applied sciences

MDPI

Review

Literature Review of Software Architectures for Developing Artificial Intelligence-Based Mobile Adaptive Learning Systems

Enrique Duran Granados [1], Eduardo López Domínguez [2,*], Saúl Domínguez Isidro [3,*], María Auxilio Medina Nieto [4], Juan Irving Vasquez-Gomez [5] and Jorge de la Calleja [4]

[1] Sección de Estudios de Postgrado e Investigación, Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del Instituto Politécnico Nacional (IPN), Ciudad de México 08400, Mexico; edurang2301@alumno.ipn.mx
[2] Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Ciudad de México 07360, Mexico
[3] Faculty of Statistics and Informatics, Universidad Veracruzana, Veracruz 91020, Mexico
[4] Postgraduate Department, Universidad Politécnica de Puebla, Puebla 72640, Mexico; maria.medina@uppuebla.edu.mx (M.A.M.N.); jorge.delacalleja@uppuebla.edu.mx (J.d.l.C.)
[5] Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC), Instituto Politécnico Nacional (IPN), Ciudad de México 07700, Mexico; jvasquezg@ipn.mx
* Correspondence: eduardo.lopez.dom@cinvestav.mx (E.L.D.); sauldominguez@uv.mx (S.D.I.)

Abstract

The widespread adoption of mobile devices has driven the development of diverse educational technologies. In this context, Artificial Intelligence-based Mobile Adaptive Learning Systems (AI-based MALS) have emerged as an efficient alternative to complement the teaching and learning process. However, the advancement of these systems is hindered by significant challenges such as insufficient software architectural support, lack of standardization, and inadequate requirements coverage and correctness. Addressing these challenges requires designing high-quality software architectures. Accordingly, research focused on the design, development, and evaluation of software architectures for AI-based MALS is critical from a software engineering standpoint. Although some secondary studies have explored related topics through literature reviews and systematic mapping, no comprehensive secondary review has identified the principal architectural aspects of AI-based MALS design. This study conducts a Literature Review (LR) of software architectures that integrate Artificial Intelligence into Mobile Adaptive Learning Systems. The main objective is to identify and compare the critical aspects of software architectures for proposed AI-based MALS in the specialized literature. The findings highlight several essential elements that software architectures for AI-based MALS should address, including requirements, quality attributes, stakeholders, user services, architectural views and styles, components, and implementation technologies. The primary finding of this study is the absence of a reference software architecture in the reviewed studies that comprehensively addresses the requirements needed to develop AI-based MALS to support adaptive learning in mobile distributed environments.

Keywords: software architectures; mobile adaptive learning systems; artificial intelligence; architectural design; reference software architecture

Academic Editor: César A. Collazos

Received: 28 August 2026
Revised: 26 October 2026
Accepted: 26 October 2026
Published: 28 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The widespread use of mobile devices has led to the development of educational technologies that help students build academic skills beyond the classroom [1–3]. In this context, Artificial Intelligence-based Mobile Adaptive Learning Systems (AI-based MALS) have emerged as an efficient alternative to complement the teaching and learning process [4–6]. AI-based MALS are software tools designed to provide a unique teaching experience for students, thereby personalizing their teaching-learning process [4–7]. From a computational perspective, AI-based MALS are distributed software systems that use artificial intelligence to dynamically personalize learning experiences on mobile devices. These systems adapt educational content, learning paths, interaction mechanisms, and feedback based on individual learners' needs and progress. By combining mobile learning, adaptive learning, software engineering, and AI techniques, MALS extend beyond conventional one-size-fits-all approaches to deliver personalized learning experiences tailored to the characteristics and constraints of smartphones. AI-based MALS offer several significant advantages for students and teachers, including individualized instruction, real-time adaptive feedback, increased flexibility, personalized teacher support, and enhanced identification of learning needs through targeted strategies. Several studies [8–26] have proposed AI-based MALS that dynamically adjust the teaching-learning process to each student's specific characteristics, skills, aptitudes, and preferences. However, significant challenges hinder the advancement of these systems, including insufficient software architectural support, a lack of standardization, and inadequate requirement coverage and correctness. Addressing these challenges requires designing high-quality software architectures. Therefore, research on the analysis, design, development, and evaluation of software architectures for developing AI-based MALS is essential. In this context, several studies [27–32] have conducted systematic literature reviews and systematic mapping studies on AI-based MALS. These secondary studies have primarily examined conceptual aspects, including contextual variables, pedagogical models, psychological dimensions, and AI-based personalization. However, these reviews do not provide a detailed characterization or comparison of the architectural mechanisms needed to implement adaptive learning models in mobile distributed environments. It is noted that no study has presented a literature analysis that identifies the key aspects that AI-based MALS must consider in the design of their software architectures regarding types of adaptation, adaptive approaches implemented, and artificial intelligence techniques included; this makes it difficult to evaluate the quality of the proposed software architectures and identify trends, areas of opportunity, and future dimensions. The research clearly lacks a comprehensive body of knowledge on developing AI-based MALS. Therefore, this paper presents a Literature Review (LR) on software architectures for developing AI-based MALS. The main objective of this study is to identify and compare the critical aspects of software architectures of proposed AI-based MALS in the specialized literature. We conducted a comprehensive search across major scientific databases to identify primary studies published in English that describe and evaluate software architectures supporting AI-based adaptive learning in mobile distributed systems. Based on the primary work selection process, we identified nineteen primary studies relevant to this work. We analyzed and compared these studies based on the requirements considered, the architectural styles and views used, the evaluation approaches, and the aspects evaluated in the proposed software architecture. The LR analysis identifies recurrent architectural elements in AI-based MALS, such as functional requirements, quality attributes, involved stakeholders, core user services, architectural views and styles, supporting technologies, and AI techniques. However, despite identifying these elements, a central finding of this study is the absence of a reference software architecture within the reviewed studies that systematically integrates them to support the development of AI-based MALS. This work is structured as follows:

https://doi.org/10.3390/app162110671

Section 2 presents an overview of related work. Section 3 describes the literature review method, and Section 4 reports the results from analyzing each study. Section 5 includes the discussion. Section 6 describes possible threats to validity. Finally, Section 7 presents the conclusions and outlines future work.

2. Related Works

Several studies have conducted LRs and mapping studies related to MALS. These studies are described below.

Vallejo-Correa et al. [27] conducted an LR to identify context-sensitive variables in mobile and ubiquitous learning between 2010 and 2019. Their study categorized contextual information into internal, external, and activity variables. Internal variables describe learner characteristics; external variables are related to the mobile context and device attributes; and activity variables capture the interaction between the learning environment and the services provided. This group reflects specific activities within the virtual learning environment's services, including usability, structure, and service quality. Although the study highlights how modern mobile devices have evolved contextual variables, it does not examine software architectures or their implications for system design. As future work, the authors propose developing an omnichannel architecture for mobile-learning-based educational environments; this concept offers a direction for enhancing education delivery via mobile devices [33]. The architecture should integrate academic, social, technological, and environmental variables.

On the other hand, Hsu et al. [28] conducted a review that categorizes and synthesizes models and frameworks for mobile learning; seventeen models were examined in their analysis, and they were divided into five categories: (i) platform and system design; (ii) pedagogical and learning environment design; (iii) technology adoption; (iv) assessment; and (v) psychological dimensions. The authors highlight that, rather than providing technical specifications for implementing mobile learning systems, these models primarily offer theoretical or pedagogical guidelines for creating mobile learning experiences. Learner-device interaction, system needs, and motivational or affective elements were considered in their classification. Moreover, the authors argued that mobile learning research would benefit from updated, more comprehensive analyses, particularly given the rapid evolution of mobile technologies. Although their work offers a broad conceptual foundation, it does not address how these models could be operationalized within AI-based MALS or how AI-based adaptivity could be supported at the architectural level.

Gumbheer et al. [29] investigated Personalized Adaptive Context-Aware Mobile Learning (PACAML), focusing on the contextual information needed to support personalization, the technological infrastructure required to capture and process that context, and the educational settings where such systems can be applied. The authors distinguish between intrinsic context, such as the learner's knowledge level, motivation, learning style, and cognitive characteristics, and extrinsic context, which reflects dynamic environmental factors that influence the learning experience and vary throughout time. The study underscores that contextual adaptation requires appropriate hardware and software resources, including sensors, mobile device capabilities, network bandwidth, and supporting system infrastructure such as cloud services. The authors identified four essential abstraction layers for PACAML systems: (i) the application layer, (ii) the context detection layer, (iii) the content adaptation layer, and (iv) the knowledge layer. Furthermore, the authors noted that existing mobile learning applications rarely incorporate learners' characteristics into context adaptation, which limits their effectiveness. Although the study provides a detailed conceptual model and highlights infrastructure requirements, it does not propose a reference software architecture or explain how the identified abstraction layers could be composed into distributed mobile learning systems.

https://doi.org/10.3390/app162110671

Kabudi et al. [30] conducted a systematic mapping study of AI-enabled adaptive learning systems to characterize the state of research on intelligent educational platforms between 2014 and 2020. The mapping analyzed 147 primary studies and identified major intervention categories, including systems, frameworks, models, approaches, and hybrid solutions. The authors found that adaptive learning systems and intelligent tutoring systems were the most frequently implemented interventions, with AI techniques used for personalizing feedback, sequencing content, and adjusting task difficulty. Despite growing interest in AI-based adaptivity, the review notes that few systems are deployed in authentic educational settings and that implementation remains largely experimental or conceptual. Furthermore, the mapping highlights an imbalance between conceptual frameworks and operational solutions that can address concrete educational challenges. The authors conclude that future work should bridge the gap between AI research and real educational deployment, particularly by integrating pedagogical concerns with data-driven adaptivity pipelines.

Similarly, Kumar et al. [31] examined the ecosystem of context-aware learning solutions, focusing on contextual dimensions, learning scenarios, and application domains. The authors categorized context into physical, learner-related, temporal, and device-centered factors, emphasizing that mobile learning environments increasingly rely on multi-source context acquisition through sensors, ambient data, and learner interaction logs. Although the mapping finds that context-awareness enhances personalization and situational relevance, it also reports a lack of standardized architectural models that can compose contextual data sources, adaptation rules, and mobile delivery channels. Moreover, most analyzed systems use isolated context variables, limiting adaptivity and constraining scalability when integrating into broader distributed learning infrastructures.

Finally, Gligorea et al. [32] analyzed AI-based adaptivity mechanisms deployed within e-learning environments, highlighting machine learning and data-driven user modeling as central enablers of personalization. The review distinguishes between cognitive, behavioral, and performance-based learner models, and surveys techniques such as clustering, classification, reinforcement learning, and recommender strategies for predicting learner needs and adjusting learning resources. However, while the review reports promising outcomes in learner performance and engagement, it stresses that existing systems seldom incorporate multimodal signals such as affective, physiological, and situational signals, nor address architectural concerns such as interoperability, data orchestration, and deployment at scale. The authors argue that research should move toward holistic models of adaptivity that integrate AI, learner context, and instructional design into operational workflows rather than treating them as isolated algorithmic components.

Table 1 summarizes existing secondary studies related to MALS by comparing their research scope, conceptual dimensions, architectural/technical considerations, and the gaps they identify in the literature.

Table 1. Overview of systematic reviews and mapping studies related to MALS.

Study	Research Scope	Conceptual Dimensions Considered	Architectural/Technical Considerations	Identified Gaps
[27]	Context-sensitive variables in mobile/ubiquitous learning (2010–2019)	Internal, external, and activity contextual variables	Mentions about the integration of academic, social, technological, and environmental variables for future architectures	This study does not examine software architectures or technical implications for system design
[28]	Models and frameworks for mobile	Platform/system design; pedagogical design; technology	Models provide conceptual/ pedagogical	Lacks discussion on operationalization in MALS and role of

	learning (17 models)	adoption; assessment; psychological dimensions	guidelines rather than implementation specifications	AI-based adaptivity at the architectural level
[29]	Personalized Adaptive Context-Aware Mobile Learning (PACAML)	Intrinsic vs. extrinsic context; application; context detection; content adaptation; knowledge layers	It defines abstraction layers and supporting hardware/software resources (sensors, network, cloud)	No reference software architecture proposed; unclear composition of layers for distributed systems
[30]	AI-enabled adaptive learning systems (2014–2020)	Systems, frameworks, models, approaches, hybrid solutions	Notes lack of real deployment; highlights experimental nature of implementations	Gap between conceptual AI research and authentic educational deployment; limited to pedagogical–adaptive
[31]	Context-aware learning solutions	Physical, learner, temporal, device-centered context	Notes multi-source context acquisition (sensors, logs, ambient data)	Absence of standardized architectural models; isolated context variables limit scalability and interoperability
[32]	AI-based adaptivity mechanisms in e-learning	Cognitive, behavioral, performance-based user models	It discusses data-driven user modeling but neglects architectural orchestration or deployment aspects	Lack of multimodal signals (affective, physiological, situational); need for holistic, scalable architectural workflows

Previous secondary studies on adaptive learning have predominantly focused on conceptual aspects such as contextual variables, pedagogical models, psychological dimensions, or AI-based personalization. However, these reviews neither characterize nor compare the architectural mechanisms required to operationalize such models in mobile distributed environments. In contrast, our review analyzes primary studies proposing AI-based MALS architectures and characterizes them by functional requirements, architectural views, quality attributes, stakeholders, evaluation strategies, deployment technologies, and implemented AI techniques.

3. Method

Following the guidance of Kitchenham et al. [34], an LR was conducted in this work; this included the following steps: (1) review planning, which entails defining the review protocol; (2) review conduction, which concerns finding and choosing primary studies, extracting, and managing data; and (3) review reporting, which involves presenting and publishing the results.

3.1. First Phase: Review Planning

This section describes the study's objective, research questions, search strategy, selection criteria for filtering primary studies, criteria for evaluating results, data extraction strategy, and strategy for synthesizing extracted data.

3.1.1. Review Objectives and Research Questions

This LR aims to identify and compare software architectures reported in the literature for developing AI-based MALS. To address this objective, the following research questions (RQs) are defined:

- RQ1: What are the functional and non-functional requirements that must be considered for developing AI-based MALS?
- RQ2: What software architectures have been proposed for developing AI-based MALS?

- RQ3: What approaches have been used to evaluate the software architectures of AI-based MALS?
- RQ4: What quality attributes have been considered in AI-based MALS?
- RQ5: What areas of study and educational levels have been addressed in AI-based MALS?
- RQ6: How is AI being implemented in the proposed software architectures?

3.1.2. Search Strategy

This study considered a wide range of digital sources, including ACM Digital Library, IEEE Xplore, Springer, Wiley, Taylor & Francis, MDPI, ScienceDirect, and Google Scholar. The search terms across these databases included "software architecture", "models", "systems", "adaptive learning", "mobile learning", "machine learning", "artificial intelligence", and related keywords. These terms were based on the keywords from the research questions proposed in this document, considering three different research areas: (1) mobile learning as the primary area, (2) mobile adaptive learning systems as a sub-research topic, and (3) software architecture as a specified category. Therefore, the main search string patterns used were as follows:

- ("SOFTWARE ARCHITECTURE" OR "FRAMEWORK" OR "EDUCATIONAL PLATFORM" OR "MODEL") AND ("ADAPTIVE" OR "CONTEXT AWARE") AND ("M-LEARNING" OR "MOBILE-LEARNING" OR "SMART LEARNING" OR "UBIQUITOUS LEARNING") AND ("MOBILE LEARNING SYSTEM" OR "MOBILE LEARNING APPLICATION" OR "SMART MOBILE LEARNING SYSTEM")
- ("SOFTWARE ARCHITECTURE" OR "FRAMEWORK" OR "EDUCATIONAL PLATFORM" OR "MODEL") AND ("ADAPTIVE" OR "CONTEXT AWARE") AND ("M-LEARNING") AND ("MOBILE")
- ("SOFTWARE ARCHITECTURE" OR "SYSTEM ARCHITECTURE" OR "ARCHITECTURAL MODEL" OR "FRAMEWORK" OR "SOFTWARE DESIGN") AND ("ADAPTIVE LEARNING" OR "PERSONALIZED LEARNING" OR "PERSONALISED LEARNING" OR "CONTEXT-AWARE LEARNING") AND ("MOBILE LEARNING" OR "M-LEARNING" OR "UBIQUITOUS LEARNING") AND ("MACHINE LEARNING" OR "DEEP LEARNING" OR "NEURAL NETWORK" OR "SUPERVISED LEARNING" OR "UNSUPERVISED LEARNING" OR "ARTIFICIAL INTELLIGENCE")

The search strings were adapted to the syntax and constraints of each digital library.

3.1.3. Selection Criteria

To select the primary studies, four inclusion criteria (IC) and five exclusion criteria (EC) were defined.

Inclusion criteria:

- IC1. Papers written in English, published in indexed and/or peer-reviewed scientific journals and conferences.
- IC2. Studies in the field of software engineering that address software architectures for AI-based MALS.
- IC3. Publications that report a detailed case study, evaluation, and a practical application involving software architectures used to support AI-based MALS.
- IC4. Studies from 2015 to 2026.

Exclusion criteria:

- EC1. Short papers, experience reports, workshop abstracts, or any document that lacks the level of detail required for a literature review.

- EC2. Documents in software engineering that do not present a model, framework, or software architecture oriented towards the development of AI-based MALS.
- EC3. Documents that match the search string pattern syntactically, but that are not related to the research topic.
- EC4. Studies that offer limited contribution to the research objectives or do not provide substantive information for answering the research questions. For example, papers do not evaluate the proposed software architecture.
- EC5. Studies that do not satisfy the inclusion criteria (IC1–IC4).

3.1.4. Quality Assessment Criteria

This literature review uses the quality criteria defined by Dybå and Dingsøyr [35]: rigor, credibility, and relevance. These quality criteria were evaluated in each primary study identified based on the questions defined in [36]:

1. Is there a clear statement of the research goals and objectives?
2. Is there an adequate description of the context in which the research was conducted?
3. Is there an adequate description of the proposed contribution, method, or approach?
4. Is there a clear statement of the findings?
5. Is the evidence obtained from experimental or observational studies?
6. Does the study have value for research or practice?

This assessment of the quality of the identified primary studies complements the inclusion and exclusion criteria described in Section 3.1.3.

3.1.5. Data Extraction and Synthesis Strategy

We thoroughly analyzed each study, extracted relevant data, and recorded information in forms designed to capture key concepts related to each study's objectives, results, and conclusions. In our case, we guided the data extraction and synthesis strategies using the characterization scheme proposed by Plaza et al. [36]. This scheme considers functional and non-functional requirements, quality attributes, architectural views, architectural styles, component types, implementation technologies, and stakeholders.

3.2. Second Phase: Review Conduction

The following steps were carried out during the review conduction phase: searching for primary studies, selecting primary studies based on inclusion and exclusion criteria, evaluating quality, rigor, credibility, relevance, extracting and synthesizing data. Each step is detailed in the following sections.

3.2.1. Search and Selection of Primary Studies

The search for primary studies was conducted following the search strategy described in Section 3.1, using the digital libraries specified in Section 3.1.2. After applying the study selection process illustrated in Figure 1, a total of 19 primary studies were included in the review. The selection process began with a review of the title, keywords, abstract, and conclusions, yielding 848 primary studies. Subsequently, a two-phase analysis and selection process was initiated. In the first phase, primary studies that did not present a software model or architecture were excluded, leaving 111 studies. In the second phase, primary studies that lacked an evaluation approach for the proposed architecture and did not consider or use AI techniques in their development were excluded, leaving nineteen studies. Finally, one iteration of backward and forward snowballing was performed; however, no additional primary studies were found. Table 2 shows the number of studies considered in both phases.

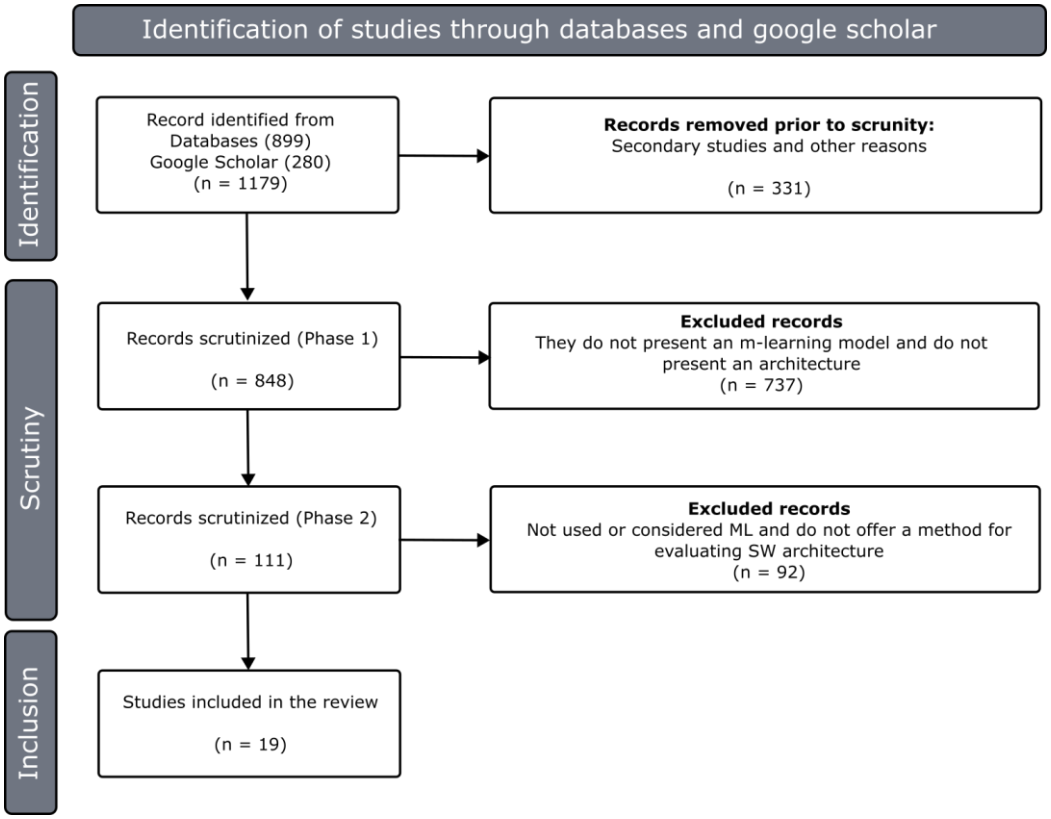


Figure 1. The primary study selection process used in this LR.

Table 2. Number of studies selected in the selection and search process.

Digital Source	Phase 1		Phase 2	
	Included	Excluded	Included	Excluded
ACM Digital Library	2	27	0	2
IEEE Xplore	8	18	1	7
Springer	37	212	6	31
Wiley	23	91	3	20
Taylor & Francis	4	101	1	3
MDPI	2	11	0	2
ScienceDirect	12	20	3	9
Google Scholar	23	257	5	18
Total	111	737	19	92
Studies considered per phase	848		111	
Studies included	19			

Each selected primary study was assigned a unique identifier to facilitate its organization and tracking throughout the review process and to distinguish it from other referenced publications (see Table 3).

Table 3. References (Ref.) and Identifiers (Id) of the studies included in the review.

Id	Year	Title	Ref.
P1	2023	A Universal Design for an Adaptive Context-Aware Mobile Cloud Learning Framework Using Machine Learning	[8]
P2	2022	Design of a Personalized Adaptive Ubiquitous Learning System	[9]
P3	2021	Leveraging the Power of Deep Learning Technique for Creating an Intelligent, Context-Aware, and Adaptive M-Learning Model	[10]
P4	2022	Mobile Learning New Trends in Emerging Computing Paradigms: An Analytical Approach Seeking Performance Efficiency	[11]
P5	2022	Mobile Learning Evolution and Emerging Computing Paradigms: An Edge-based Cloud Architecture for Reduced Latencies and Quick Response Time	[12]
P6	2024	Implementation of a Personalized Adaptive Mobile Learning System	[13]
P7	2019	Cloud-supported Machine Learning System for Context-aware Adaptive M-learning	[14]
P8	2025	Adaptive M-Learning Content on the Moodle Platform Using M-Learner Modeling Approach, Ontologies, and Machine Learning Techniques	[15]
P9	2016	A model for learning objects adaptation in light of mobile and context-aware computing	[16]
P10	2017	Constructing a User-Friendly and Smart Ubiquitous Personalized Learning Environment by Using a Context-Aware Mechanism	[17]
P11	2016	A context-aware mobile system for work-based learning	[18]
P12	2017	Model of M-Learning by Multimedia Content Delivery from mCloud to Mobile Devices	[19]
P13	2016	Application of Semantic Web Technologies to Facilitate Use of E-Learning System on Mobile Devices	[20]
P14	2021	Mudahnya BM: A Context-Aware Mobile Cloud Learning Application Using Semantic-Based Approach	[21]
P15	2020	Design and implementation of a SMART Learning environment for the Upskilling of Cybersecurity professionals in Mauritius	[22]
P16	2025	A Containerized Microservices Architecture with Reinforcement Learning for Scalable, Adaptive Learning	[23]
P17	2024	Exploring the effects of personalized recommendations on students' motivation and learning achievement in gamified mobile learning framework	[24]
P18	2025	ECGTwinMentor: A digital twin-based system for personalized ECG learning and support	[25]
P19	2025	EduXgame: Gamified learning for secondary education	[26]

3.2.2. Quality Assessment of Primary Studies

The selected primary studies were evaluated using the quality assessment criteria defined in Section 3.1.4. Six criteria were considered: research clarity, contextual description, contribution, reported findings, type of evidence, and relevance to research or practice. Studies were required to satisfy all six criteria to be included in the review. Following this assessment, all 19 primary studies met the established quality threshold and were retained for analysis.

3.2.3. Data Extraction and Synthesis

Data were extracted from the selected primary studies using a predefined form to ensure consistency and traceability throughout the extraction process. The data extraction form is presented in Table 4.

Table 4. Data extraction form.

Extracted Data	Description
Study identity	Identifier (P1–P19).
Bibliographic reference	Authors, year of publication, title, and publication source.
Study type	Type of publication (journal paper, conference paper, book chapter).
Architecture characterization	Description of the proposed architecture following the outline presented by Plaza et al. [36].
Evaluation approach	Evaluation method used in the study (case study, controlled experiment).
Abstract	Summary of the primary study used to support contextual analysis.

4. Third Phase: Review Reporting

This section presents answers to the six research questions established in Section 3.1.1.

4.1. RQ1: What Are the Functional and Non-Functional Requirements That Must Be Considered for Developing AI-Based MALS?

Based on the study carried out in [33], the following functional requirements must be considered for developing AI-based MALS:

- Adaptation of learning path, format, and content;
- Obtaining, managing, and using context, student, and domain information;
- Generating student profiles and histories;
- Managing educational and technological resources.

On the other hand, according to [33,37], the following non-functional requirements and their associated quality attributes must be considered for developing AI-based MALS:

- Functional suitability: Functional completeness and correctness;
- Compatibility: Interoperability;
- Performance efficiency: Temporal behavior, resource utilization, capacity, and scalability;
- Security: Confidentiality, integrity, and availability;
- Usability: Efficiency, effectiveness, and satisfaction.

A detailed description of each of these functional requirements can be found in [33,37].

4.2. RQ2: What Software Architectures Have Been Proposed for Developing AI-Based MALS?

To address RQ2, this section synthesizes the software architecture proposals identified in the selected primary studies. Each proposal is analyzed in terms of its architectural characteristics, including the involved stakeholders, provided services, applied architectural styles and views, evaluation approaches, application domains and educational levels, quality attributes, and artificial intelligence techniques employed.

4.2.1. Stakeholders

The selected primary studies [P1–P10], [P12], [P13], [P14], [P17], [P18], and [P19] involve university-level students as participants in the development or evaluation of AI-based MALS; see Table 5. Studies [P11] and [P15] involve professionals as students, whereas study [P14] also involves learners aged 7–12 as users. Study [P16] involves

learners of Chinese as a second language. Additionally, several studies [P1], [P2], [P3], [P4], [P7], [P10], [P12], [P13], [P14], [P15], [P16], [P17], [P18], and [P19] report involving instructors during the development process.

Table 5. Stakeholders in the selected primary works.

Stakeholder	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19
Student	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Teacher	☑	☑	☑	☑	☐	☐	☑	☐	☐	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑
Developer	☐	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Researcher	☐	☐	☐	☑	☑	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Study [P4] explicitly identifies Information Technology (IT) personnel as supervisors of the development process. It emphasizes that developers should understand the technical characteristics and implementation requirements of resource-intensive mobile learning applications before designing the architecture. Furthermore, studies [P4] and [P5] implicitly consider researchers' role in proposing new software architectures or improving existing ones. Study [P8] follows a similar pattern by involving university students, researchers, and system designers responsible for constructing learner models and adaptive mechanisms.

4.2.2. Main Components Considered in Primary Studies

The study in [P1] offers context-adapted material in accordance with Universal Design for Learning principles. The system proposed in this work consists of components that detect context changes and adapt the mobile learning application's responses and behavior. In this regard, the system provides professors with a universal learning design to develop a flexible, accessible curriculum that offers all students equal opportunities, regardless of ability, age, gender, cultural background, or language. The system applies machine learning models to predict students' concentration levels and, combined with contextual factors, the learning interval, and knowledge level, offers a specific learning model.

Study [P2] explicitly uses context to provide material tailored to preferences or concentration levels. The study focuses on developing a Bayesian network for classifying learning styles. The system architecture proposed in [P2] comprises five modules: user interface, learner, adaptation, domain, and evaluation. The user interface module provides services such as login, location detection, and a pre-test. The learner module manages students' profile information. The adaptation module personalizes learning content and activities. The domain module manages educational resources and activities to support individual students. Finally, the evaluation module records student performance logs and tracks student behavior.

The study [P3] offers educators an adaptive-learning prototype that uses algorithms to orchestrate student interactions and deliver learning content tailored to each student's needs and performance. The study classified students using machine learning based on the weight of their learning characteristics. On the other hand, the study [P4] offers IT staff a mobile cloud architecture based on four components: (1) Devices—branded smartphones such as Lenovo K8 Note, with almost identical configurations and operating systems (Android v7.1.1), (2) Learning Management System (LMS)—educational material is managed by the educator through this system and is used by instructors and students, (3) Cloud stack for performance optimization consisting of Edge-Cloud, cloudlet, (two Dell PowerEdge T320 edge and cloudlet servers with Windows Server 2012) and (4) a mini data center for distance to the cloud.

The study [P5] offers IT staff a mobile edge computing architecture in which high-performance servers distribute workloads to the edge, enabling applications (Magic Plan -iOS/Android-) to execute tasks on mobile devices. On the other hand, the architecture in [P6] adopts an expert system as an adaptation engine, which uses recommendation algorithms, rule-based engines, and machine learning models to personalize learning experiences for students. The study in [P7] proposes a cloud-based machine learning system for adaptive, context-aware learning in computer programming. The system architecture consists of three layers: context acquisition, context analysis, and learning path generation. The context acquisition layer obtains context information from students' smartphones. The context analysis layer generates a mobile learning model for each student based on the obtained context information. This layer uses the unsupervised machine learning algorithm called Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to group students based on their geographic location. Finally, the learning path generation layer uses the context analysis layer's output to provide students with personalized guidance and learning content. This layer uses a rule-based inference engine to recommend personalized programming exercises to students.

In contrast, the study [P8] proposed an m-learning architecture integrated with Moodle, including a data collection component that extracts student traces from Moodle logs, quizzes, and mobile device characteristics. A preprocessing module then cleans, encodes, anonymizes, and normalizes the data. The processed dataset is analyzed using DBSCAN and K-means to cluster learners based on behavioral and contextual characteristics. Based on these clusters, a learner model is generated that represents profiles, behavior, context, emotional state, and performance. In addition, an ontology-based model represents course components and their relationships. Finally, an adaptive delivery component recommends personalized learning resources and activities.

The study in [P9] proposed an architecture called EduAdapt, composed of five components: an LMS, a repository of learning objects, a server, a push notifier service, and a student app. The main objective of this architecture is to adapt learning objects based on learning styles, device characteristics, and other student context information. To achieve this, students must be registered in a learning management system (LMS). EduAdapt identifies students' needs based on the information stored in the LMS. Afterward, the server uses the OAI-PMH protocol to retrieve metadata for learning objects stored in the learning object repository that meet the identified needs. The server then sends a notification to the app installed on the student's mobile device via the Push Notification Service component. When the student's app receives the notification, it retrieves the student's contextual information and preferences. This information is sent via web services to the server to be used with the OntoAdapt ontology to select and adapt the learning object appropriately. At this point, the server uses rules and inferences to determine the most appropriate learning object to send to the student and, if necessary, adapts it to the required medium or format. Finally, the student's app is notified to display the adapted learning object.

The paper presented in [P10] describes a system that provides personalized learning tailored to each student's context within a ubiquitous environment designed for English-language learning. The paper proposes a software architecture with three main subsystems: multimedia learning, context-aware learning, and mobile learning recommendations. The multimedia learning subsystem manages English courses, instructional materials, and students' learning behaviors to deliver multimedia content via a streaming server. Furthermore, the context-aware subsystem, based on GPS, QR code data preprocessing, and an input module, identifies and tags the relationship between learning content and the environment. Finally, the mobile learning recommendation subsystem, comprising

student authentication components and personalized mobile recommendation agents, suggests the most appropriate learning content for students at a given time and place.

In [P11], a context-aware mobile system for work-based learning called WoBaLearn is proposed. The system architecture consists of three layers: context sensing, learning adaptation, and learning application. The context sensing layer captures context information about the activity, devices, and environment via various virtual and physical sensors. The context information obtained is predefined in the WbML context model proposed in this work. The learning adaptation layer provides context-adapted learning supports, such as learning units, learning partners, and learning activities. This layer consists of two sublayers: context management and the learning engine. The context management sublayer collects, transforms, enriches, and stores the learning context. The learning engine sublayer uses the just-in-time learning context stored in the context knowledge base to execute adaptation strategies and generate adaptive learning supports. The learning application layer focuses on student-interaction tasks such as gathering information about students and their needs, presenting adaptive learning units, creating appropriate communication platforms between students and their learning partners, and guiding students to complete relevant learning activities.

The study presented in [P12] proposed a system for delivering multimedia learning content to users based on their cognitive styles and context-aware network conditions. This system comprises the following main components: an mCloud interface, social networks, a digital library, and an adaptation engine. mCloud receives requests from users' mobile devices and performs the most processing-intensive tasks to reduce the workload on those devices. The system also determines the user's cognitive style based on their social media profiles, such as Facebook, X, and LinkedIn. The system also continuously monitors communication channels to prepare user-tailored content. The digital library sends the selected multimedia content to the content adaptation engine, which adapts it and sends it to the user's mobile device.

The paper proposed in [P13] describes the semantic web architecture of a system called Protus 2.1. This architecture mainly consists of the following components: domain ontology, learner model ontology, task ontology, teaching strategy ontology, and learner interface ontology. In this architecture, the learner interface ontology component adapts the tutoring system's user interfaces for mobile devices based on their characteristics. Based on the teaching strategy ontology's decision, the system proposed in this paper creates a presentation tailored to the specific student and device. It generates a student interface consistent with the selected design. Subsequently, the learner interface ontology specifies page content and layout and standardizes the content and query vocabulary.

The study [P14] proposed a context-aware mobile cloud learning system using a semantic-based approach. The software architecture proposed in this work consists of three main subsystems: mobile application, web server, and service provider. The mobile app subsystem consists of the following components: learning, assessment, feedback, performance, quality of service, and device status. These components collect various types of real-time contextual information, such as network status, device battery status, learner profile, learner input, and quality of service. The system then transmits the collected contextual information to the web server subsystem, which uses a web service from a framework called DACAMoL that acts as a controller and semantically transforms the context using ontologies. Rule-based techniques also discover available learning resources from the service provider subsystem. In this software architecture, the DACAMoL web service will discover the services available from the service provider. Finally, the service provider links its service when the request matches an equivalent service it offers.

[P15] proposes a Smart learning environment consisting of modules for curriculum design, initial competency assessment, learning performance evaluation, personalized

learning, visualization and feedback, recommendations, and ubiquitous computing. The curriculum design module lets content specialists, instructional designers, and trainers develop content and set assessment questions. The initial competency module identifies learners' prior knowledge. The performance evaluation module measures learning outcomes through online tests and activities. The personalized learning module adapts learning content, materials, and tasks based on learners' knowledge, performance, and objectives. The visualization and feedback module delivers timely feedback using data-driven visualizations. The recommendation module suggests effective strategies, activities, and content to help learners achieve their goals. Finally, the ubiquitous computing module ensures the learning process can happen anytime, anywhere.

The study described in [P16] proposes a web-based adaptive learning platform that uses a microservices-based architecture to support Chinese vocabulary acquisition for non-native speakers. The platform's software architecture consists of four primary layers: (1) a responsive frontend designed to capture learner interactions; (2) a collection of decoupled backend microservices responsible for content delivery, learner modeling, reinforcement learning based on the learner model, and analytics; (3) an event-streaming infrastructure that transmits fine-grained user events; and (4) a monitoring and observability layer that tracks both technical metrics and pedagogical signals.

[P17] proposes a gamified mobile learning framework integrated into two Android apps, one for learners and one for teachers, with a backend focused on managing the framework's functionalities. Subsystems and modules, including the learning management subsystem, learning subsystem, gamification module, lesson recommendation module, and problem-solving solutions recommender module, support these functionalities. The learning management subsystem provides teachers with different services such as developing, managing, and providing learning content. It integrates three modules: lesson management, theoretical test management, and problem-solving management. The learning subsystem allows students to start their learning through three levels. The gamification module provides students with diverse elements, including levels, scores, badges, leaderboards, and feedback, during the learning process to increase student engagement. The lesson recommendation module provides students with personalized lessons based on their preferences to enhance the learning process. Finally, the problem-solving solutions recommender module recommends the best problem-solving solutions to help students benefit from the solutions teachers provide.

Study [P18] proposed ECGTwinMentor, a digital twin-based educational platform that supports electrocardiogram (ECG) interpretation through interactive simulation and intelligent prediction. Its software architecture comprises three main parts: the real ECG environment, the digital twin, and cloud and edge deployment. The real ECG environment generates ECG waveforms from physiological parameters and provides ECG data to the deep learning model. The digital twin includes a data source to extract diagnostic parameters, a secure API, and an AI module that simulates ECGs and makes diagnostic predictions. The resulting predictions are presented through a user interface that provides statistics and educational feedback. The cloud deployment includes a dashboard for interacting with the system and visualizing results, while the edge deployment enables local inference on mobile and resource-constrained devices.

Study [P19] proposed EduXgame, a gamified educational platform that integrates AI-based content generation to support interactive learning activities. Its software architecture is organized into three main layers: view, business, and API. The view layer provides web and mobile interfaces for accessing educational content and interacting with learning activities. The business layer handles core learning functions and gamification, including quizzes, leaderboards, and rewards. The API layer supports communication between the platform and external services. The system also includes a backend for user

authentication, session management, and communication with the mobile application, along with a database to store user information, game scores, and generated educational content. Additionally, an AI-based content-generation service processes uploaded educational documents to create interactive learning materials, including quizzes, flashcards, and matching games.

4.2.3. Architectural Styles and Views

This section examines the architectural styles and emerging paradigms adopted in the selected primary studies. Most analyzed studies describe their software architectures using a layered and/or component-based style (see Table 6). Across these studies, a recurring set of layers includes application, context detection, content and format adaptation, and knowledge adaptation. In contrast, the studies [P2], [P3], [P9], [P10], [P12], [P13], [P14] and [P17] adopted a client–server architectural style. The selected primary studies [P1], [P3], [P6], [P7], [P9], [P12], [P14], and [P16] proposed software architectures using cloud computing. [P4], [P5], and [P18] integrated emerging paradigms such as cloud, fog, and edge into their proposed architectural designs.

Table 6. Architectural styles of the selected studies.

Architectural Styles and Emerging Paradigms	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19
Layers and components	☑	☐	☐	☑	☑	☑	☑	☑	☐	☐	☑	☐	☐	☐	☑	☑	☐	☑	☑
Client-Server	☐	☑	☑	☐	☐	☐	☐	☐	☑	☑	☐	☑	☑	☑	☐	☐	☑	☐	☐
Cloud computing	☑	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☑	☐	☑	☐	☑	☐	☑	☐
Fog computing	☐	☐	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Edge computing	☐	☐	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐

4.2.4. Technological Approaches and Technologies Used in the Implementation of Software Architecture

This section describes technologies and technological approaches used in the implementation of the proposed software architectures. The study [P1] reports the use of machine learning and cloud computing in its implementation, although no hardware specifications are provided. The study explicitly states that it uses Python to predict students' concentration levels. The study [P2] compares classification techniques, including J48, Random Tree, and Naïve Bayes, to assess learning style classification performance; however, it does not specify the implementation technologies used. In contrast, the study [P3] proposes multiple mobile learning models that use artificial neural networks to predict student performance and classify learners into different groups. At the same time, a Random Forest algorithm is used to identify the most relevant learning characteristics.

The study [P4] presents a Performance-Optimized Cloud Stack (POCS) and a cloud distance architecture composed of an edge cloud, cloudlets, and a minimal data center. Additionally, it reports the use of Long Term Evolution (LTE) broadband wireless technology and a task delegation algorithm, although the programming language is not specified. The study [P5] considers LTE wireless broadband technology, mobile devices, edge cloud nodes, and a hybrid cloud infrastructure leveraging *AWS* and *Google Cloud* services. The study [P6] employs a questionnaire based on the Felder-Silverman learning styles model, whose results are used as input data for training the machine learning components of the proposed mobile learning system. In the study [P7], data storage and mining services were provided based on the Google Firebase cloud and Google Cloud Machine Learning (ML) Engine, which were integrated with an Android application installed on students' cell phones to obtain contextual information. In this work, the context analysis layer uses the Google API client library in Python code running on the Firebase cloud to

make calls to the Google Cloud ML Engine REST APIs, and the Python code of the DBSCAN algorithm runs on Google Cloud ML Engine to determine the places that students regularly visit in the form of clusters. The study [P8] reports using ML techniques implemented in Python, including clustering algorithms and PCA, combined with ontology modeling and Moodle LMS integration, while considering device-related factors such as screen size and RAM. In contrast, the study [P9] modeled the proposed ontology in Protégé and stored it in a semantic database called StardDog. The server was developed using C# and Visual Studio 2012. Both the server and the semantic database were hosted on Amazon Web Services (AWS). The server uses SPARQL queries, the Pellet reasoner, and rules defined using SWRL. The OAI-PMH protocol was used to integrate the server with the learning object repositories. The student app was developed in Objective-C using Apple Xcode 5 for iOS. In this proposal, user authentication was performed using the OAuth 2.0 protocol.

In [P10], the proposed framework's message-transmission network architecture used 3G mobile communication networks and GPS. These networks, in conjunction with the context-aware and instructional material recommendation modules, initially determine students' potential educational needs based on their environment and learning history. Subsequently, the system recommends educational content tailored to each student. This educational content was obtained from a database of instructional materials. This work does not describe any specific technology used to build the proposed framework. In [P11], a prototype system was implemented based on the presented software architecture to provide learning support for workplace issues in two scenarios. This prototype was deployed as a web page hosted on a web server. The technologies used to implement this prototype included PHP, WordPress, Java, Jena, Web Ontology Language (OWL), XML, MySQL, etc. PHP was used to develop the system interfaces, student registration, login, and student information management. WordPress was used to build each student's personal space. Java was used to implement the system's control logic. These tasks included database manipulation and adapting instructional materials. Jena was used to manipulate student contextual information modeled in OWL and to infer indirect contextual aspects. The learning context was stored in OWL files. Learning units were stored in XML files. All student registration information, profiles, and learning histories were stored in MySQL. In [P12], a system for delivering multimedia learning content based on users' cognitive styles and context-aware network conditions is described; however, this work only mentions using OWL and the RDF/XML format to obtain information about users and their preferences. The work proposed in [P13] describes developing a mobile version of Protus 2.1 to offer the same e-learning functionalities to mobile users. This work uses Semantic Web Rule Language (SWRL), which combines OWL and RuleML. SWRL allows interoperability with major rule systems such as JESS. JESS is a Java framework for editing and applying rules, with a scripting environment and a rule engine. This work also used Protégé, an open-source ontology editor and knowledge-based framework. The Protégé OWL Plugin provides a SWRL editor, enabling the formalization of SWRL rules in conjunction with OWL ontologies.

[P14] describes a context-aware mobile cloud learning system using a semantics-based approach. This system is based on the mobile cloud learning educational paradigm, which offloads computationally intensive tasks (such as contextual reasoning) to cloud services/servers, thereby optimizing battery and memory consumption on the mobile device and enabling continuous learning and the management of learning resources both online and under conditions of variable or limited connectivity. The mobile application developed in this work was designed for the Android operating system. This application transmits the contextual information it collects to the web server subsystem via a Hypertext Transfer Protocol (HTTP) request using a Representational State Transfer (RESTful)

web service. The web server subsystem uses a framework called DACAMoL, which employs Semantic Web technologies such as OWL-S and WSDL-S.

In [P15], the proposed SMART Learning Environment was developed using a data mining methodology centered on a four-step Artificial Neural Network (ANN) process. The ANN family's backpropagation (BP) algorithm was used for supervised classification of student learning performance clusters, based on pre-test scores and completion times. The algorithm follows four steps: feed-forward computation, backpropagation to the output and hidden layers, and weight updates. Training continues until the error function meets a predefined threshold, allowing the neural network to learn from the dataset. Once trained, the system generates personalized learning pathways. The Personalized Learning Module offers six levels of instructional materials, increasing in complexity from Level 1 to Level 6. After each level, learners complete a post-test. The system evaluates both the test score and completion time; learners who pass advance to the next level; those who do not repeat the current level.

The work in [P16] presents a containerized microservices architecture for real-time adaptive language instruction composed of four layers: a responsive frontend, backend microservices, event-streaming infrastructure, and monitoring and observability. The frontend uses a React.js single-page application (SPA) that dynamically renders vocabulary exercises based on JSON payloads from the content delivery API. The backend layer consists of four microservices: content delivery, learner modeling, reinforcement learning, and analytics. Each microservice runs in a Docker container, is managed by Kubernetes, and exposes RESTful interfaces documented with OpenAPI. In the event-streaming layer, SPA batches UI events and sends them via WebSocket to an event dispatcher. For monitoring and observability, Prometheus and Grafana track each microservice's latency, throughput, and uptime to ensure system observability and reliability.

The study [P17] describes a gamified mobile learning framework implemented through two Android applications, one for students and one for teachers, and a web app for an administrator, supported by a backend system that manages the framework's core functions. In this framework, Firebase Database connects the end-user apps to the database. When a student or teacher requests a service, the Android application connects to the Web server, which retrieves data from the Firebase database using Web services. The server then encodes the data in JSON format and sends it to the Android app. The framework comprises several subsystems and modules, including the learning management subsystem, learning subsystem, gamification module, lesson recommendation module, and problem-solving solutions recommender module. The recommendation modules were implemented on two web servers using Python and the Django framework.

Study [P18] describes ECGTwinMentor, a digital twin-based educational platform implemented using Python (version >=3.8), React, and FastAPI. React supports the web interface, while FastAPI supports the cloud-based backend. The deep learning model was implemented using TensorFlow and the Keras Sequential API to classify ECG patterns based on physiological parameters. After training, the model was exported in three formats: HDF5 for Keras, TensorFlow Lite for edge devices, and ONNX for interoperability with other platforms. The system adopts a cloud–edge deployment approach, allowing centralized access through its web-based version and local inference on Android devices and Raspberry Pi using TensorFlow Lite. This approach enables offline operation in environments with limited connectivity.

In [P19], EduXgame was implemented using different technologies for its web, mobile, backend, and AI-based content generation functionalities. The web interface for administrators was developed using Next.js, while the Android mobile application was implemented in Java and XML. The backend uses Spring Boot to manage user authentication, session control, and communication with the mobile application through RESTful

APIs. MySQL is employed to store user information, game scores, and AI-generated educational content. For automated content generation, the system uses a Python Flask API to process uploaded educational documents and generate interactive learning activities. PyMuPDF and PyPDF2 extract text from PDF documents, while LangChain and the OpenAI LLM API generate quizzes, flashcards, and other learning materials. Additionally, Docker is used to support deployment across different environments, and CI/CD practices are employed to automate integration and deployment processes. SonarQube assesses software quality, including reliability, security, maintainability, code duplication, and test coverage.

4.3. RQ3: What Approaches Have Been Used to Evaluate the Software Architectures of AI-Based MALS?

The selected primary studies employ different approaches to evaluate the proposed software architectures for AI-based MALS. These approaches include architectural, system- and user-centered, educational- and learning-outcome, and AI-model evaluations. Architecture-focused assessments evaluate the underlying infrastructure, software layers, and performance in terms of response time and latency. System- and user-centered assessments evaluate usability, user experience, cognitive load, cross-platform interface adaptation, and adoption models. Evaluations of educational and learning outcomes include controlled experimental interventions that assess pedagogical impact, knowledge acquisition, pre- and post-test scores, and learning style adaptations. Evaluations of AI models focus on algorithmic validation, predictive accuracy, and cluster-quality metrics to assess AI's underlying logic. Table 7 categorizes the selected primary studies by evaluation approach. Some studies used hybrid evaluations.

Table 7. Evaluation approaches used by the selected primary studies.

Evaluation Approaches	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19
Architectural evaluation	☐	☐	☐	☑	☑	☐	☐	☐	☑	☐	☐	☐	☐	☑	☑	☑	☐	☑	☑
System/user centered evaluation	☑	☐	☑	☐	☐	☑	☑	☐	☑	☑	☑	☑	☑	☐	☑	☑	☑	☐	☐
Educational and learning outcome evaluation	☐	☑	☑	☐	☐	☐	☑	☑	☐	☑	☑	☐	☐	☐	☐	☐	☑	☐	☐
AI Model and Predictive Evaluation	☐	☐	☑	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐

In [P1], the student learning experience was evaluated for quality, acceptance, and satisfaction through an online questionnaire administered to 20 university students with prior experience in m-learning systems. The study [P2] adopted an experimental design using control and experimental groups to compare learning performance when learning styles and preferences were considered versus conventional instruction. The system proposed in [P2] was evaluated in terms of performance by a group of students in the context of a computer science course. The students were divided into two groups: groupLA and groupLB. GroupLA consisted of students who received face-to-face classes and could only take an online assessment. In contrast, students in groupLB used the system proposed in this work. This system promotes personalized learning through adaptive content, adaptive learning paths, and context awareness to meet each student's individual needs. Based on the results, students in groupLB performed better than students in groupLA. The study [P3] evaluated the predictive accuracy of artificial neural network-based learning models and conducted an experimental intervention with control and experimental groups to assess the impact on student performance and satisfaction.

In contrast, the study [P4] evaluated architectural performance through a SWOT analysis applied to the layers of a mobile cloud architecture, highlighting improvements in users' quality of experience (*QoE*). In the study [P5], researchers assessed usability and performance through a case study involving engineering students, measuring metrics such as latency and response time. The study [P6] evaluated overall satisfaction, usability, and ease of use through an online questionnaire, although details regarding the number of participants were not reported. In the study [P7], an experimental study was conducted with 150 engineering students divided into two groups to evaluate the performance and acceptance of the system proposed in this work over a period of six months. In this work, performance was evaluated through a pretest and six post-tests, and acceptance was evaluated using a modified questionnaire based on the End-User Computer Satisfaction (EUCS) model. Similarly, the study [P8] evaluates the effectiveness of its adaptive architecture through quantitative machine learning validation metrics, including clustering quality measures such as the Silhouette Score and Davies–Bouldin Index, as well as comparative analyses of student performance before and after adaptation. On the other hand, the EduAdapt architecture in [P9] was evaluated using two approaches. In the first approach, the OntoAdapt ontology was evaluated across various scenarios, and its quality and fidelity in complying with concepts related to the associated subjects were also analyzed. In the second approach, a usability evaluation was conducted on a mobile application developed using the proposed architecture to measure student satisfaction. In this evaluation, 20 undergraduate students used the developed app for one month. Subsequently, the students completed a usability questionnaire developed and validated by the authors.

The architecture described in [P10] was evaluated through experiments assessing learning performance, the effects of the main learning modules, and student satisfaction. These experiments included 102 first-year undergraduate students. The students were divided into two groups, an experimental group and a control group, to evaluate and compare differences in learning outcomes, learning styles, and student satisfaction. The experimental group, composed of 57 students, used the developed learning system for three months. In this case, the context-aware recommendation subsystem provided students with situational conversations, real-time writing exercises, fill-in-the-blank quizzes, and vocabulary acquisition activities. In contrast, the control group, comprising 45 students, used only computers or mobile phones to search the Internet for English-learning materials. In the study reported in [P11], user studies were conducted to evaluate the implemented prototype across two scenarios. The first scenario involves professionals working in office or laboratory environments. In this scenario, participants reconfigured a printer's IP address to match the organization's new intranet configuration. The second scenario involves workers at the Maped factory. Maped is a French manufacturer of school and office supplies. The task in this scenario consisted of changing the guillotine blade on a test bench for staplers and hole punches. In these scenarios, three aspects of the prototype were evaluated: (1) users' learning achievements with the system, (2) users' attitudes and acceptance of the system, and (3) the cognitive load the system imposes on users. Thirty-one professionals participated in this evaluation; 23 participated in the first scenario and 8 in the second. The 23 participants in the first user study included laboratory researchers and secretaries. The 8 participants in the second user study included interns and employees who had been in their positions for less than 6 months. The instruments used in this evaluation were a logging tool to collect data on users' learning behaviors in order to analyze their academic performance, and four questionnaires designed to analyze, respectively, users' learning attitudes, the cognitive load imposed by the learning system, acceptance of the learning system, and the learning influence.

In [P12], the developed system was evaluated by 90 students enrolled in the Database Systems course. After finishing the course, the 90 students assessed Quality of Experience using a Learning Scenario Questionnaire to estimate preferences across five learning situations. The Protus 2.1 mobile prototype developed in [P13] was evaluated in two phases. In the first phase, a group of 26 students completed a set of adaptive learning activities on mobile devices running different platforms (Apple iOS, Google Android, and Microsoft Windows) and with different screen sizes. The activities were based on a learning path previously established by a teacher. The objective of this evaluation was to assess the usability of the Protus 2.1 mobile prototype through a questionnaire. In the second phase, a group of 51 students used the Protus 2.1 mobile prototype for three weeks to provide their opinions on the role that mobile learning could play in the field of programming. After the three weeks, a questionnaire designed by the authors was administered to the students.

In [P14], the context-aware mobile cloud learning system was evaluated for its correctness in selecting appropriate learning resources across 33 simulated scenarios. These scenarios involved changes in context information and quality of service, including battery level, Internet connection, availability, and reputation. Thirty undergraduate software engineering students participated by installing the Mudahnya BM mobile app and completing a checklist for each scenario. The results showed that the app consistently provided accurate learning resources.

In [P15], the proposed SMART learning environment framework was evaluated using three distinct approaches. The first approach involved defining three experimental scenarios corresponding to different learner profiles: good, average, and weak. The framework delivered tailored learning materials and activities based on learners' pre-test and post-test results. The second approach consisted of a usability evaluation with 20 Cybersecurity professionals, which indicated that the framework was user-friendly and met participants' expectations. The third approach assessed the framework according to the Technology Acceptance Model (TAM), examining perceived ease of use, perceived usefulness, attitude towards use, behavioral intention to use, and actual system use, with 63 Cybersecurity professionals in Mauritius participating. The results were positive across all evaluated dimensions.

The microservices-based architecture proposed in [P16] was evaluated using two approaches: an architectural evaluation and a user- or system-centered evaluation, based on a real-world scenario involving 154 students of Chinese as a second language. In the architectural evaluation, this work assessed the platform's health, response latency, dynamic scaling, and operational robustness under different load scenarios. In contrast, the system- or user-centered evaluation assessed engagement metrics and user satisfaction, including average session duration, dropout rate, revisit rate, and average vocabulary proficiency on the post-test. In this work, a post-study questionnaire was also administered to students to capture students' subjective perceptions of the platform.

The study [P17] evaluated the described framework through an experiment involving 90 college students divided into three groups of 30. Over four months, each group used a different version of the proposed framework. The control group used a version of the framework without recommendation and gamification services. Another group used a version of the framework that provided only gamification services, and the third group used a version that provided both recommendation and gamification services. In this study, a pre-test and post-test were administered to measure academic performance, and a questionnaire was also given to the students to measure motivation, to analyze how the integration of personalized recommendations and gamification influences the motivation and academic performance of college students in the context of a programming course on complex data structures using mobile devices.

In [P18], ECGTwinMentor was evaluated through predictive model performance analysis, offline deployment demonstrations, and a preliminary pilot study. The deep learning model was evaluated using a synthetic dataset of 3000 ECG samples, divided into training (80%) and validation (20%) subsets. The authors assessed classification performance using accuracy, precision, recall, F1-score, log-loss, and a confusion matrix. Additionally, the system was deployed on Android and Raspberry Pi devices using TensorFlow Lite to demonstrate its ability to perform local predictions without an Internet connection. Finally, a preliminary pilot evaluation was conducted with a multi-disciplinary team comprising participants from computer science, nursing, and occupational therapy. Participants reported increased confidence in ECG interpretation and a better understanding of how changes in physiological parameters affect ECG waveforms. However, the authors indicate that formal validation with medical and health science students remains future work.

In [P19], EduXgame was evaluated through software performance testing and quality assurance analysis. The authors conducted load testing in a Docker environment to assess the system's ability to handle concurrent requests. The results showed that the platform supported up to 100 concurrent connections, with an average response time of 800 ms. Stress testing was also performed to examine system scalability and responsiveness under increased workloads. Furthermore, the authors used SonarQube to evaluate the backend, web frontend, and Android application for reliability, security, maintainability, code duplication, and code coverage. Although the authors mention that the platform operates in a controlled testing environment and undergoes iterative improvements based on feedback, no formal user-centered or educational outcome evaluation is reported.

4.4. RQ4: What Quality Attributes Have Been Considered in the AI-Based MALS?

The primary studies [P1–P19] address a range of criteria that can be systematically categorized into software quality attributes, educational outcomes, and user perceptions. Software quality attributes vary across the primary studies, ranging from baseline functional requirements to specialized non-functional attributes:

- Functional suitability and Interoperability: Consistently addressed across all selected primary studies [P1–P19].
- Usability Attributes: Studies [P1], [P6], [P7], [P11], [P15], [P16], [P18], and [P19] cover different usability-related quality attributes.
- Security (confidentiality, integrity, and availability): Some security attributes were addressed in studies [P2], [P3], [P5], [P6], [P7], [P8], [P9], [P11], [P18], and [P19]. Studies [P6] and [P16] explicitly considered confidentiality, integrity, and availability, and [P19] only integrity.
- Performance Efficiency: Studies [P4], [P5], and [P18] report performance efficiency.
- System Metrics and Architecture: Study [P9] evaluates the OntoAdapt ontology using structural quality metrics—specifically coverage, richness, and level of detail—to weight and select appropriate ontologies for specific applications.

Several studies go beyond system-level properties to evaluate pedagogical efficacy and learning performance (educational outcomes):

- Academic Performance and Achievement: Academic performance and system acceptance are evaluated in [P3], [P7], [P8], [P11], and [P17], respectively. Study [P2] focuses specifically on student performance within an online course assessment.
- Multidimensional Learning Impact: Study [P11] provides a comprehensive pedagogical evaluation, measuring just-in-time learning achievement, long-term learning achievement, learning attitudes, system-imposed cognitive load, and system acceptance.

- Instructional Effectiveness: Study [P10] evaluates the impact of learning methods on English language acquisition (everyday vocabulary, expressions, and conversation) by comparing experimental and control groups via post-tests and independent t-tests.

User perceptions and quality of experience evaluations conducted by some studies assessed how learners interact with and perceive the proposed systems based on the following quality attributes:

- Satisfaction and Effectiveness: [P3], [P9], and [P16] highlight general attributes associated with effectiveness and user satisfaction, respectively.
- Perceived Usability and Practicability: Study [P10] assesses experimental group satisfaction across ease of use, interactivity, and overall practicability.
- Quality of Experience (QoE): Study [P12] measures QoE via a domain-specific learning-scenario questionnaire primarily aimed at capturing user satisfaction with the service.
- Comprehensive Mobile Usability: Study [P13] evaluates the Protus 2.1 mobile prototype across a wide spectrum of user experience factors, including usefulness, execution speed, responsiveness, instant access, learnability, efficiency, convenience, personalization, added value, and effectiveness.
- User acceptance and new technology adoption: Study [P15] evaluates the proposed framework using the TAM model, examining perceived ease of use, perceived usefulness, attitude toward use, behavioral intention to use, and actual system use.
- User Motivation: Study [P17] assesses student motivation before and after the experiment through a motivation questionnaire. The motivation questionnaire consisted of 8 items on a 5-point Likert scale, where 5 represented strongly agree, and 1 represented strongly disagree.

4.5. RQ5: What Areas of Study and Educational Levels Have Been Addressed in AI-Based MALS?

Selected primary studies mainly addressed university-level education and/or first-year undergraduate students. Concerning application domains, several studies focus on specific areas within Information Technology (IT) such as cloud computing and databases in [P1], basic computing concepts in [P2], ubiquitous and mobile computing in [P9], computing and programming languages in [P3], [P7], [P8], [P13], database systems in [P12], software engineering in [P14], Cybersecurity in [P15], and programming complex data structures using the Java programming language in [P17]. Study [P10] describes a context-aware recommendation learning mechanism that provides English learning content to first-year undergraduate students based on their location and environment. In contrast, studies [P4], [P5], [P6], and [P11] describe using computational tasks or learning resources without specifying a particular area of study. The study in [P16] proposes a web-based adaptive learning platform with a microservices architecture to support Chinese vocabulary acquisition for non-native speakers.

Unlike the previous studies, [P18] focuses on medical education, particularly cardiology and electrocardiogram (ECG) interpretation. The proposed system is intended to support the training of medical students and early-career healthcare professionals through interactive ECG simulations and diagnostic feedback. In contrast, [P19] addresses secondary education through a gamified learning platform designed to support different school subjects. Rather than focusing on a specific discipline, this platform generates interactive educational activities from teacher-provided learning materials.

4.6. RQ6: How Is AI Being Implemented in the Proposed Software Architectures?

Based on the AI methodologies, techniques, and paradigms described in each study, the selected primary studies [P1–P19] can be classified into four categories: C1. Knowledge-based and Symbolic AI, C2. Machine Learning-based Approaches, C3. Hybrid AI Approaches, and C4. Generative AI.

4.6.1. C1. Knowledge-Based and Symbolic AI

This category includes selected primary studies in which adaptation is achieved primarily through explicit representation of knowledge, ontologies, decision rules, or inference mechanisms. These approaches model student characteristics, context, and relationships among educational resources without necessarily requiring data-trained models. The selected primary works for this category are described below.

The adaptation process described in [P9] begins by collecting real-time contextual information via the student's app. This information includes device characteristics (model, battery level, available storage, screen size, and operating system), wireless connection characteristics (signal strength and connection type), current location, and whether the student is moving or stationary. The server stores this information in the OntoAdapt ontology upon receipt. The server then uses the Pellet reasoner and predefined rules to identify the appropriate ontology topic the learning object should cover based on information about the app and the student's profile. The result of the reasoner is used to select an appropriate learning object from the learning object repositories using the learning object metadata. Once a learning object has been selected, two types of adaptations can be performed: scale and format. Scale adaptation involves modifying the resolution of the processed learning object. Format adaptation occurs when the learning object contains a document the device does not support, or that does not match the student's learning style or current context. Finally, the server sends the adapted learning object to the app for presentation to the student.

In the work proposed in [P10], the context-aware module provides students with learning materials tailored to each specific environment in real time. To achieve this, the module uses a personalized context-aware learning algorithm. This algorithm comprises the following four main rules and decision processes. Phase 1.- Identification of the student's current location via GPS; Phase 2.- Location Management (LM), which analyzes the characteristics of the student's current environment using a database of geographic environments based on location-based scenario rules. At this point, data mining is used to classify the student's location within the appropriate geographic environment database and determine the corresponding environmental information; Phase 3.- Context-aware learning and assessment, in this phase, a comprehensive learning recommendation is provided to students based on the environmental characteristics of their location; and Phase 4.- Reading and testing in the learning and assessment phase, in this phase, students review the personalized learning materials received on their mobile devices. Subsequently, the system generates a quiz with questions related to the learning content received to determine whether the student has understood the material.

In the study [P11], the adaptation engine layer applies rules and learning-based strategies to adapt and personalize learning. The input to the adaptation engine layer is the learning context specified in the WbML context model. Using adaptation strategies such as learning unit selection, sequencing, navigation through learning units, communication among learning partners, and learning activity generation, this adaptation engine layer provides appropriate support for workplace learning. This adaptation engine can also receive learner feedback and refine adaptations by recalling the learning context and/or modifying adaptation strategies. This layer is based on rules that generate adaptations using conditional IF/THEN/ELSE statements. Seven sets of rules are integrated into this

adaptation engine; these include: selection of learning units, sequencing of learning units, navigation through learning units, communication with learning peers, and generation of learning activities.

On the other hand, in [P12], the user's profile can belong to more than one profiling group. Therefore, this work uses the WOL to define and describe relationships among objects such as users, network characteristics, and access types. In this work, the adaptation engine adapts the m-learning content in two steps: first, using the user's learning preferences, and second, using the network type. The first step is modeled using long-term individual profiling, whereas the second step is modeled using short-term individual profiling.

The work proposed in [P13] uses reasoning mechanisms to infer the system's presentation and services based on the learner model and the application context. The task ontology and teaching strategy ontology components use these mechanisms to determine how adaptation is carried out. The Semantic Web Rule Language (SWRL) adaptation rules used in Protus 2.1 are learner modeling and adaptation rules. Learner modeling rules build knowledge about a learner and deduce new characteristics from the learner's current characteristics. These rules construct and update the learner model. Adaptation rules specify adaptation strategies based on domain characteristics, the system's adaptation objectives, user characteristics, context, and the presentation methods used.

The software architecture described in [P14] uses knowledge representation and rule-based semantic reasoning to support its context-aware adaptation engine. The system uses cloud-based ontologies to model relationships between intrinsic user data, such as age, language proficiency, and progress, and extrinsic environmental factors, including device metrics, network stability, and bandwidth. A semantic inference engine applies logical IF-THEN rules to this ontology, enabling real-time evaluation of current conditions and dynamically discovering, matching, and substituting educational resources from service providers. This approach ensures the application consistently delivers learning resources optimized for Quality of Service, even under limited or changing internet connectivity.

These studies show that intelligent adaptation does not depend exclusively on machine learning. It can also be implemented using formalized knowledge, contextual rules, and reasoning mechanisms. However, it is important to distinguish between architectures that use ontologies solely to represent information (P12) and those that implement explicit semantic inference mechanisms (P9, P13, and P14).

4.6.2. C2. Machine Learning-Based Approaches

This category includes studies that use machine learning (ML) algorithms for tasks such as prediction, classification, pattern recognition, and recommendation. The models process information about students, their behavior, context, or the educational domain to support the system's functions. The works classified in this category are described below.

The study [P1] implemented a neural network that uses contextual variables such as noise level, geolocation, physical activity, available study time, knowledge level, and gender to estimate the student's level of concentration. Based on this prediction, the system dynamically adjusts educational resources according to universal design for learning principles, providing content adapted to the context and the learner's individual characteristics.

On the other hand, the study [P2] employed a Bayesian network, specifically a Naive Bayes model, to identify and classify students' learning styles (active, reflective, sensing, intuitive, visual, verbal, sequential, and global). Based on this classification, the system personalizes both the learning materials and the learning sequence and continuously updates the user profile based on interactions and performance on the platform.

The study [P3] developed an m-learning model based on an Artificial Neural Network (ANN) to predict academic performance and classify students into five performance groups. The model considered behavioral characteristics such as forum participation and problem-solving; contextual characteristics such as study time and access to text or video resources; and performance characteristics such as grades obtained. The study also used Random Forest to estimate the importance of each characteristic in predicting student performance.

Studies [P4] and [P5] addressed the development of intelligent tutoring systems designed to offer individualized, adaptive learning experiences. These systems incorporated machine learning techniques that allow for dynamic adjustments to content, pedagogical strategies, and tutoring models (linear, dialogic, or exploratory) based on the student's aptitude, context, and performance.

In the study [P7], physical sensors and logical sources on students' mobile devices collect context information, which is stored in the Firebase cloud. This context information stored in Firebase is processed using the cluster-based machine learning algorithm DBSCAN in order to identify relationships between contextual learning variables. Based on the information generated by the DBSCAN algorithm, the system generates different learning patterns for each student in order to improve their academic performance in computer programming.

The SMART Learning Environment described in [P15] uses a Multi-Layer Perceptron, a feed-forward neural network with one or more hidden layers, to perform prediction and recognition tasks. The network was trained using the Backpropagation Learning algorithm. For this purpose, this work constructed a dataset. After normalization, the training set was created by assigning elements to the input and output neurons. This study used 50 training sets. The network was trained with these sets, with both the learning rate and momentum specified.

The framework proposed in [P17] provides personalized recommendations through two intelligent modules: a lesson recommendation module (LRecM) and a problem-solving solution recommender module (PSSORecM). The LRecM provides students with personalized lessons aligned with their preferences to enhance the learning process. Specifically, this module provides students with lessons ranked from highest to lowest based on predicted preferences (ratings). In contrast, the PSSORecM is behavior-based and guides students to explore other solutions. This module uses student behavior data collected by a behavior detection module that records all actions students perform while consulting the list of best solutions. The two intelligent recommendation modules developed within the framework proposed in this work use collaborative filtering and matrix factorization, applying singular value decomposition and negative matrix factorization, respectively.

In [P18], ECGTwinMentor uses a deep learning model to classify electrocardiographic patterns and provide learners with diagnostic feedback. The model was implemented using TensorFlow and the Keras Sequential API, employing a feedforward neural network with three hidden layers to process eight physiological parameters, including heart rate, cardiac intervals, rhythm, and T-wave morphology. The output consists of predicted ECG diagnostic categories, presented through the system's user interface to support interpretation and diagnostic reasoning. The AI module is incorporated into a digital twin-based architecture that also includes a parameter-driven ECG simulation component, allowing learners to modify physiological parameters, visualize the resulting waveforms, and compare them with the model's predictions. The trained model can operate through cloud services or be deployed on Android and Raspberry Pi devices using TensorFlow Lite for offline inference. Therefore, AI primarily supports diagnostic classification and educational feedback, while ECG waveform generation is performed through a deterministic physiological simulation model.

In these studies, ML primarily performs prediction, classification, pattern recognition, and recommendation. However, it is important to distinguish between systems that directly use predictions to tailor educational resources and those in which the models provide information or feedback that supports learning activities.

4.6.3. C3. Hybrid AI Approaches

This category includes architectures that integrate different AI paradigms, particularly machine learning techniques with knowledge representation mechanisms such as ontologies or decision rules. Unlike the previous categories, these studies combine data-driven processing with explicitly represented knowledge to support personalization or adaptation. The selected primary works for this category are described below.

The study [P6] implemented a personalized mobile learning expert system in which the adaptation engine combines recommendation algorithms, rule-based engines, and machine learning models to generate learning paths tailored to the user's preferences. This system uses the Felder-Silverman model to classify learning styles and, based on this, personalizes the content, interface, and interactions, also integrating continuous feedback mechanisms.

The study [P8] explicitly incorporates machine learning techniques into its architecture, particularly unsupervised learning algorithms (DBSCAN and K-means) and dimensionality reduction (PCA) to build learner models. These models then drive ontology-based adaptive content delivery, linking learner behavioral, contextual, and emotional characteristics to personalized educational resources.

The microservices-based software architecture described in [P16] incorporates a reinforcement learning engine that enables real-time user modeling, adaptive content delivery, and personalized interventions. In this architecture, the client interacts with backend microservices through an API gateway, which routes requests to ontology and reasoning services. The ontology service manages the OWL 2-based UAV mission ontology, provides endpoints for ontology retrieval and entity creation, establishes links between classes and instances, and supports SPARQL queries and ontology modification via SPARQL Update. The reasoning service uses the SWRL rule base for mission validation, conflict detection, and restriction enforcement. To minimize latency, rules are precompiled and evaluated using a custom rule engine integrated with OWLAPI. The mission interaction service orchestrates user-defined planning tasks, monitors session status, and communicates updates to the user interface. This service provides a stateless API for submitting new task graphs, requesting constraint validation, and retrieving mission summaries enhanced with semantic annotations. A dedicated visualization service subsequently converts the semantically enriched mission data into dynamic, browser-native visual representations.

These studies integrate different AI approaches into a single architecture. In particular, [P8] and [P16] illustrate how explicit knowledge representation can coexist with data-driven learning mechanisms. This combination separates the semantic organization of information from the processes that identify patterns or make adaptive decisions.

4.6.4. C4. Generative AI

This category includes studies that incorporate generative models—particularly large language models (LLMs)—to produce educational content and resources. Only one selected primary study fell into this category.

In [P19], the system employs LLMs, accessed through the OpenAI API and orchestrated using LangChain, within a Python Flask-based content generation service. Uploaded PDF documents are processed to extract textual information, which is subsequently converted into vector embeddings and stored using FAISS to enable semantic

retrieval. LangChain uses a retrieval-based question-answering chain to retrieve relevant document content and provide it to the language model. The output is organized into structured educational materials, including quizzes, flashcards, matching games, and chapter descriptions, available through the platform's web and mobile interfaces. Therefore, AI is primarily implemented as a content-generation and semantic-retrieval mechanism intended to support the creation of interactive learning activities and reduce the effort teachers need to prepare educational resources.

This approach proposes a different role for AI within educational architectures. While previous approaches have focused primarily on selecting, recommending, or adapting existing resources, Generative AI can produce new materials based on information provided to the system. This expands the possibilities for implementing AI in educational settings, although content generation does not necessarily imply individualized adaptation.

5. Discussion

Based on the comparative analysis in Table 8, most selected primary studies [P1–P19] consider functional requirements for format and/or content adaptation. Learning path adaptation is considered only in studies [P1–P3], [P6–P7], [P11], and [P15–P17]. The acquisition, management, and use of student, context, and domain information, the management of technological and educational resources, and the generation of student history and profiles are also considered in the studies [P1], [P3], [P6], [P8], [P9], [P10], [P11], [P13], and [P14]. Studies [P4] and [P5] focus their functional requirements on acquiring, managing, and using contextual and student information; adapting format and content; and managing technological resources. In contrast, the studies proposed in [P2] and [P7] consider neither the functional requirements of format adaptation nor the management of technological resources. The study [P12] considers neither the functional requirements of learning path adaptation nor the management of technological resources. Similarly, study [P8] incorporates format and content adaptation through clustering-based learner modeling and ontology-driven personalization, while explicitly addressing the acquisition, preprocessing, and management of contextual, behavioral, and performance data to generate dynamic learner profiles. The study [P15] considers neither the functional requirements for acquiring, managing, and using contextual information nor the management of technological resources. The study [P16] does not consider the functional requirement for managing technological resources. In contrast, the study [P17] does not consider functional requirements for format and content adaptation, acquisition, management, and use of contextual information, or management of technological resources. In [P18], the proposed system focuses on acquiring and using student-provided physiological parameters and domain-specific information to generate ECG simulations and diagnostic feedback. It also considers the functional requirements for generating student histories and profiles and managing technological resources through cloud and edge deployment. However, it does not explicitly describe learning path adaptation; the acquisition, management, and use of contextual information; or the management of educational resources. In contrast, [P19] addresses format and content adaptation by using AI to generate educational activities, including quizzes, flashcards, and matching games. The platform also supports managing educational and technological resources, collecting and using student performance information, and tracking learning progress and activity history. Nevertheless, although personalization is discussed, the implementation of automated learning path adaptation and the acquisition, management, and use of contextual information are not specified. Inconsistent requirement coverage may lead to suboptimal design decisions, limited standardization, and lower quality in AI-based MALS with respect to requirement correctness and coverage.

Table 8. The main functional requirements considered in the selected primary studies are.

Functional Requirements		Selected Study																		
		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19
Types of adaptation	Learning path	☑	☑	☑	☐	☐	☑	☑	☐	☐	☐	☑	☐	☐	☐	☑	☑	☑	☐	☐
	Format	☑	☐	☑	☑	☑	☑	☐	☑	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☑
	Content	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑
Acquisition, management, and use of information	Contextual	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☐	☐	☐
	Student	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
	Dominion	☑	☑	☑	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Management of technological resources		☑	☐	☑	☑	☑	☑	☐	☑	☑	☑	☑	☐	☑	☑	☐	☑	☐	☑	☑
Management of educational resources		☑	☑	☑	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
Generation of student history		☑	☑	☑	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Generation of a student profile		☑	☑	☑	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

Regarding stakeholders, the selected primary studies predominantly involve university students; some also include teaching staff [P1], [P2], [P3], [P4], [P7], [P10], [P12], [P13], [P14], [P15], [P16], [P17], [P18], and [P19]. In contrast, developer and researcher participation is limited; only studies [P4], [P5], and [P8] explicitly consider these roles. Moreover, study [P4] is the only one that clearly reports the involvement of Information Technology (IT) staff in supervising the development process.

From an architectural perspective, the LR indicates a preference for layered and component-based software architectures, which approximately 58% of the selected primary studies adopted. This architectural choice contrasts with the client–server approach proposed in [P2], [P3], [P9], [P10], [P12], [P13], [P14], and [P17], underscoring the limited diversity of architectural styles explored in the literature. On the other hand, the selected primary studies [P1], [P3], [P6], [P7], [P9], [P12], [P14], and [P16] proposed cloud-based software architectures, whereas [P4], [P5], and [P18] incorporated emerging paradigms, including cloud, fog, and edge computing, into their architectural designs. In this case, only the study [P16] proposed a containerized microservices architecture. The selected primary studies do not incorporate architectural software design patterns such as micro-frontends and mobile-backend-as-a-service with serverless machine learning. This omission reveals a gap in adopting contemporary architectural approaches. Microservices and serverless have expanded rapidly in recent years, offering benefits such as agile development, independent deployments, strong fault isolation, and tech-stack flexibility. These capabilities can significantly reduce development lead times, optimize operational costs, and streamline infrastructure management.

The software architectures analyzed in these studies highlight key software engineering trade-offs, particularly around computation distribution, architectural decoupling, and adaptability mechanisms. Architectural proposals that depend on centralized cloud infrastructure and edge servers offload resource-intensive tasks and deep learning inference from mobile devices, thereby preserving battery life and sustaining processing performance. However, these systems are susceptible to network latency, privacy risks, and strict connectivity requirements. In contrast, edge-focused or device-local execution models enhance responsiveness and offline resilience but are limited by hardware constraints. Architecturally, traditional multi-layer and modular designs integrate well with standardized institutional environments such as learning management systems. Yet, they often exhibit tight coupling that limits independent scaling of adaptation and recommendation

modules. Conversely, contemporary microservices and decoupled approaches support agile deployment, fine-grained event streaming, and fault isolation for AI components, although they introduce considerable operational overhead and distributed-system complexity. Additionally, balancing machine learning models and generative AI, which offer high personalization accuracy but demand substantial computational resources and pose interpretability challenges for educators, with deterministic rule-based and ontological inference engines underscores a fundamental tension between dynamic system adaptability and explainable, predictable behavior.

Regarding evaluation approaches, the selected primary studies predominantly used system- and user-centered evaluations, as well as educational and learning-outcome evaluations. Specifically, [P1], [P6], [P12], and [P13] used a system- and user-centered approach; [P2] relied exclusively on an educational and learning-outcome approach; and [P4], [P5], [P14], and [P19] used an architectural approach. Several studies employed multiple methods: [P7], [P10], [P11], and [P17] used system- and user-centered and educational and learning-outcome approaches; [P8] used educational and learning-outcome, and ML-model and predictive methods; [P9], [P15], and [P16] employed system- and user-centered and architectural approaches. [P3] utilized a tri-fold evaluation strategy incorporating system- and user-centered, educational and learning-outcome, and ML-model and predictive evaluation approaches. More recently, [P18] combined ML-model and predictive and architectural evaluations. In contrast, [P19] focused primarily on architectural evaluation through performance testing and software quality analysis, without reporting a formal evaluation of user-centered or educational outcomes.

The selected primary studies [P1–P19] examined criteria across software quality attributes, educational outcomes, and user perceptions. All studies consistently evaluated core system capabilities, specifically functional suitability and interoperability [P1–P19]. In contrast, primary studies selectively assessed usability, security, performance efficiency, and architectural structure. For example, studies [P1], [P6], [P7], [P11], [P15], [P16], [P18], and [P19] conducted usability assessments. Several studies addressed some security attributes [P2], [P3], [P5], [P6], [P7], [P8], [P9], [P11], [P18], and [P19]. Studies [P6] and [P16] explicitly considered confidentiality, integrity, and availability, and [P19] considered only integrity. [P4], [P5], [P18], and [P9] considered performance efficiency and structural ontology quality metrics, respectively. Beyond technical functionality, multiple studies evaluated educational outcomes, such as academic performance and achievement [P2], [P3], [P7], [P8], [P11], [P17], and linguistic gains measured through controlled post-testing [P10]. Study [P11] presented a holistic pedagogical assessment, incorporating cognitive load, learning attitudes, and both short- and long-term achievement. User perceptions and Quality of Experience (QoE) were assessed through subjective feedback on system interaction, highlighting general satisfaction and effectiveness [P3], [P9], [P12], [P16], ease of use and interactivity [P10], comprehensive user-experience evaluations addressing learnability, responsiveness, convenience, and personalization in mobile contexts [P13], user acceptance and new technology adoption [P15], and user motivation [P17]. Additionally, [P18] addressed security through encrypted communication, performance efficiency through local inference, and deployment flexibility through its cloud–edge architecture. The study also reported preliminary feedback concerning learners' confidence and understanding of ECG interpretation, although no formal educational outcome assessment was conducted. In [P19], performance efficiency was examined through load and stress testing, while reliability, security, maintainability, code duplication, and test coverage were assessed using SonarQube.

Regarding the educational levels and application domains addressed in the selected primary studies, most focus exclusively on university-level students. The reported areas of study are mainly concentrated in Information Technology–related domains such as

Computer Science and programming languages (C++ and Java). Studies [P4] and [P5] use computational tasks and learning resources without specifying a particular academic discipline, while [P6] and [P11] report no specific area. The study [P10] provides English learning content for first-year undergraduate students tailored to their location and environment. The work [P12] developed a system to deliver multimedia learning content related to course database systems to students. The study in [P16] proposes a web-based adaptive learning platform to support Chinese vocabulary acquisition for non-native speakers. Nonetheless, no primary studies address AI-based MALS in other disciplinary contexts, such as mathematics or chemistry. Beyond these domains, [P18] addresses medical education, particularly electrocardiogram (ECG) interpretation, through interactive simulations and AI-based diagnostic feedback designed for medical students and healthcare trainees. Furthermore, [P19] extends the educational scope to secondary education by introducing a gamified learning platform that supports multiple school subjects through AI-generated educational activities. Nevertheless, the selected studies remain predominantly focused on computing-related disciplines, with limited representation of other academic fields, such as mathematics and chemistry. This concentration limits the generalizability of current architectural solutions across different educational contexts and highlights opportunities for future research to explore broader disciplinary domains and more diverse learner populations.

The findings reveal that AI is incorporated into MALS architectures through four main implementation approaches: (i) knowledge-based and symbolic AI, (ii) machine learning, (iii) hybrid AI, and (iv) generative AI. As summarized in Table 9, these approaches support different educational functions, including learner modeling, performance prediction, contextual reasoning, learning resource recommendation, content adaptation, diagnostic feedback, and educational content generation. Knowledge-based approaches primarily employ ontologies and rule-based mechanisms to represent contextual information and support adaptation decisions. In contrast, ML-based approaches use data-driven models for prediction, classification, and recommendation. Hybrid approaches combine knowledge representation with learning-based mechanisms, while generative AI adds the ability to produce educational resources. From an architectural perspective, AI is integrated through different mechanisms, ranging from adaptation engines and learner modeling components to dedicated inference and recommendation services. However, architectural detail varies across studies; some explicitly describe AI components and their interactions, while others report only their functional responsibilities.

Table 9. AI implementation approaches, architectural integration, and educational functions in MALS.

Implementation Approach	Study	AI Technique	Architectural Integration	Educational Function
C1. Knowledge-based and Symbolic AI	P9	Ontologies, Pellet, rules	OntoAdapt ontology and server-side reasoning engine	Learning object selection and format adaptation
	P10	Context-aware decision rules	IPCAL algorithm within context-aware and recommendation subsystems	Location-based learning material recommendation
	P11	IF/THEN/ELSE rules	Rule-based adaptation engine	Learning unit selection, sequencing, and activity adaptation
	P12	OWL-based knowledge representation	Ontology-supported profiling and adaptation engine	Content adaptation according to learner preferences and network conditions

	P13	Ontologies, SWRL rules	Learner modeling and adaptation reasoning mechanisms	Learner model updating and adaptation strategy selection
	P14	Ontologies, semantic reasoning, IF-THEN rules	Cloud-based ontologies and semantic inference engine	Context-aware resource selection and substitution
C2. Machine Learning-based Approaches	P1	ANN	Neural network processing learner and contextual variables	Concentration prediction and context-aware adaptation
	P2	Naïve Bayes	Learning-style classification linked to learner profiling	Learning material and sequence personalization
	P3	ANN, Random Forest	Performance prediction and feature-importance analysis	Academic performance prediction and learner grouping
	P4–P5	ML techniques (unspecified)	Intelligent tutoring systems incorporating learning-based adaptation	Personalized content and pedagogical strategies
	P7	DBSCAN	Contextual data processing using Firebase-stored information	Identification of contextual learning patterns
	P15	MLP, backpropagation	Neural network within the SMART learning environment	Prediction and recognition tasks
	P17	Collaborative filtering, SVD, NMF	Lesson and problem-solving recommendation modules	Personalized lesson ranking and solution recommendation
	P18	Feedforward neural network	AI classification module within a digital twin architecture	ECG classification and diagnostic feedback
C3. Hybrid AI Approaches	P6	ML, recommendation algorithms, rule-based reasoning	Adaptation engine integrating learning models and decision rules	Personalized learning paths, content, and interfaces
	P8	DBSCAN, K-means, PCA, ontologies	Learner modeling connected to ontology-based adaptation	Behavioral, contextual, and emotional personalization
	P16	Reinforcement learning (RL), OWL 2	RL recommendation and educational ontology services within a microservices architecture	Adaptive vocabulary and exercise recommendation
C4. Generative AI	P19	LLM, embeddings, semantic retrieval	Flask content-generation service integrating LangChain, FAISS, and an LLM API	Generation of quizzes, flashcards, and interactive learning materials

Although several studies identify specific AI techniques and describe their integration into software architectures, information regarding model configuration, training procedures, optimization strategies, and AI-specific evaluation remains heterogeneous. Consequently, the available evidence supports a comparative analysis of AI's architectural integration and educational functions. However, it does not consistently enable comparisons of ML pipelines, hyperparameter configurations, or experimental settings across studies. Furthermore, although some architectures process heterogeneous data sources, the available descriptions do not consistently show multimodal learning through joint modeling of different modalities. The incorporation of LLMs in [P19] represents a distinct implementation approach, in which AI supports semantic retrieval and the generation of educational materials rather than exclusively predicting learner characteristics or selecting existing resources. This development is also reflected in emerging research on the

integration of LLMs into learning management systems for instructional content and assessment generation [38–42]. Nevertheless, the evidence from [P19] does not demonstrate learner-specific adaptation, since the reported implementation primarily focuses on document-based educational content generation.

AI-based MALS is a specialized, emerging research area at the intersection of m-learning, educational technologies, software architectures, and artificial intelligence. This LR was deliberately structured to ensure rigor and focus, emphasizing studies that specifically address the architectural dimensions of AI-based MALS. As a result, numerous related works were excluded due to insufficient direct relevance. The final corpus comprises a select set of mature, focused contributions, reflecting the specificity of the inclusion criteria rather than any limitation of the search methodology. These selected studies contain the principal approaches and architectural paradigms that continue to shape advancements in the field. The identified research gaps remain substantial, underscoring under-explored avenues within educational computing. Based on the results, the selected primary works [P1–P19] have limitations due to inadequate architectural support, including a lack of standardization and limited coverage and correctness of requirements. Our findings show conceptual fragmentation and extend this by showing the absence of reference software architecture capable of systematically integrating all necessary aspects for developing AI-based MALS to facilitate adaptive learning in mobile distributed systems. This moves the research frontier from conceptual gaps to architectural ones, suggesting the need for unifying frameworks that combine AI-based adaptivity, contextual sensing, and distributed deployment mechanisms.

6. Threats to Validity

Tricco et al. [43] identify three types of bias that can occur in an LR: identification bias, selection bias, and data acquisition bias. Publication bias may also arise if the methods for identifying relevant studies lack sufficient sensitivity, potentially resulting in the omission of pertinent research [44]. To address these concerns, the methodological process outlined in [34] was followed, and additional measures were considered throughout the review. To reduce identification and selection bias, the research team used a collaborative, iterative approach to plan the review, define the search strategy, screen retrieved publications, and select primary studies. The researchers discussed discrepancies to ensure consistent application of the inclusion and exclusion criteria. The search was conducted using recognized scientific digital libraries, and the selection was restricted to peer-reviewed journal articles and conference proceedings. Although this decision was intended to prioritize academically reviewed research, it may have excluded relevant evidence from gray literature, introducing a potential source of publication bias.

Data acquisition and classification bias may also affect the validity of the findings, particularly because the review examines heterogeneous architectural descriptions, functional requirements, quality attributes, evaluation approaches, and AI implementation mechanisms. Differences in terminology and reporting detail across primary studies may lead to inconsistencies in data extraction and categorization. To mitigate this threat, the extracted information was analyzed collaboratively, and classifications were based on the descriptions and evidence reported in the primary studies. Nevertheless, some degree of interpretation remains unavoidable when architectural or technical characteristics are not explicitly defined by the original authors.

Finally, the findings are limited by the selected digital libraries and the search period. Relevant studies published outside the selected sources or after the search date may not have been identified. Moreover, the diversity of educational contexts, technological approaches, and evaluation methods reported in the primary studies may limit the generalizability of the findings. Consequently, the results should be interpreted in relation to the

available evidence rather than as an exhaustive representation of all possible architectures for AI-based mobile adaptive learning systems.

7. Conclusions and Future Work

This study reviewed software architectures for developing AL-based MALS. The main objective was to characterize and compare these software architectures across critical aspects, including requirements, quality attributes, involved stakeholders, core user services, architectural views and styles, supporting technologies, and AI techniques. Our study revealed the following findings: 1. Only three selected primary studies (16%) address the main functional requirements for AI-based MALS. 2. Most reviewed studies focus on university students as stakeholders, with fourteen including teachers, revealing limited research on elementary and high school students in core subjects such as mathematics and physics. 3. Software architectures in this domain are predominantly layered, client-server, and cloud-centric, with a growing trend toward multi-tier cloud–fog–edge paradigms to meet real-time responsiveness and distributed context-processing requirements. 4. The selected primary studies predominantly use system- and user-centered, and educational and learning-outcome approaches to evaluate proposed software architectures for AI-based MALS based on distinct criteria related to software quality attributes, educational outcomes, and user perceptions. 5. Most AI-based MALS rely on Machine Learning-based Approaches (47.4%) and Knowledge-based and Symbolic AI (31.6%) to perform adaptive learning.

Inconsistent coverage of functional and non-functional requirements, and limited integration of AI as an architectural organizing principle, may lead to poor architectural design decisions. Because software architecture defines component interactions, data storage, and business logic, these omissions can lead to costly refactoring, flawed data models, unclear component boundaries, integration challenges, user dissatisfaction, system unreliability, technical debt, security vulnerabilities, and compliance issues. These challenges hinder effective, standardized development of AI-based MALS and often prevent successful implementation of adaptive learning. Therefore, the principal finding is that the reviewed literature lacks a reference software architecture that systematically addresses all architecturally significant requirements for developing AI-based MALS. This LR gives researchers insights into the advantages, limitations, and challenges in the analyzed studies and helps development teams identify essential requirements for AI-based MALS in real-world scenarios. For future work, we propose designing a reference software architecture for AI-based MALS that explicitly integrates AI-driven student, domain, and context modeling; ontology-based adaptation; data security mechanisms; usability-oriented evaluation metrics; emerging architectural software design patterns; and Large Language Models to address the fragmentation identified in the current literature.

Author Contributions: Conceptualization and design, E.D.G. and E.L.D.; methodology, E.D.G., E.L.D. and S.D.I.; validation, E.D.G., E.L.D., S.D.I., M.A.M.N., J.I.V.G. and J.d.l.C.; formal analysis, E.D.G., E.L.D., S.D.I., M.A.M.N., J.I.V.G. and J.d.l.C.; investigation, E.D.G., E.L.D. and S.D.I.; resources, E.D.G. and E.L.D.; data curation, E.D.G., E.L.D., S.D.I., M.A.M.N., J.I.V.G. and J.d.l.C.; writing—original draft preparation, E.D.G., E.L.D., S.D.I., M.A.M.N., J.I.V.G. and J.d.l.C.; writing—review and editing, E.D.G., E.L.D., S.D.I., M.A.M.N., J.I.V.G. and J.d.l.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mejeh, M.; Rehm, M. Taking adaptive learning in educational settings to the next level: Leveraging natural language processing for improved personalization. *Educ. Technol. Res. Dev.* **2024**, *72*, 1597–1621.
2. Garzón, J.; Burgos, D.; Kinshuk; Tlili, A. Mobile learning significantly enhances student learning gains: A meta-analysis and research synthesis. *Comput. Educ.* **2025**, *238*, 105415.
3. Dong, Z.; Chiu, M.; Zhou, S.; Zhang, Z. The effect of mobile learning on school-aged students' science achievement: A meta-analysis. *Educ. Inf. Technol*. **2024**, *29*, 517–544.
4. Yaghmour, S.; Qureshi, M.I.; Ullah, I.; Perumal, R.K.; Khan, M.M. AI-driven adaptive learning systems in mobile education: A systematic review of personalization strategies, effectiveness, and user interaction. *Int. J. Interact. Mobile Technol*. **2025**, *19*, 70–86. https://doi.org/10.3991/ijim.v19i19.57695.
5. Ennouamani, S.; Akharraz, L.; Mahani, Z. Integrating ICT in Education: An Adaptive Learning System Based on Users' Context in Mobile Environments. In *Proceedings of the ICBDSDE: International Conference on Big Data and Smart Digital Environment, Casablanca, Morocco, 29–30 November 2019*; Springer: Cham, Switzerland; pp. 15–19.
6. Pugliese, L. Adaptive Learning Systems: Surviving the Storm, Educause Review. 2016. Available online: https://er.educause.edu/articles/2016/10/adaptive-learning-systems-surviving-the-storm (accessed on 26 August 2026).
7. Ennouamani, S.; Mahani, Z. A comparative study of the learner model in adaptive mobile learning systems. In *Proceedings of the 2nd International Conference on Networking, Information Systems & Security, Rabat, Morocco, 27–29 March 2019*; Association for Computing Machinery: New York, NY, USA; pp. 1–11.
8. Awwad, A.M. A universal design for an adaptive context-aware mobile cloud learning framework using machine learning. *J. Mobile Multimedia*. **2023**, *19*, 707–738. https://doi.org/10.13052/jmm1550-4646.1934.
9. Adewale, O.S.; Agbonifo, O.C.; Ibam, E.O.; Makinde, A.I.; Boyinbode, O.K.; Ojokoh, B.A.; Olatunji, S.O. Design of a personalized adaptive ubiquitous learning system. *Interact. Learn. Environ.* **2022**, *32*, 208–228. https://doi.org/10.1080/10494820.2022.2084114.
10. Adnan, M.; AlSaeed, D.H.; Al-Baity, H.H.; Rehman, A. Leveraging the power of deep learning technique for creating an intelligent, context-aware, and adaptive m-learning model. *Complexity.* **2021**, *2021*, 5519769. https://doi.org/10.1155/2021/5519769.
11. Mohiuddin, K.; Miladi, M.N.; Ali-Khan, M.; Abdul-Khaleel, M.; Nasr, O.A.; Shahwar, S.; Islam, M.A. Mobile learning new trends in emerging computing paradigms: An analytical approach seeking performance efficiency. *Wireless Commun. Mobile Comput.* **2022**, *2022*, 6151168. https://doi.org/10.1155/2022/6151168.
12. Mohiuddin, K.; Fatima, H.; Khan, M.A.; Khaleel, M.A.; Nasr, O.A.; Shahwar, S. Mobile learning evolution and emerging computing paradigms: An edge-based cloud architecture for reduced latencies and quick response time. *Array* **2022**, *16*, 100259. https://doi.org/10.1016/j.array.2022.100259.
13. Akinwole, P.E.; Boyinbode, O.K.; Kinga, M.T.; Olotu, P.K. Implementation of a personalized adaptive mobile learning system. *Int. J. Innov. Sci. Res. Technol.* **2024**, *9*, 570–577. https://doi.org/10.38124/ijisrt/IJISRT24APR535.
14. Adnan, M.; Habib, A.; Ashraf, J.; Mussadiq, S. Cloud-supported machine learning system for context-aware adaptive M-learning. *Turk. J. Electr. Eng. Comput. Sci.* **2019**, *27*, 2798–2816. https://doi.org/10.3906/elk-1811-196.
15. Fezazi, N.E.E.; Miloud, S.E.; Oumaira, M.D.; Daoudi, M. Adaptive M-Learning content on the Moodle platform using M-Learner modeling approach, ontologies, and machine learning techniques, *Educ. Process Int. J.* **2025**, *16*, e2025218. https://doi.org/10.22521/edupij.2025.16.218.
16. Abech, M.; Costa, C.A.; Barbosa, J.L.V.; Rigo, S.J.; Righi, R.R. A model for learning objects adaptation in light of mobile and context-aware computing. *Pers Ubiquit Comput.* **2016**, *20*, 167–184. https://doi.org/10.1007/s00779-016-0902-3.
17. Yao, C.B. Constructing a user-friendly and smart ubiquitous personalized learning environment by using a context-aware mechanism. *IEEE Trans. Learn. Technol.* **2017**, *10*, 104–114. https://doi.org/10.1109/tlt.2015.2487977.
18. Zhang, B.; Yin, C.; David, B.; Chalon, R.; Xiong, Z. A context-aware mobile system for work-based learning. *Comput. Appl. Eng. Educ*. **2016**, *24*, 263–276. https://doi.org/10.1002/cae.21704.
19. Davcev, D.; Jakimovski, G.; Scepanovic, S. Model of M-Learning by multimedia content delivery from mCloud to mobile devices. In *Teaching and Learning in a Digital World. ICL 2017*; Advances in Intelligent Systems and Computing; Auer, M., Guralnick, D., Simonics, I., Eds.; Springer: Cham, Switzerland, 2018; Volume 715. https://doi.org/10.1007/978-3-319-73210-7_92.

20. Vesin, B.; Klašnja-Milićević, A.; Ivanović, M. Application of semantic web technologies to facilitate use of E-Learning system on mobile devices. In *Smart Education and e-Learning 2016*; Smart Innovation, Systems and Technologies; Uskov, V., Howlett, R., Jain, L., Eds.; Springer: Cham, Switzerland, 2016; Volume 59. https://doi.org/10.1007/978-3-319-39690-3_42.
21. Muhammad, S.; Admodisastro, N.; Osman, H.; Mohd Ali, N. Mudahnya BM: A Context-Aware Mobile Cloud Learning Application Using Semantic-Based Approach. In *Advances in Visual Informatics. IVIC 2021*; Lecture Notes in Computer Science; Badioze Zaman, H., Ed.; Springer: Cham, Switzerland, 2021; Volume 13051. https://doi.org/10.1007/978-3-030-90235-3_20.
22. Sungkur, R.K.; Maharaj, M.S. Design and implementation of a SMART Learning environment for the Upskilling of Cybersecurity professionals in Mauritius. *Educ. Inf. Technol.* **2021**, *26*, 3175–3201. https://doi.org/10.1007/s10639-020-10408-9.
23. Gong, Y.; Cai, K. A Containerized Microservices Architecture with Reinforcement Learning for Scalable, Adaptive Learning. *J. Web Eng.*. **2025**, *24*, 1203–1230. https://doi.org/10.13052/jwe1540-9589.2482.
24. Drissi, S.; Chefrour, A.; Boussaha, K.; Zarzour, H. Exploring the effects of personalized recommendations on student's motivation and learning achievement in gamified mobile learning framework. *Educ. Inf. Technol*. **2024**, *29*, 15463–15500. https://doi.org/10.1007/s10639-024-12477-6.
25. Flores-Martin, D.; Díaz-Barrancas, F.; Pardo, P.J.; Berrocal, J.; Murillo, J.M. ECGTwinMentor: A digital twin-based system for personalized ECG learning and support. *SoftwareX* **2025**, *32*, 102398. https://doi.org/10.1016/j.softx.2025.102398.
26. Achour, K.; Lachgar, M.; Elloubab, A.; Ait Ouahda, Y.; Laanaoui, M.D.; Ourahay, M. EduXgame: Gamified learning for secondary education. *Softw. Impacts* **2025**, *24*, 100761. https://doi.org/10.1016/j.simpa.2025.100761.
27. Vallejo-Correa, P.; Monsalve-Pulido, J.; Tabares-Betancur, M. A systematic mapping review of context-aware analysis and its approach to mobile learning and ubiquitous learning processes. *Comput. Sci. Rev.* **2021**, *39*, 100335. https://doi.org/10.1016/j.cosrev.2020.100335.
28. Hsu, Y.C.; Ching, Y.H. A review of models and frameworks for designing mobile learning experiences and environments. *Can. J. Learn. Technol.* 2015, 41. https://doi.org/10.21432/T2V616.
29. Gumbheer, C.P.; Khedo, K.K.; Bungaleea, A. Personalized and adaptive context-aware mobile learning: Review, challenges and future directions. *Educ. Inf. Technol.* **2022**, *27*, 7491–7517. https://doi.org/10.1007/s10639-022-10942-8.
30. Kabudi, T.; Pappas, I.O.; Olsen, D.H. AI-enabled adaptive learning systems: A systematic mapping of the literature. *Comput. Educ. Artif. Intell.* **2021**, *2*, 100017. https://doi.org/10.1016/j.caeai.2021.100017.
31. Kumar, B.A.; Sharma, B.; Nakagawa, E.Y. Context aware mobile learning: A systematic mapping study. *Educ. Inf. Technol.* **2021**, *26*, 2033–2052. https://doi.org/10.1007/s10639-020-10347-5.
32. Gligorea, I.; Cioca, M.; Oancea, R.; Gorski, A.T.; Gorski, H.; Tudorache, P. Adaptive learning using artificial intelligence in e-Learning: A literature review. *Educ. Sci.* **2023**, *13*, 1216. https://doi.org/10.3390/educsci13121216.
33. Nepomuceno, A.R.; Domínguez, E.L.; Isidro, S.D.; Nieto, M.A.M.; Meneses-Viveros, A.; Calleja, J. Software architectures for adaptive mobile learning systems: A systematic literature review. *Appl. Sci.* **2024**, *14*, 4540. https://doi.org/10.3390/app14114540.
34. Kitchenham, B. *Procedures for Performing Systematic Reviews*; Technical Report; Keele University: Keele, UK, 2004; pp. 1–33.
35. Dybå, T.; Dingsøyr, T. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.* **2008**, *50*, 833–859. https://doi.org/10.1016/j.infsof.2008.01.006.
36. Plaza, A.M.; Díaz, J.; Pérez, J. Software architectures for health care cyber-physical systems: A systematic literature review. *J. Softw. Evol. Process*. **2018**, *30*, e1930. https://doi.org/10.1002/smr.1930.
37. *ISO/IEC/IEEE 24765:2017*; International standard—Systems and software engineering—Vocabulary (ISO/IEC/IEEE 24765:2017). ISO: Geneva, Switzerland;, 2017. Available online: https://standards.ieee.org/standard/24765-2017.html (accessed on 25 August 2025).
38. Cao, Z.; Zhang, X.; Zeng, D.D. Large language models: Technology, intelligence, and thought. *Front. Eng. Manag.* **2025**, *12*, 710–715. https://doi.org/10.1007/s42524-025-5004-3.
39. Shi, Y.; Yu, K.; Dong, Y.; Chen, F. Large language models in education: A systematic review of empirical applications, benefits, and challenges. *Comput. Educ. Artif. Intell.* **2026**, *10*, 100529. https://doi.org/10.1016/j.caeai.2025.100529.
40. Naveed, H.; Khan, A.U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Akhtar, N.; Barnes, N.; Mian, A. A comprehensive overview of large language models. *ACM Trans. Intell. Syst. Technol.* **2025**, *16*, 1–72. https://doi.org/10.1145/3744746.
41. Lopez-Gazpio, I. Integrating large language models into accessible and inclusive education: Access democratization and individualized learning enhancement supported by generative artificial intelligence. *Information* **2025**, *16*, 473. https://doi.org/10.3390/info16060473.
42. Pesovski, I.; Santos, R.; Henriques, R.; Trajkovik, V. Generative AI for customizable learning experiences. *Sustainability* **2024**, *16*, 3034. https://doi.org/10.3390/su16073034.

43. Tricco, A.C.; Tetzlaff, J.; Sampson, M.; Fergusson, D.; Cogo, E.; Horsley, T.; Moher, D. Few systematic reviews exist documenting the extent of bias: A systematic review. *J. Clin. Epidemiol.* **2008**, *61*, 422–434. https://doi.org/10.1016/j.jclinepi.2007.10.017.
44. Nightingale, A. A guide to systematic literature reviews. *Surgery* **2009**, *27*, 381–384. https://doi.org/10.1016/j.mpsur.2009.07.005.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.