International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

AI-Based Predective Self-Adaptive Optimization System

Mr. Manas Kumar Majhi
Amity School of Engineering & Technology
Amity University Chhattisgarh
Raipur, Chhattisgarh

Dr. Poonam Mishra (Mentor)
Department of Computer Science and Technology,
Amity School of Engineering & Technology
Amity University Chhattisgarh
Raipur, Chhattisgarh

Ms. Neha
Amity School of Engineering & Technology
Amity University Chhattisgarh
Raipur, Chhattisgarh

Md. Nayab Hussain
Amity School of Engineering & Technology
Amity University Chhattisgarh
Raipur, Chhattisgarh

Abstract - Modern computer systems frequently experience performance degradation, excessive battery consumption, overheating, and inefficient background resource utilization due to continuously changing user workloads and static optimization mechanisms. Conventional system monitoring applications are limited to displaying hardware statistics and fail to provide intelligent decision-making or autonomous adaptation. To address this limitation, this research proposes an **AI Driven Self-Adaptive System**, an intelligent desktop-based optimization framework capable of continuously monitoring critical system parameters and dynamically adjusting system behavior according to real-time conditions.

The proposed system collects live operational data such as CPU utilization, RAM consumption, battery level, temperature variation, running process load, and user activity patterns. This monitored information is stored as a historical dataset and processed using Machine Learning techniques to identify usage trends, forecast battery behavior, and detect abnormal system conditions. In addition, anomaly detection algorithms are employed to recognize sudden spikes, overheating events, and unusual background process activities. To enhance autonomous decision-making, a Reinforcement Learning based adaptive engine is integrated, enabling the software to learn from previous

optimization outcomes and automatically switch between Performance Mode, Power Saving Mode, and Focus Mode for maximum efficiency.
The system further includes an intelligent process recommendation mechanism to identify unnecessary high-resource applications and a cloud-enabled monitoring dashboard that allows remote observation through mobile devices. By combining predictive analytics, self-learning adaptation, and real-time automation, the proposed framework reduces manual system management efforts while improving performance stability, energy efficiency, and personalized user experience. The developed model demonstrates how AI can transform conventional monitoring software into a proactive and self-optimizing computing assistant.

Keywords - Artificial Intelligence, Self-Adaptive System, Machine Learning, Reinforcement Learning, System Resource Monitoring, Predictive Battery Forecasting, Anomaly Detection, Intelligent Process Optimization, Autonomous Decision Making, Cloud Monitoring.

1. INTRODUCTION

In the modern computing environment, desktop and laptop systems are required to perform a wide variety of operations such as software execution, multitasking, media processing, online communication, and data handling while maintaining high speed and stability. As the complexity of user workload increases, computer systems often experience performance degradation, high CPU consumption, memory congestion, overheating, rapid battery discharge, and excessive background process activity. These issues negatively influence both the efficiency of the machine and the productivity of the user.

Conventional monitoring utilities are commonly used to observe system parameters such as CPU usage, RAM usage, battery level, and running applications. However, such tools are mainly limited to static visualization and manual inspection. They only display numerical values or graphical

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

reports without providing any intelligent interpretation, predictive analysis, or autonomous optimization support. Therefore, users are still forced to manually identify resource-intensive applications, switch operating modes, or terminate unnecessary tasks in order to maintain system performance. Recent studies have highlighted that modern adaptive software systems require a transition from passive monitoring to continuous self-learning and automatic decision-making frameworks [1], [2].

The development of self-adaptive AI systems has seen significant strides in recent years, especially within the domains of autonomous decision-making and dynamic environmental interactions. These systems are designed to continuously learn from their surroundings, adapt to changes in real-time, and autonomously make decisions, which is essential for complex and dynamic environments. Self-adaptation is particularly relevant in fields such as autonomous vehicles, robotics, and smart grids, where the environment is constantly evolving and often unpredictable. The integration of AI systems capable of self-adaptation has been facilitated by advances in machine learning, neural networks, and reinforcement learning (RL), enabling systems to improve their decision-making capabilities based on accumulated experience. The emergence of self-adaptive artificial intelligence (AI) systems has revolutionized the landscape of autonomous decision-making across dynamic environments. As technological advancements continue to accelerate, the demand for AI systems capable of making real-time, autonomous decisions in highly unpredictable and complex settings has surged. With the rapid growth of Artificial Intelligence (AI) and Machine Learning (ML), the concept of self-adaptive computing has gained significant attention in current research. AI-based adaptive systems have the capability to continuously collect runtime information, analyze behavioral patterns, and make intelligent decisions based on changing environmental conditions. Machine learning algorithms can identify hidden usage trends and forecast future resource behavior, while anomaly detection models can recognize unusual spikes, overheating conditions, or suspicious process activity before severe system failure occurs [3], [4]. This makes intelligent automation far more efficient than traditional rule-based monitoring.

In addition to predictive learning, Reinforcement Learning (RL) has emerged as an effective approach for dynamic software adaptation because it allows a system to learn the most beneficial optimization action through reward-based experience. Instead of relying on fixed predefined rules, RL-based adaptive software can gradually improve its future decisions by observing previous performance outcomes. Several recent adaptive computing studies have shown that reinforcement learning significantly enhances autonomous mode selection, anomaly response, and runtime efficiency in intelligent monitoring systems [5], [6].

Motivated by these technological advancements, this research proposes an **AI Driven Self-Adaptive System**, an intelligent desktop-based software platform designed to transform ordinary monitoring tools into a proactive optimization assistant. The proposed framework continuously monitors critical system resources including CPU utilization, RAM consumption, battery percentage, temperature status, running processes, and user activity behavior. The collected data is stored as a historical dataset and further processed using machine learning models to perform battery prediction, anomaly detection, and usage pattern analysis.

To provide autonomous optimization, the proposed system incorporates a Reinforcement Learning based decision engine that dynamically switches among Performance Mode, Power Saving Mode, and Focus Mode according to the current resource condition and previously obtained optimization rewards. Furthermore, the framework also provides smart process recommendations to identify unnecessary high-resource applications and integrates a cloud-enabled mobile dashboard for remote monitoring access. The core principle behind self-adaptive AI is its ability to modify its decision-making processes dynamically, based on real-time feedback from its environment. This capability is underpinned by advanced machine learning techniques such as reinforcement learning (RL), neural networks, and evolutionary algorithms, which enable systems to learn from past experiences and adapt their behaviors accordingly. By combining predictive analytics, adaptive anomaly awareness, self-learning decision making, and real-time automation, the developed model aims to reduce manual system management effort and create a more efficient, intelligent, and personalized computing environment.

Therefore, the objective of this work is to develop a unified AI-powered self-optimizing desktop assistant capable of improving system performance, conserving battery power, detecting abnormal behavior, and adapting itself continuously according to user workload requirements. The proposed research contributes to the growing field of intelligent self-management systems where software is expected not only to monitor but also to think, learn, and react autonomously [1], [5].

2. LITERATURE REVIEW

The concept of self-adaptive software systems has attracted significant attention in recent years due to the growing demand for intelligent applications that can automatically respond to changing runtime conditions. Early research in this field primarily focused on the theoretical foundation of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

autonomic and self-managing software, where systems were expected to monitor their environment, analyze internal states, and execute corrective actions without continuous human involvement [9], [10]. These studies established the idea that future software should not remain static but must evolve dynamically according to operational requirements.

As research progressed, several scholars emphasized that traditional software monitoring architectures are no longer sufficient for modern computing workloads. Salehie and Tahvildari [21] explained that static software systems often fail when environmental conditions become highly dynamic because they lack runtime learning and adaptation capabilities. Similarly, Gomaa [20] discussed the importance of feedback control loops in adaptive software engineering, where continuous sensing, analysis, planning, and execution are required to maintain system stability. Arcaini et al. [22] further strengthened this perspective by showing that MAPE-K based adaptive loops provide an effective structural backbone for intelligent self-management systems.

The integration of Machine Learning into self-adaptive systems has recently become a major research direction. Gheibi et al. [1] conducted a systematic review and concluded that machine learning enables adaptive software to move beyond predefined rules by learning hidden runtime patterns from collected data. According to Pahl and Azimi [12], data-driven software equipped with learning mechanisms can significantly improve predictive decision making compared to conventional monitoring-only applications. This indicates that ML models are increasingly becoming central to intelligent resource management systems.

Several works have specifically investigated predictive analytics for system resource behavior. Singh and Kumar [27] demonstrated that battery consumption trends can be effectively estimated using historical performance parameters, enabling early power-saving actions before critical battery depletion occurs. In a similar direction, Hwang and Kim [28] proposed a machine learning based CPU and memory load prediction mechanism for smart computing systems, proving that future workload forecasting can reduce sudden performance bottlenecks. Brownlee [31] also highlighted that predictive modeling techniques are particularly suitable for environments where system parameters change continuously over time.

Another critical area explored by researchers is anomaly detection in runtime computing environments. Islam et al. [29] applied Isolation Forest based anomaly detection to identify abnormal CPU and RAM utilization patterns and reported improved early fault awareness. Lavin and Ahmad [30] compared several streaming anomaly detection methods and found that real-time abnormality identification is highly effective in preventing delayed manual troubleshooting. Additionally, HaddadPajouh et al. [26] showed that deep behavioral analysis can detect suspicious hidden activities that are often missed by standard process monitoring tools. These findings suggest that anomaly-aware systems provide a stronger preventive layer than simple resource visualization.

Reinforcement Learning has emerged as one of the most promising approaches for autonomous runtime adaptation because it allows software to learn optimization decisions through experience rather than depending entirely on hard-coded thresholds. Metzger et al. [2] explained that online reinforcement learning helps self-adaptive systems select runtime actions according to changing environmental states. Zhang et al. [3] further proposed a meta reinforcement learning model that improves adaptive decision flexibility in uncertain software conditions. In addition, Shevtsov et al. [17] and Feit et al. [18] demonstrated that reinforcement learning significantly improves trustable autonomous decision making when systems must repeatedly choose among multiple optimization alternatives.

The explainability and practical robustness of reinforcement learning based adaptive software have also been studied in recent years. Feit et al. [4] reported that self-adaptive systems equipped with explainable online RL are more reliable because the software can justify why a particular adaptation action was selected. Alhijawi et al. [8] introduced a continual reinforcement learning expert system for self-optimization in real-time software and found that continuous reward feedback improves long-term adaptation accuracy. These works confirm that RL can serve as a practical engine for dynamic mode switching and self-corrective optimization.

Cloud-oriented and distributed resource management studies further contribute to the development of intelligent monitoring frameworks. Chen and Bahsoon [23] proposed online quality-of-service modeling for cloud software services, where adaptive decisions are made continuously according to runtime demand. Zhou et al. [14] reviewed deep reinforcement learning based resource scheduling approaches in cloud environments and concluded that intelligent scheduling provides better efficiency than static allocation policies. Such research supports the inclusion of remote monitoring and adaptive cloud synchronization in modern self-management systems.

From the implementation perspective, multiple researchers have also emphasized the importance of combining machine learning frameworks with practical software development platforms. Raschka and Mirjalili [33], Géron [35], and Chollet [32] provided strong evidence that Python-based machine learning libraries enable scalable predictive analytics, anomaly modeling, and adaptive automation within

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

desktop applications. Similarly, Grinberg [36] discussed that lightweight web frameworks such as Flask can effectively extend desktop intelligence platforms into cloud-accessible monitoring dashboards.

Although the reviewed literature provides substantial progress in self-adaptive software, most existing studies focus on isolated components such as only anomaly detection, only resource forecasting, or only reinforcement learning adaptation. Very limited work attempts to integrate predictive battery analysis, anomaly identification, smart process recommendation, autonomous mode switching, and cloud-based monitoring into one unified desktop-oriented platform. Therefore, there remains a clear research gap in developing a comprehensive AI-powered self-adaptive desktop assistant capable of simultaneously learning, predicting, optimizing, and remotely monitoring system behavior. This gap forms the primary motivation behind the proposed AI Driven Self-Adaptive System.

3. METHODOLOGY

- Proposed System Architecture
- Real-Time Resource Monitoring
- Historical Dataset Preparation
- Predictive Battery Forecasting
- Anomaly Detection
- Smart Process Recommendation
- Reinforcement Learning Adaptive Switching
- Usage Pattern Prediction
- Desktop Dashboard UI
- Cloud Monitoring Integration
- System Workflow Algorithm
- Performance Evaluation

3.1 Proposed System Architecture

The proposed AI Driven Self-Adaptive System is designed as an integrated multi-layer desktop intelligence framework that continuously monitors system behavior, analyzes collected data, generates adaptive decisions, and provides remote accessibility. Unlike conventional monitoring tools that only display resource statistics, the proposed architecture follows a self-learning feedback mechanism where runtime information is continuously sensed, processed, and converted into optimization actions [2], [9]. The complete architecture is composed of six major functional layers, as illustrated in Figure 1.

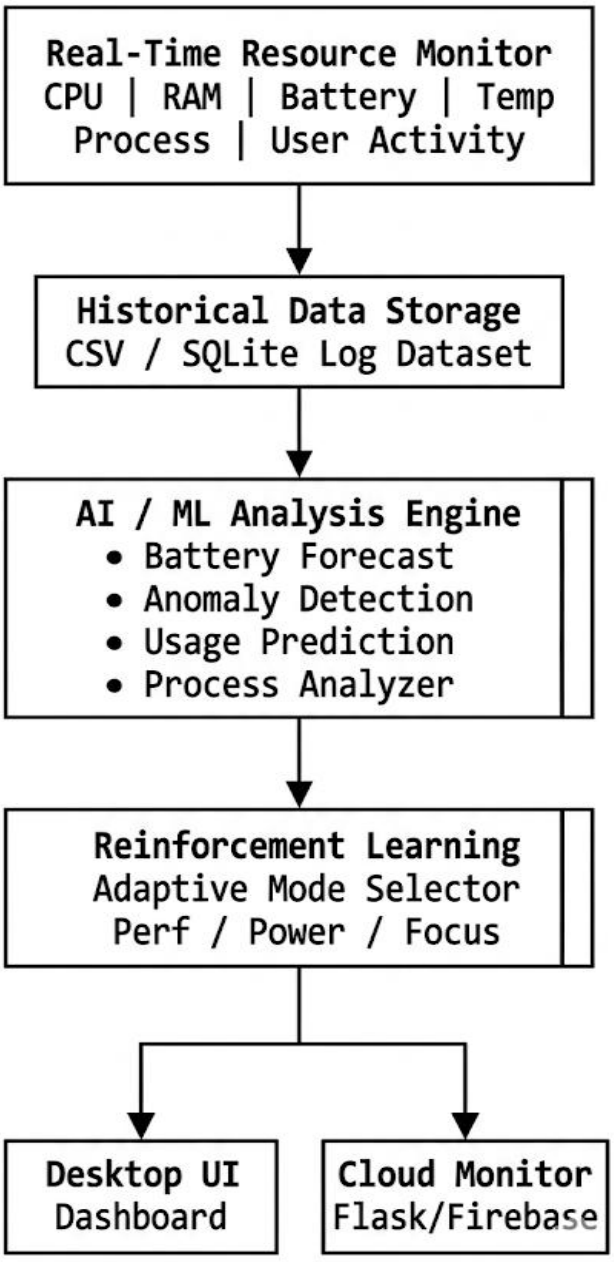

Figure 1: Proposed system architecture

Architecture Description

1. Real-Time Resource Monitoring Layer

This layer acts as the sensing component of the system. It continuously captures live desktop parameters such as CPU utilization, RAM usage, battery percentage, temperature status, active background processes, and user activity. Continuous monitoring is essential in self-adaptive software because all intelligent decisions depend on accurate runtime observations [10], [21].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

2. Historical Data Storage Layer

All monitored values are stored in a timestamp-based dataset using CSV or SQLite logging. This layer maintains historical system behavior records which are later used to train machine learning models. Previous adaptive computing studies have shown that long-term data storage improves prediction quality and autonomous decision consistency [1], [22].

3. AI / Machine Learning Analysis Engine

The stored data is processed through an internal AI engine consisting of four analytical modules:

i. **Predictive Battery Forecasting** – estimates future battery drain trend.

ii. **Anomaly Detection** – identifies overheating, CPU spikes, abnormal RAM, or suspicious activity.

iii. **Usage Pattern Prediction** – studies previous user workload behavior.

iv. **Smart Process Analyzer** – detects unnecessary high-resource applications.

This analytical division helps the software move from simple monitoring toward intelligent understanding of system conditions [14], [23].

4. Reinforcement Learning Decision Engine

The outputs of the AI engine are sent to a Q-learning based adaptive controller. This module automatically selects the most suitable optimization state:

i. Performance Mode

ii. Power Saving Mode

iii. Focus Mode

The RL engine learns from previous rewards and continuously improves future optimization choices, making the software more autonomous than fixed threshold systems [2], [17], [18].

5. Desktop Dashboard Visualization Layer

After adaptive processing, all outputs are displayed on a dark-theme desktop dashboard including:

i. live CPU/RAM graphs

ii. battery prediction

iii. anomaly alerts

iv. current active mode

v. smart recommendations

This keeps the user visually informed about system intelligence activities.

6. Cloud and Mobile Monitoring Layer

A cloud synchronization module using Flask and Firebase sends important runtime information to a remote mobile dashboard. Through this, users can monitor:

i. CPU status

ii. RAM condition

iii. battery level

iv. anomaly alerts

v. active optimization mode

from any mobile browser in real time.

Overall, the proposed architecture creates a closed-loop adaptive desktop ecosystem where monitoring, data storage, machine learning analysis, reinforcement learning optimization, user visualization, and cloud connectivity work together continuously. Such modular intelligent architectures are considered highly effective for modern self-managing software platforms because they provide both autonomous adaptation and user transparency [3], [8].

3.2 Real-Time Resource Monitoring Module

The Real-Time Resource Monitoring Module serves as the primary sensing unit of the proposed AI Driven Self-Adaptive System. The main responsibility of this module is to continuously observe the live operational state of the computer and provide updated runtime information to the intelligent analysis engine. Since all adaptive decisions depend on current system behavior, accurate and uninterrupted monitoring becomes the foundation of the entire framework [10], [21].

Unlike traditional static monitoring utilities that only display occasional system statistics, the proposed module performs continuous background scanning at fixed time intervals. During each monitoring cycle, the software captures multiple critical resource parameters that directly influence system performance, battery efficiency, and user experience. The monitored parameters include:

i. CPU utilization percentage

ii. RAM consumption level

iii. Battery charging/discharging percentage

iv. System temperature

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

v. Number of active running processes

vi. High-resource background applications

vii. User activity state (active/idle)

These parameters are selected because they represent the most significant indicators of desktop workload intensity and hardware stress. Previous adaptive computing studies indicate that multi-parameter runtime sensing provides more reliable decision input than single-resource observation models [1], [22].

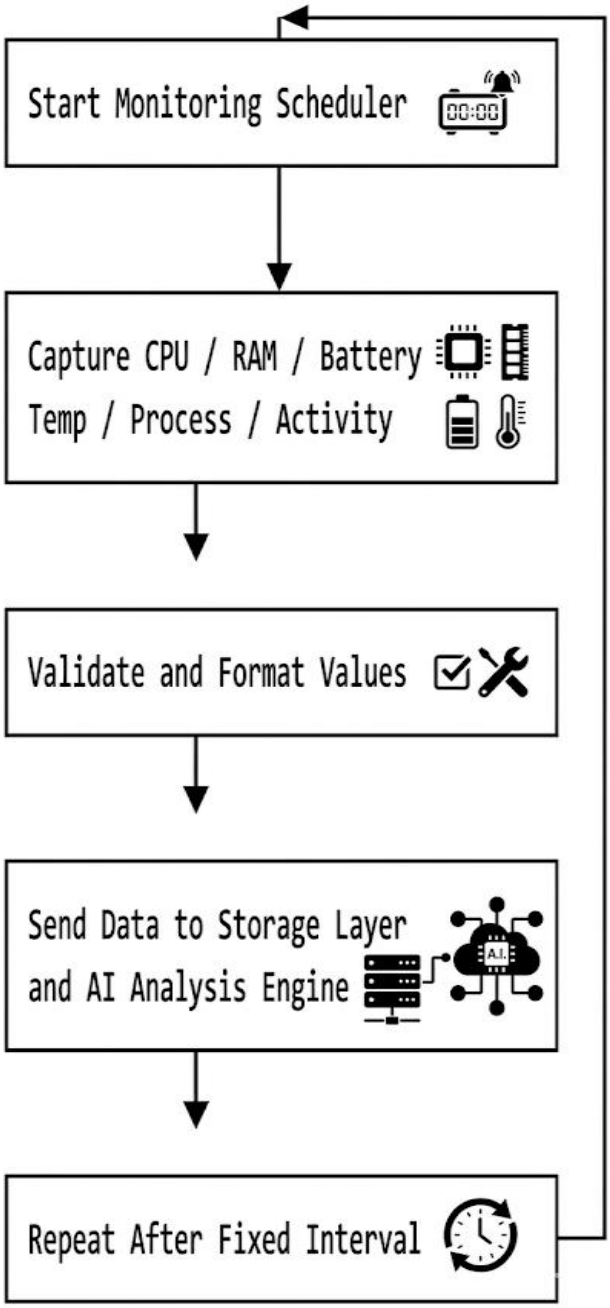

Figure 2: Workflow of Real-Time Resource Monitoring

Module Working Process

The monitoring module follows an automated cyclic execution model. Initially, a scheduler starts the monitoring service immediately after the desktop application is launched. At every predefined interval, the software fetches fresh hardware and process information from the operating system using Python-based monitoring libraries. CPU and RAM percentages provide a direct measure of processing load, while battery and temperature values help identify power drain and overheating conditions. Simultaneously, process scanning identifies applications consuming unusually high resources, and user activity tracking determines whether the machine is actively used or remains idle.

After collection, all monitored values are validated and converted into a structured numerical format so that they can be easily stored and processed by machine learning algorithms. Clean runtime data formatting is important because inconsistent sensor readings can reduce the prediction accuracy of the adaptive engine [12], [31]. Once formatted, the monitoring results are immediately transferred to two destinations:

- The Historical Data Storage Module for long-term dataset creation, and
- The AI Analysis Layer for instant prediction and anomaly evaluation.

Thus, the same monitoring stream supports both current decision making and future learning.

Technical Implementation Basis

The practical implementation of this module is based on lightweight Python system monitoring libraries such as:

- psutil for CPU, RAM, battery, and process statistics
- WMI / OS sensor APIs for temperature readings
- Time scheduler functions for interval-based repeated scanning

These tools enable low-overhead data acquisition without significantly affecting system performance. Lightweight monitoring is necessary because the software itself should not become a burden on system resources while trying to optimize them [23], [33].

Overall, the Real-Time Resource Monitoring Module acts as the continuous observation backbone of the proposed system. It ensures that the software always remains aware of the current machine condition and continuously feeds reliable runtime information into the higher intelligent layers. Without this uninterrupted sensing mechanism, predictive analysis, anomaly detection, and reinforcement learning adaptation

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

would not be possible. Therefore, this module establishes the first and most essential step toward building a fully autonomous self-adaptive desktop environment [2], [8].

3.3 Historical Data Collection and Dataset Preparation

The continuous runtime information collected from the monitoring module is not only used for immediate observation but is also stored for future intelligent analysis. For this purpose, the proposed system includes a dedicated Historical Data Collection and Dataset Preparation Module that records every monitored system reading in a structured format. This stored information acts as the learning foundation for machine learning prediction, anomaly detection, and reinforcement learning based adaptive decisions [1], [22].

Each monitoring cycle generates a new record containing the current system state. The major attributes stored in the dataset are:

- Timestamp
- CPU utilization
- RAM usage
- Battery percentage
- Temperature value
- Running process count
- High-resource application load
- User activity status

These values are automatically logged at fixed intervals and stored in CSV or SQLite format. Maintaining timestamp-based historical records allows the software to observe how system behavior changes over time under different user workloads. Previous studies have reported that adaptive systems with long-term runtime datasets produce more stable and accurate learning outputs compared to systems that rely only on current sensor readings [12], [23].

Before sending this information to the AI engine, the raw collected data undergoes a simple preparation process. In this step:

- missing values are removed,
- abnormal null readings are filtered,
- numerical parameters are normalized, and
- categorical activity labels are converted into machine-readable form.

This preprocessing is necessary because clean and consistent data improves the efficiency of predictive machine learning models and reduces false anomaly detection [31], [33].

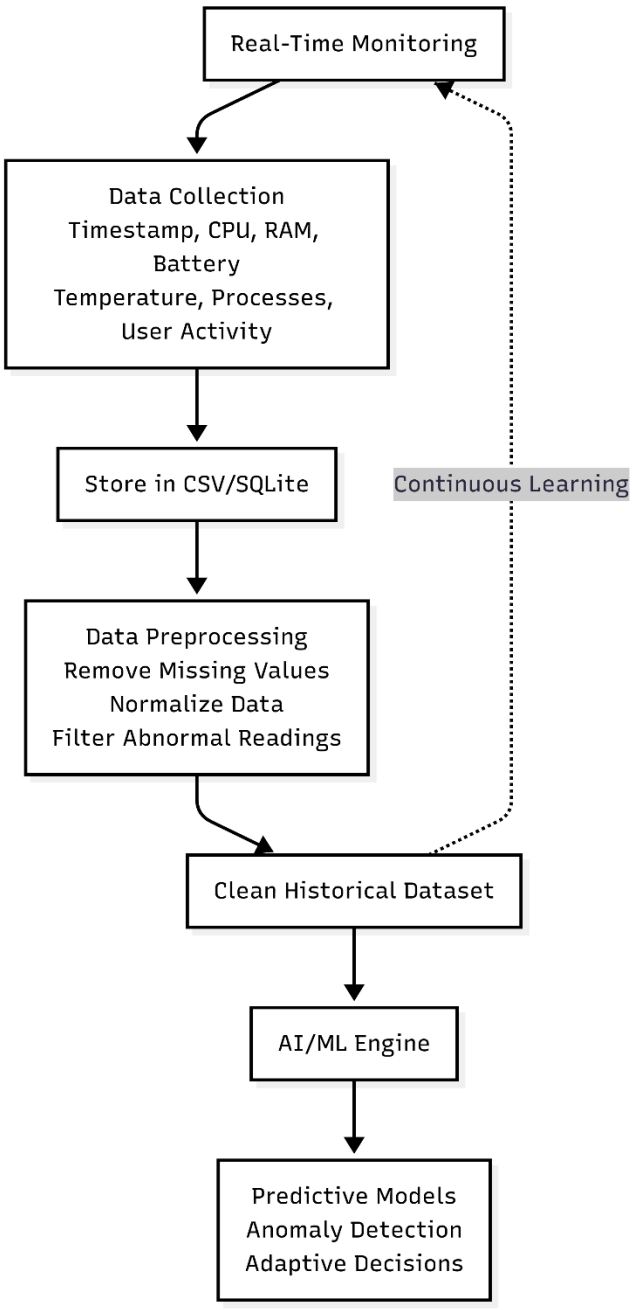

Figure 3. Historical Data Preparation Flow

Thus, this module creates a continuously expanding historical knowledge base that enables the proposed system to move from simple real-time monitoring toward data-driven intelligent adaptation. By preserving past behavior patterns, the software gains the ability to learn, compare, and make more informed optimization decisions in future runtime conditions [2], [17].

3.4 Machine Learning Based Predictive Battery Forecasting

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Battery efficiency is one of the most important concerns in modern desktop and laptop systems, especially when users perform continuous multitasking or high-resource activities. In many cases, battery drain is noticed only after the power level becomes critically low, which reduces user productivity and may interrupt ongoing work. To overcome this limitation, the proposed system includes a Machine Learning Based Predictive Battery Forecasting Module that estimates future battery status before severe discharge occurs [27], [31].

This module uses the historical dataset generated from the monitoring layer and analyzes several battery-related runtime parameters such as:

- current battery percentage,
- CPU utilization,
- charging or discharging state,
- running heavy process load, and
- previous battery drain trend.

By studying the relationship among these variables, the machine learning model predicts the approximate battery level after a future interval such as 30 minutes or 1 hour. Such prediction allows the system to understand whether the current workload may lead to rapid battery loss.

A lightweight regression-based predictive model is used for this purpose because regression algorithms are effective in estimating continuous numerical outputs from previous behavioral records. Similar studies have shown that battery forecasting through machine learning significantly improves early energy-saving decisions compared to manual observation methods [27], [35].

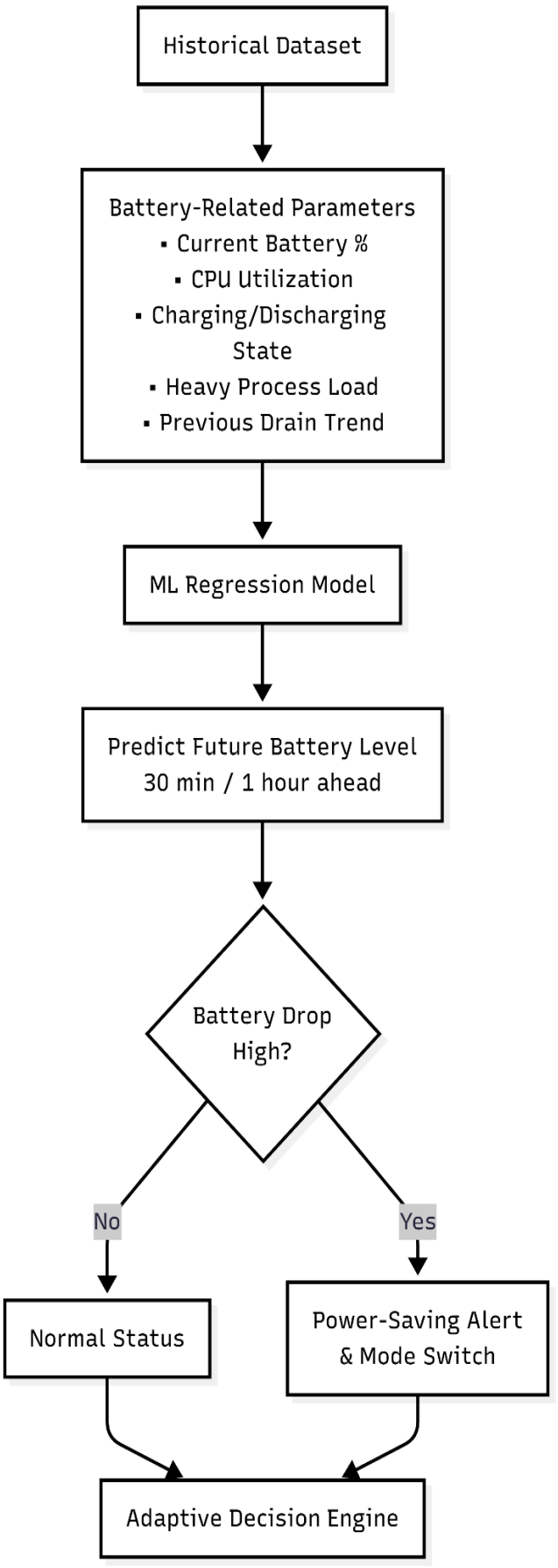

Figure 4. Battery Prediction Workflow

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Once the future battery percentage is estimated, the result is passed to the adaptive decision engine. If the predicted battery drop is unusually high, the system can proactively recommend power-saving actions or prepare for automatic mode switching. Therefore, this module makes the proposed software predictive rather than reactive, as it enables the computer to respond before battery problems become critical [2], [23].

3.5 Anomaly Detection Mechanism

In desktop computing systems, abnormal conditions such as sudden CPU spikes, unusual RAM consumption, excessive heating, rapid battery drain, or suspicious background processes can reduce performance and may lead to system instability. Conventional monitoring software usually displays these changes only as raw numbers, leaving the user responsible for manually identifying the problem. To provide early automated fault awareness, the proposed framework includes an Anomaly Detection Mechanism that continuously checks whether the current system behavior deviates from learned normal patterns [29], [30].

This module receives live and historical monitoring values from the dataset and compares the current resource state with previously observed regular usage behavior. The major abnormal situations identified by the system include:

- unusually high CPU usage,
- excessive RAM occupancy,
- overheating condition,
- abnormal battery drainage, and
- hidden high-resource process activity.

Instead of using only fixed threshold values, the system applies a lightweight machine learning anomaly model that can recognize irregular combinations of resource values. This makes the detection process more flexible because some abnormal conditions may not be visible through single-parameter checking alone. Previous studies have shown that machine learning based anomaly detection provides faster and more reliable identification of runtime irregularities than manual monitoring methods [26], [29].

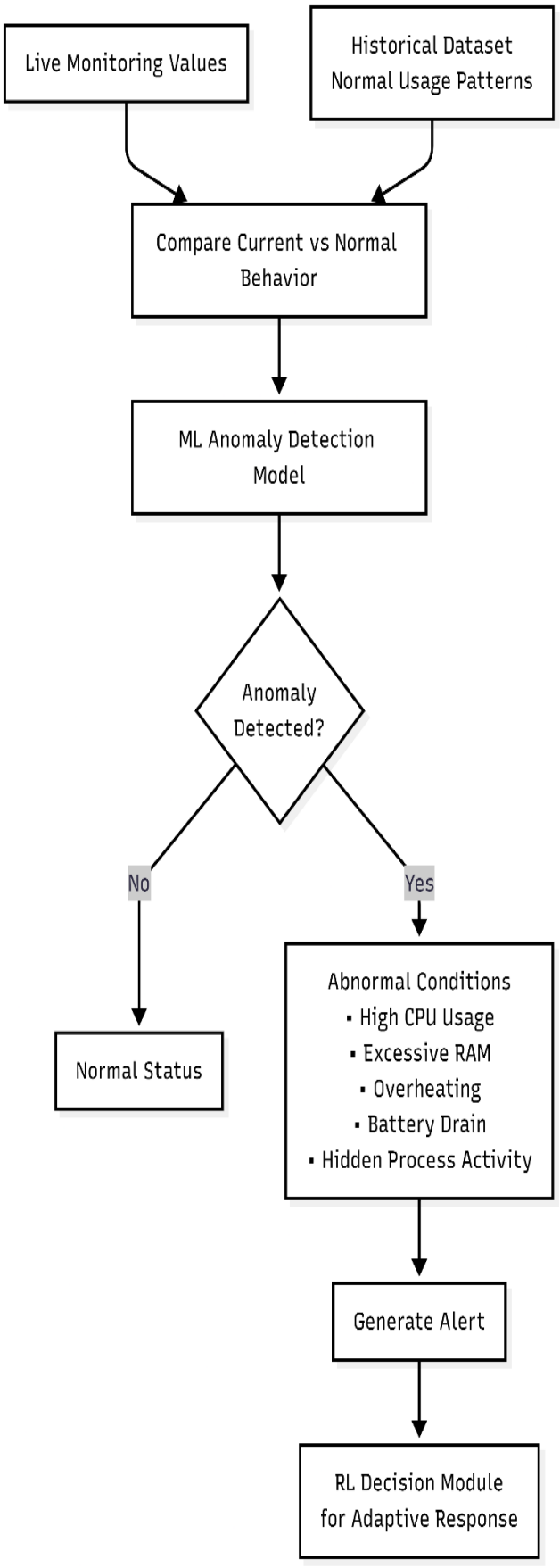

Figure 5. Anomaly Detection Flow

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Whenever an abnormal condition is detected, the software immediately generates an alert and forwards this information to the reinforcement learning decision module for possible optimization action. In this way, the system does not simply inform the user about the issue but also prepares the higher intelligent layers for adaptive response. Therefore, the anomaly detection mechanism adds a preventive safety layer to the proposed self-adaptive desktop environment by identifying performance threats before they become severe [4], [17].

3.6 Smart Process Recommendation Engine

One of the common reasons behind desktop slowdowns is the presence of unnecessary background applications that silently consume high CPU power, excessive memory, and battery resources. In normal cases, users are unable to identify which running process is actually responsible for the performance drop because traditional task managers display only technical process names without any intelligent recommendation. To solve this issue, the proposed system includes a Smart Process Recommendation Engine that automatically scans active applications and identifies resource-intensive tasks requiring user attention [12], [23].

This module continuously receives live process information from the monitoring layer and evaluates each running application on the basis of:

- CPU consumption,
- RAM usage,
- execution duration,
- background priority, and
- battery impact.

After comparing these values, the engine detects applications that are consuming unusually high resources for a long period or are not essential to the current user activity. Such applications are marked as heavy or unnecessary processes.

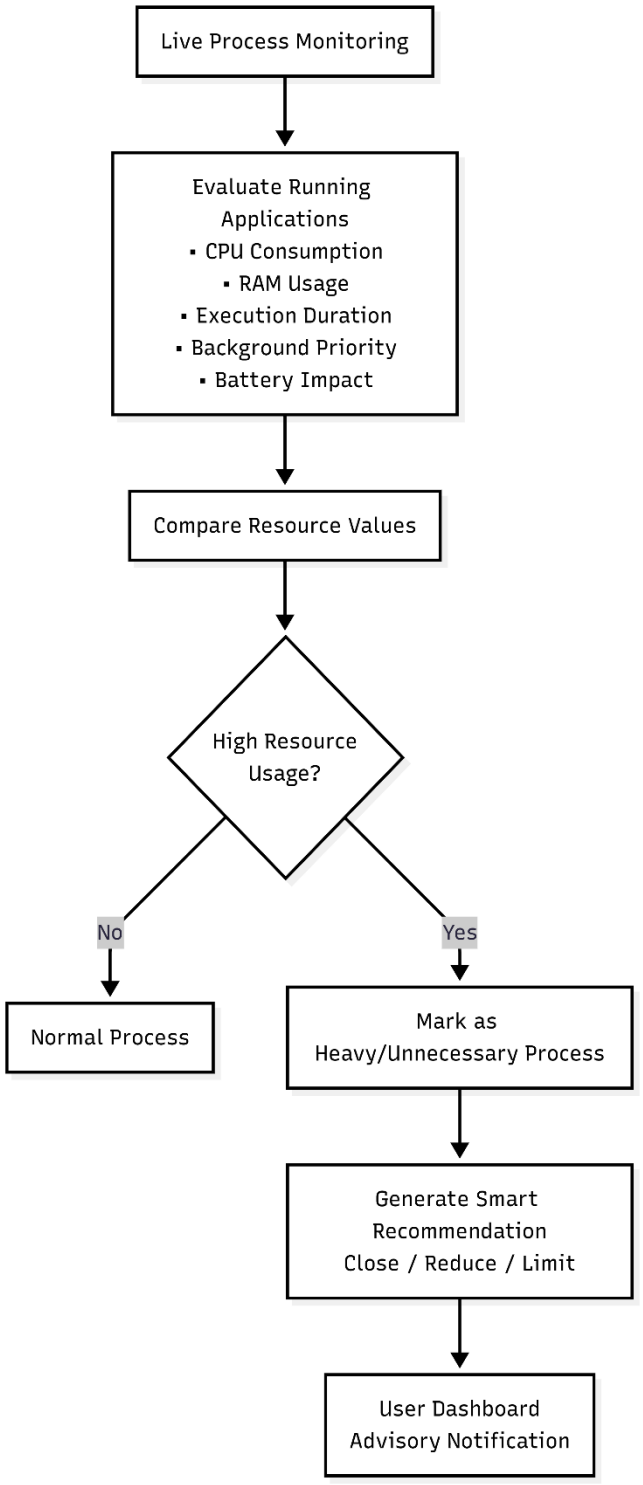

Figure 6. Smart Process Recommendation Flow

Instead of directly terminating the task, the software generates an intelligent recommendation for the user such as closing, reducing, or limiting the selected process. This makes the system more user-friendly because optimization is performed with guidance rather than unsafe forced shutdown. Recent adaptive software research supports the use of intelligent recommendation layers because advisory

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

optimization improves usability while still reducing manual troubleshooting effort [1], [18].

Thus, the Smart Process Recommendation Engine acts as a bridge between low-level process monitoring and practical user optimization by converting technical process statistics into understandable resource-saving suggestions.

3.7 Reinforcement Learning Based Adaptive Mode Switching

To make the proposed software truly self-adaptive, it is not sufficient to only monitor resources and generate alerts. The system must also be capable of selecting the most suitable optimization response according to changing runtime conditions. For this purpose, a Reinforcement Learning Based Adaptive Mode Switching Module is integrated as the main decision-making controller of the framework. This module enables the software to learn from previous optimization outcomes and automatically choose the best operating mode without constant manual intervention [2], [17].

The adaptive controller continuously receives processed information from the monitoring, battery prediction, anomaly detection, and process recommendation layers. Based on the current system state, it selects one of the following optimization modes:

- **Performance Mode** – for high workload and intensive processing
- **Power Saving Mode** – for low battery or overheating conditions
- **Focus Mode** – for moderate balanced usage with reduced background disturbance

Instead of relying on fixed predefined rules, the system uses a Q-learning based reinforcement strategy where every optimization action receives a reward according to its effectiveness. If the selected mode improves system speed, reduces battery drain, or lowers abnormal load, the reward becomes positive; otherwise, the model gradually learns to avoid that action in similar future conditions. Reinforcement learning studies confirm that reward-based adaptation produces more flexible and intelligent runtime decisions than static threshold systems [3], [18].

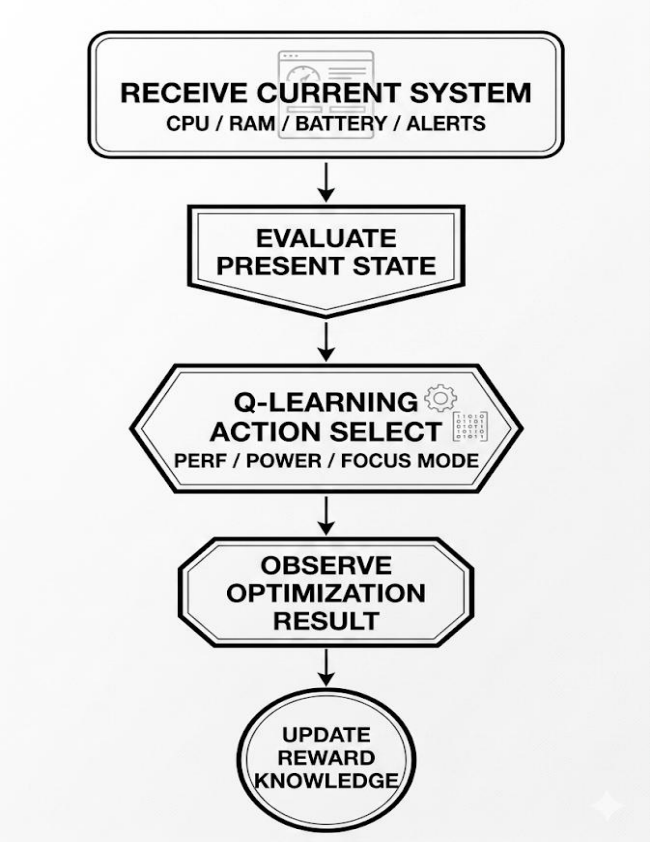

Figure 7. RL Adaptive Decision Flow

Through continuous reward updating, the software gradually becomes more accurate in selecting the proper adaptive mode under different workload situations. This transforms the proposed framework from a simple monitoring application into a learning optimization assistant that improves its future decisions with experience. Therefore, this module serves as the autonomous intelligence core of the entire AI Driven Self-Adaptive System [4], [8].

3.8 AI Usage Pattern Prediction

System performance requirements do not remain constant throughout the day because user activities continuously change from one workload condition to another. Sometimes the user may perform heavy tasks such as coding, gaming, or multimedia processing, while at other times the system may remain idle or under light usage. If the software can estimate these changing usage patterns in advance, optimization can be applied more effectively. Therefore, the proposed framework includes an **AI Usage Pattern Prediction Module** that studies previous system behavior and predicts possible future workload trends [1], [27].

This module analyzes the historical dataset generated during long-term monitoring and observes repeated relationships among:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

- CPU usage trend,
- RAM occupancy,
- process intensity,
- active session duration, and
- idle time behavior.

Using these recurring records, the machine learning model classifies whether the upcoming system condition is likely to be:

- high workload,
- normal workload, or
- idle/light usage.

Such early prediction helps the adaptive engine prepare optimization decisions before the actual resource stress begins.

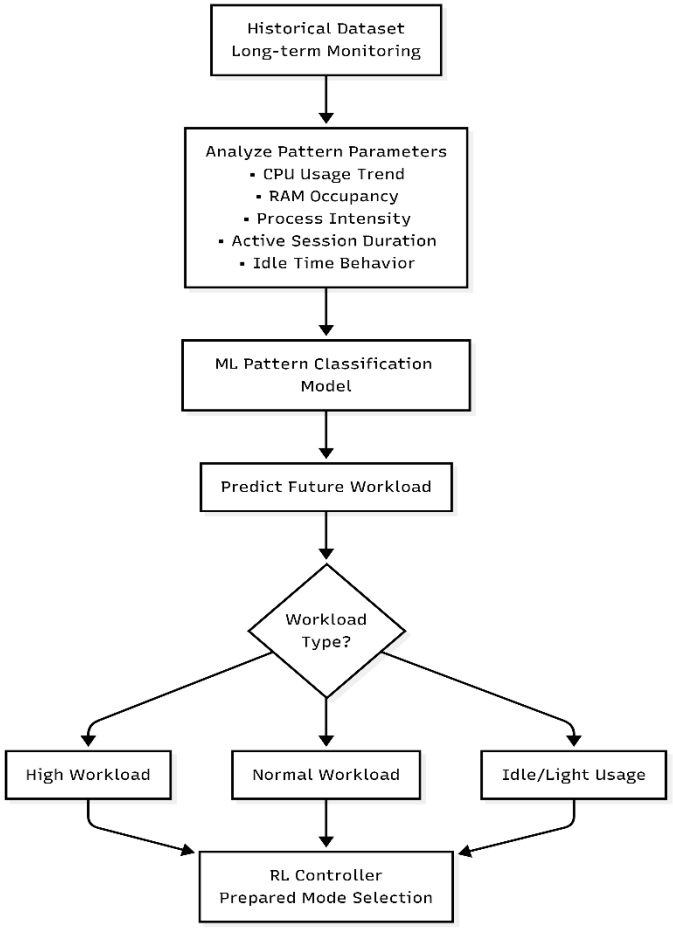

Figure 8. Usage Prediction Flow

By receiving this predicted workload information, the reinforcement learning controller can make more prepared mode selections instead of reacting only after system pressure increases. Recent adaptive learning studies indicate that anticipatory workload prediction improves the responsiveness and smoothness of self-managing software systems [14], [35].

Hence, this module adds future-awareness to the proposed framework and helps the software shift from reactive optimization to predictive adaptive management [2], [17].

3.9 Desktop Dashboard User Interface Design

To ensure that all intelligent monitoring and adaptive outputs remain easily understandable to the user, the proposed system provides a dedicated Desktop Dashboard User Interface. This interface acts as the visual communication layer between the internal AI engine and the end user. Instead of presenting technical background processing in hidden form, the dashboard displays all major system activities in a simple, organized, and real-time graphical format. Research in adaptive software usability suggests that visual dashboards improve user trust and transparency because the user can observe how the intelligent system is making decisions [18].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

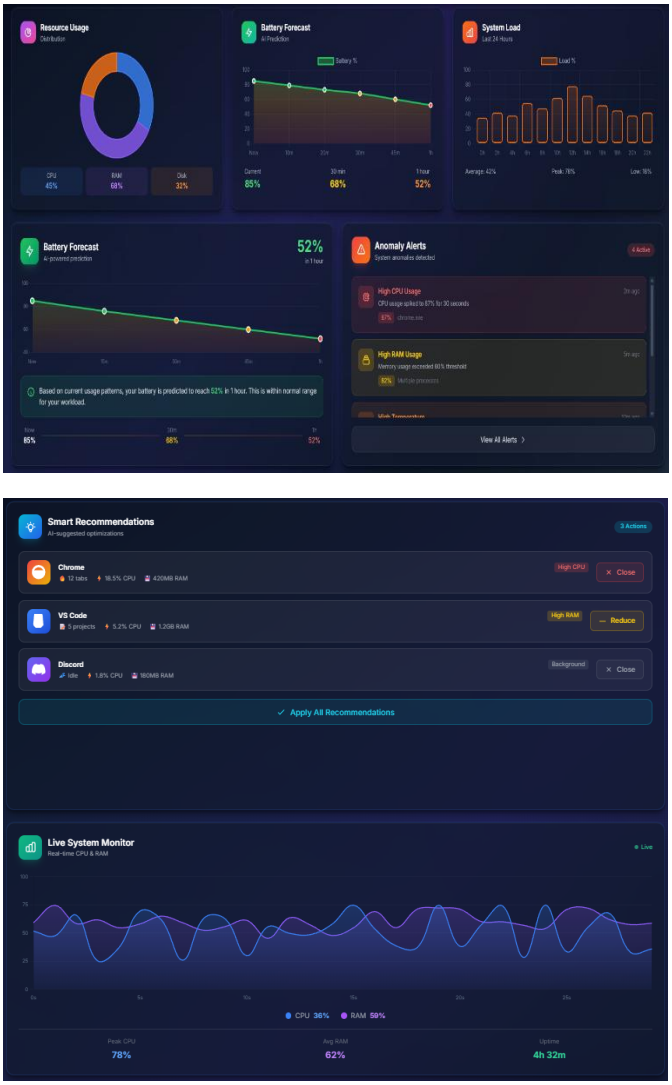

Figure 9. Desktop Dashboard Interface of Proposed System

The dashboard is designed using a modern dark-theme desktop layout in order to provide better readability and continuous monitoring comfort. It contains multiple visual panels that collectively represent the live health status of the machine. The major components displayed in the interface are:

- live CPU utilization graph,
- RAM usage graph,
- battery percentage and future battery forecast,
- anomaly warning notifications,
- smart process recommendations,
- current adaptive optimization mode, and
- system status summary.

Each panel is updated continuously as new monitoring values are received from the runtime sensing module. As a result, the user can instantly observe both the present system condition and the intelligent actions suggested by the software.

Special attention is given to keeping the interface simple and non-technical so that even users without deep hardware knowledge can understand the software recommendations. Whenever an abnormal condition or heavy background process is detected, the warning message is shown directly on the dashboard. Similarly, whenever the reinforcement learning controller changes the optimization mode, the selected adaptive state is highlighted visually.

Thus, the Desktop Dashboard User Interface not only improves monitoring visibility but also makes the proposed AI Driven Self-Adaptive System more interactive, transparent, and user-friendly by converting complex backend analysis into understandable visual information.

3.10 Cloud Monitoring Integration

Modern intelligent systems are expected to provide accessibility beyond the local machine so that important runtime information can be observed remotely whenever required. In order to extend the proposed desktop monitoring framework outside the physical computer, a Cloud Monitoring Integration Module is incorporated. This module allows the live adaptive status of the system to be synchronized to a lightweight cloud server and viewed from a mobile browser in real time [23], [36].

The cloud layer receives processed outputs from the desktop application such as:

- CPU usage,
- RAM status,
- battery percentage,
- anomaly alerts,
- active optimization mode, and
- smart system recommendations.

These values are transmitted through a Flask-based web service and stored in Firebase cloud storage for instant remote access. By using this communication mechanism, the software creates a parallel mobile monitoring dashboard that reflects the current health condition of the desktop even when the user is physically away from the machine.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

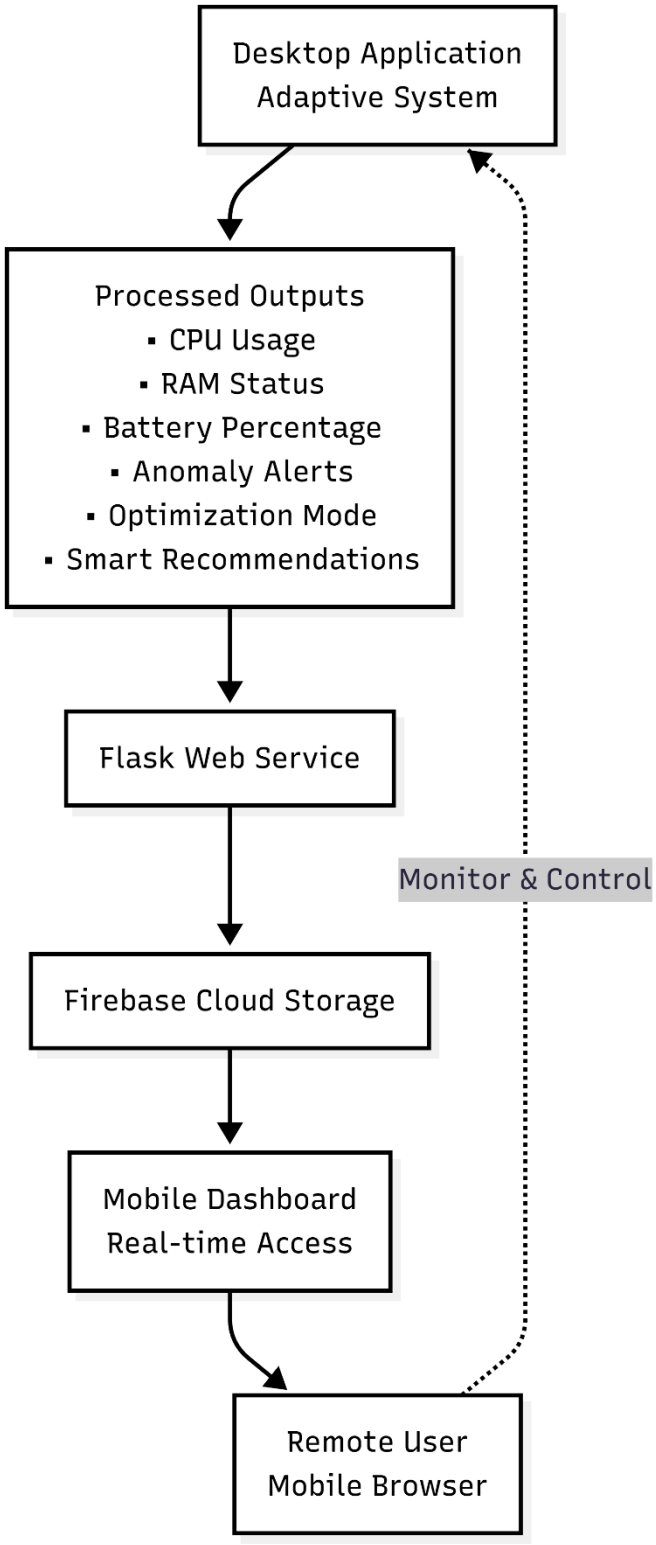

Figure 10. Cloud Monitoring Communication Flow

This integration improves the practical usability of the proposed system because users can receive warnings and monitor adaptive changes without continuously sitting in front of the desktop. Recent cloud-based adaptive monitoring studies indicate that remote visibility increases user control, alert responsiveness, and overall system transparency [14], [23].

Therefore, the Cloud Monitoring Integration module adds mobility and real-time remote supervision to the proposed AI Driven Self-Adaptive System, making it more flexible than traditional desktop-only monitoring applications.

3.11 Overall System Workflow Algorithm

The overall working of the proposed AI Driven Self-Adaptive System follows a continuous closed-loop adaptive process in which monitoring, learning, prediction, decision making, and remote synchronization operate together. The workflow begins with real-time collection of system resource parameters and gradually moves through different intelligent layers until an adaptive optimization response is generated [2], [22].

Initially, the Real-Time Monitoring Module captures live CPU, RAM, battery, temperature, process, and user activity values. These readings are then stored in the historical dataset where previous runtime behavior is maintained for machine learning analysis. After storage, the AI engine performs battery forecasting, anomaly identification, smart process evaluation, and usage pattern prediction using the collected information.

The processed outputs are then forwarded to the Reinforcement Learning controller, which selects the most suitable adaptive mode according to the present system condition. Once the decision is made, the generated optimization result is displayed on the desktop dashboard and simultaneously synchronized to the cloud monitoring platform for mobile access. This cycle is repeated continuously at every monitoring interval, allowing the software to learn and adapt in real time [17], [23].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

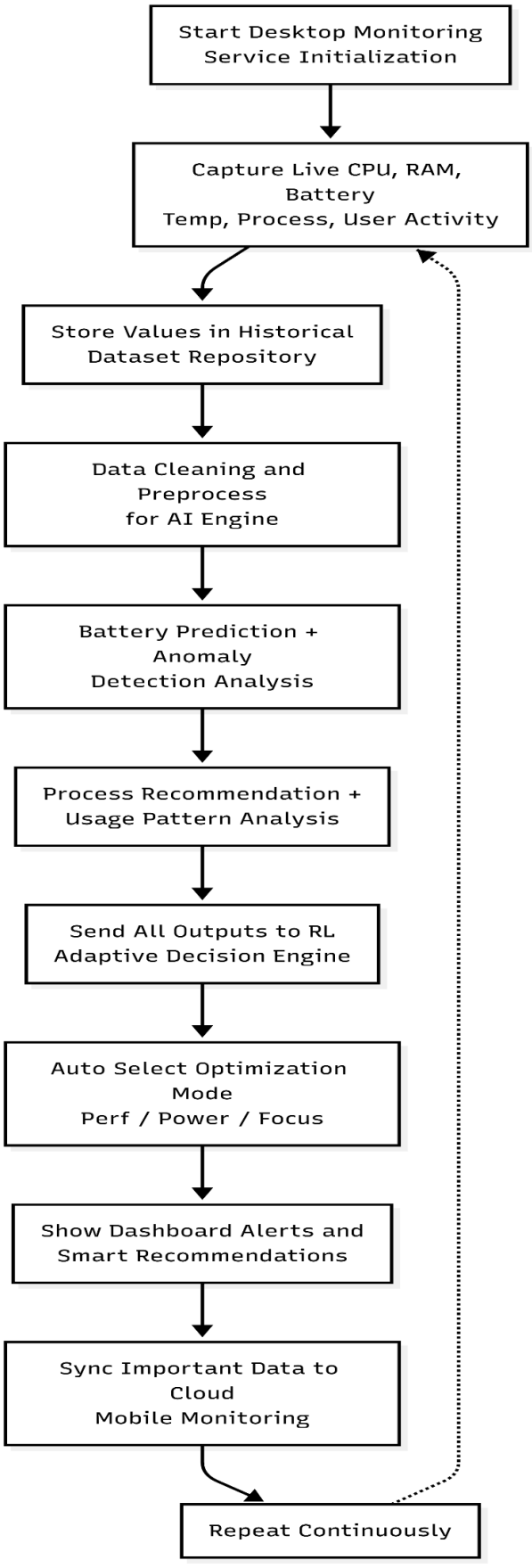

Figure 11. Overall System Workflow Algorithm

Thus, the proposed workflow ensures that the system does not remain limited to passive observation, but continuously learns from runtime behavior and produces autonomous optimization actions whenever required [3], [8].

4. RESULTS AND DISCUSSION

After implementing the proposed AI Driven Self-Adaptive System, the developed framework demonstrated that intelligent monitoring combined with machine learning and reinforcement learning can significantly improve desktop resource awareness and adaptive optimization. Unlike conventional monitoring software that only presents hardware values, the proposed system was able to generate predictive outputs, identify abnormal conditions, recommend optimization actions, and provide autonomous mode switching based on runtime analysis. The obtained results indicate that integrating multiple AI modules into a single desktop platform improves both system responsiveness and user convenience [2], [17].

4.1 Real-Time Monitoring Performance

The first observable result of the developed system was successful continuous monitoring of major desktop resources. The software was able to capture and update:

- CPU utilization,
- RAM usage,
- battery percentage,
- temperature condition,
- running process activity, and
- user idle/active status

at regular intervals without interrupting system performance.

This confirmed that the sensing layer was capable of generating stable runtime input for higher intelligent modules.

4.2 Battery Forecasting Result

The machine learning battery forecasting model successfully analyzed previous battery drain patterns and generated future battery estimates for upcoming usage intervals. The output showed that the system could predict whether the battery level was likely to decrease rapidly during heavy workload sessions.

As a result:

- early low-battery warnings were generated,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

- power-saving suggestions became possible, and
- adaptive mode preparation improved.

This demonstrates that predictive monitoring is more effective than waiting for manual low-battery observation [27], [35].

4.3 Anomaly Detection Output

The anomaly detection module successfully identified abnormal system behaviors during testing conditions such as:

- sudden CPU spike,
- unusual RAM increase,
- overheating trend, and
- hidden heavy background processes.

Whenever such irregular conditions occurred, the software generated instant warning notifications on the dashboard. This reduced the delay between issue occurrence and user awareness, proving the usefulness of AI-based abnormality recognition [29], [30].

4.4 Smart Process Recommendation Result

During continuous background scanning, the Smart Process Recommendation Engine was able to identify applications consuming unnecessarily high resources for extended periods. Based on this analysis, the dashboard displayed user-friendly recommendations for reducing or closing selected tasks.

This helped in:

- minimizing unwanted CPU load,
- reducing RAM congestion, and
- lowering background battery consumption.

Hence, the recommendation layer improved practical resource management without forcing unsafe automatic task termination.

4.5 Reinforcement Learning Adaptive Mode Switching Result

The Reinforcement Learning controller showed progressive improvement in selecting the proper optimization mode according to changing desktop conditions. During repeated runtime cycles, the software learned when to switch among:

- Performance Mode,
- Power Saving Mode, and
- Focus Mode.

Reward-based learning enabled more suitable mode selection over time, especially during high-load and low-battery scenarios. This confirms that self-learning adaptive decision making is more dynamic than fixed threshold based optimization [3], [18].

4.6 Cloud Monitoring Result

The cloud integration module successfully transmitted important runtime values from the desktop application to the remote mobile dashboard. Users were able to monitor:

- CPU status,
- battery level,
- anomaly alerts, and
- active adaptive mode

through mobile browser access. This improved the flexibility of the proposed system by extending monitoring beyond the local desktop environment [23], [36].

4.7 Overall Discussion

From the combined results, it can be observed that the proposed framework performed as more than a simple monitoring utility. The integration of prediction, anomaly awareness, smart recommendation, reinforcement learning adaptation, and cloud synchronization created a practical self-managing desktop assistant. Each module contributed to reducing manual supervision and improving runtime awareness.

Therefore, the developed results strongly suggest that AI-powered adaptive monitoring systems can provide:

- better performance transparency,
- improved battery efficiency,
- faster abnormality response, and
- more personalized optimization support

than conventional desktop monitoring applications [1], [8].

5. CONCLUSION

This research presented an AI Driven Self-Adaptive System as an intelligent desktop-based solution capable of transforming conventional system monitoring into an autonomous optimization framework. The developed model successfully combines real-time resource monitoring, historical data learning, predictive battery analysis, anomaly

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

detection, smart process recommendation, reinforcement learning based adaptive mode switching, and cloud monitoring into one unified platform. Unlike traditional monitoring tools that only provide passive hardware statistics, the proposed system demonstrates the ability to observe, learn, predict, and respond according to continuously changing runtime conditions [1], [2].

The overall implementation shows that the integration of Machine Learning and Reinforcement Learning can significantly reduce manual monitoring effort while improving system awareness, battery efficiency, and adaptive performance management. In addition, the cloud-enabled remote accessibility further enhances the practical usability of the framework by allowing users to remain connected with system health information beyond the local desktop environment [17], [23].

Therefore, it can be concluded that the proposed AI Driven Self-Adaptive System provides a more intelligent, proactive, and user- centered alternative to traditional desktop monitoring applications. The developed framework highlights how modern AI techniques can be effectively used to build self-learning software capable of making continuous optimization decisions in real time [3], [8].

6. AREAS OF RESEARCH: FUTURE DIRECTIONS

Although the proposed AI Driven Self-Adaptive System provides an intelligent framework for desktop monitoring, predictive analysis, anomaly detection, and adaptive optimization, there are still several advanced research possibilities that can further improve its intelligence, autonomy, and practical usability. As Artificial Intelligence and self-managing software continue to evolve, future enhancements can make the system more accurate, more personalized, and capable of handling wider computing environments. Current adaptive software studies also suggest that continuous learning, deeper automation, and multi-device intelligence are the next major directions in self-optimizing systems [9], [17].

6.1 Deep Learning Based Resource Prediction

In the current framework, lightweight machine learning models are used for battery and workload forecasting. In future research, deep learning models can be integrated to study more complex long-term behavioural relationships among CPU, RAM, battery, and user workload patterns. This may improve prediction accuracy during highly dynamic system conditions [32], [35].

6.2 Fully Automatic Process Optimization

At present, the software generates smart recommendations for high-resource applications. A future version may include controlled automatic background process management where the system can suspend or limit unnecessary applications without waiting for user confirmation. Such autonomous optimization can reduce manual effort further [1], [23].

6.3 Advanced Cyber Threat and Security Monitoring

The anomaly detection module currently focuses mainly on abnormal resource behavior. Future work can extend this intelligence toward hidden malware activity, suspicious unauthorized programs, and unusual network consumption. This would transform the software from a performance assistant into a combined adaptive security monitoring platform [26], [30].

6.4 Multi-Device and IoT Integration

The present model is designed mainly for desktop and mobile browser monitoring. Future research may expand the architecture to synchronize across multiple laptops, office systems, or IoT devices under one intelligent dashboard. This will make the framework more scalable for enterprise-level adaptive supervision [14], [23].

6.5 More Advanced Reinforcement Learning Models

Currently, a basic Q-learning adaptive engine is considered for mode switching. In future studies, more advanced deep reinforcement learning algorithms can be applied to improve faster self-learning, more complex decision policies, and better long-term optimization performance under uncertain runtime environments [3], [8].

6.6 Personalized User Behavior Adaptation

Another promising direction is the development of highly personalized adaptive profiles. The software may learn individual user habits such as study time, gaming time, idle periods, or charging behavior and generate custom optimization policies specifically suited to that user. This would increase the human-centered intelligence of the framework [2], [18].

6.7 Voice Assistant and Notification Automation

Future implementation can also include voice-based warnings, intelligent desktop notifications, and conversational assistant support so that the software can interact with the user more naturally during anomaly situations or adaptive changes. This would improve accessibility and real-time user engagement.

Overall, these future directions indicate that the proposed AI Driven Self-Adaptive System has strong research scalability

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

beyond its current implementation. With deeper learning models, stronger automation, security intelligence, and broader device integration, the framework can evolve into a highly autonomous self-managing computing ecosystem suitable for next-generation smart environments [4], [9].

7. REFERENCES

[1] O. Gheibi, D. Weyns, and F. Quin, "Applying Machine Learning in Self-Adaptive Systems: A Systematic Literature Review," *arXiv preprint arXiv:2103.04112*, pp. 1–34, 2021.

[2] A. Metzger, C. Quinton, Z. A. Mann, L. Baresi, and K. Pohl, "Realizing Self-Adaptive Systems via Online Reinforcement Learning and Feature-Model Guided Exploration," *Computing*, vol. 106, no. 5, pp. 1251–1272, 2024.

[3] M. Zhang, J. Li, H. Zhao, K. Tei, S. Honiden, and Z. Jin, "A Meta Reinforcement Learning-based Approach for Self-Adaptive System," *Proceedings of IEEE ACSOS*, pp. 1–10, 2021.

[4] F. Feit, A. Metzger, and K. Pohl, "Explaining Online Reinforcement Learning Decisions of Self-Adaptive Systems," *arXiv preprint arXiv:2210.05931*, pp. 1–18, 2022.

[5] M. Waqar, A. H. Khan, and I. Bhatti, "Self-Adaptive AI Systems for Autonomous Decision-Making in Dynamic Environments," *International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence*, vol. 2, no. 3, pp. 45–57, 2024.

[6] T. Shaik, X. Tao, L. Li, H. Xie, H. N. Dai, F. Zhao, and J. Yong, "AI-Driven Multi-Agent Reinforcement Learning Framework for Real-Time Monitoring of Physiological Signals," *Brain Informatics*, vol. 12, no. 14, pp. 1–20, 2025.

[7] A. K. Aleti, "Reinforcement Learning Driven Adaptive Software Testing with Continuous Fault Anticipation and Self-Healing Feedback Loops," *International Journal of Artificial Intelligence, Data Science and Machine Learning*, vol. 6, no. 4, pp. 21–28, 2025.

[8] M. M. Alhijawi, A. Alenezi, and S. Alshammari, "A Novel Continual Reinforcement Learning-Based Expert System for Self-Optimization of Soft Real-Time Systems," *Expert Systems with Applications*, vol. 235, pp. 122309, 2024.

[9] D. Weyns, "Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges," *Handbook of Software Engineering*, Springer, pp. 399–443, 2021.

[10] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[11] R. R. Al-Nima, T. Han, S. A. Al-Sumaidaee, and T. Chen, "Robustness and Performance of Deep Reinforcement Learning," *Applied Soft Computing*, vol. 105, pp. 107295, 2021.

[12] C. Pahl and S. Azimi, "Constructing Dependable Data-Driven Software with Machine Learning," *IEEE Software*, vol. 38, no. 5, pp. 88–97, 2021.

[13] R. Wohlrab, J. Cámara, D. Garlan, and B. Schmerl, "Explaining Quality Attribute Tradeoffs in Automated Planning for Self-Adaptive Systems," *Journal of Systems and Software*, vol. 198, pp. 111538, 2023.

[14] G. Zhou, W. Tian, and R. Buyya, "Deep Reinforcement Learning-Based Methods for Resource Scheduling in Cloud Computing: A Review," *Artificial Intelligence Review*, vol. 57, no. 124, pp. 1–29, 2024.

[15] S. Saboori, G. Jiang, and H. Chen, "Autotuning Configurations in Distributed Systems for Performance Improvements," *IEEE ICDCS*, pp. 769–776, 2008.

[16] J. Faustino, "Quality Management for AI-Generated Self-Adaptive Resource Controllers," *Machines*, vol. 14, no. 1, pp. 25–41, 2026.

[17] S. Shevtsov, R. Calinescu, and C. Paterson, "Explainable Reinforcement Learning for Trusted Self-Adaptive Systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 16, no. 4, pp. 1–27, 2022.

[18] F. Feit, A. Metzger, and K. Pohl, "A User Study on Explainable Online Reinforcement Learning for Adaptive Systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 19, no. 3, pp. 1–44, 2024.

[19] L. Baresi and C. Ghezzi, "The Disappearing Boundary Between Development-Time and Run-Time," *Proceedings of FSE*, pp. 17–22, 2010.

[20] H. Gomaa, "Engineering Self-Adaptive Systems with Feedback Control Loops," *Proceedings of ICSE Workshop*, pp. 1–8, 2019.

[21] M. Salehie and L. Tahvildari, "Self-Adaptive Software: Landscape and Research Challenges," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 2, pp. 1–42, 2009.

[22] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptive Systems," *Software and Systems Modeling*, vol. 14, no. 1, pp. 1–24, 2021.

[23] T. Chen and R. Bahsoon, "Self-Adaptive and Online QoS Modeling for Cloud-Based Software Services," *IEEE Transactions on Software Engineering*, vol. 43, no. 5, pp. 453–475, 2021.

[24] A. Bauer and M. Truscan, "Runtime Verification for Adaptive Software Systems," *Formal Methods in System Design*, vol. 55, no. 3, pp. 233–261, 2020.

[25] J. Cámara, D. Garlan, and B. Schmerl, "Adaptation Impact and Environment Models for Architecture-Based Self-Adaptive Systems," *Science of Computer Programming*, vol. 136, pp. 50–72, 2021.

[26] H. M. HaddadPajouh, A. Dehghantanha, and K. K. R. Choo, "A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2020.

[27] D. Singh and V. Kumar, "Predictive Analytics for Battery Consumption in Intelligent Computing Systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 7, pp. 3451–3464, 2022.

[28] S. H. Hwang and J. Kim, "Machine Learning Based CPU and Memory Load Prediction for Smart Computing Environments," *Sensors*, vol. 22, no. 14, pp. 5112, 2022.

[29] M. M. Islam, A. Rahman, and N. Rahman, "Anomaly Detection in Computer Resource Utilization Using Isolation Forest," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 6, pp. 211–219, 2022.

[30] A. Lavin and S. Ahmad, "Evaluating Real-Time Anomaly Detection Algorithms for Streaming System Data," *Proceedings of ICDM Workshop*, pp. 1–9, 2021.

[31] J. Brownlee, *Machine Learning Algorithms for Predictive Modeling*. Melbourne, Australia: Machine Learning Mastery, 2021.

[32] F. Chollet, *Deep Learning with Python*. 2nd ed. New York, NY, USA: Manning Publications, 2021.

[33] S. Raschka and V. Mirjalili, *Python Machine Learning*. 3rd ed. Birmingham, UK: Packt Publishing, 2022.

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[35] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow*. 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.

[36] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*. 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.