Ingénierie des Systèmes d'Information
Vol. 31, No. 8, August, 2026, pp. 2831-2845
Journal homepage: http://iieta.org/journals/isi

A Microservice-Based E-Government Service Management System Framework Using Fast Collaboration Competencies 2.0 Model

Muhammad Yusuf[1*], Moch Kautsar Sophan[2], Arif Muntasa[2], Andharini Dwi Cahyani[2], Deshinta Arrova Dewi[3], Kazeem Oluwakemi Oseni[4]

[1] Department of Information Systems, University of Trunodjoyo Madura, Bangkalan 69162, Indonesia
[2] Department of Informatics Engineering, University of Trunodjoyo Madura, Bangkalan 69162, Indonesia
[3] Faculty of Data Science and Information Technology, INTI International University, Nilai 71800, Malaysia
[4] School of Computing, Engineering, and Creative Industries, University of Bedfordshire, Luton LU1 3JU, United Kingdom

Corresponding Author Email: muhammadyusuf@trunojoyo.ac.id

https://doi.org/10.18280/isi.310826

Received: 22 May 2026
Revised: 29 July 2026
Accepted: 11 August 2026
Available online: 31 August 2026

Keywords:
microservices, e-government, service management system, Fast Collaboration Competencies, Peace, Justice and Strong Institutions, good governance

ABSTRACT

The Indonesian private sector, including large enterprises, already implements service management using information technology. Local government units in Indonesia have not yet fully implemented integrated information systems for service management, particularly microservice-based architectures. This research aims to develop a microservice-based e-government service management system (M-EGSMS) for the Department of Communication and Informatics in Sampang, Madura. The system was developed using the Fast Collaboration Competencies (FCC) 2.0 method. This paper presents two main contributions: the M-EGSMS architectural framework and the generalised FCC 2.0 development methodology. This work has implications for both theory and practice. This review broadens the theoretical understanding of service management, microservices, and e-government domains. In practice, the Department of Communication and Informatics in Sampang, Madura, and other practitioners could use the M-EGSMS for complaint handling. It supports the 16th Sustainable Development Goal (SDG), Peace, Justice and Strong Institutions, by promoting good governance. Future research will involve implementing a service management information system in the Department of Communication and Informatics of Sumenep, Madura, Indonesia.

1. INTRODUCTION

The Indonesian private sector, particularly large enterprises, already implements service management using information technology. Local governments in Indonesia are still far from fully implementing an information system for service management, particularly in the use of microservices. This research aims to develop a microservice-based e-government service management system (M-EGSMS) using microservices for the Department of Communication and Informatics in Sampang, Madura. It supports the 16th Sustainable Development Goal (SDG), Peace, Justice and Strong Institutions, with a focus on Governance. The research also aims to improve the software development methodology from Fast Collaboration Competencies (FCC) 1.0 to FCC 2.0.

The theoretical contribution of this research lies in developing the M-EGSMS framework and refining FCC 1.0 into FCC 2.0. The proposed M-EGSMS provides a conceptual basis for organising e-government service management functions within a microservices architecture, particularly for complaint handling, monitoring, reporting, notification, and user management. Furthermore, FCC 2.0 extends the original FCC 1.0 model by incorporating microservices or Application Programming Interface (API) development and API testing as explicit stages. Therefore, this research contributes not only at the implementation level, but also at the conceptual level by linking microservices, service management, and e-government development in an integrated framework.

The research question that addresses theoretical depth on service decomposition and governance principles is: Does the proposed framework offer a novel service decomposition method, new governance principles, or an integrated contribution that encompasses both?

This work offers both theoretical and practical implications by broadening the understanding of microservices, service management, and e-government domains. In practice, the Department of Communication and Informatics in Sampang, Madura, as well as other practitioners, could use the M-EGSMS for complaint handling.

Some literature related to this research analyses and discusses various opportunities and obstacles related to the adoption and deployment of microservices, given the paucity of empirical data on the subject. The paper's findings are based on in-depth interviews with 19 Information and Communication Technology (ICT) architects with substantial backgrounds in middleware, large-scale corporate systems, service-oriented architectures (SOAs), and, to a lesser degree, microservices [1]. Furthermore, this paper presents the concept of an Infrastructure Intelligent Service System from an information-flow perspective to enable intelligent

infrastructure management. It is based on research on infrastructure digitalisation and the integration of construction and maintenance [2]. Additionally, the research outlined six specific research challenges to get things started: (1) feature identification; (2) variability modelling; (3) variable microservice architectures; (4) interchangeability; (5) deep customisation; and (6) re-engineering a Software Product Line (SPL). Our goal is to avoid reinventing concepts from one area of study to another by using these issues as a springboard for future research in this area of study [3]. Furthermore, Universitas Muslim Indonesia (UMI) has created many applications to facilitate internal and external management of the campus's digital information and management systems. However, these apps were not well-suited for continuous use due to their complexity and lack of integration. Thus, by applying microservices, UMI hopes to develop a fully connected and well-managed campus information system. Using the microservices concept, large applications are broken into smaller, networked components [4]. This project aims to use the web service implementation methodology to develop an e-course learning management system that uses web services to enable online learning. MySQL is used for the database. Eventually, this application will serve as a tool to support online education; members, or users, will act as student actors and administrators, handling mentor and course data that will be visible [5]. This research described the SigSaude information system, used in student-run clinics to administer health services and store electronic medical data [6]. In this research, a new microservice reliability model (MSRM) for health information system (HIS) is proposed, based on the Predicate Petri net (PrT net) formalism [7].

This study proposes a distributed microservice-based architecture. This design is scalable because it breaks down complex professional analytic functions into microservices that can be deployed across multiple machines in various locations [8]. The website provides access to the certification data procedure. Often constructed using a monolithic approach, which entails a large package including an application [9]. To make the enterprise's original enterprise resource planning (ERP) system compatible with the new system and progressively upgrade its functional modules, this study employs Docker [10]. This study aims to provide a preliminary system design, along with a thorough description of the specification criteria, so that the decision support system can be designed as expected in the space science centre environment [11]. Additionally, the goal of this research is to highlight the primary security features of this architecture in the current cost-effective period [12]. This research describes the system's primary realisation and approach. Front-end and back-end decoupling has significantly increased the system's load capacity, and the system performs well, demonstrating the benefits of microservices and front-end/back-end separation in design [13]. The research then delves into the growing security solutions enabled by microservices, namely for legacy systems already in place, as well as the latest technical trends as seen through the organisational perspective of new development components [14]. This study aims to systematically identify, classify, and compare the body of research on microservices and cloud applications. We have conducted a systematic mapping of 21 selected studies published between the last two years and the end of 2015, since the microservices pattern first appeared [15]. This paper presents a systematic mapping study on microservices. It aims to identify emerging trends, potential research gaps, the motivation for microservice research, and current trends in this field [16]. This paper presents a classification framework for research studies in microservices architecture, a systematic map of the current research landscape, an assessment of the potential for industrial adoption of research outcomes, and a discussion of emerging findings and implications for future research [17]. This study aims to conduct a structured literature review concerning current research on Information Technology Service Management (ITSM) and Information Technology Infrastructure Library (ITIL). The review results indicate that motives, critical success factors, implementation status, and benefits are the most frequently investigated topics and warrant further investigation [18]. This paper presents a systematic mapping study on microservices architectures and their implementation. Our analysis focuses on the architectural challenges, architectural diagrams/views, and quality attributes related to microservice systems [19]. This paper presents an integrative review of the literature on the quality of e-government services, providing a basis for further development of related models and ontologies. We consider 18 approaches concerning the quality of service in the broad public sector and specific areas of e-government. We then categorised those as introverted and extroverted depending on their focus of interest on either organisational matters or the front end of service [20]. The research results showed that a microservice-based API could enable data interoperability among e-government services and be developed using multiple programming languages and base codebases. This API can be further enhanced by adding more data objects, using Amazon Web Services (AWS) Cognito for managing authorisation, integrating AWS Elastic search to load and filter data, and showing real-time data objects on the front end [21]. This paper presents a method for objectively evaluating the change in service quality resulting from implementing e-government projects. We specifically propose the Analytical Hierarchy Process as a tool to assess e-government-induced alterations in public service quality [22]. This research aims to develop a service design for an e-government framework that supports the Indonesian government in addressing issues arising from platform and application diversity. This paper presents the conceptual services of the Travel System using a SOA with Enterprise Service Bus (ESB) technology. It comprises a variety of services and business activities that can be combined [23]. This study aims to assess Data Quality Management (DQM) maturity, offer best practice recommendations, and develop a practical improvement plan with organisation-tailored indicators, a gap not addressed by earlier research. This study proposes three plans to enhance DQM by addressing 49 identified weaknesses. These are to be adopted progressively and sequentially over three years [24]. This research developed an IT service system in line with the company's requirements. This result can serve as the foundation for IT service development at an outsourcing security company. During this research, stakeholder involvement has been most significant; all deliverables have been validated and verified by stakeholders [25]. The suggested method can improve the reusability of microservices and thereby increase the application's concurrency, thereby improving performance [26]. This project proposes redesigning the official SBD website to improve the quality of information dissemination and the user experience. Web development is founded on the user-centred design (UCD) methodology. UCD also allows for the identification of the user and the respective requirements more

comprehensively. One example is the inclusion of specific requirements for people with colour blindness and dyslexia on the new SHD website [27]. The research explored factors that influence e-government adoption in Indonesia. Policymakers and government institutions need to enhance the system [28]. This study seeks to analyse the impact of government intervention. The influence of unemployment and spending on the Indonesia Human Development Index (IHDI) in Indonesia between 2010 and 2013. The method of analysis employed in this case is fixed-effects regression [29]. There is also a reference to the FCC 1.0 model [30]. Also, there is research on measuring effectiveness using the revised DeLone and McLean IS success model [31]. Additionally, research examines factors and social media in digital government [32].

Table 1. Categorization of the literatures

Category	Focus	References
1. Microservice architecture, technical transformation, and adoption.	Concepts, benefits, challenges, migration from monolith	[1-4, 8-10, 13, 26]
2. Applications of microservices in specific domains	Education, healthcare, science, enterprise	[5-7, 11]
3. Public digital services and interoperability	e-government, citizen-centred design, service quality	[20-23, 27, 28, 32]
4. IT service management and quality governance	Service/data-quality management, monitoring, evaluation	[18, 24, 25, 29, 31]
5. Microservices as supporting systems	Governance, coordination, public-service support	[12, 14-17, 19, 30]

The literature review could be categorised into five main groups according to the focus as shown in Table 1:

1. Microservice architecture, technical transformation, and adoption.

This literature examines the concept, architectural benefits, implementation challenges, adoption, and transformation from monoliths to modularity, scalability, and reusability of microservices. Microservice architectures facilitate integration across distributed systems.

2. Applications of microservices in specific domains.

Furthermore, domain-specific implementations address operational challenges in education, healthcare, and public administration.

3. Public digital services and their interoperability.

The reviews capture e-government systems, interoperability, citizen-centeredness of digital services, and service quality,

4. IT service management and quality governance.

The referenced studies cover service management and handling data quality. Furthermore, the studies examined monitoring, evaluation, and management dimensions of the systems.

5. Microservices as the supporting systems.

The studies capture the role of microservices as supporting systems in the governance and coordination of public service systems.

Moreover, the critical analysis of the literature is presented below. The literature captures implementation, architecture, service quality, interoperability, and IT governance. Therefore, microservices are part of broader digital transformation, not only a technical architecture. However, some studies capture microservices, while others examine web services, data quality, IT service management, SOA, public service quality, and digital government. These are related areas, but they are not interchangeable.

The literature focuses on architecture rather than operational aspects. Also, the studies examine the benefits of microservices on scalability, modularity, interoperability, maintainability, decoupling, reusability, and system improvements. Therefore, microservices are the solution for integration and disconnected problems. However, there are limited studies to prove that microservices improve service performance and outcomes after deployment.

Reviews show that microservices are promising; however, the field is still developing in various aspects, including methodological ones. Furthermore, microservices are applied in some domains, such as education, healthcare, enterprise systems, science, and others. However, there is a problem of transferability. The microservices system that works well in a non-government organisation might not automatically work in the same way in a government system. Government systems have higher security and compliance concerns, broader citizen-friendly and accessibility requirements, stricter accountability, and more complex stakeholders. The literature shows that interoperability in microservices applications is not only a design problem but also a governance and institutional coordination problem. Microservices security is important, as an attack could happen across multiple services, APIs, containers, and communication channels. However, it seems security is still a supporting issue in the literature. Overall, microservices are a widely promising architecture for addressing complexity and integration problems nowadays. Furthermore, they require service coordination and interoperability when applied across domains to improve service quality and effectiveness in government systems.

This work includes an introduction, literature review, research methodology, results, discussion, and conclusion.

2. METHODOLOGY

The step-by-step process of this study, as shown in Figure 1, began with searching Google Scholar for references using the keywords e-government, service management systems, and microservices. Furthermore, we reviewed 35 references from various reputable journals and proceedings.

Then, we identified existing conditions and user requirements. Moreover, we conducted analysis and design of the e-government service management system using microservices. Furthermore, we implemented the e-government service management system as a microservices architecture, tested it, and drew conclusions.

3. RESULTS AND DISCUSSIONS

This research developed an M-EGSMS for the Department of Communication and Informatics in the Sampang region, as shown in Figure 2. Previously, the department lacked a service management system to handle hardware, software, network, and other infrastructure issues in the Sampang government.

First, the existing conditions have been identified based on interviews with staff of the Department of Communication and Informatics in Sampang, Madura, and the results are:

(a) There are ICT services, which are not yet detailed.
(b) There is no procurement or data collection for the application yet.
(c) Application issue reporting exists; however, instant messaging integration (e.g., WhatsApp) is lacking.
(d) Report on network service.
(e) There was no add or remove application feature.
(f) There is a need to prepare a form of response as evidence.
(g) There is a need for a special application from the Department of Communication and Informatics in Sampang, Madura, that can be reported if there is trouble.
(h) Data collection or backup of all new application procurements from the Department of Communication and Informatics in Sampang, Madura, or other departments.
(i) Application onboarding procedures currently rely on informal messaging channels.
(j) Application development needs a formal letter signed by the Head of Department.
(k) The report should be for every department, application, issue, week, month, and year.
(l) There are existing data, such as related departments, application data, and network complaint data.

The system also requires additional features, such as WhatsApp notifications, an API, and the ability to upload action evidence or progress updates until completion.

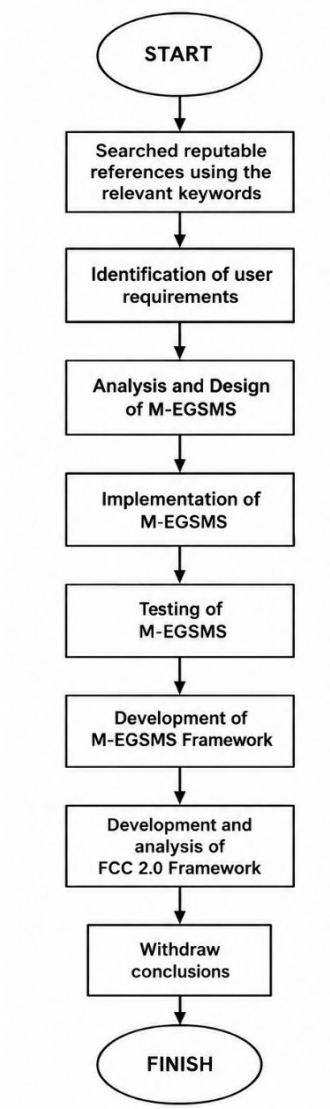

Figure 1. Step-by-step research methods

Figure 2. Interface of microservice-based e-government service management system (M-EGSMS)

Furthermore, Figure 3 presents the system use case diagram. The system includes actors such as the super admin, the operator or admin of the Department of Communication and Informatics, the head of the Department of Communication and Informatics, and other departments. Moreover, the super admin has access to the right data, registered applications, registered department data, and login. The operator or admin can log in, view the number of reports, manage complaint data and reports, and view registered accounts. The head of the Department of Communication and Informatics can also log in, view report notifications, and submit complaints. Additionally, other departments can log in to manage account registration, reports, and complaints.

Moreover, Table 2 presents the system requirements of the system, where the super admin manages all system and application registration, and the admin of the Department of Communication and Informatics can delete, update, and search for all application data and see the number of complaint data for each day, month, and year, as well as statistics of data submission and registered applications. Additionally, the Head of the Department of Communication and Informatics can search application data, view complaint data, and report on applications. Other departments can also search existing application data, select the complained application, enter complaint data, and submit the complaint-related report.

Furthermore, Figure 4 presents a sequence diagram of the input reporting data, which consists of logging in with a registered user account, then navigating to the user dashboard and opening the report menu. The user can then add a complaint report and enter evidence data into the report form. The system then submits the new data with a new report status.

Table 2. User requirements

User	System Requirements
Super Admin	Manage all system and application registration
Admin of the Department of Communication and Informatics	Delete, update, and search for all application data and see the number of complaint data each day, month, and year, as well as statistics of data submission and registered applications
Head of the Department of Communication and Informatics	Search for application data, see complaint data, and report on the application
Other Departments	Search for existing application data, select the complained application, enter the complaint data, and submit the report related to the complaint and application

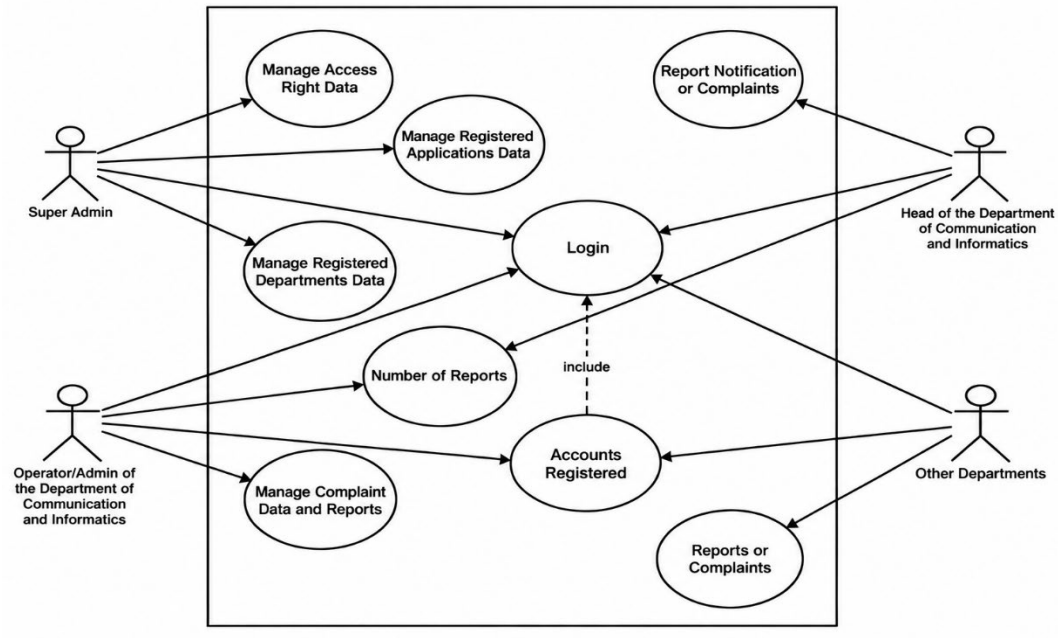

Figure 3. Use case diagram

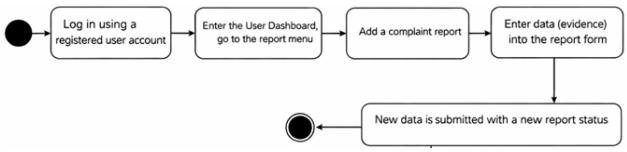

Figure 4. Sequence diagram of input reporting data

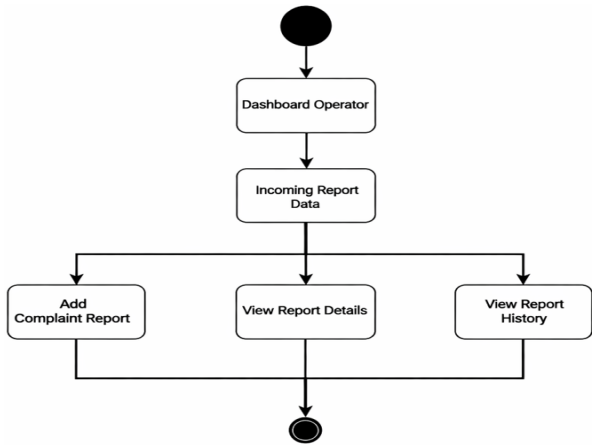

Figure 5. Sequence diagram of the management of report data

Moreover, Figure 5 shows the sequence diagram for managing report data. It starts by opening the operator user's dashboard. Then, the operator enters the incoming report data. This stage includes adding a complaint report, viewing report details, and viewing report history.

Additionally, Figure 6 shows a sequence diagram for reporting. It starts with the incoming report, and then the system validates the complaint data. If the data is valid, the admin or operator reviews the report. Additionally, if the report is accepted, the system follows up on the complaint with related actions and continues to resolve it. Moreover, if the report is not accepted and the data is invalid, the admin or operator will update the report data and synchronise the statistics.

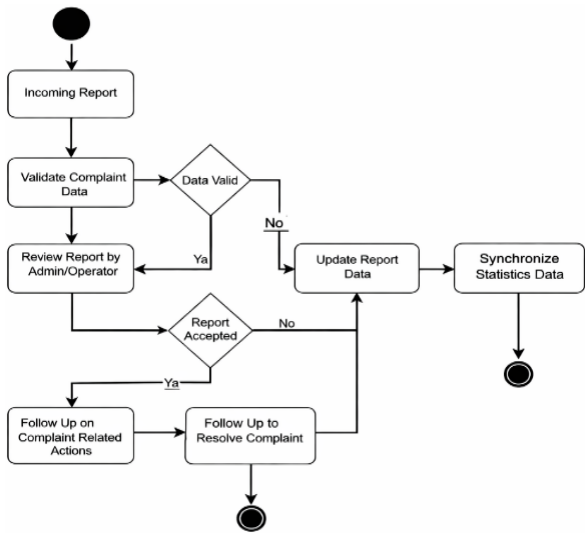

Figure 6. Sequence diagram of reporting

Table 3 presents the API key as one of the service security measures, shared with permitted third parties to access the service through the API. Technically, when a client requests to post/get data through the API, it uses the previously obtained API key. After that, the server checks the API key; if it is valid, the process continues. However, if it is not valid, the process will be refused.

Table 3. Application Programming Interface (API) key

Key	X-API-Key
Value	7045365d2a631eabd9ac28fffcc63e69fc280fe22b1aa a58d6a484e466070863

When the API is accessed successfully, the log is saved to the API request log, including the API key, request method, endpoint, and creation date. Figure 7 presents the JSON format for the API response.

Moreover, Table 4 shows the API request log, which includes API key data such as ID, API key, request method,

endpoint, and creation time. The API key will be updated periodically and randomly. The user periodically retrieves the API key and stores it in the system for security reasons. Although API logging has been implemented to support service monitoring and traceability, this research has not yet carried out a statistical analysis of the generated log data, as the current log records were captured primarily during the functional testing phase rather than over a sustained period of real operational use.

Therefore, interpretation of system behaviour based on API logs remains limited and preliminary. Future research is expected to analyse the log records quantitatively once sufficient operational data has accumulated, examining:

(1) Request distribution, such as the frequency of calls per endpoint and per time interval, to identify usage patterns and peak-load periods;

(2) Response characteristics, including the proportion of successful versus failed responses per endpoint;

(3) Error patterns, such as the most frequent error types, invalid API key, malformed request, and whether errors cluster around specific endpoints, time periods, or client sources. This quantitative log analysis would complement the architectural and functional evaluation presented in this study by providing empirical evidence of actual API usage behaviour under real deployment conditions.

Moreover, Figure 8 presents an API test that simulates an action performed by an external system to access the API and view complaint data previously entered in the main system. In addition, Figure 9 illustrates the complaint data results in graphical form. Figure 10 shows the API for retrieving user IDs and their metadata, including category, application name, complaint, status, and other relevant fields. Figure 11 presents the API test results, including user ID, token, ID, creation date, category name, application name, complaint, file, and status.


```
"status": "success",
"data": {
    "data_pengaduan": [
        {
            "id_users": "……",
            "token": "……",
            "id1": "…",
            "createdate": "……",
            "nama_kategori": "……",
            "nama_aplikasi": "……",
            "keluhan": "……",
            "file1": …… .png",
            "status_pengaduan": "baru"
        },
    ]
},
"maxPge": 2,
"currentPage": 1
```


Figure 7. JavaScript Object Notation (JSON) format for Application Programming Interface (API) response

Table 4. Application Programming Interface (API) request log

ID	API Key	Request Method	Endpoint	Creation Time
26	7045365d2a631eabd9ac28ffcc63e69fc280fe22b1aaa58d6…	GET	http://localhost:8080/index.php/api/pengaduan/5578	2024-10-28 09:54:55
27	7045365d2a631eabd9ac28ffcc63e69fc280fe22b1aaa58d6…	GET	http://localhost:8080/index.php/api/pengaduan/5578	2024-10-28 09:55:27

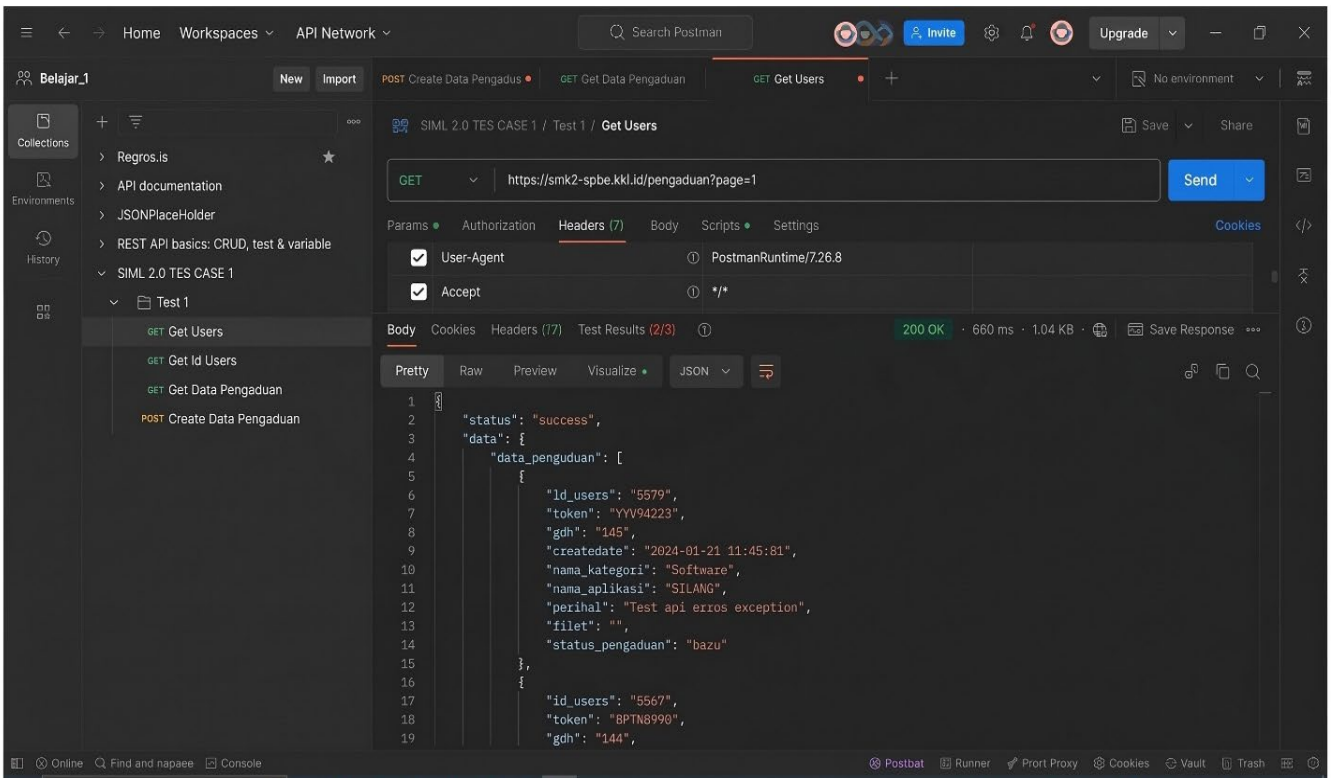

Figure 8. Application Programming Interface (API) test to view the existing complaint data

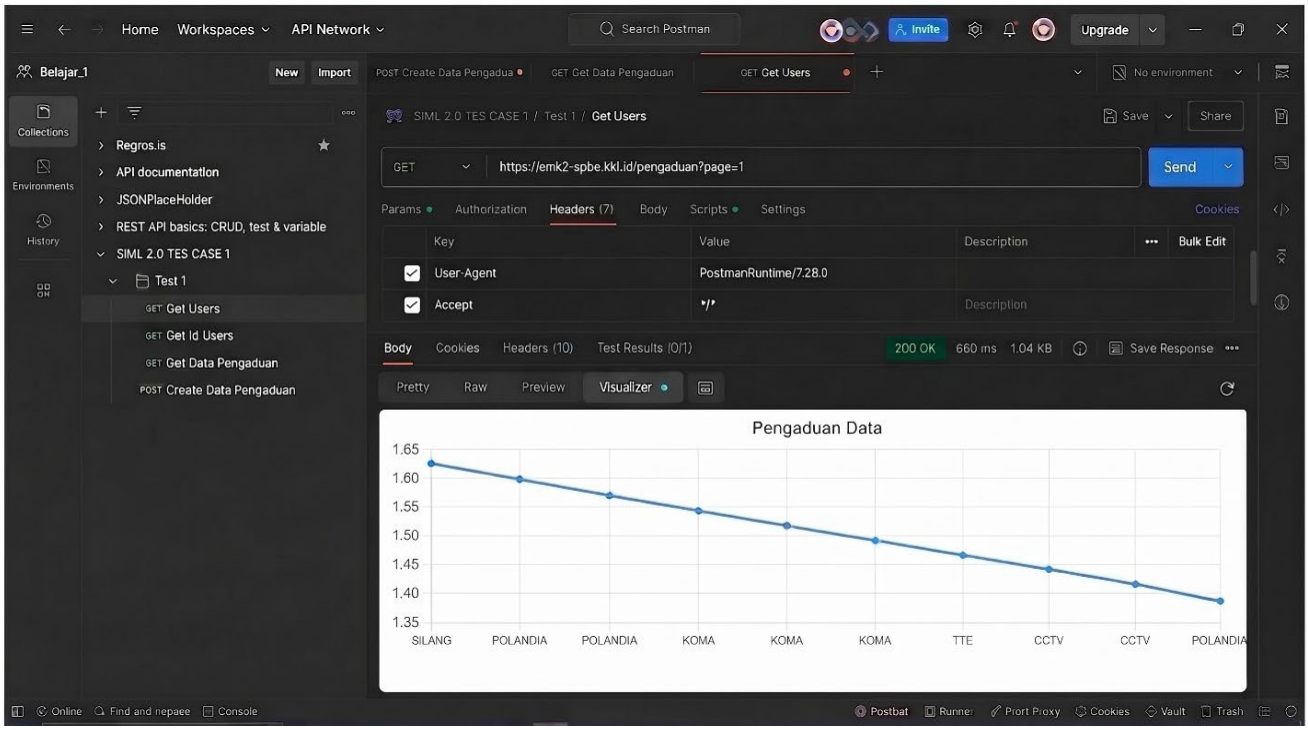

Figure 9. Data statistics from testing

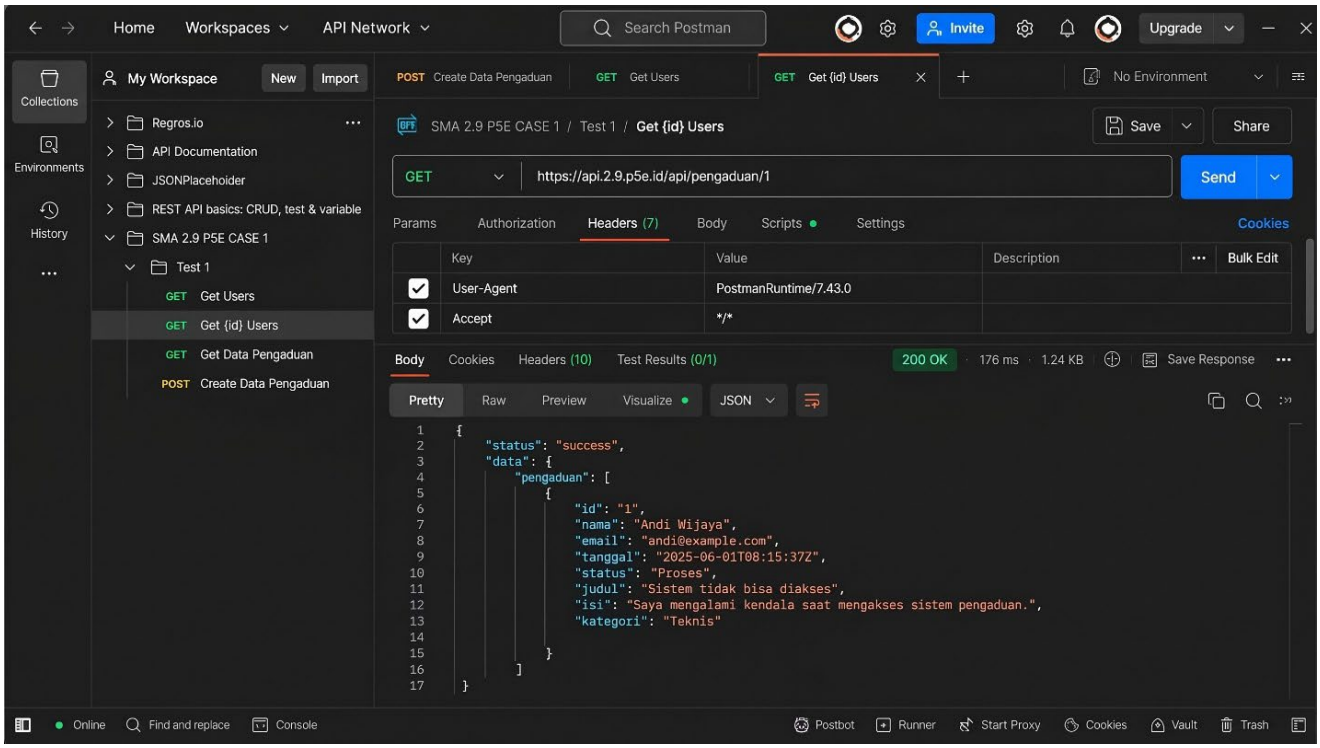

Figure 10. Application Programming Interface (API) testing with user ID = 5578

Additionally, Figure 12 shows API testing for retrieving complaint data across several applications, such as application name, address, institution, and other related information. Figure 13 presents the retrieved complaint data, including ID, application name, platform, address, and department name. Furthermore, Figure 14 shows API testing for adding complaint data, including user ID, complaint content, category ID, and other required fields. Figure 15 presents the results of the add-complaint test, including user ID, application ID, institution ID, category ID, and other relevant information.

M-EGSMS is essential for managing IT services, especially when using microservices. Furthermore, the system provides some core services as follows:

•Incident management: Manages knowledge management, assignment, reporting, and resolution of incidents.

•Problem management: Identifies and fixes underlying issues by analysing incident patterns.

•Change management: Controls IT service modifications to ensure no disruption.

•Service request management: Responds to routine service inquiries.

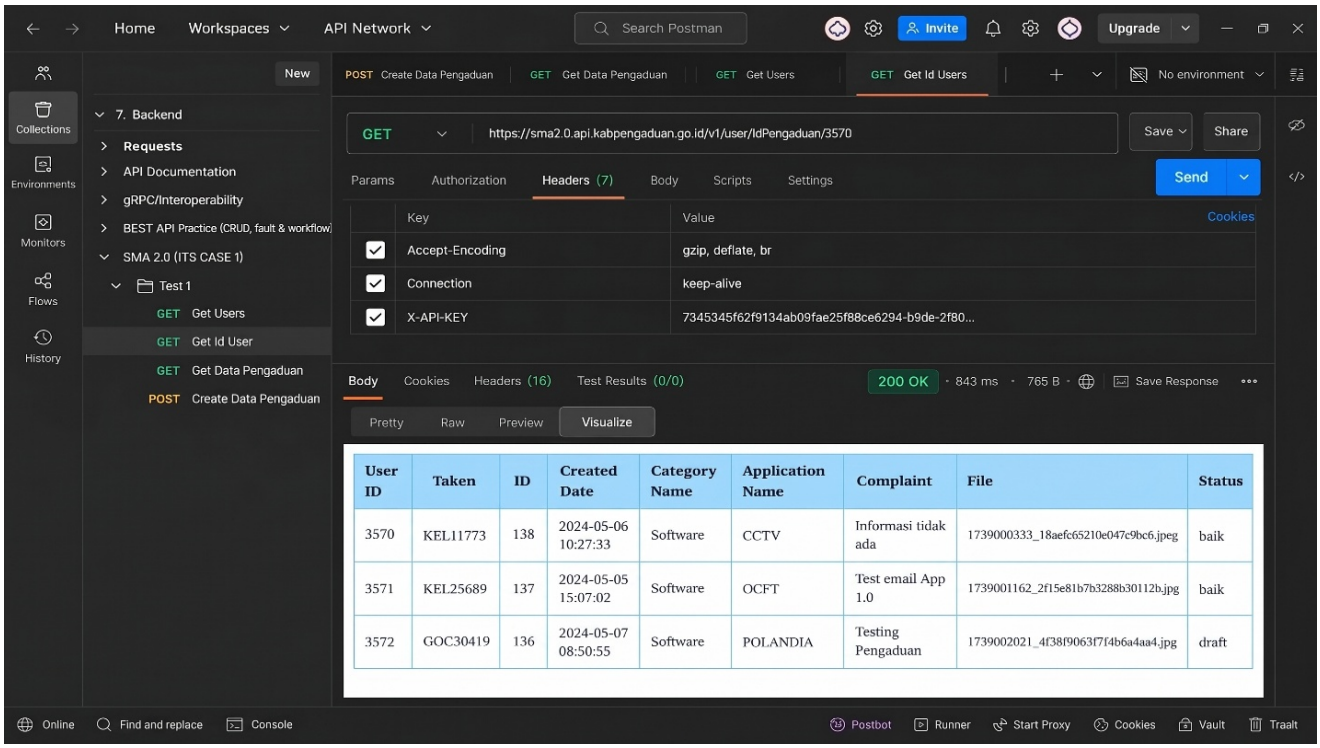

Figure 11. Data analysis from testing

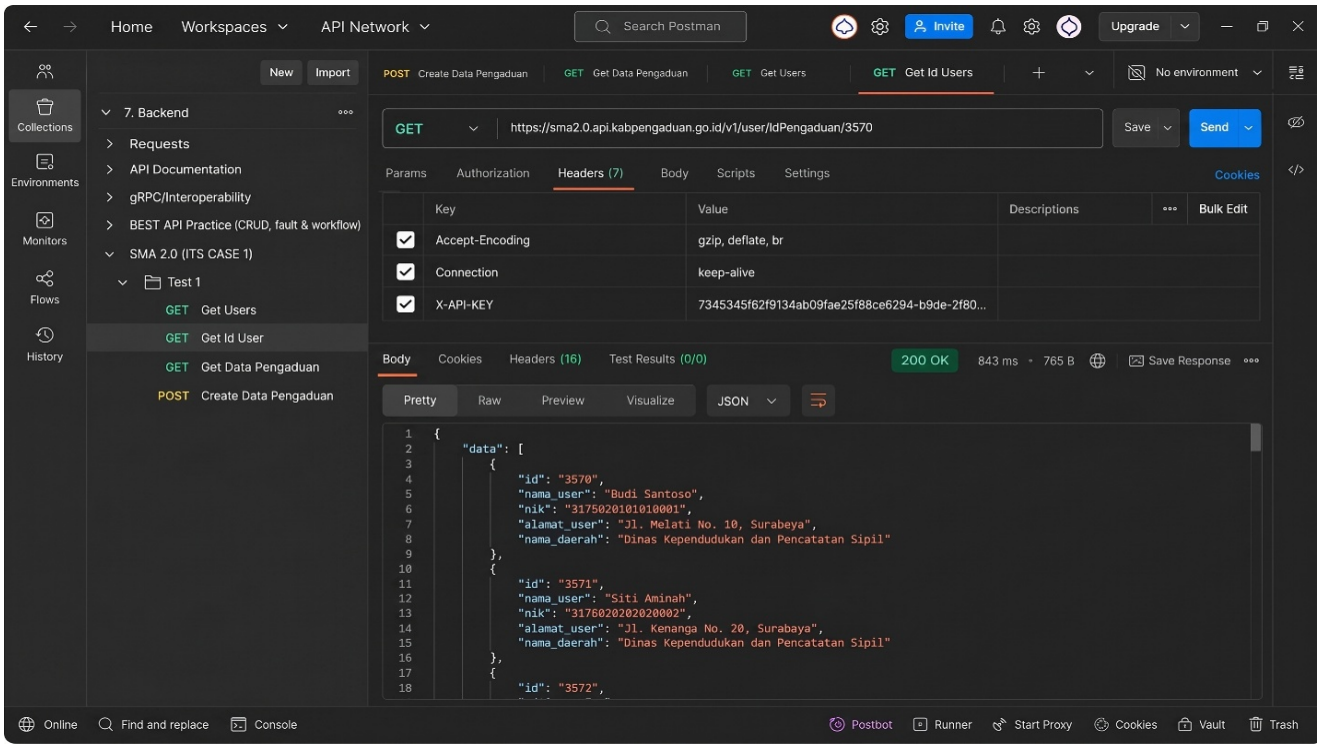

Figure 12. Application Programming Interface (API) testing to retrieve complaint data in each related institution

Table 5. Sample of testing report

Number of Tests	Testing Feature	Error	Error Part
1	Login	Operator of multiple device logins Double input	Login successful for more than 1 device at 1 time in the operator part. Data was affected when 2 or more operators are processing the data.
2	Registration	None	None
3	Admin Dashboard	Data unsynchronizing	Additional new data did not appear. Data was not yet synchronized in the operator interface.
4	User	Dismiss/failed data input	Failed data input
5	Others	Error exception code	The feature could not be opened because of a code error.

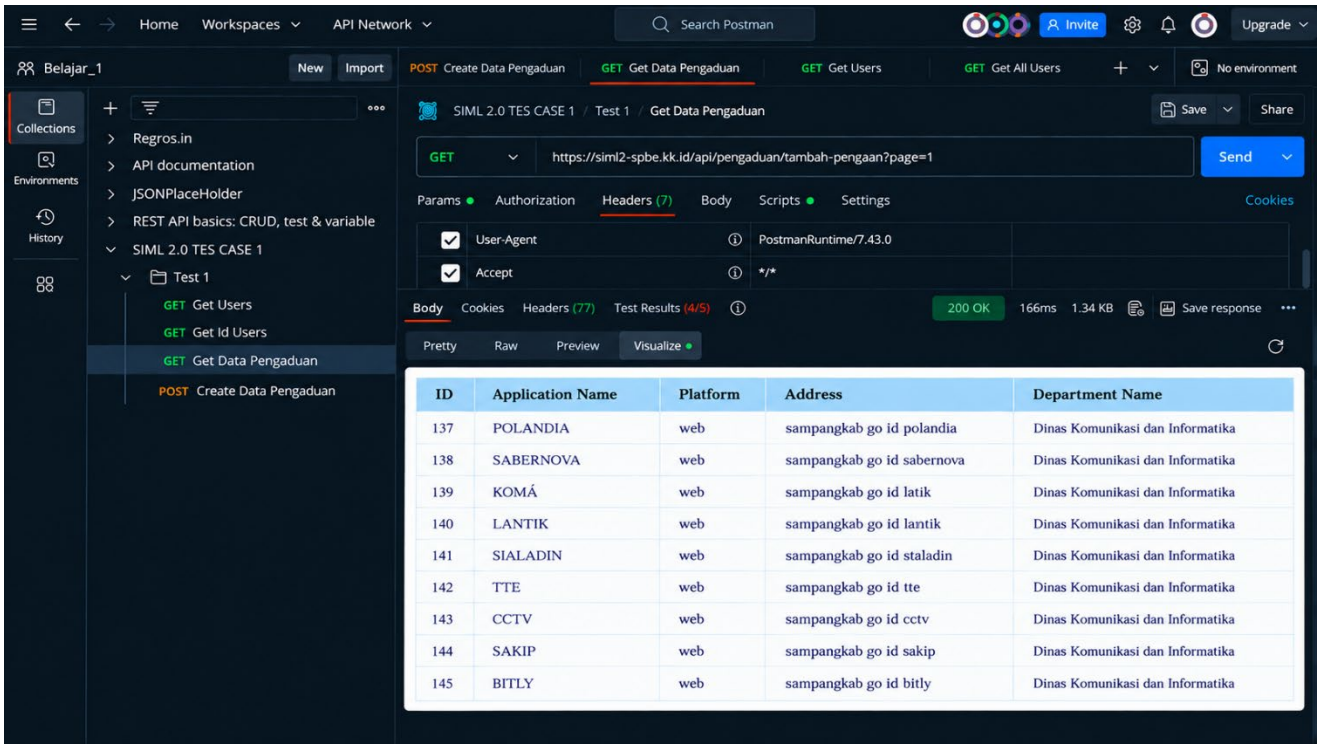

Figure 13. Data analysis from testing

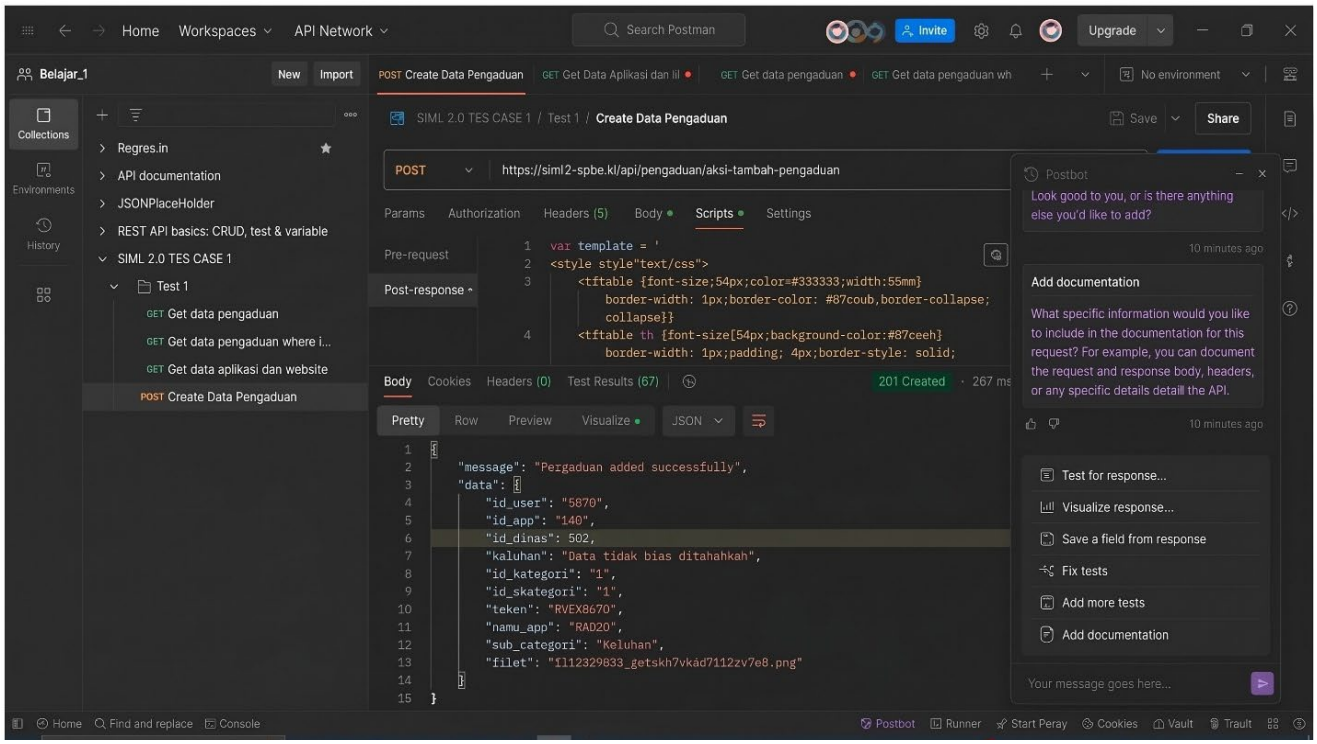

Figure 14. Application Programming Interface (API) testing for adding complaint data

Moreover, the core components include incident, problem, and change management, as well as configuration management and reporting and data analytics to support decision-making.

Microservices-based systems offer benefits such as scalability, resilience, flexibility, and cost efficiency. Scalability means services can scale independently and utilise resources efficiently. Moreover, resilience means one service failure does not affect others. Additionally, flexibility means it allows diverse tech stacks. Cost efficiency also means resource optimisation reduces operational expenses. However, challenges and complexities remain, including managing distributed systems, ensuring data consistency, testing complexity, and the overhead of maintaining service interdependencies and security.

Two testers tested M-EGSMS on 25th October 2024. The M-EGSMS was also tested, and a sample of the report is presented in Table 5. There are five test cases of testing as follows:

1. Register as a guest

2. Simulation of an application complaint
3. Simulation of an application request
4. Simulation of an application feature addition Log in as an operator to give a response

Table 5 indicates domains that need refinement and defect detection. Conflicting or duplicate data processing can occur due to concurrent operator logins from multiple devices. Therefore, defect reporting should be followed by root-cause analysis, corrective response, and post-fix validation. It is necessary to validate that the login/session management mechanism avoids conflicting parallel access and duplicate input.

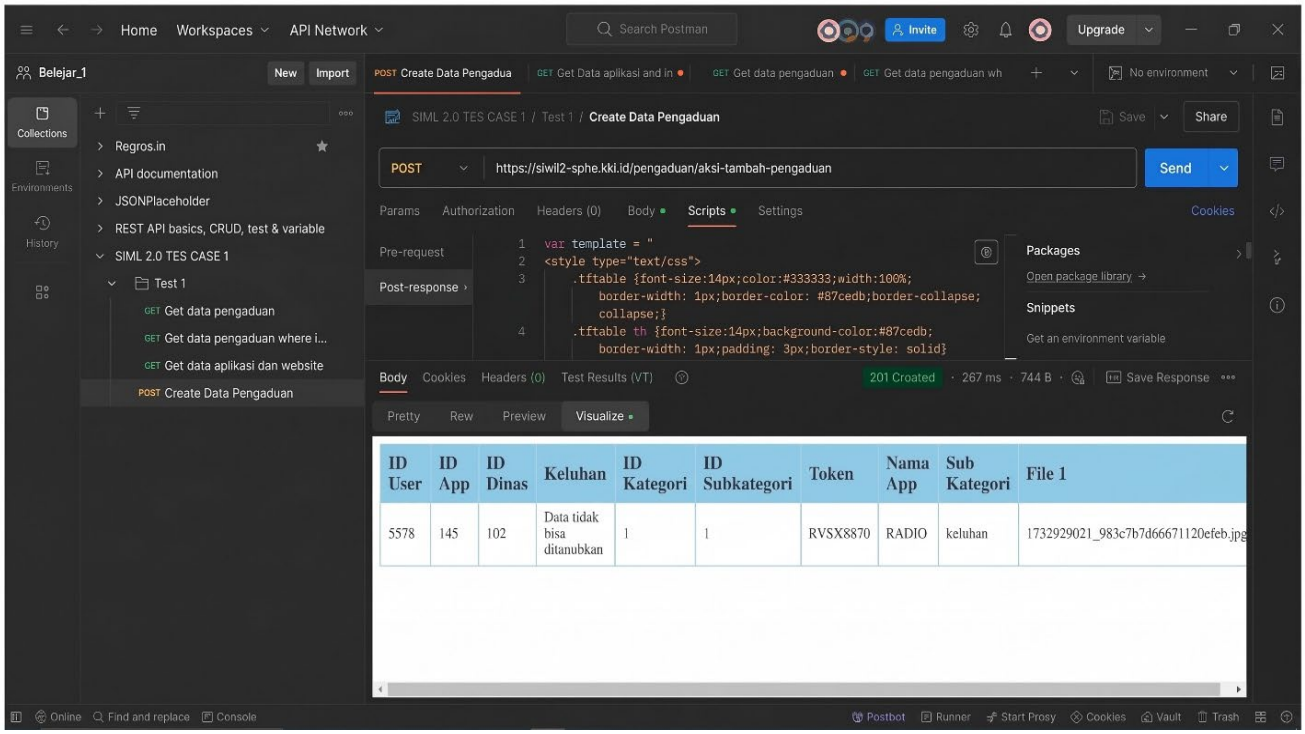

Figure 15. Added data analysis

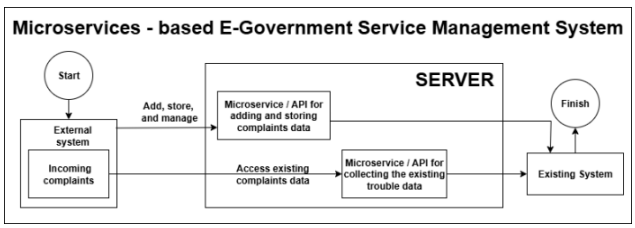

Figure 16. Microservice-based e-government service management system (M-EGSMS) framework

Based on this research, the M-EGSMS Framework is developed as shown in Figure 16. The system starts receiving incoming complaints from the external system. It then adds, stores, and manages complaint data on the server and in the existing system. The existing system can also access the complaint data through the microservice/API.

The proposed framework adopts a layered microservices architecture for e-government service management, where citizens submit complaints and public service requests through a unified API Gateway that serves as the main entry point for request routing, security control, authentication, and access management. Within this architecture, the system is decomposed into independent microservices, including complaint management, user and profile management, workflow coordination, notification handling, and analytics, so that each service can operate, scale, and evolve according to its own business function. To support dynamic interaction among distributed services, the framework incorporates service discovery to enable automatic service registration and instance lookup, improving flexibility, interoperability, and fault tolerance. The framework also applies data partitioning through a database-per-service approach, where each microservice owns its data repository to reduce coupling, strengthen autonomy, and support independent maintenance and scalability.

Under the database-per-service approach, each microservice exclusively owns its schema and data store, enforcing encapsulation and preventing direct cross-service database access. Consequently, inter-service data consistency is maintained via API-mediated communication through the Unified API Gateway, balancing strict ACID consistency with service autonomy and scalability.

Beyond access control, microservice governance in M-EGSMS also encompasses API versioning to manage backward compatibility as services evolve, service-level policy enforcement at the Unified API Gateway (e.g., rate limiting and request validation), and lifecycle management for service registration and deregistration through the service discovery component. These governance mechanisms are essential to maintain consistency, traceability, and controlled evolution as new services are added or existing ones are modified.

In addition, the deployment topology organises the architecture into client, gateway, service, and data layers deployed on container-based infrastructure or orchestration platforms such as Kubernetes, which enhances system availability, resilience, and operational efficiency. Within this deployment topology, the orchestration layer handles container scheduling, horizontal scaling of service instances

based on load, health-check-driven fault recovery, and rolling updates without service downtime. This orchestration capability is essential for the microservices layer, where independently deployed services (complaint management, user and profile management, workflow orchestration, notification, and service discovery) must be coordinated reliably despite operating as separate deployable units.

Overall, integrating API Gateway control, service discovery, domain-oriented microservices, partitioned data management, and a scalable deployment topology provides a modular, implementation-oriented foundation for modernising e-government service management and integrating it with existing government information systems.

Service decomposition in M-EGSMS follows three operational criteria. First, functional cohesion groups operations that share a single business purpose and are frequently invoked together into one service; for example, complaint intake, validation, and status tracking are cohesive functions kept within the Complaint Management Service rather than split further, since separating them would introduce unnecessary inter-service calls for a single logical operation. Second, business capability boundaries align each service with a distinct organisational responsibility rather than a technical layer; for instance, user authentication and profile data are grouped under the User and Profile Management Service because they represent a single capability (identity and profile ownership) independent of how complaints or workflows are processed. Third, autonomous service design requires each resulting service to own its data exclusively and be independently deployable and scalable; this criterion drove the separation of Workflow Orchestration from Notification, since workflow state transitions and outbound notifications have different scaling profiles and failure characteristics and therefore should not share a deployment unit or database. Applying these three criteria to the department's core functions yields the six services shown in Figure 16 and Table 6: Complaint Management, User and Profile Management, Workflow Orchestration, Notification, Service Registry, and Service Discovery, each corresponding to a distinct business capability with minimal inter-service coupling.

Table 6. Service decomposition mapping

Service	Business Capability	Decomposition Rationale
Complaint Management Service	Complaint intake, validation, status tracking	High internal cohesion; frequently co-invoked operations kept together
User & Profile Management Service	Identity, authentication, profile ownership	Distinct capability independent of complaint/workflow logic
Workflow Orchestration Service	Complaint routing and process-state coordination	Separated from Notification due to differing scaling and failure profiles
Notification Service	Outbound alerts to citizens/staff	Independently scalable; decoupled from workflow state to avoid cascading failure
Service Registry	Service registration	Infrastructure-level capability, distinct from business-domain services
Service Discovery	Instance lookup for inter-service communication	Infrastructure-level capability, supports runtime coordination

The deployment strategy of M-EGSMS follows a container-based approach, in which each microservice, such as Complaint Management, User and Profile Management, Workflow Orchestration, Notification, Service Registry, and Service Discovery, is packaged into an independent container image with its own runtime dependencies and configuration. Each service is containerised using Docker and deployed to a Kubernetes cluster managed by the Department of Communication and Informatics. The deployment workflow proceeds in stages: (1) building and versioning the container image for each service after implementation and API testing are completed; (2) deploying the image to the target environment, where the Unified API Gateway is updated to route requests to the new service instance; (3) performing a health check to confirm the new instance is responding correctly before directing production traffic to it; and (4) retaining the previous container version to allow rollback in case the new deployment introduces errors. Services can be deployed and updated independently, allowing individual services, such as notification, to be redeployed without affecting the availability of others, such as complaint management. While the current implementation runs on the existing setup, the architecture is designed to be compatible with container orchestration platforms such as Kubernetes for future production-scale deployment, supporting automated scaling and fault recovery.

In M-EGSMS, a single business process, such as complaint submission, spans multiple independent services: the complaint management service, which records the complaint; the workflow orchestration service, which routes it for review; and the notification service, which alerts the relevant staff. Because each service owns its own database under the database-per-service approach, this process cannot rely on a single atomic transaction spanning all services. Instead, M-EGSMS manages cross-service transactions using an orchestrator-coordinated Saga pattern via the workflow orchestration service, in which each service performs its local transaction and publishes an event indicating success or failure, allowing the next service in the sequence to proceed accordingly. If the notification service cannot deliver an alert after a complaint has been recorded and routed, it triggers a compensating action rather than attempting to roll back the entire distributed transaction. This approach favours eventual consistency over strict atomicity: each service's local data remains consistent immediately after its own operation, while overall workflow consistency across services is achieved progressively as compensating or corrective actions complete.

The testing in this research should be regarded as an initial functional validation of the M-EGSMS prototype. The evaluation primarily emphasised API testing, feature validation, and defect identification through representative test scenarios conducted by two expert testers. Although this testing was adequate to confirm the basic functionality of the proposed framework, it is insufficient to provide a comprehensive assessment of performance, scalability, and concurrency within a distributed microservices environment.

In particular, the scope of the present research did not include stress testing, load testing, or systematic multi-user concurrency evaluation. Therefore, future research should broaden the evaluation by expanding the range of testing scenarios, involving more participants, and incorporating quantitative performance analysis under realistic workload conditions.

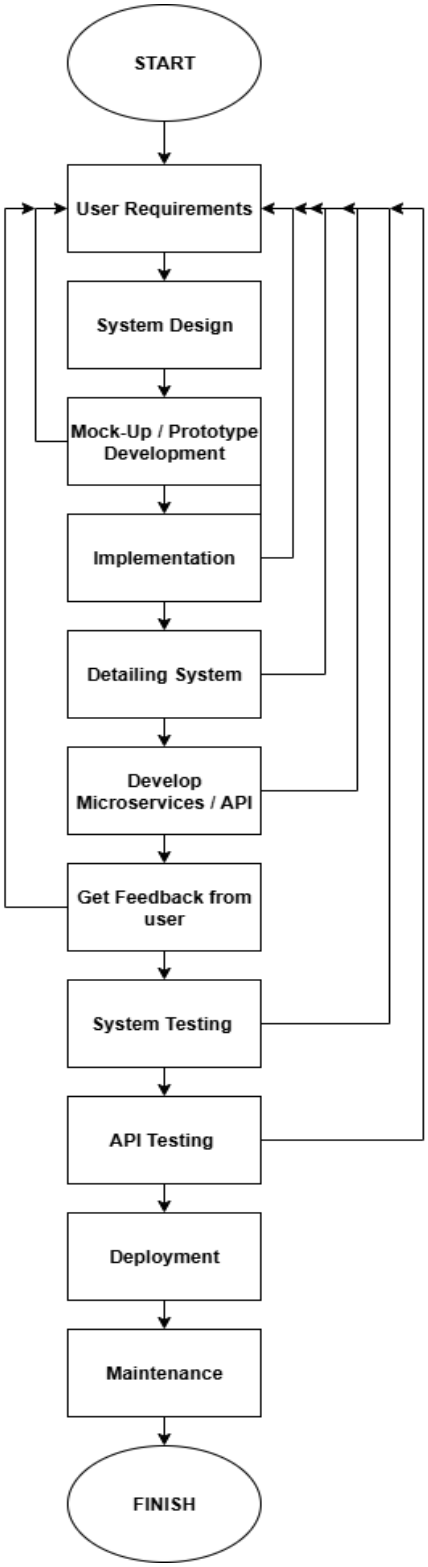

Figure 17. Proposed Fast Collaboration Competencies 2.0 (FCC 2.0) model

Furthermore, to strengthen the empirical validation of the proposed M-EGSMS framework, the evaluation should include performance metrics covering response time, throughput, and scalability. In particular, response time should be measured in terms of the average, median, and 95th percentile response time, and should be recorded separately for each core endpoint, such as complaint submission, complaint retrieval, user login, and reporting, so that performance can be examined per functional service rather than as a single aggregate figure. Moreover, throughput should be evaluated in requests per second and completed transactions per minute, along with the error rate under load, to indicate the system's request-handling capacity and reliability as demand increases. Additionally, because M-EGSMS adopts a microservices-based architecture, assess scalability by observing performance under increasing numbers of concurrent users (e.g., 10, 50, 100, and 200 users) and by examining whether adding service instances improves throughput or stabilises response time. Where available, CPU and memory utilisation should also be recorded to provide further insight into resource consumption under varying workloads. These metrics matter because they show not only the framework's functional correctness, but also its operational feasibility for deployment in real e-government environments.

The present research mainly focuses on the architectural design, system performance measurement, and functional validation of the proposed M-EGSMS framework. The study has not yet obtained quantitative user feedback through a structured evaluation instrument. The current testing instead relied on expert-based functional validation, which is appropriate for confirming system correctness but does not capture end-user perceptions of usability, satisfaction, or perceived usefulness. Therefore, the findings on usability and user acceptance remain limited and should be considered preliminary. Future research is expected to include a questionnaire-based user evaluation involving both citizens submitting complaints and staff of the Department of Communication and Informatics who operate the system, using a validated instrument such as the System Usability Scale (SUS) or a Technology Acceptance Model (TAM)-based questionnaire covering perceived usefulness, perceived ease of use, and behavioural intention to use. A minimum sample of 30–50 respondents across both user groups would allow descriptive statistical analysis, such as mean usability scores and item-level ratings, to complement the functional and architectural evaluation presented in this study and provide a more complete assessment of M-EGSMS from both technical and user-centred perspectives.

Table 7. Comparison between FCC 1.0 and FCC 2.0

FCC 1.0 Stages	FCC 2.0 Stages
1. User Requirements 2. System Design 3. Mock-Up/Prototype Development 4. Implementation 5. Detailing System 6. Develop Microservices/API 7. Testing 8. Deployment 9. Maintenance	1. User Requirements 2. System Design 3. Mock-Up/Prototype Development 4. Implementation 5. Detailing System 6. Develop Microservices/API 7. Get Feedback from User 8. System Testing 9. API Testing 10. Deployment 11. Maintenance

Note: FCC: Fast Collaboration Competencies; API: Application Programming Interface.

This research also improved the FCC 1.0 Diagram into FCC 2.0, as shown in Figure 17. The proposed FCC 2.0 consists of

several stages: user requirements, system design, mock-up/prototype development, implementation, system detailing, microservices/API development, user feedback, system testing, API testing, deployment, and maintenance. As shown in Table 7, additional stages include microservices/API development and API testing.

Furthermore, Table 8 compares this research with other research. The four studies represent distinct approaches to advancing e-government services, differing sharply in research focus, architectural design, and treatment of interoperability as explained below:

1) Research focus: M-EGSMS is implementation-oriented, centred on building a working microservices-based complaint management system and generalising the development process itself through FCC 2.0. This contrasts with study [33] and study [34], which are technology-adoption studies that examine how and to what maturity level blockchain and AI, respectively, have been applied in e-government contexts, rather than delivering a new system architecture. Study [35] departs furthest from the others, focusing entirely on citizen behaviour rather than any technical system.

2) Architecture: M-EGSMS is the only study among the four with a concrete, layered technical architecture, such as client, gateway, service, and data, backed by an explicit service-decomposition rationale and a defined deployment model, such as containers/Kubernetes. Study [33]'s architecture is structural but domain-specific to blockchain components, such as network, consensus, and ledger, and study [34]'s is similarly technology-specific, centred on AI/deep-learning infrastructure. Study [35] has no technical architecture at all.

3) Interoperability: Three of the four studies treat interoperability as a core design concern, but through different mechanisms. M-EGSMS achieves it via API-based integration and service discovery; study [33] achieves it via a shared blockchain ledger enabling trusted cross-agency data exchange; study [34] achieves it via a centralised AI-enabled hub connecting heterogeneous systems and legacy platforms. Study [35] is the outlier as it does not address interoperability at all, since its unit of analysis is the individual citizen, not the system.

Overall, the comparison shows that e-government research spans a spectrum from purely behavioural/adoption studies [35] to technology-specific architectural studies [33, 34] to a full implementation-and-methodology contribution (M-EGSMS). This positions M-EGSMS's contribution as distinct in scope: unlike studies [33, 34], which centre on a single enabling technology (blockchain, AI), M-EGSMS proposes a general-purpose architectural pattern (microservices) paired with a reusable development methodology (FCC 2.0), making its contribution both technical and methodological rather than technology-adoption-focused.

Table 8. Comparison with other studies

Research	Research Focus	Architecture	Interoperability
M-EGSMS using FCC model	Microservices-based complaint management system; refines FCC 1.0 into FCC 2.0.	Layered architecture (client, gateway, service, data). Six microservices; database per service; Kubernetes deployment.	Unified API Gateway routes requests and enforces access control. Service discovery supports instance lookup.
The use of blockchain technology in e-government services [33]	Blockchain adoption in e-government, studied across cases by maturity level.	Five components: Users, blockchain network, consensus, smart contracts, distributed ledger.	Shared blockchain infrastructure enables trusted data sharing among agencies.
Automating e-government services with artificial intelligence [34]	Shared blockchain infrastructure enables trusted data sharing among agencies.	Centralized information-management architecture and an AI-enabled platform.	AI-enabled architecture links systems, agencies, and legacy platforms.
Adoption and use of e-government services: The case of Romania [35]	Factors shaping citizens' adoption of e-government, using Romania as a case.	Factors shaping citizens' adoption of e-government, using Romania as a case.	Factors shaping citizens' adoption of e-government, using Romania as a case.

Moreover, there were some challenges in the M-EGSMS development process, such as:

•Junior programmers with minimal experience, who have slowed M-EGSMS development and have difficulties writing efficient and bug-free code, as well as not understanding basic principles of a good software development process

•Slow input of data from the staff of the Department of Communication and Informatics in Sampang.

Furthermore, there are some proposed solutions to overcome the challenges above as follows:

•Code review regularly, as junior programmers need to be monitored and send a daily report to mitigate errors earlier

•Manual and beta testing by the quality assurance team and selected users to detect bugs

•Detailed and complete testing scenarios to anticipate various situations that are experienced by the system

4. CONCLUSIONS

The M-EGSMS framework provides a structured approach for public institutions to streamline service complaint handling through a scalable microservices architecture. Microservices can improve the system's benefits, including scalability, resilience, agility, technological flexibility, heterogeneity, and cost efficiency. Furthermore, M-EGSMS should include elements such as business capability alignment, communication control, service discovery, traffic distribution, circuit breakers, data management, security, containerization, and orchestration tools. Implementing microservices also brings advantages and challenges, including distributed systems, overhead costs, and testing complexity. Beyond its practical implementation, this research contributes theoretically by extending service decomposition principles into the e-government context and by generalising the FCC 2.0 model as a reusable methodological framework for future microservices-based public-service systems.

Building on the current implementation-focused contribution, future research will pursue the following concrete directions. First, a quantitative performance evaluation of M-EGSMS will be conducted, measuring response times, such as average, median, and 95th percentile per core endpoint, as well as throughput, such as requests per second and completed transactions per minute, and scalability under increasing concurrent-user loads for 10, 50, 100, and 200 users, together with CPU and memory utilisation. Second, a structured, questionnaire-based user evaluation will be conducted with both citizens and staff of the Department of Communication and Informatics, using a validated instrument such as the SUS or a TAM-based questionnaire, to assess usability, satisfaction, and perceived usefulness. Third, the accumulated API request logs will be statistically analysed to characterise request distribution, response patterns, and error trends once sufficient operational data has been collected. Fourth, the current API key-based security mechanism will be extended with modern authentication standards, including OAuth 2.0, JSON Web Token (JWT), and mutual Transport Layer Security (mTLS), to strengthen access control and support finer-grained authorisation. Finally, the deployment and orchestration strategy will be validated at production scale, including containerised deployment on Kubernetes with defined auto-scaling and fault-recovery policies, as part of the planned implementation of a service management information system for the Department of Communication and Informatics of Sumenep, Madura, Indonesia, using the FCC 2.0 model.

ACKNOWLEDGMENTS

We would like to thank the Institute for Research and Community Service, Universitas Trunodjoyo Madura, for the research funding in the National Collaboration scheme; our partner Department of Communication and Informatics of Sampang, Madura, Indonesia, as well as our research team, Sevin Dias Andika, Yohan Fadhillah Jibraltar, and Muhammad Nadda Khatani.

REFERENCES

[1] Baškarada, S., Nguyen, V., Koronios, A. (2020). Architecting microservices: Practical opportunities and challenges. Journal of Computer Information Systems, 60(5): 428-436. https://doi.org/10.1080/08874417.2018.1520056

[2] Lv, Y., Tan, W. (2022). Infrastructure smart service system based on microservice architecture from the perspective of informatization. Mobile Information Systems, 2022(1): 1344720. https://doi.org/10.1155/2022/1344720

[3] Assunção, W.K.G., Krüger, J., Mendonça, W.D.F. (2020). Variability management meets microservices: Six challenges of re-engineering microservice-based webshops. In Proceedings of the 24th ACM Conference on Systems and Software Product Line, pp. 1-6. https://doi.org/10.1145/3382025.3414942

[4] Salim, Y., Muis, I., Syafie, L., Azis, H., Manga, A.R. (2023). One-gateway system in managing campus information system using microservices architecture. Bulletin of Social Informatics Theory and Application, 7(2): 83-91. https://doi.org/10.31763/businta.v7i2.635

[5] Dahri, F., El Hanafi, A.M., Handoko, D., Wulan, N. (2022). Implementation of microservices architecture in learning management system e-course using web service method. Sinkron, 7(1): 76-82. https://doi.org/10.33395/sinkron.v7i1.11229

[6] Barroca Filho, I.D.M., Sampaio, S.C., Junior, G.S.A., et al. (2019). A microservice-based health information system for student-run clinics. In Computational Science and Its Applications – ICCSA 2019, Saint Petersburg, Russia, pp. 3-16. https://doi.org/10.1007/978-3-030-24308-1_1

[7] Liu, Z., Yu, H., Fan, G., Chen, L. (2021). Reliability modeling and analysis of hospital information system based on microservices. In 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, pp. 313-318. https://doi.org/10.1109/PIC53636.2021.9687027

[8] Li, X., Xi, Y., Zhu, H., Ling, J., Zhang, Q. (2020). Infrastructure smart service system based on microservice architecture. In Information Technology in Geo-Engineering: Proceedings of the 3rd International Conference (ICITG), Guimarães, Portugal, pp. 131-143. https://doi.org/10.1007/978-3-030-32029-4_12

[9] Rozi, I.F., Ariyanto, R., Pramudita, A.N., Yunianto, D.R., Putra, I.F. (2020). Implementation of microservices architecture on certification information system (case study: LSP P1 State Polytechnic of Malang). IOP Conference Series: Materials Science and Engineering, 732(1): 012085. https://doi.org/10.1088/1757-899X/732/1/012085

[10] Tang, W., Wang, L., Xue, G. (2019). Design of information system architecture of garment enterprises based on microservices. Journal of Physics: Conference Series, 1168(3): 032128. https://doi.org/10.1088/1742-6596/1168/3/032128

[11] Elyyani, Andrian, Y., Utama, A.Z., Saputro, M.F.E., Fatimah, S.K. (2022). Design of decision support system service in the Space Science Center using microservices approach. Journal of Physics: Conference Series, 2214(1): 012029. https://doi.org/10.1088/1742-6596/2214/1/012029

[12] Mateus-Coelho, N., Cruz-Cunha, M., Ferreira, L.G. (2021). Security in microservices architectures. Procedia Computer Science, 181: 1225-1236. https://doi.org/10.1016/j.procs.2021.01.320

[13] Gong, Y., Gu, F., Chen, K., Wang, F. (2020). The architecture of micro-services and the separation of front-end and back-end applied in a campus information system. In 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, pp. 321-324. https://doi.org/10.1109/AEECA49918.2020.9213662

[14] Cavallari, M., Tornieri, F. (2018). Information systems architecture and organization in the era of microservices. In Network, Smart and Open: Three Keywords for Information Systems Innovation, pp. 165-177. https://doi.org/10.1007/978-3-319-62012-1_10

[15] van Steen, M., Ferguson, D., Pahl, C. (2022). Proceedings of the 12th International Conference on Cloud Computing and Services Science: CLOSER 2022. SCITEPRESS. https://doi.org/10.5220/0000159300003200

[16] Vural, H., Koyuncu, M., Guney, S. (2017). A systematic literature review on microservices. In Software

Architecture, Lecture Notes in Computer Science, pp. 203-217. https://doi.org/10.1007/978-3-319-62407-5_14
[17] Di Francesco, P., Lago, P., Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software, 150: 77-97. https://doi.org/10.1016/j.jss.2019.01.001
[18] Iden, J., Eikebrokk, T.R. (2013). Implementing IT service management: A systematic literature review. International Journal of Information Management, 33(3): 512-523. https://doi.org/10.1016/j.ijinfomgt.2013.01.004
[19] Alshuqayran, N., Ali, N., Evans, R. (2016). A systematic mapping study in microservice architecture. In Proceedings of 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, pp. 44-51. https://doi.org/10.1109/SOCA.2016.15
[20] Papadomichelaki, X., Magoutas, B., Halaris, C., Apostolou, D., Mentzas, G. (2006). A review of quality dimensions in e-government services. In International Conference on Electronic Government, Krakow, Poland, pp. 128-138. https://doi.org/10.1007/11823100_12
[21] Puspitasari, N., Budiman, E., Sulaiman, Y.N., Firdaus, M.B. (2021). Microservice API implementation for e-government service interoperability. Journal of Physics: Conference Series, 1807(1): 012005. https://doi.org/10.1088/1742-6596/1807/1/012005
[22] Ray, S., Rao, V.V. (2004). Evaluating government service: A customers' perspective of e-government. In Proceedings of the 4th European Conference on E-government, Dublin, Ireland, pp. 17-18.
[23] Fajar, A.N., Shofi, I.M. (2019). Service oriented design for Indonesian e-government system using SOA. IOP Conference Series: Materials Science and Engineering, 598(1): 012106. https://doi.org/10.1088/1757-899X/598/1/012106
[24] Nugraha, T.F., Wibowo, W.S., Genia, V., Fadhil, A., Ruldeviyani, Y. (2024). A practical approach to enhance data quality management in government: Case study of Indonesian customs and excise office. Journal of Information Systems Engineering and Business Intelligence, 10(1): 51-69. https://doi.org/10.20473/jisebi.10.1.51-69
[25] Wijaya, L., Raharjana, I.K., Purwanti, E. (2018). Strategic management for IT services on outsourcing security company. Journal of Information Systems Engineering and Business Intelligence, 4(1): 46-56. https://doi.org/10.20473/jisebi.4.1.46-56
[26] Wang, F.J., Fahmi, F. (2018). Constructing a service software with microservices. In Proceedings - 2018 IEEE World Congress on Services (SERVICES), San Francisco, CA, USA, pp. 33-34. https://doi.org/10.1109/SERVICES.2018.00035
[27] Puspitasari, I., Cahyani, D.I. (2018). A user-centered design for redesigning e-government website in public health sector: An approach to improve the user experience. In 2018 International Seminar on Application for Technology of Information and Communication, Semarang, Indonesia, pp. 219-224. https://doi.org/10.1109/ISEMANTIC.2018.8549726
[28] Fakhruzzaman, M.N., Dimitrova, D.V. (2020). Factors influencing e-government adoption in Indonesia: The importance of perceived risk. Journal of Advanced Research in Dynamical and Control Systems, 12(6): 125-131. https://doi.org/10.5373/JARDCS/V12SP6/SP20201015
[29] Herianingrum, S., Muhammad Nafik, H., Fauzi, Q., Afifa, F.U., Laila, N. (2019). The effect of government expenditure on Islamic human development index. Opcion, 35(88): 685-703.
[30] Yusuf, M., Sophan, M.K., Darmawan, A.K., Satoto, B.D., Anamisa, D.R., Agustiono, W. (2023). Fast collaboration competencies model for software development life cycle (SDLC). In Proceeding of IEEE 9th Information Technology International Seminar (ITIS), pp. 1-6. https://doi.org/10.1109/ITIS59651.2023.10420226
[31] Rahmatullah, R., Habibi, A., Khaeruddin, K., et al. (2025). A study of user satisfaction and net benefits in Indonesia through the DeLone and McLean model for e-government success. Discover Sustainability, 6(1): 710. https://doi.org/10.1007/s43621-025-01645-4
[32] Ting, T.T., Lee, M.Y., Chok, S.X., et al. (2024). Digital government: Social media as a mediator in technology acceptance with political knowledge, interest, and participation. Online Journal of Communication and Media Technologies, 14(4): e202454. https://doi.org/10.30935/ojcmt/15145
[33] Lykidis, I., Drosatos, G., Rantos, K. (2021). The use of blockchain technology in e-government services. Computers, 10(12): 168. https://doi.org/10.3390/computers10120168
[34] Al-Mushayt, O.S. (2019). Automating e-government services with artificial intelligence. IEEE Access, 7: 146821-146829. https://doi.org/10.1109/ACCESS.2019.2946204
[35] Colesca, S.E., Dobrica, L. (2008). Adoption and use of e-government services: The case of Romania. Journal of Applied Research and Technology, 6: 204-217.