Alqalam Journal of Medical and Applied Sciences. 2026;9(7):2174-2186
https://doi.org/10.54361/ajmas.269748

Original article

Formal Specifications of Context-Aware Systems using the Calculus of Context-Aware Ambient

Najat Atbaiga[1]*, Ghazala Jabir[1], Mabroka Al-Shebany[2]

[1]*Department of Software Engineering, Faculty of Information Technology, University of Sirte, Sirte, Libya*
[2]*Department of Computer Science, Faculty of Computer Science, University of Sirte, Sirte, Libya*
Corresponding email. *najatatbaiga@su.edu.ly*

Abstract
A context-aware laboratory system (CaLab) proposes several services in smart laboratories, including automating critical functions such as accurate attendance logging, intelligent energy management, and personalized instructional setup. The CaLab system is designed to adapt intelligently to user and environmental context. The behavioral rules, contextual relationships, and adaptive processes of the CaLab System are defined to support accurate simulation and validation. This paper proposes the formal specification and simulation of the Context-aware Computer Laboratory System. The formal specification ensures the logical correctness and integrity of the design. The Calculus of Context-aware Ambients (CCA, in short) is adopted as the formal modelling language. CCA provides a structured framework for representing the contextual interactions and adaptive processes within ubiquitous computing environments. The Generic Context-Aware Architectural Framework provides a robust structure for translating the CCA rules into a modular system. A Python simulation is utilized for validating the behavior of the proposed system, which provides a controlled evaluation of its capabilities. The main objective of this work is to deliver a formally verified, scalable, and intelligent framework that significantly enhances the operational quality and resource efficiency of university computer laboratories. Furthermore, a formal specification of a CA system using CCA can be translated into any high-level programming language.
Keywords. Pervasive System, Calculus, Context-Aware Ambients, Formal Specification.

Introduction

Mark Weiser's vision of computers of the 21st century was " they weave themselves into the fabric of everyday life until they are indistinguishable from it" [1]. This concept led to the beginning of ubiquitous computing, sometimes referred to as pervasive computing, which is a new paradigm for distributed systems. Appliances should disappear into the background and detect the user's context to provide the necessary assistance to the user at any time and from any location, making the user and his tasks the central focus rather than computers and technological problems [2]. Context-aware systems are a synonym of such systems. Utilization of pervasive systems can be in any domain, including education, by saving time and acting as a fast and easy way of producing services.

Due to the organizational processes of a university, such as attendance registration and laboratory customization, interruptions can occur during the university practical lecture. A context-aware laboratory system consists of components representing lab resources, students, instructors, and devices, which interact according to context-aware rules. The CaLab System provides several services in a smart laboratory, including detecting the presence of a lecturer, which triggers a reasoning process, and logging lecturers into the Blackboard virtual learning environment at the beginning of each practical lecture. The authorized access allows lecturers to interact with teaching materials such as slides and videos stored in the Blackboard virtual learning environment, activates the laboratory computers, and adjusts environmental settings such as lighting and student attendance management as they enter the laboratory. The reasoning mechanism logs the lecturer out at the end of the practical lecture and ensures that adaptive operations, such as automatic device shutdown. This system also notifies lecturers of a list of students' attendance automatically.

Using information from the timetable, the system is aware of the practical lectures that are scheduled in the laboratory and the authorized students who attend the lecture. In addition, if no presence is detected after a short predefined period, the system automatically initiates energy-saving actions such as turning off computers, lights, and other devices to conserve power and maintain operational safety. The system is formally modeled using the Context Calculus Approach (CCA, in short), a powerful method for representing and verifying contextual dependencies. This enables formal analysis and simulation of the laboratory system using the execution environment of Python. This paper models a context-aware laboratory system that responds dynamically to real-time environmental and user context, enhancing operational efficiency and user comfort. Several important properties of a context-aware laboratory system have been validated.

Related works

A hybrid approach was described for evaluating students' foreign language knowledge in a cyber–physical educational environment. The proposed assessment approach, using the mathematical theory of fuzzy functions, ensures a fair evaluation of students. Ambient-oriented CCA modeling provides a variety of opportunities for preliminary testing, verification, and analysis of the basis scenarios relevant to student evolution and providing the entire learning process in the cyber–physical educational environment [3]. Siewe demonstrates that CCA can be used to specify any system that can be specified in Petri nets. The proposed algorithm transforms a Petri net into a process in CCA. The CCA process can be analyzed and verified using the CCA simulator ccaPL. The algorithm is implemented in Python, and therefore the translation

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

of a Petri net into a CCA process. This article concluded that the specification of complex pervasive and IoT systems can be specified using the Petri net graphical notations together with the CCA language [4]. A formal specification of the Whiteboard system was produced in the calculus of context-aware Ambient (CCA). This system provides a variety of services in a smart classroom, such as the ability to log lecturers and students into the Blackboard virtual learning environment at the beginning of each lecture and log them out at the end of the lecture. Additionally, the students have been notified by the system of their absence from a lecture, and it maintains a list of attendance automatically. As proof-of-concept, important properties of a classroom whiteboard system have been validated using the execution environment of CCA called ccaPL [5]. An algorithm was proposed for translating a context-aware use case diagram into a CCA process. The use of the CCA tools to analyze the specification of context-aware systems. The paper demonstrated how the CCA interpreter can be used to validate and execute various scenarios of a use case diagram. A real-world example of a context-aware pedestrian collision avoidance system is used to illustrate the pragmatics of the approach [6].

Overview of Context-Aware Laboratory

A context-aware laboratory (CaLab) system is designed as an intelligent and context-aware framework for managing university computer laboratories. Its primary objective is to automate and optimize laboratory operations, including student attendance tracking, device control, and personalized content delivery, while ensuring efficient and adaptive interactions within the lab environment. The system integrates smart devices and context-sensitive mechanisms to respond dynamically to environmental changes and user actions. The CaWB represents the central intelligence of the system and serves as the control and coordination hub. It monitors contextual inputs, interprets events, and triggers corresponding actions across the laboratory. It is assumed that each lecturer and student is equipped with a smart ID. The sensors detect the lecturer and student ID inside the lab, where contextual parameters such as user identity, presence, and environmental states (light, window, occupancy) are detected.

Once a signal is captured, it is transmitted to the Context-Aware Whiteboard (CaWB), which requests login through the BB. The BB acts as a context server and authentication repository. It stores user credentials, schedules, and module-related information. The CaWB communicates with the Blackboard (BB) component after obtaining a user presence signal to authenticate the detected identity. The BB validates the user and responds to CaWB with an authorization message. If authentication is successful, CaWB proceeds to notify the Attendance List (aList) module, which records the presence of the corresponding user and updates the attendance register dynamically.

CaWB triggers multiple parallel interactions: sending activation commands to the Computer Control Module (CaPC) to power on the laboratory computers. Also, it issues context-based configuration commands to the Environmental Control Subsystem (Checker, Light, Win) to monitor lighting and windows. Activating the Display and Audio-Visual Systems (AVS) to load the lecturer's materials automatically and prepare the lab for teaching activities. Throughout this process, the Checker continuously monitors the operational status of environmental components and reports their states back to CaWB. This feedback ensures that all devices remain synchronized with the current context. Additionally, the Reporting Module within Checker may generate periodic summaries (e.g., energy usage or session logs) and send them to administrative interfaces for review. At the end of the practical lecture, or when all users leave the lab, the sensors detect the absence and transfer this update to CaWB. Consequently, CaWB instructs CaPC to shut down the computers and turn off the lights. This automatic adjustment guarantees energy conservation and maintains system integrity. Designing a context-aware system requires a well-structured architectural framework that enables the acquisition, processing, and utilization of contextual information. This general architecture shows the main components of the system [2]. The ID card should be carried by lecturers and students. Generic context-aware architecture typically consists of three fundamental layers: the context acquisition layer, where the sensor hardware senses the data from the surrounding environment and communicates with the middleware to transfer the data wirelessly; and the context reasoning (or middleware) layer, which communicates with the context server using a network based on TCP/IP, for example. The middleware of this system is whiteboard (CaWB), and the context server updates and synchronizes information across the system and the application layer. Each layer plays a specific role in transforming raw environmental data into meaningful adaptive actions [7]. (Fig 1) depicts the general architecture of the CaLab system.

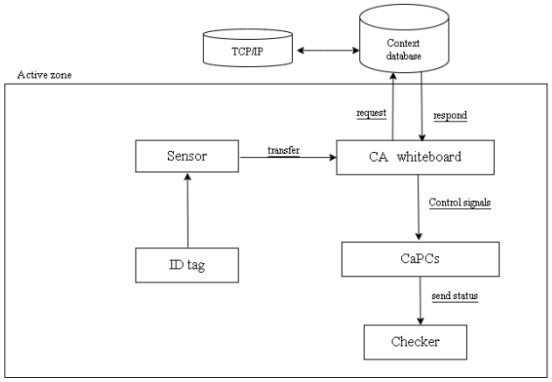

Fig.1. Architecture of CaLab system

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

Overview of CCA

This section demonstrates the informal semantics and syntax of CCA. The Calculus of Context-aware Ambients (CCA) is a formal modelling language designed to specify and reason about systems whose behavior depends on changing contextual information. It extends the Ambient Calculus, originally introduced by Cardelli and Gordon in 1998, by incorporating context-awareness into its computational model. CCA provides a formal notation for high-level description of mobile and Ca systems, enables formal reasoning about properties of Ca systems, and helps understanding the behavior of mobile and Ca systems through simulation. For more details about CCA, see reference [8]. The syntax of CCA is presented in (Table 2), based on three syntactic categories: processes (denoted by P or Q), context expressions (denoted by k), and capabilities (denoted by M). We suppose a countably infinite set of names, where elements are written in lowercase letters, e.g., n, g, x, and y. There are reviews of the meaning of the three syntactics.

CCA processes

Denote a computation, describe the behavior of an ambient, and execute specific capabilities after the environment is satisfied.

Capabilities

which are elementary actions a process can perform. Ambient exchanges messages using output/ input capabilities, some form capabilities used for process abstraction call.

Context model

In CCA, context is modelled as a process with a hole in it. The hole (ʘ) in a context represents the position of the process of that context. (Table 1) A property of a context can be described by a formula called a context expression (CE, short).

Table 1. The syntax of Context

$C ::= 0 \mid \Theta \mid n[C] \mid 'C\|P' \mid (\nu\, n)\, C$

Context expressions

The CE always holds true. Which is a formula expressing some property of a process's context. The CE ● holds solely for the whole context, as an example, the position of the process assessing that context expression. Some symbols of CE, such as A CE ⊕κ, hold for a context if that context contains a child context which k maintains. A CE ✧κ holds for a context if there is a sub-context for which k holds and exists somewhere in that context. Some examples of predicates that can be used to indicate common context properties are the location of the user and the individuals the user is with.

- has (n) = ⊕(● | n [true] | true)
  means if ʎ is upper ambient and contains an ambient called n
- at (n) = n [⊕(● | true)] | true
  means if ʎ is located at the upper ambient called n
- with(n) = n[true] |⊕ (● |true)
  means if ʎ is (co-located) with an ambient called n at the upper ambient
- at2(n,m) = n [m [true] | true] | true
  means if the ambient m is inside the ambient n

Formal Specification of CaLab System

The CaLab system is formally specified in CCA. It is assumed that the CaWB system is in a computer Lab (CaLab), which acts as the active zone of the system. i.e any ID card within the active zone can be detected by the system. The CaLab is modelled in CCA using the following ambients:

- The lab ambient caLab, which represents the active zone of the system.
- The blackboard ambient BB, which models the context server and hosts the user login accounts.
- The display system Ambient Display, for PowerPoint presentations and tutorials.
- The audio-visual system, Ambient AVS, for playing audio content and videos.
- The attendance monitoring system aList, which manages student attendance.
- The context-aware whiteboard ambient CaWB controls the whole system.
- The lecturer ambient lect1, representing a lecturer of ID number lect1.
- The student ambient st1, representing a student of ID number st1.
- The computer ambient CaPC, which represents the personal computer in the laboratory and manages its on/off state based on occupancy.
- The context-aware window and lighting ambients (Win, light) represent the environmental controls of windows and lighting, respectively.
- The check ambient, a logical parent of light and win, which collects their state (on/off, open/closed).

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

Table 2. The syntax of CCA

P, Q ::=		**Process**
	0	inactivity
	P|Q	Parallel composition
	(ν *n*) *P*	name restriction
	!P	replication
	n[*P*]	ambient
	κ?*M*.*P*	context-guarded prefix
	$x\triangleright(y_1,\ldots,y_i).P$	process abstraction
	if κ_1? $M_1.P_1$ ... $\kappa_i.P_i$ fi	selection
κ ::=		**Context Expression**
	true	true
	n=*m*	name match
	●	hole
	¬ *κ*	negation
	$\kappa_1 \mid \kappa_2$	parallel composition
	κ_1 ^ κ_2	*conjunction*
	n[*κ*]	location context
	⊕*κ*	spatial next modality
	✧*κ*	somewhere modality
M ::=		**Capabilities**
	del *n*	delete ambient n
	in *n*	move into ambient n
	out	move out of parent
	α $x(z_1, \ldots, z_i)$	process abstraction call
	α rec($y_1,\ldots,y_i$)	input
	α send($z_1,\ldots,z_i$)	output
α ::=		**Locations**
	↑	any parent
	n↑	parent n
	↓	any child
	n↓	child n
	::	any sibling
	n ::	sibling n
	∈	locally

The overall model of the system is given in (Figure.2), where Pi is a process specifying the behavior of the corresponding ambient that contains it. The textual representation of the model is given in Eq. (1).

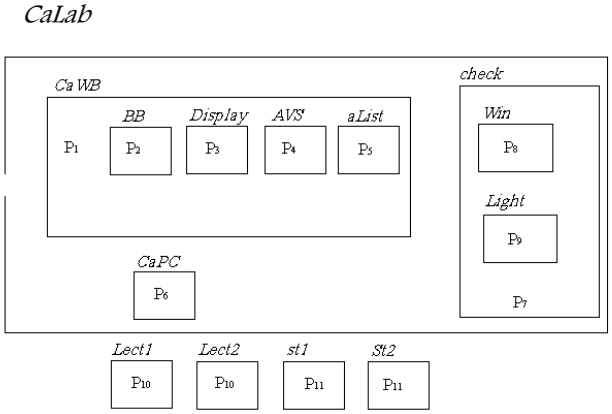

Figure 2. The model of the CaLab system in CCA

Received: *18-05-2026* - ***Accepted***: *16-07-2026* - ***Published***: *21-07-2026*

$$
\begin{array}{l}
caLab[\\
\quad CaWB[\\
\quad\quad P_1 \\
\quad\quad |BB[P_2] \\
\quad\quad |Display[P_3] \\
\quad\quad |AVS[P_4] \\
\quad\quad |aList[P_5] \\
\quad\quad] \\
| \ CaPC[P_6] \\
|check[\\
P_7 \\
|win[P_8] \\
|Light[P_9] \\
\quad]
\end{array}
\qquad (1)
$$

The context-aware whiteboard ambient CaWB

This ambient continuously monitors the active zone of the laboratory (CaLab) to detect the presence of lecturers and students through their smart IDs. Once a user is detected, CaWB initiates a login request to Blackboard (BB). This behavior is specified in Eq. (2).

$$! :: rec(src,req) \ . \ BB↓send(src,req).0 \qquad (2)$$

If the login is accepted, the user's ID is added to the attendance list. Otherwise, the user is informed that access has been denied. In addition to authentication, CaWB handles requests for teaching resources, such as lecture slides and tutorials (sent to Display), and audio or video content (sent to AVS). This behavior is specified in Eq. (3).

$$
\begin{array}{l}
!↓ rec(dest,req,reply) \ .if \\
\quad (req=slides \vee req=tutorials)? \ Display↓ send(req,reply).0 \\
\quad (req=audio \vee req=video)? \ AVS↓ send(req,reply).0 \\
\quad (reply=allowed \wedge ✧ at2(alist,dest))? \ dest:: send(req,reply).CaPC::send(power\_on).0 \\
\quad (reply=allowed \wedge \neg ✧ at2(alist,dest))? \ aList↓send(dest) \ . \ dest :: send(req,reply).0 \ | \\
CaPC::send(power\_on).0 \\
\quad fi \\
! \neg at2(caLab,lecturer)? \ CaPC :: send(power\_off).0
\end{array}
\qquad (3)
$$

Therefore, the whole behavior P1 of the CaWB is the parallel composition of the process in Eq. (2) and the process in Eq. (3), as collected in Eq. (4).

$$P1 \ = \ Eq. \ (2) \ | \ Eq. \ (3) \qquad (4)$$

The blackboard ambient BB

The BB ambient serves as the system's database for lecturers and students enrolled in a specific module. When CaWB sends a login request, BB checks if the user ID exists in its records. If the ID is present, the login is approved; and denied if not. Besides handling authentication, BB also provides access to teaching resources such as slides, tutorials, audio, and video files. These resources are forwarded through CaWB to the appropriate ambients, ensuring that only authenticated users can access the learning materials. The behavior of P2 is specified in Eq. (5).

$$
\begin{array}{l}
! \uparrow rec(src,req) \ .if \\
(req=login \wedge ✧ \ has \ (src))? \uparrow send(src,req,allowed).0 \\
(req=login \wedge \neg ✧ \ has \ (src))? \uparrow send(src,req,denied).0 \\
\quad (reply=slides)? \uparrow send(src,req,slides\_ppt).0 \\
\quad (reply=tutorials)? \uparrow send(src,req,tutorial\_pdf).0 \\
\quad (reply=audio)? \uparrow send(src,req,mm01\_wav).0 \\
\quad (reply=video)? \uparrow send(src,req,mm01\_wmv).0 \\
\quad fi \\
|st1[0] \ | \ lect1[0]
\end{array}
\qquad (5)
$$

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

The lecturer ambient Lect

The Lect1 ambient represents a lecturer equipped with a smart ID. When the lecturer enters the lab, the ID is detected by CaWB, which requests login through the BB. A successful login grants the lecturer access to teaching materials (slides, tutorials, audio, video). If the login fails, the lecturer's access is denied, and they cannot proceed with the lecture. The general behavior P10 of a lecturer is depicted in Eq. (6).

$$P10 = \begin{pmatrix} \text{In } CaLab.\ CaWB\text{:: send}(lecturer,login). \\ CaWB\text{::rec}(req,reply).\ \text{if} \\ (reply=allowed)?\ CaWB\text{:: send}(lecturer,slides). \\ CaWB\text{:: send}(lecturer,tutorial). \\ CaWB\text{:: send}(lecturer,audio). \\ CaWB\text{:: send}(lecturer,video).\ \text{Out.0} \\ (reply=denied)?\ \text{Out.0} \\ \text{fi} \end{pmatrix} \quad (6)$$

The student ambient St

It is assumed that each student carries a smart badge that communicates with the CaWB system. Once a student enters the laboratory, CaWB reads the student's ID and forwards it to the BB for verification. If the ID is confirmed, the student is added automatically to the attendance list (aList). If the login attempt fails, the student is notified and eventually leaves the lab. The general behavior P11 of a student is specified in Eq. (7).

$$P11 = \begin{pmatrix} \text{In } caLab\ .\ CaWB\text{::send}(student,login). \\ CaWB\text{::rec}(req,reply).\ \text{if} \\ (reply=allowed)?\ \text{send().out.0} \\ (reply=denied)?\ \text{out.0} \\ \text{fi} \end{pmatrix} \quad (7)$$

The CaPC ambient

The CaPC ambient models the laboratory's personal computers. Each computer can be switched on or off according to lab occupancy and user presence. When students are authenticated, computers are enabled for use. If the lab is empty, computers are automatically turned off to save energy. The behavior of P6 is specified in Eq. (8).

$$P_6 = \begin{pmatrix} CaWB\text{:: rec}(req).\ \text{if} \\ (req=power\_on)?\ \text{Switch to}<on>.0 \\ (req=power\_off)?\ \text{Switch to}<off>.0 \\ \text{fi} \\ checker\text{:: send}(CaPC,\ status).0 \end{pmatrix} \quad (8)$$

The other ambients

The Check ambient serves as a logical parent to the Win and Light ambients. It collects their current states and coordinates them to maintain consistency in environmental control. For example, ensuring the changes in lighting and ventilation in the lab. The behavior of P7 is specified in Eq. (9).

$$P7 = !\text{rec(src,status). gen(report).0} \quad (9)$$

The Win ambient models the state of laboratory windows (open/closed), while the Light ambient manages illumination (on/off). Both adjust their state depending on occupancy and context, ensuring proper environmental conditions in the lab. The behavior of P8 and P9 is specified in Eq. (10) and Eq. (11), respectively.

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

$$P8= !\uparrow send(win,status).0 \quad (10)$$

$$P_9= !\uparrow send(\textit{light,status}).0 \quad (11)$$

The Display ambient represents the screen used to present teaching materials such as slides and tutorials. It receives documents from CaWB and visualizes them to the users. The behavior of P3 is specified in Eq. (12).

$$P3= !CaWB\uparrow rec(req,reply).0 \quad (12)$$

The AVS ambient is responsible for multimedia playback in the lab. It receives requests for audio or video files from CaWB and plays the content accordingly. The behavior of P4 is specified in Eq. (13).

$$P4= !CaWB\uparrow rec(req,reply).0 \quad (13)$$

The aList ambient maintains the records of student attendance. Once CaWB confirms a valid student login, the student's ID is forwarded to aList, where a new child ambient is created for that ID. The attendance log thus reflects all authenticated students present in the lab. The behavior of P5 is specified in Eq. (14).

$$P5= !CaWB\uparrow rec(n).n[0].0 \quad (14)$$

The context reasoning mechanism of the CaLab system defines how environmental information is interpreted and how appropriate actions are determined using formal Context Calculus Approach (CCA) rules. The system continuously evaluates contextual data such as user identity, role, and environmental state represented as formal CCA facts, allowing it to derive higher-level contextual knowledge and make intelligent decisions. For instance, when a lecturer enters the laboratory, the Context-Aware Whiteboard (CaWB) acquires the user's institutional ID and generates the CCA fact User (lecturer, present). This fact, together with existing context knowledge (LabOccupied = FALSE), triggers a reasoning sequence based on CCA rules. This approach ensures that the CaLab system not only reacts to immediate events but also maintains a formally verified, intelligent, and adaptive operation under all conditions, directly linking contextual facts to decision logic through CCA.

IF User (lecturer, present) AND LabOccupied = FALSE

THEN

authorize_access(CaWB) AND activate_computers() AND
load_lecture_materials() AND adjust_environment(lighting=ON)

Validation and Implementation

The UML Sequence Diagram is employed to model the dynamic interactions and message exchanges among the components of the Context-Aware Laboratory (CaLab) system. Each diagram represents a specific scenario within the system, such as attendance detection, instructor authentication, or environmental control. The visualization of these scenarios, the diagram helps to clarify how contextual events trigger operations and how responses transfer through the different system layers. This form of modelling also assists in validating the logical sequence of interactions defined in the formal specification, ensuring consistency between conceptual and architectural designs [9].

The implementation is conducted as a software-based simulation rather than a physical deployment. This section details the implementation strategy, development environment, mapping between CCA constructs and executable components, and the realization of core system modules. In addition, illustrative execution scenarios are presented to demonstrate the correct operation of the system under typical laboratory conditions. The CaLab system is simulated using Python due to its expressive syntax, rapid prototyping capabilities, and suitability for modelling event-driven systems [10]. Python allows the abstraction of physical entities such as sensors and devices into logical objects while preserving system behavior [11]. A scenario-based functional testing approach is adopted to assess the system under realistic operational conditions. Each test scenario represents a meaningful sequence of contextual events, such as user entry, authentication, resource allocation, or laboratory exit. These scenarios are derived directly from the formal CCA rules and the functional requirements to provide traceability between specification and simulation [12].

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

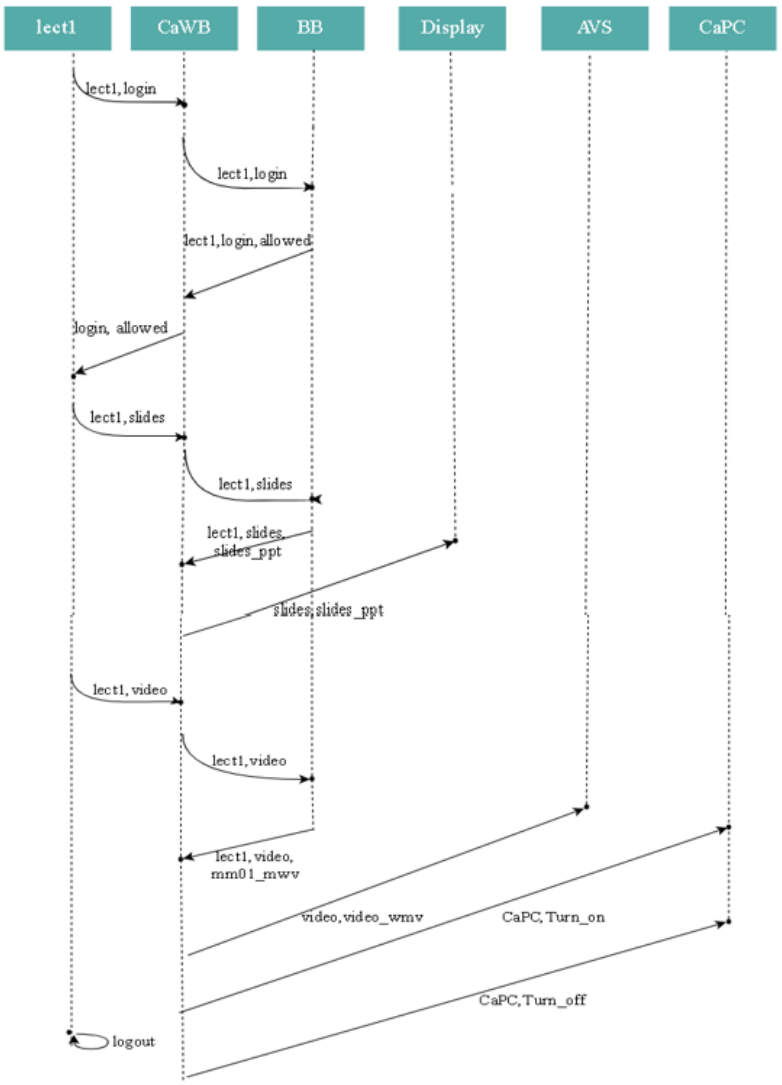

Fig.3. Sequence Diagram of Instructor-specific Context Delivery

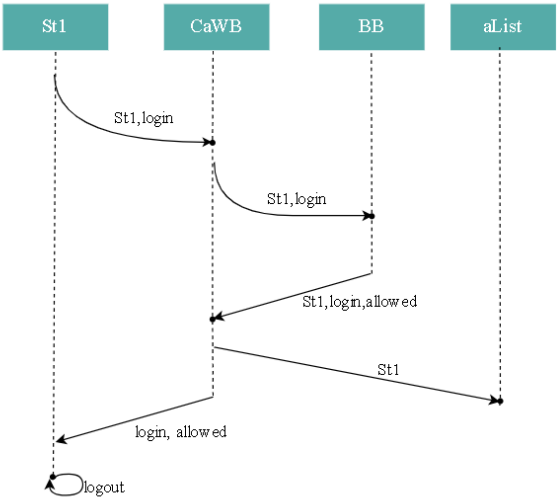

Fig.4. Sequence Diagram of Automated Attendance Logging

Scenario 1: Instructor Entry and Session Initialization

This scenario evaluates the system's ability to correctly recognize the instructor's context and initialize a teaching session. The system verifies the authorization of entered instructors against the internal knowledge base. (Fig.3) Illustrates the sequence of interactions that occur when a lecturer enters the laboratory. After detecting the lecturer's ID, the CaWB initiates a login request to the BB, which validates the lecturer's identity and authorizes access. Once authenticated, CaWB triggers the Display and AVS modules to load the corresponding lecture materials automatically, such as presentation

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

slides, tutorials, audio, or video resources. At the same time, CaPC ensures all workstations are ready to power on for attended students. The environmental control subsystem adjusts lighting for optimal classroom conditions, as shown in (Fig. 5). This automation ensures that each session begins efficiently and with minimal setup effort, aligning the environment and resources with the authenticated instructor's context.

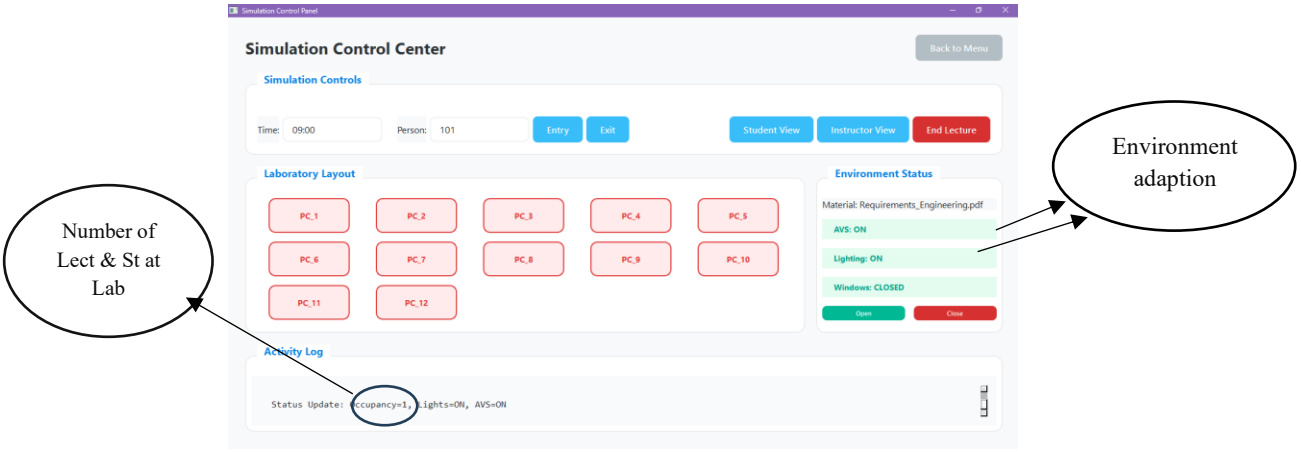

Fig.5. Authorized Instructor entry and session initialization

Scenario 2: Student Entry and Automated Attendance Recording

The scenario validates the system's automated attendance mechanism and its ability to allocate laboratory resources dynamically. As shown in (Fig.4), A registered student enters the laboratory; the system correctly validates authorization and records attendance automatically. We assume that the authenticated instructor in the lab, a student carrying a valid institutional ID, enters the lab; the CaWB detects the user's ID and sends an authentication request to the BB, which confirms the user's identity and returns an authorization message. The CaWB forwards the verified student ID to the aList module, which records the student's attendance. The CaPC module is instructed to activate the computers assigned to the detected users as shown in (Fig. 6). This process eliminates manual attendance recording, ensuring accuracy and immediacy while maintaining synchronization between user authentication and resource activation.

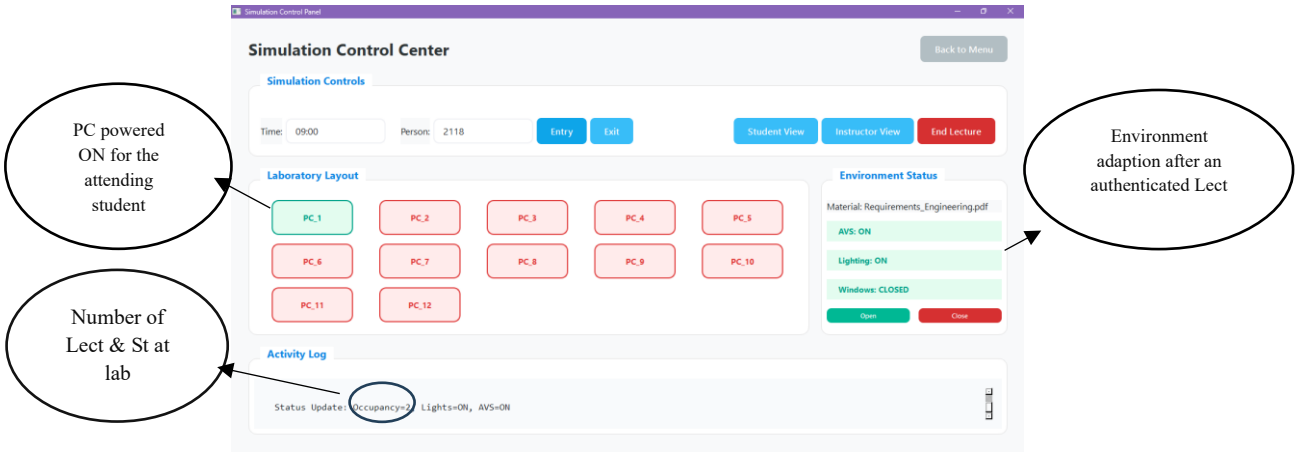

Fig.6. Automated attendance logging and device allocation

Scenario 3: Instructor Entry Outside Scheduled Session/ Student Entry Outside Assigned Session

The first part of this scenario tests the system's ability to handle a valid user (Instructor) attempting to access the laboratory outside their scheduled teaching time. Once the instructor's entry is made, the system successfully authenticates the user. However, the context engine detects a mismatch between the current time and the instructor's schedule. The system grants general access but displays a notification on the Instructor Dashboard as depicted in (Fig.7), and a dedicated screen stating: "Welcome, Dr. [Instructor Name]. You are not scheduled for a session at the current time, as shown in (Fig.8).

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

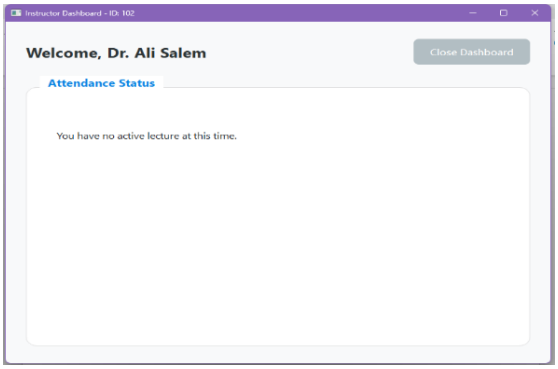

Fig.7. System notification for instructor on Dashboard

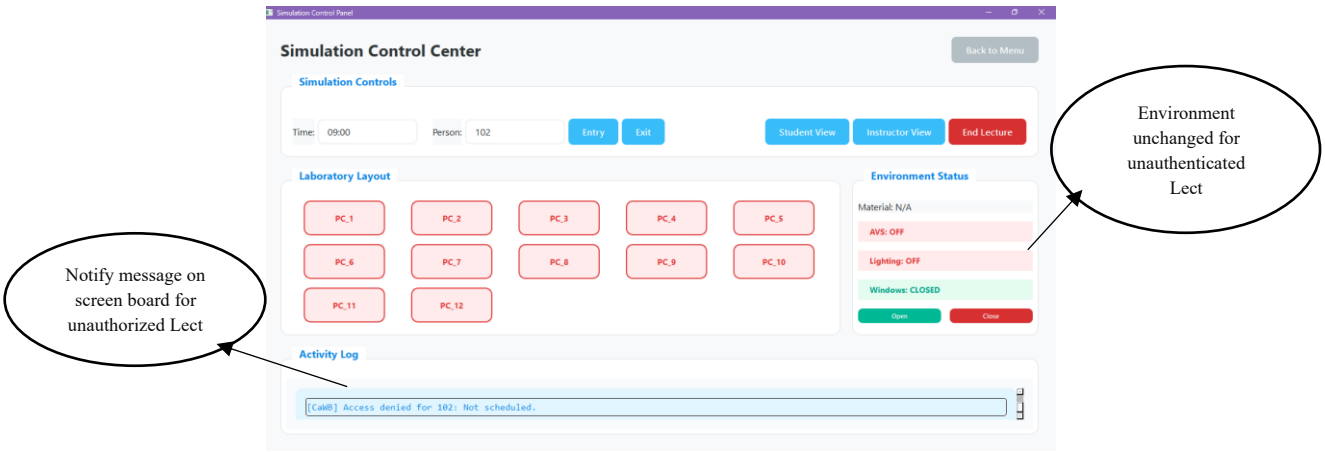

Fig.8. System notification for the instructor on the screen board

The second part of the scenario evaluates the CaLab system's ability to handle a contextually invalid access attempt by a student (outside their assigned lecture time**)** who attempts to enter the laboratory. The student enters the laboratory during an active session, and the CaWB identifies the student and verifies their identity through the BB. Although the student is a registered user, the scheduling context indicates that the student is not assigned to the lecture at the current time. The system does not grant access to the active session, and no attendance record. Laboratory devices remain powered off, and environmental settings remain unchanged as fig.9 shown. In addition, the system displays a notification informing the student that access is restricted due to the absence of an assigned session at the current time, as (Fig.10) displayed.

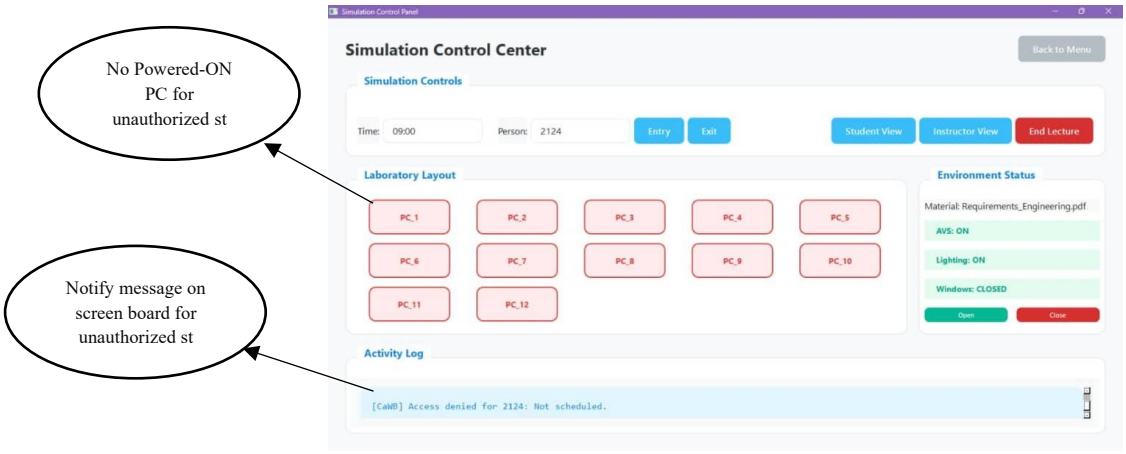

Fig.9. The system dedicated to unassigned students

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

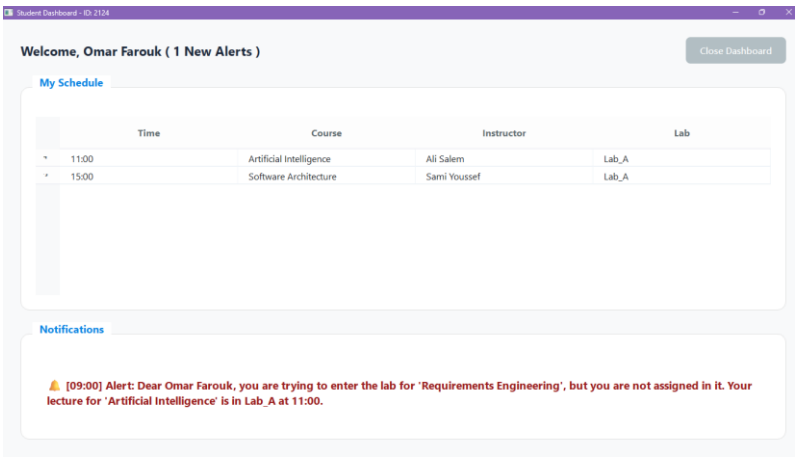

Fig.10. System Notification for unauthorized student on Dashboard

Scenario 4: Student exits active session and Resource Reorganization

The following scenario examines how the system reacts when a student leaves the lab during a session. (Fig.11) shows that the session was initialized with two assigned students with instructor. Once the student's departure is detected, CaWB updates the attendance list to indicate the time of the student's leaving, as depicted in (Fig.12). The system turns off the assigned PC for the left student, and environmental settings continue to be unchanged, as (Fig. 13) displayed. Moreover, the planned laboratory session continues undisturbed for the remaining users.

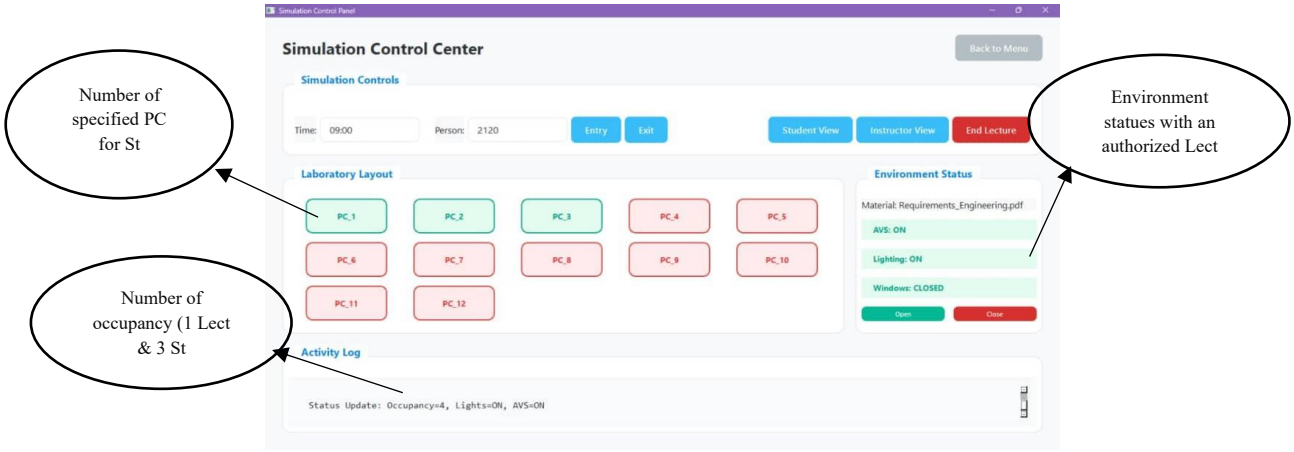

Fig.11. Environment and PCs status before student exit

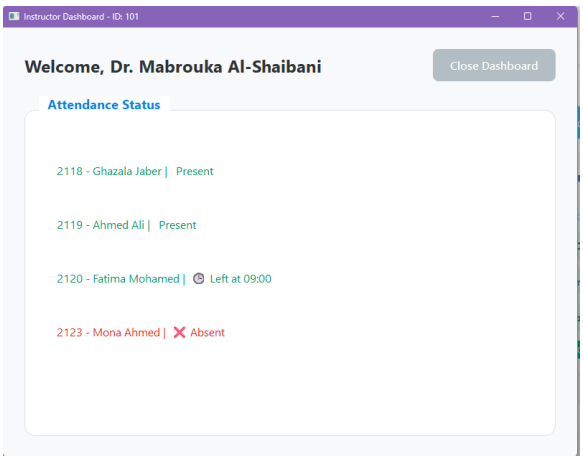

Fig.12. Attendance list shows time of leaving

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

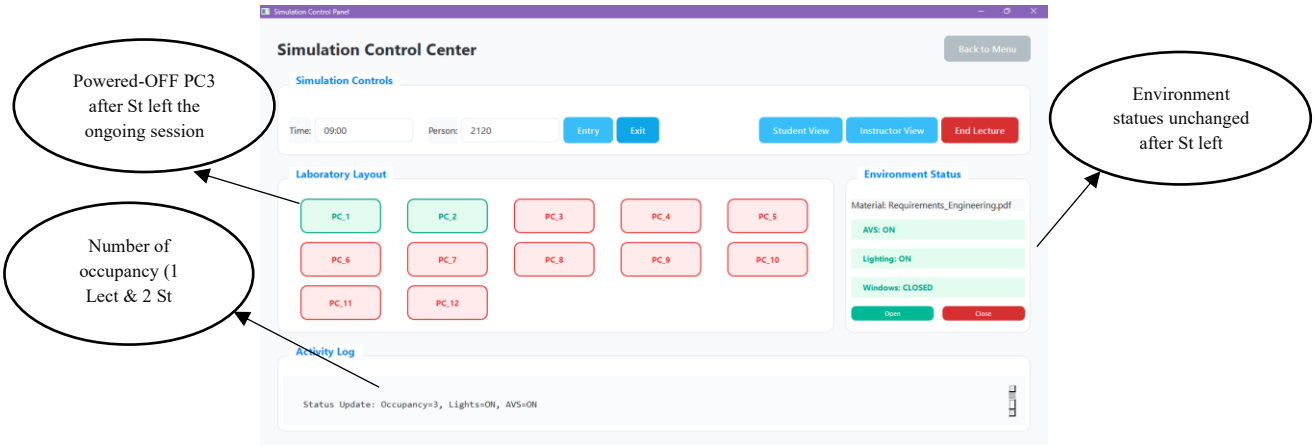

Fig.13. Environment and PCs status after student exit

Scenario 5: session Termination and final report

The evaluation of how the system behaves after a session terminates and the lab is completely empty is presented in this scenario. The CaLab system detects the leaving of all users once the last user has left the lab, following a brief verification period, then the computers are turned off, and the lighting is automatically turned off. The result is shown in (Fig.14). Once the session is done and all users have left the laboratory, the checker ensures the environmental conditions in the lab, such as light and windows. The CaLab system generates the final report at the end of each session; includes a technical report and an attendance report. The technical report provides the current state of light and windows, the time of the session, the label of the lab, the number of operated PCs, and is submitted to the administrative interfaces room for review. The attendance report details the attendance list of the module is sent to the instructor at the end of each session. The final report is indicated in (Fig.15).

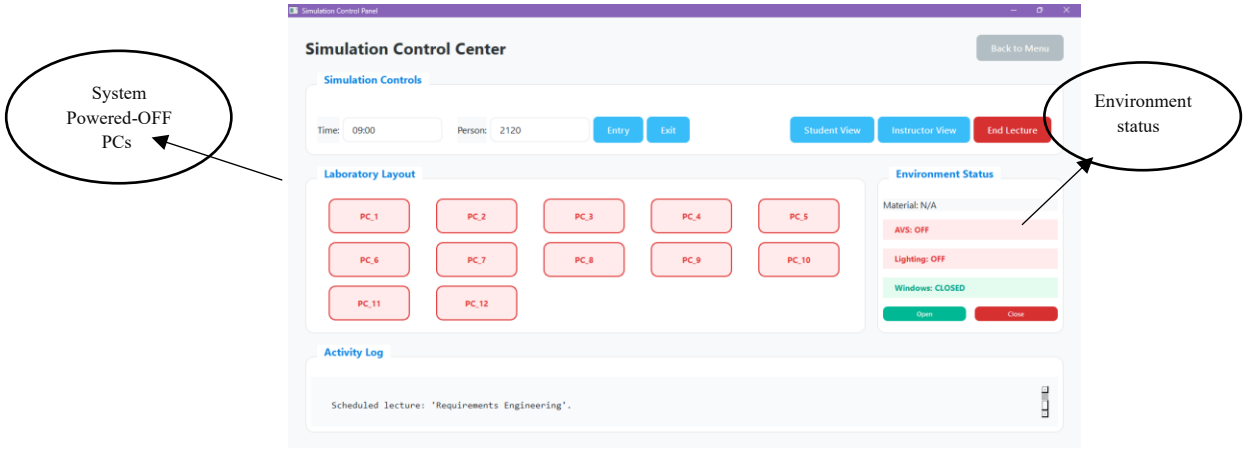

Fig.14. System shutdown after session terminated

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*

Fig.15. System final report after session terminated

Discussion, Conclusion, and Future Work

This work introduced the formalization of a context-aware laboratory system in CCA. The specification of the CaLab system was simulated using the execution environment of Python, based on scenario-based testing. The CaLab system was designed as a framework capable of intelligently adapting to user identity, environmental conditions, and activity context. The architectural design was realized as a complete software simulation using the Python programming language, featuring a graphical user interface for visualization. This phase successfully illustrated that the formal rules can be translated into functional code that responds dynamically to contextual events. The Testing Phase involved evaluating the system through a series of realistic, scenario-based tests that simulated user entry and exit under various conditions. The results demonstrated the system's robustness and adaptability in managing various laboratory scenarios, confirming its potential for enhancing operational efficiency and user experience in smart laboratory environments. In terms of future work will focus on expanding the system to include more complex contextual rules and real-world deployment. Using ccaRV tools for formal verification [13]. This tool can be used during the development of a context-aware system to validate that the system design expressed in CCA satisfies the specification of the system before the implementation in a high-level programming language.

References

1. Weiser M. The computer for the twenty-first century. Sci Am. 1991;265(3):94–104.
2. Al-Sammarraie MH. Policy-based approach for context-aware systems [doctoral dissertation]. Leicester (UK): De Montfort University; 2011.
3. Glushkova T, Ivanova V, Zlatanov B. Beyond traditional assessment: A fuzzy logic-infused hybrid approach to equitable proficiency evaluation via online practice tests. Mathematics. 2024;12(3):371.
4. Siewe F, Germanos V, Zeng W. Mapping Petri nets onto a calculus of context-aware ambients. Software. 2024;3(3):284–309.
5. Atbaiga NA, Siewe F. Formal specification of a context-aware whiteboard system in CCA. In: Proceedings of the International Conference on Pervasive Systems and Computing (LICEET 2018); Tripoli, Libya; 2018.
6. Siewe F, Al-Alshuhai A, Almutairi S, Almutairi A. Analyzing use case diagrams in a calculus of context-aware ambients. Int J Intell Comput Res. 2016;7(1). doi: 10.20533/ijicr.2042.4655.2016.0081.
7. Richards M, Ford N. Fundamentals of software architecture: An engineering approach. Sebastopol (CA): O'Reilly Media; 2020.
8. Siewe F, Zedan H, Cau A. The calculus of context-aware ambients. J Comput Syst Sci. 2011;77:597–620.
9. Sommerville I. Software engineering. 11th ed. London (UK): Pearson; 2021.
10. Ramalho L. Fluent Python. 2nd ed. Sebastopol (CA): O'Reilly Media; 2022.
11. Fitzpatrick M. Create GUI applications with Python & Qt6. 6th ed. Sebastopol (CA): O'Reilly Media; 2022.
12. Al-Sammarraie A, Ahmed M, Al-Dahash S. Automated environmental control in academic institutions: Energy savings and device longevity. J Sustain Comput. 2020;15:100–12.
13. Siewe F. Runtime verification tool for the calculus of context-aware ambients. Mathematics. 2025;13(22):3606.

Received*: 18-05-2026 -* ***Accepted****: 16-07-2026 -* ***Published****: 21-07-2026*