 ISSN:2302-8149
e-ISSN:2540-9719

Implementation of a Face Recognition API for an Automated Web-based Attendance System using Extreme Programming

[1]Rizky Wandri*, [2]Mutia Fadhilla, [3]Dwi Fiqri Qurniawan, [4]Eka Deddy Saputra
[1,2,3]Program Studi Teknik Informatika, Universitas Islam Riau
[4] Independent Researcher, Pekanbaru, Indonesia
[1,2,3]Jl. Kaharuddin Nst No.113, Riau, Indonesia
*e-mail: rizkywandri@eng.uir.ac.id

(***received*:** 30 April 2026, ***revised*:** 8 June 2026, ***accepted*:** 10 June 2026)

Abstract

Manual attendance recording is still vulnerable to data manipulation, entry delays, and high reliance on manual input. This research focuses on the implementation and integration of the Facial Recognition REST API into a web-based attendance information system to automate attendance recording. The novelty of this research lies in the integration of REST API-based facial recognition services with a three-layer architecture consisting of a camera device, a Raspberry Pi 5-based edge server, and a Laravel-based application server, developed using the Extreme Programming (XP) method, resulting in a modular, responsive, and easily scalable attendance system compared to previous approaches. The system was developed in four sprints over one month with an average speed of 15.5 story points per sprint, based on twelve user stories defined in the planning stage. In the facial recognition module, the identification process utilizes a combination of Haar Cascade for facial position detection, FaceNet for facial embedding formation, and Cosine Similarity for similarity measurement against a reference database consisting of 18 students with 6 reference photos per student. System testing was conducted using three class videos with a resolution of 1920×1080 pixels with an average duration of 5 minutes and 28 seconds. Evaluation of the accuracy of the facial recognition model was part of a collaborative study reviewed separately; this paper focuses on the system's integration performance and functionality. Functional testing using black box methods across 20 scenarios demonstrated that all modules—authentication, master data management, facial registration, facial recognition-based attendance, reporting, and error handling—performed according to specifications, with a consistent system response time of 2–4 seconds. The results demonstrate that the system reduces reliance on manual input, improves record-keeping efficiency, and provides real-time attendance data, and is ready for further development for broader adoption in educational settings.

Keywords: *automated attendance system, face recognition, rest api integration, edge computing, extreme programming*

1 Introduction

The development of information and communication technology has brought significant changes in various aspects of life, including in the field of education [1], [2]. One of the challenges faced by educational institutions is effective and efficient attendance management. The manual attendance system that is still widely used is prone to recording errors, data manipulation, late entry, and does not support real-time monitoring. The main problem raised in this study is how to develop an automatic attendance recording system that can reduce dependence on manual input while providing real-time attendance data through the integration of facial recognition technology into a web-based information system.

Various technology-based solutions have been developed to address the limitations of manual systems, including the use of facial recognition technology and computing services. However, most existing solutions are still developed using traditional development models and monolithic approaches, limiting the flexibility, scalability, and maintainability of

 ISSN:2302-8149
e-ISSN:2540-9719

systems when deployed on a larger institutional scale. These limitations highlight the need for a more adaptive development approach and a more modular system architecture.

To address these needs, this study adopted the Extreme Programming (XP) method, a leading agile software development methodology that emphasizes flexibility, iterative development, and active user involvement. This method integrates practices and values that encourage simplicity, communication, and responsiveness to feedback, which are crucial in today's fast-paced development environment [3], [4], [5]. XP is well-suited for projects involving small teams and requiring high flexibility in responding to changing requirements [6], [7], [8], and enables teams to produce final products that better align with users' real needs [9], [10]. Research [11] also shows that XP facilitates user-centered application design even when initial requirements are unclear.

Although various studies have successfully developed facial recognition-based attendance systems, most still focus on local implementations with a monolithic approach and have not optimized the integration of the Facial Recognition API directly with web applications. Unlike previous studies, this study combines agile Extreme Programming methods with REST API-based facial recognition services to produce a more responsive, flexible, and scalable attendance system. This integration enables real-time attendance data processing, increased development efficiency through iterative cycles, and ease of system scalability and maintenance. Further elaboration on this research gap is described in the Literature Review section.

Based on the background and gaps, the objectives of this research are: (1) to implement the integration of Facial Recognition API into a web-based attendance information system as the main identification module; (2) to develop a system using the Extreme Programming method with a three-layer architecture consisting of a camera device, an edge server, and an application server; and (3) to test the functionality and performance of the developed system in automating attendance recording. Validation and functional testing are performed at each iteration to ensure the reliability and feasibility of the system when applied to real-world scenarios.

2 Literature Review

The development of automated attendance systems is inseparable from the advancement of supporting computing technologies, particularly cloud and edge computing. In the field of education, the concept of cloud has gained attention, where institutions combine resources to access shared cloud services, reduce costs, and improve service accessibility [12]. This trend illustrates the growing recognition of the collaborative potential of cloud computing [13], [14], and as cloud adoption increases across various sectors, especially in the public sphere, it becomes crucial to maximize the benefits of cloud technology [15]. Edge computing exists to alleviate the centralization pressures faced by traditional cloud models to improve performance [16], [17], [18], by promising solutions to latency and bandwidth constraints, thus facilitating real-time processing [19]. Edge computing strategically places computing resources closer to the data source, thus enabling lower latency and improving data handling efficiency [20], [21]. For example, the application of edge computing in applications such as smart water systems enables local data processing that improves service efficiency and reliability [22]. Edge computing architecture not only overcomes the shortcomings of traditional cloud systems, but also leverages the advantages of local data processing for the development of robust frameworks in various sectors, including healthcare, smart cities, and the Metaverse [23], [24]. Thus, while cloud computing is becoming the cornerstone of modern IT infrastructure, the evolution of edge computing offers valuable technologies to overcome the inherent limitations of centralized data processing [14], [17], [18], [25], [26].

The development of digital technology-based attendance systems shows a significant shift from manual methods to more efficient and accurate automated systems. Manual attendance systems still have various limitations, such as being prone to recording errors, data manipulation, and not supporting real-time monitoring. Based on a bibliometric analysis conducted by [27] of 161

 ISSN:2302-8149
e-ISSN:2540-9719

publications from the Scopus database for the 2019–2024 period, research related to facial recognition-based attendance systems has experienced a significant increase with an annual growth rate of 20.11%, with a primary focus on the utilization of deep learning technology, convolutional neural networks (CNN), and the integration of automated systems in educational environments.

One concrete implementation of facial recognition technology for attendance systems is the system developed by [28] which utilizes a Raspberry Pi device and a camera module to perform real-time facial identification. The system works through several stages, starting with image acquisition, preprocessing, facial detection using the Haar-like features algorithm, feature extraction, and facial recognition using the Eigenfaces and Fisherfaces methods. Attendance results are stored in a spreadsheet database that can be accessed remotely. The study demonstrated that the system was capable of improving the efficiency and accuracy of attendance recording, while supporting the concept of contactless attendance, which is safer and more practical. However, this approach was still local and based on specialized hardware, thus limiting its scalability for implementation on a larger institutional scale.

In terms of software development methods, [29] demonstrated the application of the Extreme Programming (XP) method in developing a web-based helpdesk information system in an industrial environment. The study implemented the complete XP phases, including exploration, planning, iteration, production, and maintenance, resulting in a system capable of improving data management efficiency and supporting accurate report generation. Functional testing was conducted using black box testing methods that validated all usage scenarios [5]. This study demonstrated that XP is effective for web-based systems that require the integration of various components and high flexibility to meet changing needs.

The application of Extreme Programming has also proven effective in various contexts of information system development in educational environments. Research by Mutia [30] on a learning process document management information system shows that XP enables efficient system development with support for adaptability to user feedback. Similarly, Priambodo and Muhajirin [31] applied XP in designing a chatbot for student registration that requires rapid and responsive design changes. Both studies confirm that the iterative XP approach is suitable for systems that require gradual adjustments according to user needs, as required in the development of a facial recognition-based attendance system in this study.

Although various studies have successfully developed facial recognition-based attendance systems, a bibliometric review by [27] revealed that most studies still focus on local implementations with limited datasets and have not optimized the integration of web-based systems or the use of external APIs to improve system scalability and flexibility. Furthermore, international collaboration in this area is still low, reaching only 5.59%, indicating that model generalization across different environments and demographics remains a major challenge. The bibliometric study also identified that the integration of edge computing and Internet of Things (IoT) technologies with facial recognition systems represents a significant research opportunity for further development. Based on this gap, this study positions its contribution on the integration of REST API-based facial recognition into a web-based attendance system using Extreme Programming, with a three-layer architecture consisting of a camera device, an edge server as a local computing intermediary, and a Laravel-based application server. This approach not only overcomes the limitations of local system scalability but also leverages the real-time processing advantages of edge architectures as recommended in recent bibliometric studies, thus hopefully resulting in a more adaptive and scalable solution for the needs of today's educational institutions.

3 Research Method

This study uses an applied research approach, focusing on the application of the Face Recognition API to automate attendance recording. The research methodology is designed to ensure that the API integration, facial detection process, and attendance recording can run accurately, stably, and be operational through a web-based information system.

 ISSN:2302-8149
e-ISSN:2540-9719

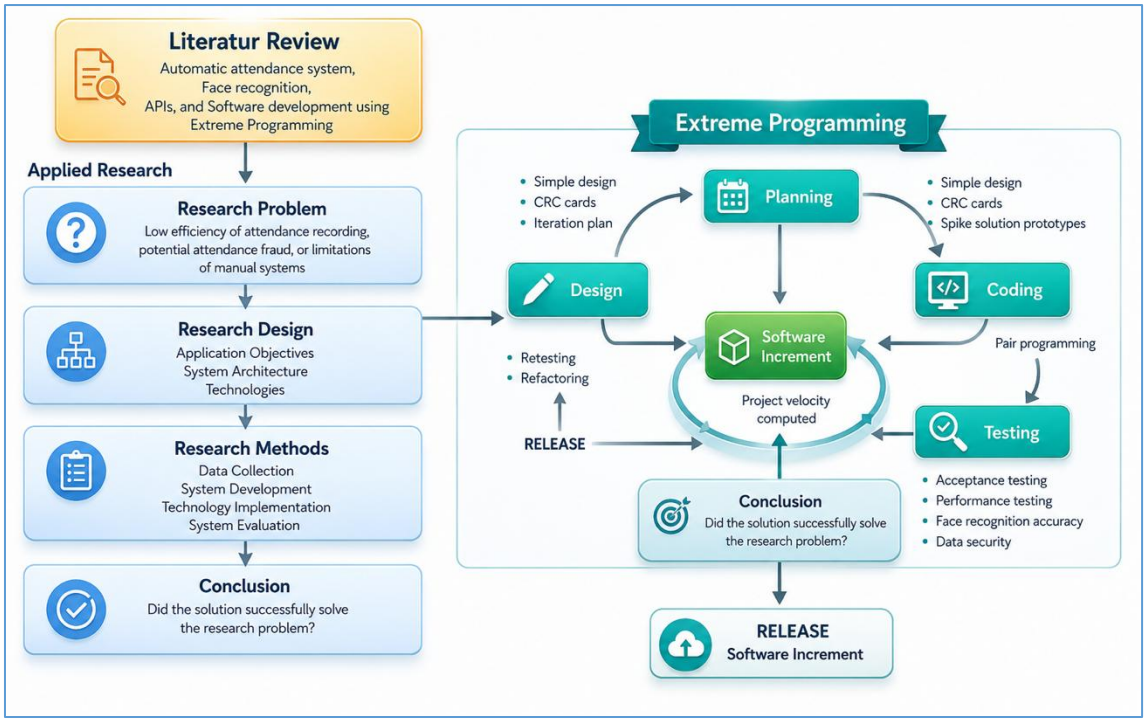

Figure 1 Research stages

Figure 1 depicts the research flow from beginning to end along with the tasks of each member at each stage as follows:

a. Literature Review
   In-depth literature study to understand the concepts, technologies, and methods relevant to the research topic, namely automatic attendance system, face recognition, API, and software development using XP methodology.
b. Research Problem
   This stage focuses on identifying real-world problems, such as low attendance recording efficiency, potential attendance fraud, or limitations of manual systems. Researchers analyze user needs, the operational context, and relevant technical constraints to ensure that the identified problems truly require a feasible solution.
c. Research Design
   This stage involves developing a research design that describes how the solution will be developed and tested. The research design includes:
   1) Formulation of application objectives (face recognition accuracy, recording speed, integration with web systems).
   2) Selection of system architecture, such as client-server, edge computing, or API-based architecture.
   3) Selection of technology, Face Recognition API, web framework (Laravel), and evaluation methods such as measuring accuracy and response time.
   4) Designing implementation scenarios, such as camera placement locations, data flows, and verification mechanisms.
   5) Experimental design, including how the system will be tested, who the users will be, and what conditions will be compared.

   Based on the system requirements analysis, a use case diagram was designed that illustrates the interactions between actors and system functionality. This system involves four main actors: Admin, Lecturer, Student, and Edge Server. This can be seen in the following figure:

 ISSN:2302-8149
e-ISSN:2540-9719

Figure 2 Use case diagram

Admins have full access to master data management, including student data, courses, semesters, and course offerings. Lecturers are responsible for scheduling class sessions, registering student facial data, and monitoring attendance reports. Students can automatically take attendance through facial recognition and access personal attendance reports. The Edge Server acts as a system actor, handling the real-time facial recognition process, from capturing images through the camera, sending data to the API, receiving identification results, and recording attendance into the database. The "include" relationship is used to illustrate the dependencies between processes in the automated attendance flow.
Based on the designed use cases, system requirements are further elaborated in the form of user stories. User stories are used in the Extreme Programming approach to operationally define system features from a user perspective. Table 1 below presents user stories along with acceptance criteria, priorities, and sprint targets for each feature.

Table 1 User story

Id	Actor	User story	Acceptance criteria	Priority	SP	Sprint
US-01	Admin	As an admin, I want to login to the system to access the data management features.	Login successful with valid credentials; rejected if incorrect.	High	3	Sprint 1
US-02	Admin	As an admin, I want to register student data so that students can be identified during attendance.	Student data is saved; duplication is rejected by the system.	High	5	Sprint 1
...	...	...	...	...	...	...
US-11	Edge Server	As an edge server, I want to monitor the attendance status from the edge dashboard so that the synchronization	The dashboard displays active devices, pending sessions, and processed attendance.	Medium	5	Sprint 4

 ISSN:2302-8149
e-ISSN:2540-9719

		process can be monitored.				
US-12	Admin	As an admin, I want the system to handle API connection failures automatically so that attendance doesn't stop.	The system performs an automatic retry; failures are logged and reported to the admin.	Low	4	Sprint 4

d. Research Methods

This stage describes the operational steps for implementing the research design. This section includes:

1) Data Collection Method.
   Data collection was conducted for two purposes: establishing a facial reference database and providing system testing data. The testing data consisted of three live class videos recorded during lectures with a resolution of 1920×1080 pixels and an average duration of 5 minutes and 28 seconds per video. The reference database (student embedded library) consisted of 18 students, each with six reference photos. The reference photos were taken in frontal positions with varying facial angles: facing forward (with and without a smile), tilted to the left and right by approximately 20–30°, and facing upward and downward. These angle variations were intended to improve the reliability of the identification process under various facial positions. An in-depth evaluation of the accuracy of the facial recognition model was part of a collaborative study conducted separately; in this study, the data were used as input for the integration and attendance recording process in the web-based system.
2) System Development Method: Extreme Programming (XP).
3) Technology Implementation Method: Face Recognition API integration, backend module development (REST API), frontend (attendance UI), and server deployment.
4) System Evaluation Method: Measurement of face detection accuracy, error rates, API response time, usability testing, and performance analysis.
5) Validation Method: Trials with users (teachers/others) and data reliability verification.

e. Conclusion

The final stage involves compiling a conclusion that answers whether the solution successfully addressed the research problem. In applied research, this section also summarizes the technology's effectiveness, immediate benefits, and recommendations for practical implementation.

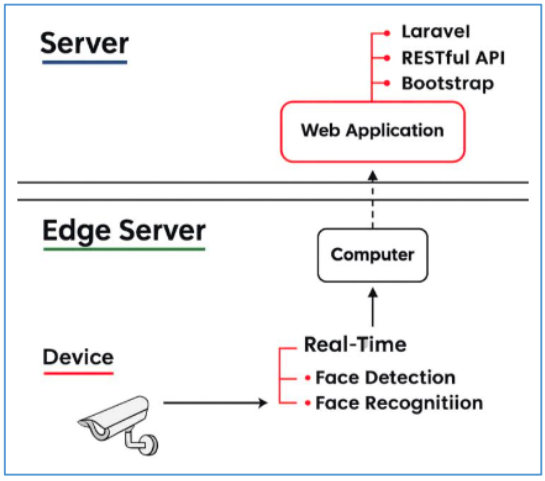

Figure 3 System architecture diagram

Figure 3 above shows how the facial detection and recognition process begins with the device, which serves as the primary input point. The camera performs face detection and recognition in real time, enabling the system to capture facial images and perform identification on-the-spot. This initial

 ISSN:2302-8149
e-ISSN:2540-9719

process ensures that the data sent to the server is more concise and relevant. The facial recognition data is then sent to an edge server, a computer that acts as an intermediary for initial processing before it reaches the main server. Edge servers help reduce the computational load on the central server by executing lighter processes, performing initial validation, and ensuring more efficient data transmission. This approach improves system response time and reduces latency, especially when processing is performed in real time. On the main server, a web application built using Laravel, RESTful API, and Bootstrap serves as the central data management and user interface. All data received from the edge server is then stored, processed, and displayed through a responsive web application. With this architecture, the system can perform detection on the local device, intermediate processing on the edge server, and centralized data storage and management on the application server.

Integration between system components is facilitated by a REST API that acts as a centralized integration layer between the web application, academic database, and facial recognition module on edge devices. This API provides a number of services, including JWT Bearer-based authentication, system status monitoring, academic data management, student data management and facial embedding, and synchronization of attendance results from edge devices. With this structure, the API functions as a centralized backend that connects academic administration processes with facial recognition-based attendance execution on Raspberry Pi devices. On the facial recognition module side, the identification process utilizes a combination of Haar Cascade for facial position detection, FaceNet for generating numerical representations (embeddings) of faces, and Cosine Similarity for measuring the similarity of facial images to a reference database. The following table 2 presents the main service groups and endpoints provided by the API.

Table 2 List of main API endpoints

Service Group	Primary Endpoint	Function
Authentication	POST /api/auth/token	Generating JWT tokens for API access
Status	GET /api/status	Displays application status, server time, and user/client identity.
Runtime	GET /api/runtime/edge-status	Checking the readiness of an edge device or service
Students	GET, POST, PUT /api/students	Student data management
Student Embeddings	GET, POST, DELETE /api/students/{id}/embeddings	Student facial image/embedding management
Edge Attendance	POST /api/edge/attendance/sync	Synchronize attendance results from edge devices to the system
Edge Attendance	GET /api/edge/attendance/sessions	Displays a list of saved attendance sessions
Admin (Academic)	GET/POST/PUT/DELETE /api/admin/{resource}	CRUD data for academic periods, courses, lecturers, class offerings, registrations, sessions, users, and client API

To support the claim of low-latency local processing, system tests were run on edge devices with the specifications detailed in Table 3. The use of Docker containers was intended to maintain consistency of the experimental environment while supporting reproducibility.

Table 3 Edge server specifications

Aspect	Specification
Edge device	Raspberry Pi 5
Memory	8 GB RAM
Cooling system	Active cooler
Operating system	Raspberry Pi OS 64-bit Bookworm
Runtime environment	Docker container
Base image	python:3.11-slim
The purpose of the experiment	Running the system pipeline on the target edge device
Reproducibility support	Docker containers maintain consistency of experimental environments

 ISSN:2302-8149
e-ISSN:2540-9719

To maintain the internal validity of the testing, several variables were held constant across all configurations and videos tested, so that differences in results could be attributed to the system optimization strategy rather than differences in data acquisition conditions. Data acquisition was performed using a smartphone camera with a fixed resolution of 1920x1080 pixels at a frame rate of 60 fps. The camera was placed statically at the front of the classroom on the wall facing the students (the whiteboard area), with a constant position, angle, and height throughout the recording. Lighting, using natural classroom light, was maintained consistent across all three videos; because of its natural nature, there is an inherent intensity gradient inherent to the room conditions, with students in the front rows receiving brighter light while those in the back rows receive dimmer light.
All three videos were recorded under the same acquisition conditions, but with student seating positions randomly varying between videos. This variation was designed to avoid bias in the evaluation toward any particular position, allowing system performance to be tested across the diverse distances and lighting conditions experienced by each student. Computationally, all tests were run on the same hardware, operating system, and Docker container. Because the testing uses pre-recorded video, each configuration is tested on identical inputs, so any differences in results are solely due to changes in system configuration. The reference database and parameters not being tested are also kept constant according to the baseline configuration.

4 Results and Analysis

This research focuses on the implementation of a facial recognition API to automate the attendance recording process in a web-based attendance information system. Several key achievements have been achieved in accordance with the research plan. This research was conducted using an Extreme Programming (XP) approach that emphasizes iterative system development, adapting to changing needs, and supported by continuous testing. The XP implementation in this research consists of the stages of Planning, Design, Coding, Testing, and Software Improvement.

a. Planning

The planning phase of this research was conducted systematically following the Extreme Programming (XP) approach, which emphasizes the identification of adaptive and user-driven requirements. Based on the proposal analysis, the main issues that emerged were the low efficiency of the manual attendance system, the potential for data manipulation, and limitations in real-time attendance monitoring. At this stage, system requirements were formulated in the form of more operational user stories, such as the system's ability to register users, capture facial data, integrate with the Facial Recognition API, and automatically record attendance. Next, acceptance criteria were defined to ensure each feature could be measurably validated, such as the system's success in recording attendance without manual input and its ability to respond to requests within a specified timeframe. Quantitative evaluation of the accuracy of the facial recognition model was part of a collaborative study reviewed separately; in this study, the focus of performance measurement was directed at system integration metrics, namely response time and functional success. As an initial implementation of this phase, a student registration module was developed as the system's foundation. Before system development began, a series of short experiments (solution spikes) were conducted to investigate technical questions that could potentially hinder the sprint. In XP, a solution spike is a short experiment aimed at reducing technical risk and ensuring implementation feasibility. Table 4 presents the spike solutions implemented, along with their results and technical decisions.

Table 4 Spike solution

Id	Spike name	Technical questions investigated	Results / decisions	Sprint	Duration
SP-01	REST API connectivity	Can the facial recognition REST API be integrated directly with Laravel without additional libraries?	Valid - Laravel HTTP Client successfully sends multipart requests to API endpoints with API key authentication in	Sprint 1	2 day

 ISSN:2302-8149
e-ISSN:2540-9719

			the header.		
SP-02	Edge server feasibility	Can a local computer act as an intermediate edge server between the camera and the main server without excessive latency?	Valid - Edge server successfully processes local images and forwards them to API with an average latency of < 1 second.	Sprint 1	2 day
SP-03	Face capture quality	What is the optimal image format and resolution to send to the REST API for maximum facial recognition accuracy?	Valid - JPEG format with a resolution of 640×480, size < 200KB gives the best results. Images are sent as base64 encoded strings.	Sprint 2	1 day
SP-04	Real-time sync mechanism	What is the most efficient mechanism for synchronizing presence data from edge servers to the main server in real-time?	Valid - A RESTful API POST endpoint on the primary server receives data from the edge server each time recognition is successful. No WebSocket is required.	Sprint 3	1 day
SP-05	API error handling	How should the system respond if the REST API is unreachable or returns an error during the attendance process?	Valid - The system performs a maximum of three retries with 2-second intervals. If unsuccessful, the request is queued for reprocessing.	Sprint 4	1 day

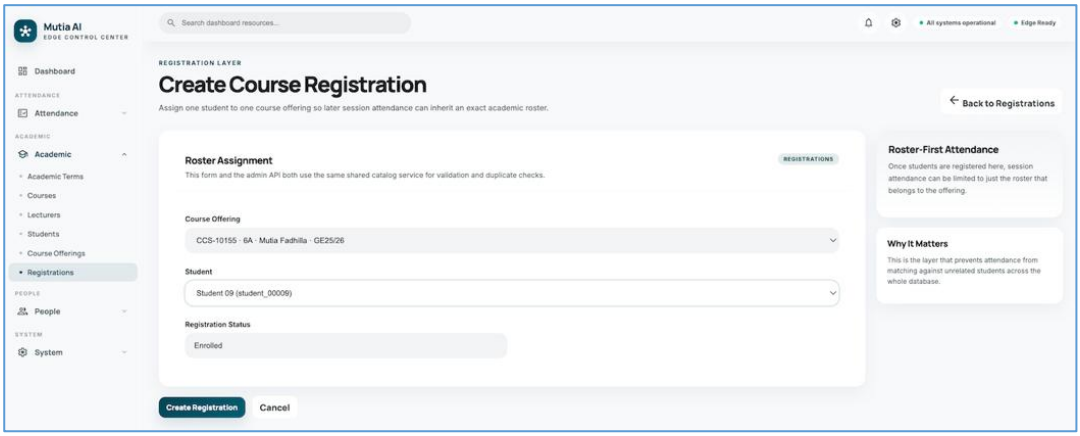

Figure 4 Student registration display

This display shows the student data input process which will be used as the main identity in the system, as well as being the basis for the facial recognition process.

b. Design

The design phase was conducted using a simple design approach based on XP principles, while adhering to the system architecture outlined in the proposal, which is client-server based with API integration. The system was designed as a web application that functions as a

 ISSN:2302-8149
e-ISSN:2540-9719

data manager, while facial identification is handled through an external API service. The system design includes several key components:

1) User and academic data management module
2) Facial image capture module
3) Face recognition API integration module
4) Attendance recording and storage module

The data flow is designed starting from capturing facial images using a camera, sending them to an API, and receiving the identification results, which are then stored in a database. The main focus in this phase is the design of the facial embedding process, which is the foundation of the facial recognition process. The collected facial data is not only stored as images but also processed into a numerical representation (embedding) through an API using the FaceNet model, to be further compared against a reference database using Cosine Similarity.

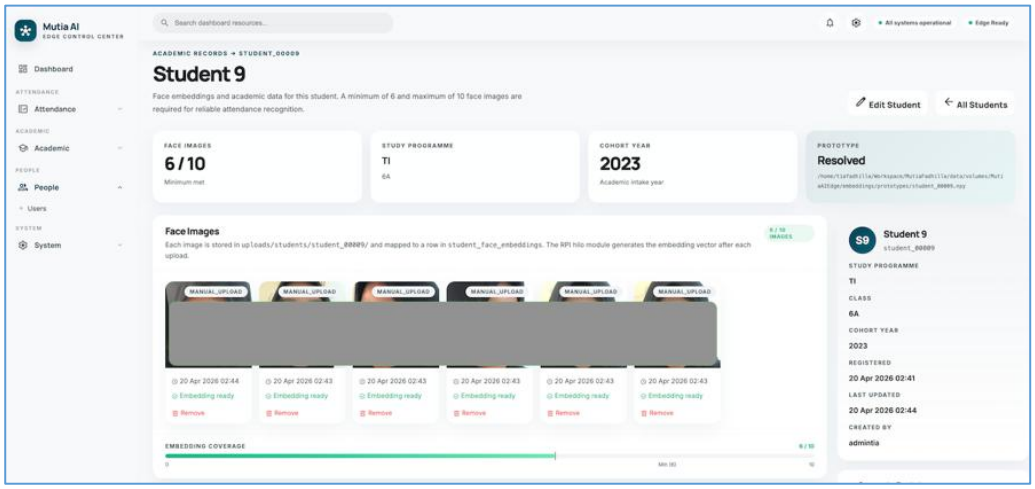

Figure 5 Face embedding process

Figure 5 shows how the system connects the face input process with the API as part of the integration design.

c. Coding

The implementation phase was conducted iteratively according to XP principles, where each feature was developed based on prioritized requirements and tested immediately upon completion. The coding process focused not only on feature development but also on integration between modules to ensure the complete system's operation. Some key modules that have been successfully implemented include:

1) Course Management

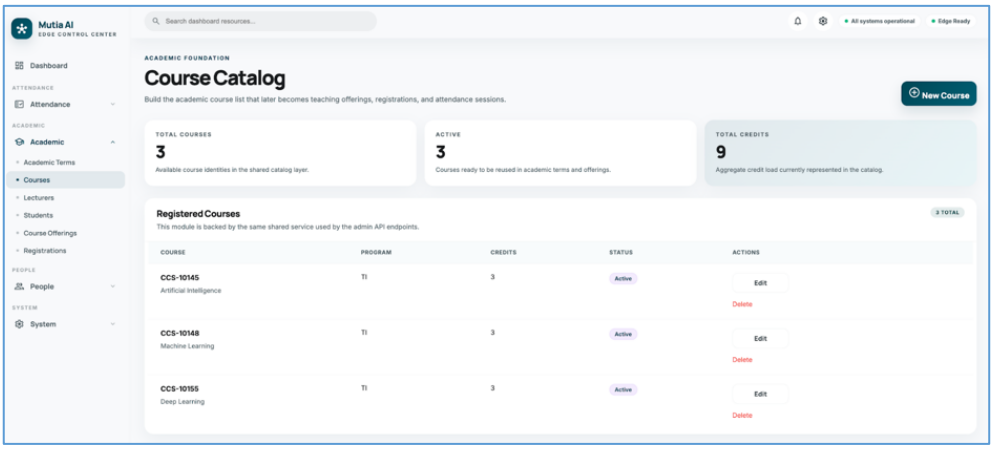

Figure 6 Course list display

 ISSN:2302-8149
e-ISSN:2540-9719

This module manages course data as part of the attendance system structure.

2) Semester Management and Course Offerings

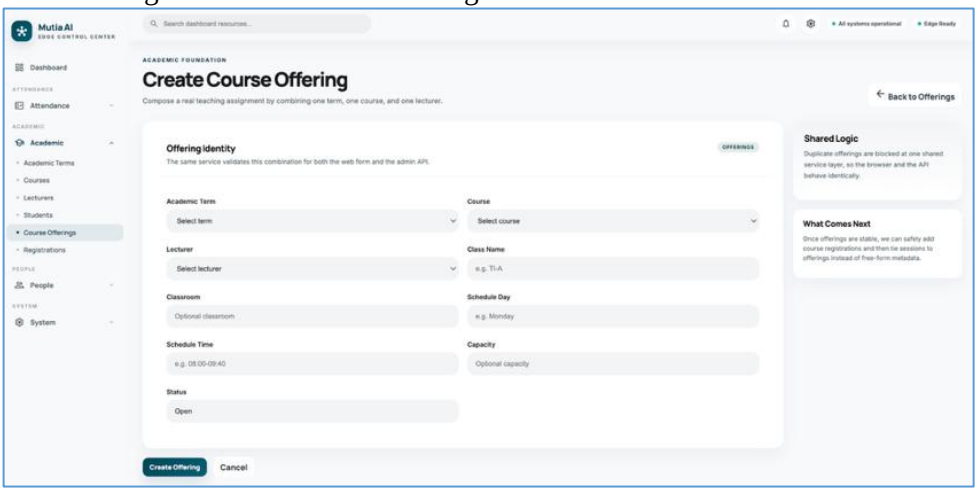

Figure 7 Course Settings for a specific semester

Used to determine the context of absences based on the academic period.

3) Class Scheduling

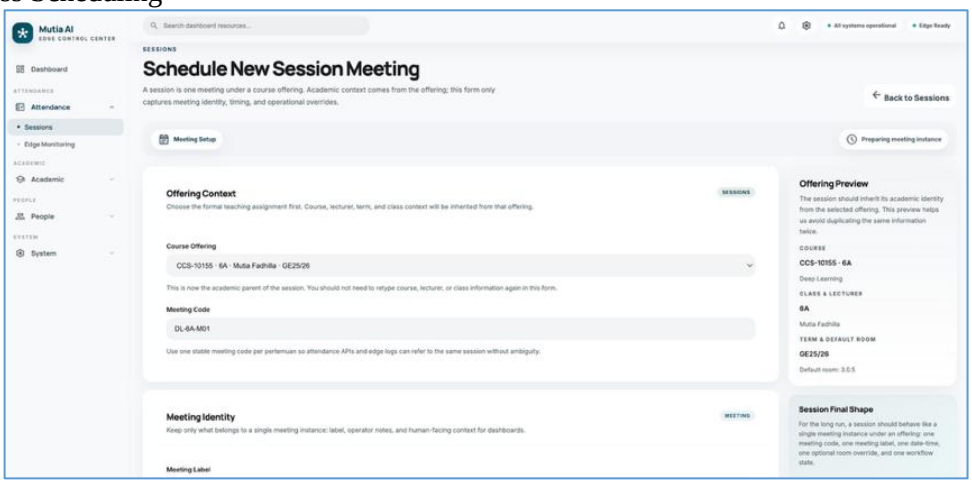

Figure 8 Additional lecture sessions

This serves as a time reference for the attendance process.

4) Face Recognition-Based Attendance

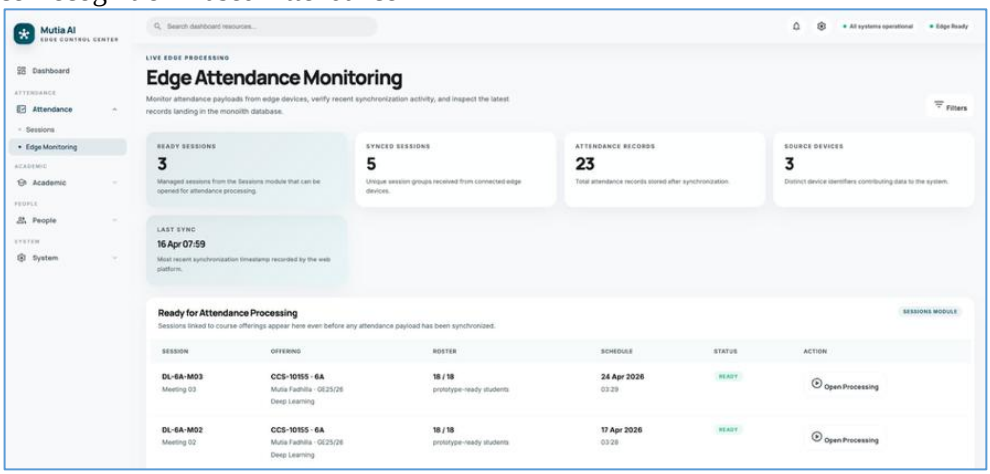

Figure 9 Lecture session attendance display

 ISSN:2302-8149
e-ISSN:2540-9719

The core feature of this system is that the user's face is captured by the camera, sent to the API for identification, and the results are synchronized to the application server via the POST /api/edge/attendance/sync endpoint. This implementation demonstrates the successful integration of the API into a web application, enabling the attendance process to be automated without manual input.

d. Testing

The testing phase was conducted to ensure the system not only functioned effectively but also met the established performance indicators. Testing was conducted in stages, including unit testing, integration testing, and functional testing. Integration testing focused on communication between the web system and the Facial Recognition API, specifically the process of sending facial image data and receiving identification results. Initial test results showed that the system was capable of automatically identifying users and recording attendance into the database based on active lecture sessions. Functional testing was conducted using black box testing methods to verify that each system feature operated according to the requirements defined in the user story. This testing included 20 test scenarios covering the authentication module, master data management, facial registration, facial recognition attendance, attendance reporting, and error handling. A summary of the test results is presented in Table 5.

Table 5 Blackbox testing

No	Test scenario	Input	Expected output	Actual output	Status
Authentication module					
TC-01	Login with valid credentials	Registered email & password	Successfully logged into the dashboard	Successfully logged into the dashboard	Valid
TC-02	Login with wrong password	Valid email, wrong password	The message "Invalid credentials" appears.	The message "Invalid credentials" appears.	Valid
...	...	...	...	...	...
Error handling module					
TC-19	Input blank form	Submit the form without filling in the fields	Validation displays error messages per field	Validation displays error messages per field	Valid
TC-20	Access to admin features by students	Login as student, access admin URL	System denies access; redirect to dashboard	System denies access; redirect to dashboard	Valid

All 20 test scenarios were declared Valid, indicating that all system modules ran according to the user story specifications and acceptance criteria established in the Planning phase.

 ISSN:2302-8149
e-ISSN:2540-9719

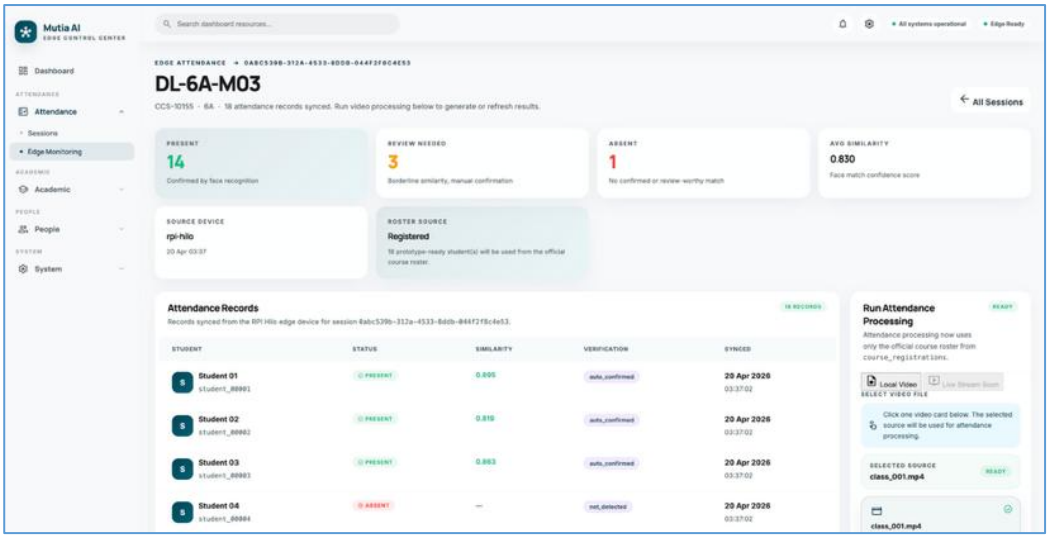

Figure 10 Student attendance recording results

To meet performance indicators, response time testing was conducted by measuring the time the system takes from image capture to recording attendance results in the database. In the context of this research, real-time is operationally defined as the total capture-to-database duration consistently within the user's tolerable range during class attendance sessions, which is under 5 seconds per identification. Testing was conducted on an edge device with the specifications described in Section 3 (Raspberry Pi 5, 8 GB RAM, Raspberry Pi OS 64-bit, Docker container runtime). Measurement results showed an average system response time of 2–4 seconds, depending on network conditions and server load, thus meeting the set response time target. Quantitative evaluation of the accuracy of the facial recognition model (Haar Cascade + FaceNet + Cosine Similarity) is part of a collaborative study reviewed separately. Therefore, the testing phase of this research focused on system functional validation and integration performance, as the primary contribution to information system development.

To place the contribution of this research in the context of similar studies, Table 6 presents a brief comparison with some previous approaches to facial recognition-based attendance systems.

Table 6 Comparison with previous studies

Aspect	Previous research		This research
	Ughade et al. [28]	Widiarsa et al. [29]	
Main focus	Local facial recognition	Web-based helpdesk system	API integrated web-based attendance system
Architecture	Local (Raspberry Pi)	Centralized web application	Three layers: camera–edge–application server
Development methods	Not specific (hardware)	Extreme Programming	Extreme Programming
API Integration	No	No	REST API with JWT authentication
Processing	Local on device	Centralized server	Edge + application server
Saving results	Local spreadsheet	Centralized database	Centralized database (real-time sync)
Performance indicators tested	Recognition accuracy	Functionality	Functionality + response time (2–4 seconds)

This comparison demonstrates that this research complements previous studies by combining the strengths of agile development (XP) methods, a three-layer edge-based architecture, and API integration that enables real-time attendance data synchronization. This is a key differentiator from previous approaches, which were generally local or web-based without edge processing.

 ISSN:2302-8149
e-ISSN:2540-9719

e. Software Increment

At this stage, the system has reached a functional prototype stage that can be used in real-world simulations. All major modules have been integrated, from user registration and academic data management to automatic attendance recording based on facial recognition. Preliminary results indicate that the system is capable of:

1) Reducing reliance on manual input
2) Increasing attendance recording efficiency
3) Providing real-time attendance data

The XP approach allows systems to be developed incrementally while still producing a usable product at each iteration. This is an advantage over traditional approaches, which tend to wait until a system is fully developed. The concrete benefit of implementing XP in this study was seen in the team's ability to validate the REST API integration as early as Sprint 1 through the SP-01 solution spike. This allowed technical integration risks to be identified and addressed earlier than in the waterfall approach, which only tests integration at the end of the development cycle.

As part of the development process evaluation, project velocity was calculated to measure team productivity during each sprint, based on the number of story points successfully completed in a single sprint. Table 7 and Figure 11 present velocity data for four system development sprints, complemented by a cumulative progress visualization (burn-up) to show the accumulation of story points completed over time.

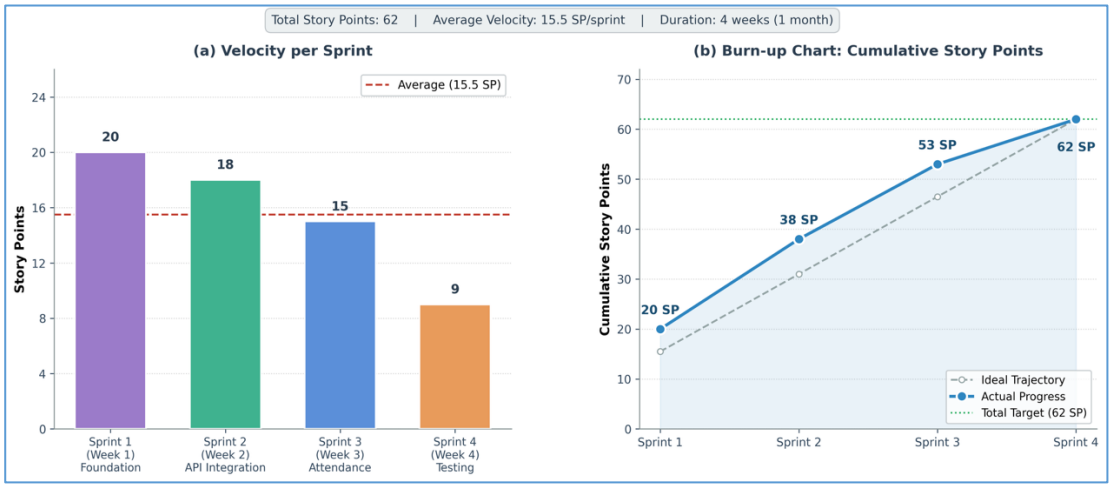

Figure 11 (a) velocity per sprint (b) burn-up chart: cumulative story points

Velocity and Burn-up Chart of System Development Across Four Sprints (a) Velocity per sprint shows the distribution of completed story points with an average of 15.5 SP/sprint; (b) The burn-up chart displays the cumulative story points consistently at or above the ideal trajectory, reaching the total target of 62 SP at the end of Sprint 4. The total scope of 62 SP was estimated based on the 12 user stories defined during the planning phase.

Table 7 Velocity

Sprint	Focus	US finished	Story points	Notes
Sprint 1	Foundation & data master	US-01, 02, 03, 04	20	Highest - many modules are worked on at once at the start
Sprint 2	Face registration & API	US-05, 06, 07	18	Spike solution shortens API integration time
Sprint 3	Attendance & reporting	US-08, 09, 10	15	Stable, end-to-end integration successfully completed
Sprint 4	Testing & finalization	US-11, 12	9	Velocity drops naturally - focus on testing and bug fixes

 ISSN:2302-8149
e-ISSN:2540-9719

Based on velocity data, the average team completed 15.5 story points per sprint, with a total of 62 story points completed across the four sprints. Velocity was highest in Sprint 1 (20 SP) as many foundation modules were worked on in parallel and some technical risks were addressed through spike solutions. Velocity in Sprint 4 decreased to 9 SP, which is understandable considering the focus of the final sprint shifted from adding new features to testing, bug fixing, and finalizing the system. This decreasing pattern is common in sprint-based XP projects and indicates a healthy transition from the development phase to the stabilization phase.

Although the system has been successfully implemented, further development is still needed, particularly in:

1). Accuracy testing based on a broader dataset (as part of a collaborative study)
2). Expanding the scope of API integration to other services
3). Quantitative analysis of system performance across network conditions
4). Improving system security and scalability

5 Conclusion

This research successfully implemented the integration of a Facial Recognition REST API into a web-based attendance information system using a three-layer architecture consisting of a camera device, a Raspberry Pi 5-based edge server, and a Laravel-based application server. The developed system is capable of real-time facial identification and automatic attendance recording, thereby reducing reliance on manual processes and minimizing the potential for errors and data manipulation. Functional testing results using black-box methods across 20 scenarios showed that all modules performed according to specifications, with a consistent system response time of 2–4 seconds, meeting the target response time.

The main contribution of this research lies in the integration of Extreme Programming methods with a three-layer REST API-based architecture for a facial recognition-based attendance system. Unlike previous approaches, which were generally localized to specific hardware or web-based without edge processing, the developed integration enables real-time attendance data synchronization between the edge device and the application server through a number of structured endpoints, as documented in Section 3. This approach results in a more modular, responsive, and scalable system for institutional contexts. The application of the Extreme Programming method also provided concrete benefits to this project through API integration validation, which could be performed as early as Sprint 1 through a spike solution. This allowed technical risks to be identified and addressed earlier than with a waterfall approach, which only tests integration at the end of the development cycle. The development process of four sprints over one month, with an average velocity of 15.5 story points per sprint, demonstrated that the iterative XP approach effectively produced a functional system in a short time.

For educational institutions, the implementation of this system has the potential to have practical impacts, including a significant reduction in the time required for manual attendance recording for each lecture session, a reduction in data entry errors, and the availability of real-time attendance reports that can be directly accessed by lecturers and students. To support this practical adoption, system deployment is recommended using an edge device with minimum specifications equivalent to a Raspberry Pi 5 (8 GB RAM) with an active cooler, a camera with a minimum resolution of 1920 x 1080 pixels, placed statically at the front of the classroom, and consistent classroom lighting. The three-layer architecture and Extreme Programming process applied in this study are structured, allowing the development process to be replicated in other institutional contexts.

This study has several important limitations to note. First, the quantitative evaluation of facial recognition model accuracy was part of a collaborative study reviewed separately, so this paper focuses on system integration performance and functional validation. Second, the test dataset used consisted of 18 students with a reference database of 6 photos per student and three class videos with an average duration of 5 minutes and 28 seconds, so the generalizability of the system performance to a larger population requires further testing. Third, structured usability testing of end users has not been conducted at this stage and is positioned as part of further development.

 ISSN:2302-8149
e-ISSN:2540-9719

References

[1] R. Wandri, S. Daulay, Y. Arta, A. Hanafiah, and J. Mardafora, "Pengenalan dan Pelatihan Algoritma Pemrograman menggunakan Aplikasi *Scratch* untuk Siswa SMK YKWI Pekanbaru," *Jurnal Pengabdian Masyarakat dan Penerapan Ilmu Pengetahuan*, Vol. 04, No. 01, 2023.

[2] R. Wandri, P. R. Setiawan, Y. Arta, and A. Hanafiah, "*Designing a Learning Game for Elementary School Students in Learning Mathematics using a Mobile Platform*," *Sistemasi: Jurnal Sistem Informasi*, Vol. 13, No. 3, pp. 1139–1146, Apr. 2024, [Online]. Available: http://sistemasi.ftik.unisi.ac.id

[3] E. Supriyati and M. A. Gustalika, "*Extreme Programming Method for Integrated Service System Website Development in Rejosari Village*," *Jurnal Riset Informatika*, Vol. 5, No. 3, pp. 263–268, 2023, DOI: 10.34288/jri.v5i3.202.

[4] A. D. Putra, F. Syakti, A. Ananda, and V. O. Armelia, "*Digitalization of Archipelago Cultural Insight Education using Extreme Programming Method*," *Journal of Information Systems and Informatics*, Vol. 6, No. 1, pp. 579–588, 2024, DOI: 10.51519/journalisi.v6i1.695.

[5] H. Wicaksono, J. Shadiq, D. I. Putri, and R. Sayekti, "*The Power of Adaptability: Achieving Agility and Quality in Web based Company Profiles Through Extreme Programming*," *Jurnal Techno Nusa Mandiri*, Vol. 20, No. 1, pp. 54–62, 2023, DOI: 10.33480/techno.v20i1.4219.

[6] P. A. Nani and N. M. R. Mamulak, "Implementasi *Extreme Programming* dalam Pengembangan Sistem Pencatatan dan Pelaporan Kasus pada Gugus Tugas Percepatan Penanganan Covid-19 Provinsi NTT," *JSII (Jurnal Sistem Informasi)*, Vol. 10, No. 1, pp. 20–24, 2023, DOI: 10.30656/jsii.v10i1.6039.

[7] S. Oktaviani, A. Priyanto, and C. Wiguna, "Implementasi *Extreme Programming* pada Sistem Informasi Program Kreativitas Mahasiswa berbasis *Web*," *Jsii (Jurnal Sistem Informasi)*, Vol. 9, No. 1, pp. 89–94, 2022, DOI: 10.30656/jsii.v9i1.3666.

[8] R. Nurfalah and A. Lattu, "Perancangan Sistem Informasi Sekolah berbasis *Website* (Study Kasus SD Negeri Cisarua)," *Jurnal Informatika Teknologi dan Sains*, Vol. 5, No. 1, pp. 54–59, 2023, DOI: 10.51401/jinteks.v5i1.2234.

[9] A. R. Isnain, M. Indigo, and A. T. Priandika, "Pemanfaatan *E-Commerce* Model *Business to Consumer* pada Putri Tapis Lampung," *Jurnal Teknoinfo*, Vol. 17, No. 1, p. 253, 2023, DOI: 10.33365/jti.v17i1.2368.

[10] N. Nopriadi, R. Harman, A. Amrizal, and R. Fauzi, "Pendekatan Sistem *Pakar Forward Chaining* dengan *Extreme Programming* pada Seleksi Karyawan PT. Enka Mandiri Sukses," *Jurnal Desain dan Analisis Teknologi*, Vol. 3, No. 1, pp. 59–67, 2024, DOI: 10.58520/jddat.v3i1.53.

[11] M. F. Anasara, "*Design of Android-based Muna Language Dictionary Application using Extreme Programming Method*," *Matics Jurnal Ilmu Komputer dan Teknologi Informasi (Journal of Computer Science and Information Technology)*, Vol. 15, No. 1, pp. 51–57, 2023, DOI: 10.18860/mat.v15i1.23151.

[12] N. S. Aldahwan and M. S. Ramzan, "*The Descriptive Data Analysis for the Adoption of Community Cloud in Saudi HEI-based Factor Adoption*," *Biomed Res. Int.*, Vol. 2022, No. 1, 2022, DOI: 10.1155/2022/7765204.

[13] B. Mkhatshwa and T. Mawela, "*Cloud Computing Adoption in the South African Public Sector*," *Indonesian Journal of Electrical Engineering and Informatics (Ijeei)*, Vol. 11, No. 2, 2023, DOI: 10.52549/ijeei.v11i2.4464.

[14] A. P. Vumo, "*A Study Into Privacy and Legal Issues in Cloud Computing: The Mozambican Context*," *International Conference on Cyber Warfare and Security*, Vol. 19, No. 1, pp. 511–517, 2024, DOI: 10.34190/iccws.19.1.2095.

[15] A. F. Ali, A. A. Hassan, H. O. Abdullahi, and R. H. Abdulah, "*Analyzing the Factors Influencing the Adoption of Cloud Computing by SMEs using the SEM Approach*," *International Journal of Advanced and Applied Sciences*, Vol. 10, No. 7, pp. 66–79, 2023, DOI: 10.21833/ijaas.2023.07.009.

[16] X. Jiang, "*Energy Consumption Optimization of Edge Computing based on Reinforcement Learning*," *J. Phys. Conf. Ser.*, Vol. 2246, No. 1, p. 012076, 2022, DOI: 10.1088/1742-6596/2246/1/012076.

[17] Z. Zhang, W. Sun, and Y. Yu, "*Research on Intelligent Scheduling Mechanism in Edge Network for Industrial Internet of Things*," *Security and Communication Networks*, Vol. 2022, pp. 1–14, 2022, DOI: 10.1155/2022/5358873.

[18] Y. Xiao, L. Wei, J. Feng, and E. Wang, "*Two-Tier End-Edge Collaborative Computation Offloading for Edge Computing*," *Journal of Computational Methods in Sciences and Engineering*, Vol. 22, No. 2, pp. 677–688, 2022, DOI: 10.3233/jcm-215923.

[19] Y. Zhang, H. Yu, W. Zhou, and M. Man, "*Application and Research of IoT Architecture for End-Net-Cloud Edge Computing*," *Electronics (Basel).*, Vol. 12, No. 1, p. 1, 2022, DOI: 10.3390/electronics12010001.

[20] L. A. Haibeh, M. C. E. Yagoub, and A. Jarray, "*A Survey on Mobile Edge Computing Infrastructure: Design, Resource Management, and Optimization Approaches*," *IEEE Access*, Vol. 10, pp. 27591–27610, 2022, DOI: 10.1109/access.2022.3152787.

[21] W. Jin, S. Lim, Y. Suh, C. Park, and D. Kim, "*Transparent Access to Heterogeneous IoT based on Virtual Resources*," *Computers Materials & Continua*, Vol. 74, No. 3, pp. 4983–4999, 2023, DOI: 10.32604/cmc.2023.020851.

[22] L. Ye, J. Zhong, and X. Chen, "*Application of Edge Computing in Smart Water*," 2023, DOI: 10.3233/faia230052.

[23] F. Sepulveda, J. S. Thangraj, and J. Pulliam, "*The Edge of Exploration: An Edge Storage and Computing Framework for Ambient Noise Seismic Interferometry using Internet of Things based Sensor Networks*," *Sensors*, Vol. 22, No. 10, p. 3615, 2022, DOI: 10.3390/s22103615.

[24] Y. Liang and T. Li, "*Ubiquitous Power Internet of Things-Oriented Low-Latency Edge Task Scheduling Optimization Strategy*," *Front. Energy Res.*, Vol. 10, 2022, DOI: 10.3389/fenrg.2022.947298.

[25] S. Das, "*A Comparative Analysis of Cloud Computing and Fog Computing: Advancing Towards Edge Intelligence*," *International Journal for Multidisciplinary Research*, Vol. 6, No. 1, Jan. 2024, DOI: 10.36948/ijfmr.2024.v06i01.12569.

[26] B. Tank and V. Gandhi, "*A Comparative Study on Cloud Computing, Edge Computing and Fog Computing*," 2023, DOI: 10.3233/atde221329.

[27] A. Agustiyar, R. R. Isnanto, and C. E. Widodo, "*Face Recognition for Attendance Systems: A Bibliometric Review of Research Trends and Opportunities*," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, Vol. 15, No. 01, pp. 8–13, Dec. 2025, DOI: 10.32736/sisfokom.v15i01.2529.

[28] U. R. Ughade, S. M. Gikwad, A. N. Yeole, and Dr. A. O. Mulani, "*Automatic Attendance System using Face Recognition*," *Journal of Image Processing and Intelligent Remote Sensing*, No. 34, pp. 11–18, Jul. 2023, DOI: 10.55529/jipirs.34.11.18.

[29] B. L. Widiarsa, H. Hendri, B. O. Lubis, B. Santoso, R. T. Yunandar, and A. M. B. Aji, "Penerapan Metode XP (*Extreme Programming*) pada Sistem Informasi *Help Des*k Pengajuan Gambar berbasis *Website*," *Jurnal Teknologi Sistem Informasi*, Vol. 6, No. 1, pp. 21–35, Apr. 2025, DOI: 10.35957/jtsi.v6i1.9267.

[30] C. C. Mutia, "Sistem Informasi Pengelolaan Dokumen Proses Pembelajaran (SIPDOPP) Program Studi Teknologi Informasi," *Jurnal Teknologi Informasi*, Vol. 2, No. 1, p. 19, 2023, DOI: 10.35308/jti.v2i1.7546.

[31] H. Priambodo and A. Muhajirin, "Perancangan *ChatBot* Pendaftaran Siswa dengan Telegram *BOT Design a Chatbot for Student Registration using Telegram BOT*," *Journal of Informatics and Information Security*, Vol. 3, No. 1, pp. 73–88, 2022, DOI: 10.31599/jiforty.v3i1.1332.