. Additionally, the `GET /api/service-estimations` API endpoint provides data on all active services along with estimated completion times and default priorities.

1.2. Community Panel

The Citizen Portal (`/warga`) provides an interface for citizens to submit service requests, upload required documents, and track the status of their requests. The dashboard displays three widgets: request statistics (total, in progress, completed, rejected), a timeline of recent requests with tracking numbers and color-coded statuses, and a service catalog with categories and estimated processing times. The application form automatically integrates smart features: service information, recommendations for related services (collapsible), and real-time validation of the format (PDF, JPEG, PNG) and maximum size (2 MB) of required documents.

1.3. Staff Panel

The Agent Panel (`/agent`) is designed to process service requests through a multi-step approval workflow. The dashboard displays four widgets: task statistics (active, completed, SLA alerts, SLA violations), a 30-day task trend chart, a table of recent tasks sorted by SLA deadline with priority badges and status, and SLA alerts for critical requests (overdue or approaching the 48-hour deadline). Agents have four actions: Approve, Request Revision, Reject (with confirmation), and Mark as Complete. Each action is recorded in the `approval_actions` table as an audit trail and sends an automatic notification to the citizen.

1.4. Administrator Panel

The Administrator Panel (`/admin`) provides full control over the entire platform configuration. The dashboard displays three widgets: summary statistics (total submissions with a mini chart, completed, pending agent review, SLA violations), a graph showing service trends over the last 30 days, and an agent performance table with color-coded indicators for workload and SLA violations. Administrators can manage four resources: service categories (icons, order), services (requirements documents and approval stages per service via repeater), users (roles, status), and audit trails (audit logs) of system activities. Table 6 summarizes the features of each panel.

Table 6. Feature Comparison by User Panel

Score Range	Residents	Officer	Admin
Dashboard & statistics	✓	✓	✓
Trend & performance charts	-	✓	✓
Submit & upload documents	✓	-	-
Request processing (approve/revise/reject)	-	✓	✓
Alerts & SLA tracking	-	✓	✓
Service catalog & recommendations	✓	-	-
System management (services, users, categories)	-	-	✓
Monitoring (audit logs, agent performance)	-	-	✓

2. Smart Feature Test Results

Smart feature testing was conducted to validate that each algorithm functions as designed, as outlined in the Methods section. The testing utilized seeder data covering 6 services across 5 categories, with varying document requirements and approval stages.

2.1. Automated Document Completeness Validation

Document validation testing was conducted at two levels. On the client side, the system successfully validated the file format and size prior to upload. On the server side, the `ValidateDocumentCompleteness` algorithm successfully verified the completeness of all required documents before the application was submitted. Testing on the New ID Card Creation service, which requires 4 mandatory documents, resulted in a 100% success rate (7 out of 7 scenarios): the system accurately detected document completeness (0/4, 1/4, 3/4, 4/4), rejected files larger than 2 MB, and distinguished valid formats (PDF, PNG, JPEG) from invalid formats (EXE).

2.2. Duplicate Application Detection

The `DetectDuplicate` algorithm was tested by varying the temporal conditions and request statuses, resulting in a 100% success rate (8 out of 8 scenarios). The system correctly rejected 3 duplicate requests with active statuses (submitted, in progress, verification) within the 30-day window, while accepting valid requests: requests outside the 30-day window, requests with terminal statuses (completed/rejected), requests for different services, or requests with no prior history.

2.3. Related Service Recommendations

The `GetRelatedServices` algorithm was tested on 6 services across various categories, achieving a 100% success rate (6 out of 6 scenarios). In the Civil Registration category, which has 2 services, the system accurately recommended the other service (New ID Card ↔ Family Card Update). In the 4 other categories that have only 1 service (Permits, Health, Social, Education), the system displays the message “No related services in this category.” The algorithm is designed to display a maximum of 5 recommendations and automatically adjusts when service categories are added by the administrator.

2.4. Automatic Priority Calculation

The `CalculatePriority` algorithm was tested on all 6 services available in the data seeder. Table 7 shows the priority calculation results for each service.

Table 7. Automatic Priority Calculation

Service	SLA (days)	Total	Priorities
New ID Card	7	5	medium
Changes to the Family Card	5	6	high
IUMK	7	4	medium
Certificate of Good Health	2	7	high
Emergency Social Assistance	21	3	low
Educational Scholarships	45	2	low

All 6 scenarios produced priority scores that matched the manual calculations based on the multi-criteria scoring algorithm (100% success rate). The results show that the algorithm effectively prioritizes services with short SLAs and high-urgency categories: the Health Certificate received a High priority (score of 7) despite its default priority being Low, due to the combination of a very short SLA (2 days) and a high weight for the Health category (3 points). Conversely, Social Assistance, which has an inherent High priority, only received a score of 3 (Low) due to its long SLA (21 days) and the Social category having no additional weight. The distribution of priorities from the test results is shown in Figure 3.

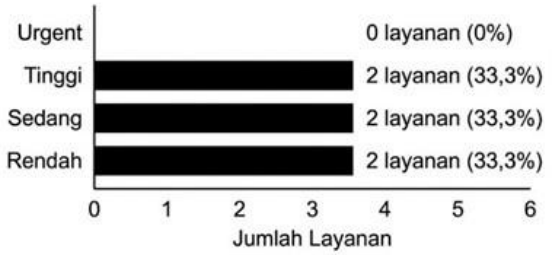

Figure 3. Service Priority Distribution

3. Black Box Test Results

Black-box testing using equivalence partitioning was performed on all functional modules of the PANDU platform. The testing covered 14 test scenarios designed in the Methods chapter, including: (a) authentication of three user roles using valid and invalid credentials; (b) service requests with variations in file format and size; (c) duplication detection with variations in time windows; (d) priority calculation with combinations of SLAs and categories; (e) a multi-step approval workflow; and (f) status tracking with valid and invalid tracking numbers. All 14 scenarios produced outputs consistent with the expected results, resulting in a 100% success rate (14/14). Each equivalence class—whether valid or invalid—is handled correctly by the system. Table 8 summarizes the test results per module.

Table 8. Summary of Black-Box Testing Results by Module

Service	Scenario	Status
Authentication	4	100%
Service Request	3	100%
Duplicate Detection	2	100%
Automatic Prioritization	1	100%
Approval Workflow	2	100%
Status Tracking	2	100%
Total	**14**	**100%**

4. Usability Testing Results

Usability testing was conducted using the System Usability Scale (SUS) with 15 participants, consisting of 5 administrators/government officials and 10 members of the public. Respondents were asked to complete a series of tasks on the PANDU platform, then fill out the SUS questionnaire, which consisted of 10 statements rated on a scale from 1 (Strongly Disagree) to 5 (Strongly Agree). Table 9 presents the results of the PANDU platform usability evaluation.

Table 9. Technical Specifications of the PANDU Platform

Metric	Value
SUS Score (N=15)	**83.97** (SD: 10.59, Range: 67.5–100.0)
Evaluation Results	**Grade A** (Excellent, Acceptable): 86.7% of respondents in categories A and B

Based on the data in Table 9, the PANDU platform received an average SUS score of 83.97, earning a Grade A and an "Excellent" rating, indicating a very high level of usability. The distribution of respondents' SUS scores is shown in Figure 4.

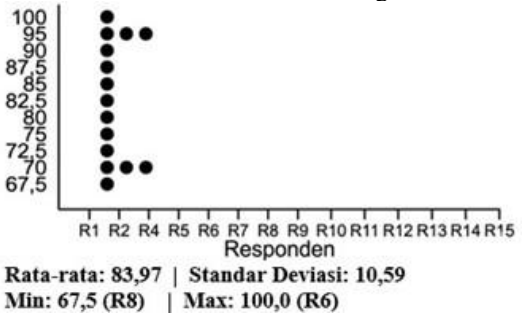

Figure 4. Distribution of Respondents' SUS Scores

A total of 86.7% of respondents (13 out of 15) rated the platform in the *Acceptable* category (Grade A: 9 respondents or 60%, Grade B: 4 respondents or 26.7%), indicating that the PANDU platform offers

good to very good usability for users. The two respondents who rated the platform in the D category (67.5 and 70.0) were from the over-50 age group, suggesting a potential need to simplify the interface for users with limited digital literacy.

5. Discussion

5.1. Value of Novelty and Innovation

The PANDU platform introduces several innovative features that set it apart from existing e-government systems and previous research. First, the integration of four smart features—automatic document validation, duplicate detection, service recommendations, and automatic priority calculation—into a single unified platform represents an approach that is still rare among digital public service systems in Indonesia. Unlike the SmartASN platform, which focuses on transforming the management of civil service human resources, PANDU directly automates the interaction process between citizens and government agencies in the delivery of public services.

Second, the multi-criteria scoring algorithm for automatic priority calculation applies weighting based on three factors that take into account the context of public services in Indonesia: the inherent priority of the service, the SLA duration, and the urgency category of the service. This approach enables more objective and consistent prioritization compared to manual prioritization by staff, which is prone to subjectivity and inconsistency.

Third, the load-balancing mechanism for task assignment ensures that the workload is distributed evenly, thereby reducing the risk of delays caused by a backlog of tasks assigned to specific staff members. This mechanism is complemented by a tiered SLA alert system (48-hour alerts and delay alerts) that enables proactive measures to prevent SLA violations.

Fourth, a multi-panel architecture (users, agents, administrators) with configurable multi-step approval workflows per service provides a high degree of flexibility. Administrators can independently define the number of approval steps, the responsible agents, and the SLAs for each step for every service, without requiring any code changes.

5.2. Practical Implications

The test results reveal several significant practical implications for the delivery of public services. First, the automatic document validation feature can reduce the rate of file returns due to incomplete documentation, which has long been the primary cause of delays in service processing. Two-tier validation (client and server) ensures that submitted applications meet all requirements, allowing staff to process them immediately without the need for manual verification of document completeness.

Second, a real-time tracking system with unique tracking numbers and a public API allows citizens to monitor the status of their applications without having to physically visit government offices. This aligns with the principle of 24/7 service accessibility and has the potential to reduce physical lines at service offices.

Third, the duplicate detection feature effectively prevents the waste of resources caused by duplicate requests, which, on a large scale, can have a significant impact on the operational efficiency of government agencies. The 30-day window strikes a balance between preventing duplication and providing citizens with the flexibility to resubmit a request if their previous one has already been processed.

Fourth, the average SUS score of 83.97 (Grade A, Excellent) indicates that the PANDU platform has an excellent level of usability. This finding suggests that although the platform integrates technically complex smart features, the user interface successfully hides that complexity and continues to provide an intuitive experience for end users.

5.3. Comparison with Previous Research

Compared to previous research, the PANDU platform offers a more comprehensive approach to public service automation. Most existing research, such as the SMARTGOV maturity model, focuses on theoretical frameworks for e-government transformation without incorporating the implementation of operational platforms. Research on interconnected government services proposes weighted graph models for the design of connected services, but has not yet implemented intelligent features such as automatic validation and adaptive priority calculation.

The PANDU platform bridges this gap by providing an operational implementation that combines the concept of an integrated service ecosystem with intelligent automation features. Compared to smart city initiatives such as those implemented in Bandung, PANDU offers advantages in terms of: (a) service modularity that can be configured without recoding; (b) multi-criteria-based automatic prioritization algorithms; (c) integrated duplication prevention mechanisms; and (d) a *load balancing* system for assigning personnel.

However, this platform has several limitations that need to be noted. First, the service recommendation algorithm still uses a simple content-based filtering approach based on category similarity, which could be enhanced with collaborative filtering techniques or AI-based approaches to generate more personalized recommendations. Second, usability testing was conducted on a limited sample (15 respondents), so the results should be generalized with caution. Third, the platform has not yet integrated digital signature and payment gateway features, which are essential components for the full implementation of public services. Further development in these areas is expected to enhance the comprehensiveness and operational readiness of the PANDU platform for deployment in a real-world government environment.

CONCLUSION

This research developed the PANDU (Integrated Digital Public Services) platform to address the issues with public services in Indonesia, which are still manual, error-prone, and lack a real-time tracking system. The platform is built as a web-based smart ecosystem with a Model-View-Controller architecture that integrates four key intelligent features: automatic document validation, duplicate request detection, related service recommendations, and automatic priority calculation based on multi-criteria scoring. Implementation results show that the platform was successfully developed with three separate panels for citizens, staff, and administrators that accommodate the needs of all stakeholders. Black-box testing using equivalence partitioning confirmed that all modules operate correctly, while the System Usability Scale evaluation yielded ratings in the Excellent and Acceptable categories, indicating that the platform has a very high level of usability despite integrating technically complex smart features.

The novelty of this research lies in the integration of intelligent automation features into a unified ecosystem that bridges the gap between the theoretical framework of smart government and the implementation of operational platforms. The PANDU platform provides practical solutions with support for configurable multi-stage approval workflows, load-balancing mechanisms for staff assignments, real-time tracking via public APIs, and comprehensive audit trails to ensure transparency and accountability. Limitations of the study include: (1) a recommendation algorithm that is still simple and can be improved with machine learning; (2) usability evaluation with a limited sample that needs to be expanded; and (3) lack of integration with digital signatures and payment gateways. For future research, it is recommended to develop a mobile app version, implement AI for chatbots and predictive analytics, conduct comparative studies across various local government agencies, and explore the integration of blockchain technology to enhance data security and integrity. The development of these aspects is expected to strengthen the PANDU platform's contribution to accelerating the sustainable digital transformation of public services in Indonesia.

ACKNOWLEDGEMENT

The acknowledgement should contain thanks to all individuals and institutions that have contributed to the research, the work, and the writing of the manuscript.

REFERENCES

Al-Ansi, A. M., Garad, A., & Jaboob, M. (2024). Elevating e-government: Unleashing the power of AI and IoT for enhanced public services. Heliyon, 10(23), e40591. https://doi.org/10.1016/j.heliyon.2024.e40591

Azizah, D. N., Mahendar, I. A., Alfatih, M. F., Anwar, S. I., Al Hapid, N. M., & Wicaksono, A. (2024). Analysis and testing of the Combox web application system using black box testing with the equivalence partitioning method. International Journal of Electrical Engineering, Mathematics and Computer Science, 1(4), 37–43. https://doi.org/10.62951/ijeemcs.v1i4.118

Bojović, Ž., Klipa, Đ., Bojović, P. D., Jovanović, I. M., Šuh, J., & Šenk, V. (2023). Interconnected government services: An approach toward smart government. Applied Sciences, 13(2), 1062. https://doi.org/10.3390/app13021062

Gräfe, P., Marienfeldt, J., Wehmeier, L. M., & Kuhlmann, S. (2024). Changing tasks and changing public servants? The digitalisation and automation of public administrative work. Information Polity. Advance online publication. https://doi.org/10.1177/13837605241289773

Gusman, S. W. (2024). Development of the Indonesian government's digital transformation. Dinasti International Journal of Education Management and Social Science, 5(5), 1128–1141. https://doi.org/10.38035/dijemss.v5i5.2868

Haug, N., Dan, S., & Mergel, I. (2023). Digitally-induced change in the public sector: A systematic review and research agenda. Public Management Review, 26(7), 1963–1987. https://doi.org/10.1080/14719037.2023.2234917

Hujran, O., Alarabiat, A., Al-Adwan, A. S., & Al-Debei, M. (2023). Digitally transforming electronic governments into smart governments: SMARTGOV, an extended maturity model. Information Development, 39(4), 811–834. https://doi.org/10.1177/02666669211054188

Huynh, T. S., Tran, D. T., Nguyen, L. A. T., & Vu, Q. H. (2022). Design and implementation of web application based on MVC Laravel architecture. European Journal of Electrical Engineering and Computer Science, 6(4), 23–29. https://doi.org/10.24018/ejece.2022.6.4.448

Latupeirissa, J. J. P., Dewi, N. L. Y., Prayana, I. K. R., Srikandi, M. B., Ramadiansyah, S. A., & Pramana, I. B. G. A. Y. (2024). Transforming public service delivery: A comprehensive review of digitization initiatives. Sustainability, 16(7), 2818. https://doi.org/10.3390/su16072818

Nyeleker, K. P., Mutiarin, D., & Barrow, E. (2025). From e-government to digital government: SmartASN as a sustainable digital innovation in Indonesia's public sector. Public Accounting and Sustainability, 2(1), 1–18. https://doi.org/10.18196/pas.v2i1.23

Saravanos, A., & Curinga, M. X. (2023). Simulating the software development lifecycle: The waterfall model. Modelling, 4(4), 396–408. https://doi.org/10.3390/modelling4040021

Sofyan, A. (2024). Digital innovation in public service delivery: An implementation study of smart city initiatives in Bandung City, Indonesia. VISIONER: Jurnal Pemerintahan Daerah di Indonesia, 16(3), 204–215. https://doi.org/10.54783/jv.v16i3.1259

Suria, O. (2024). A statistical analysis of system usability scale (SUS) evaluations in online learning platform. Journal of Information Systems and Informatics, 6(2), 992–1007. https://doi.org/10.51519/journalisi.v6i2.750

Yas, Q. M., ALazzawi, A., & Rahmatullah, B. (2023). A comprehensive review of software development life cycle methodologies: Pros, cons, and future directions. Iraqi Journal for Computer Science and Mathematics, 4(4). https://doi.org/10.52866/ijcsm.2023.04.04.014

Ye, X., Su, X., Yao, Z., Dong, L., Lin, Q., & Yu, S. (2023). How do citizens view digital government services? Study on digital government service quality based on citizen feedback. Mathematics, 11(14), 3122. https://doi.org/10.3390/math11143122