International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

Acadbot: AI-Driven Automation of Academic Services

Nandhitha B R[1], Ayesha Siddiqah Khanam[2], Bharath Moger[3], Bhavana G M[4], Keerthana M D[5]
[1]Associate professor Nandhitha B R, Dept of ISE, Malnad College of Engineering, Hassan, Karnataka India.
[2,3,4,5]UG Scholar Ayesha Siddiqah Khanam, Dept of ISE, Malnad College of Engineering, Hassan, Karnataka, India.
Emails: *brn@mcehassan.ac.in[1], ayesha4siddiqah@gmail.com[2], 03bharatmgr@gmail.com[3], bhavanagm004@gmail.com[4], keerthanagowda9336@gmail.com[5]*

Abstract
Accessing timely and accurate information in academic institutions is often challenging, as students, faculty, and administrators struggle to locate dynamic data such as event schedules, meeting details, and academic notifications, which are typically scattered across multiple systems. Traditional university portals lack intuitive, centralized, and fast query mechanisms, leading to inefficiency, delays, and user dissatisfaction. To address this issue, we present Acadbot, a full-stack web application built using the MERN (MongoDB, Express, React, Node.js) stack, offering a unified and user-friendly dashboard for managing academic information, supported by a secure Role-Based Access Control (RBAC) system that ensures proper authorization for different user groups. The core innovation of this project is its lightweight, database-driven AI chat assistant, which deviates from conventional systems that depend on large vector embeddings, FAISS indexes, and separate retrieval pipelines. Instead, Acadbot employs a practical heuristic-based retrieval engine that performs case-insensitive, token-based, and typo-tolerant regex searches directly on the live MongoDB database. By querying real-time operational data rather than relying on preprocessed vector stores, the system reduces complexity and avoids issues related to outdated or unsynchronized information. This approach enables Acadbot to deliver fast, accurate, and context-aware responses tailored to academic environments. The paper further discusses the system architecture, the implementation of the heuristic retrieval algorithm, and the benefits of adopting this efficient approach for domain-specific academic chatbots. Keywords: MERN Stack, AI Chatbot, Academic Portal, Heuristic Retrieval, Information Retrieval, Role-Based Access Control (RBAC).
Keywords: *Academic portal; Ai chatbot; Heuristiretrieval; Information retrieval; Mern stack.*

1. Introduction
In modern academic institutions, sharing information effectively is a crucial and ongoing challenge. Universities create a constant stream of time-sensitive data, which includes event schedules, academic deadlines, meeting announcements, and course-related documents. This information is often spread out across different departmental websites, hard-to-navigate portals, or scattered through email inboxes. This fragmentation results in a poor to solve this issue, we developed and implemented Acadbot, a full-stack web application built on the MERN (MongoDB, Express, React, Node.js) stack. Acadbot creates a unified portal where all key academic functions—from managing events and distributing schedules to sharing documents and sending university-wide notifications—can be handled and accessed. A key feature of our system is a strong Role-Based Access Control (RBAC) model. Thisexperience for everyone involved. Students may miss important deadlines, faculty face excessive administrative tasks, and administrators struggle to maintain consistent communication. Current Learning Management Systems (LMS) are typically complex and lack the intuitive, on- demand query interfaces that users expect today. Without a centralized, responsive, and easy-to-query "single source of truth," there is a significant information gap that leads to inefficiency, user frustration, and missed opportunities. model customizes the app's functionality and data visibility according to the

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

specific needs of different user roles: administrators, faculty, and students. To solve this issue, we developed and implemented Acadbot, a full-stack web application built on the MERN (MongoDB, Express, React, Node.js) stack. Acadbot creates a unified portal where all key academic functions—from managing events and distributing schedules to sharing documents and sending university-wide notifications—can be handled and accessed. A key feature of our system is a strong Role-Based Access Control (RBAC) model. This model customizes the app's functionality and data visibility according to the specific needs of different user roles: administrators, faculty, and students. It ensures that sensitive functions, like creating new events or uploading official documents, are limited to authorized users while general information stays accessible. The main focus of this paper is the design of the system's AI-powered chat [1].

2. Related Works

Integrated Knowledge Management System Based Mentoring for new University Staff Development Topic:

This paper, written by Puangpet Srivichai, Komsak Meksamoot, Anchalee Jengjalern, and Nopasit Chakpitak, presents a framework for a system that combines a Knowledge Management System (KMS) with a mentoring process. The main goal is to improve and speed up the professional development of new university staff. The authors tackle a major issue faced by academic institutions: the loss of experienced staff due to retirement, which creates a long gap before new hires reach a similar level of expertise. Their proposed system aims to effectively and efficiently close this knowledge gap. The framework is based on the idea that a structured, technology-driven environment can help transfer both tacit and explicit knowledge from experienced faculty to new staff. The system matches new staff with established mentors who are recognized experts in their fields. This matching goes beyond formal guidance; it helps facilitate a variety of professional development methods, including mentoring, training sessions, coaching, and counseling. The KMS component acts as a knowledge repository, containing user profiles, educational content, and records of various professional development activities. This material is organized for easy access, offering new staff the information and resources needed to navigate their roles and advance their careers. A key contribution of this work is its focus on a facilitated approach. The system includes a coordinator role to ensure effective mentoring relationships and active engagement from new staff members. This human aspect is essential to support the technological structure, making sure the system functions as an active tool for building professional relationships and promoting a culture of continuous learning. The authors collected data through interviews with academic staff from both a newly established and a well-established university to inform their design, showcasing a practical base for their conceptual model. They conclude that this integrated approach is a valuable resource for educational organizations, providing a solid solution to cut down the time needed for new staff development and address the larger issues of staff succession and knowledge retention. The paper effectively argues that a combined KMS and mentoring program offers a more structured and quicker path to expertise for new university personnel compared to traditional, less formal methods [2].

Efficiency Engine: Designing and Implementing an Academic Management System

Topic:

The paper "Efficiency Engine: Designing and Implementing an Academic Management System," by Nishu Sethi and Anshu Malhotra, outlines the creation and deployment of a software solution for educational institutions. This system is seen as a centralized platform that digitalizes and automates a range of administrative and academic processes. The main goal is to improve efficiency, accuracy, and communication across the entire institution, benefiting administrators, faculty, staff, and students. The system is built with a modular design, integrating key functionalities that are important for daily college or university operations. These modules cover areas like student information management, course registration, attendance tracking, and maintaining

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

academic records. By bringing these functions into one platform, the system eliminates manual paperwork and reduces the likelihood of human error—common issues in traditional administrative workflows. The paper emphasizes how this digital approach allows real-time access to information, so all users can get up-to-date data when they need it, which improves decision-making speed and quality. A significant part of the paper addresses the practical challenges and solutions involved in implementing such a system. The authors stress the importance of security, explaining the measures taken to protect data, verify user identities, and ensure privacy. These considerations are essential given the sensitive nature of academic and personal information handled by the system. The paper also highlights the need for a well-planned implementation strategy that includes user training and ongoing technical support. These components are crucial for making sure the system is not only technically sound but also widely adopted by users, maximizing its potential benefits. The "Efficiency Engine" is presented as a transformative tool that can streamline operations, improve communication, and create a more integrated and efficient academic environment. The authors provide a practical guide for institutions looking to modernize their administrative and academic functions through a strong and secure digital platform [3].

Academic Management System Admin Reference Manual Topic:

The "Academic Management System Admin Reference Manual," created by the ICAR- Indian Agricultural Statistics Research Institute (IASRI) for agricultural universities, is a practical guide to a web-enabled management system. The focus of the paper is on the administrative features of this system, which was developed under the National Agricultural Higher Education Project (NAHEP). Its aim is to improve governance and management of academic activities in these universities by offering a clear, automated framework for various tasks. The manual guides administrators through the system's different modules and functionalities. It outlines the specific responsibilities tied to various user roles, such as Dean, Registrar, Professor, and Student, along with the Administrator. The paper details a range of administrative tasks that the system automates, including the approval of registered students and faculty, adding new disciplines, and managing courses and reports. A key feature highlighted is the system's built-in workflow, which automatically informs users of their next steps. This is vital for ensuring that complex processes, like student registration or course approval, are completed on time and in an organized manner, improving overall efficiency. The document provides a step-by-step guide for common administrative tasks, such as updating administrator profiles and adding new faculty members. It also explains how administrators can change their roles to manage other user profiles, which is helpful if a user cannot complete their own work. The manual describes search functions that help administrators quickly locate student and faculty profiles. Essentially, this paper serves as an operational guide, showing how the system cuts down the time and effort involved in manual academic processes. It highlights the system's modular design and how it enhances academic management efficiency through automation and clear, role-based duties. The work emphasizes the importance of a user-friendly digital platform for modern educational governance [4].

Lecture Management Notification System. Topic:

The paper "LectureManagement Notification System," written by Prof. VaishaliGedam, Vinay Nikose, Neha Dhargave, Tejas Kripal, Shruti Patil, Ayush Jharbade, and Khushi Gawande, introduces an Android application designed as a complete solution for college management. The main theme of this work is the use of mobile technology to streamline administrative tasks and improve communication within educational institutions. The application is presented as a single platform that meets the diverse needs of various user groups, including administrators, faculty, students, and parents. The system is founded on the idea of providing a single digital platform for all users. For administrators, the application includes modules for managing admissions, collecting fees, keeping student records,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

and overseeing staff. By automating these complex tasks, the system frees up time and resources for strategic planning and institutional growth. Faculty members gain from tools like attendance management, grade books, course planning, and communication features, which help them manage classes efficiently and interact better with students. This set of tools allows educators to spend more time on teaching and less on administrative tasks. For students, the application serves as a central hub for their academic life. They can access course materials, lecture notes, assignments, and other resources, helping them stay organized, track their academic progress, and engage with their coursework. The paper also points out the collaborative features of the system, such as discussion forums and chats, which promote a more interactive learning experience. A unique feature of the system is the real-time notification module for parents. This ensures that parents receive updates on their children's academic progress and attendance, strengthening the important relationship among parents, teachers, and students. The application is a powerful tool for improving the efficiency and effectiveness of educational institutions [5].

3. Methodology

Our project, Acadbot, is a full-stack web application built using the MERN stack. This section describes the technologies and architectural patterns we used to create the system, focusing on its three main layers: the frontend, the backend, and the database. It also breaks down the Role-Based Access Control (RBAC) framework that secures the application and customizes the user experience.

3.1 Full-Stack System Architecture

The system follows a classic three-tier architecture, separating presentation, logic, and data [6].

3.2 Frontend (Client-Side)

The frontend is a modern, responsive Single Page Application (SPA) developed with the React.js library. This component-based approach allows for reusable UI elements, which improves maintainability and scalability. The structure of the application is defined in App.js, which serves as the main entry point for all visual components. These components represent distinct features of the application, such as Dashboard.js for the user's home page, Login.js and Register.js for authentication, and other feature-specific components like Events.js, Schedules.js, Meetings.js, Documents.js, and the AIChat.js interface..Client-side navigation is handled by react-router- dom, enabling seamless transitions between views without a full page reload. The routing logic is defined within the <Router>, <Routes>, and <Route> components, mapping URL paths to their corresponding React components. A key part of our security model is implemented with the custom <ProtectedRoute> component, which serves as a guard forsensitive routes. It checks localStorage for a valid token; ifthe token is not there, the user is redirected to the login page using the <Navigate to="/" replace component, preventing unauthorized access to dashboard pages. State management occurs at the component level using React Hooks. The useState hook manages local state, such as user inputs in forms (for example, username and password in Login.js) and stores data fetched from the backend (like the events array in Events.js). The useEffect hook manages by using mobile technology to create a connected academic environment side effects, especially for fetching initial data when a component mounts (for example, fetchEvents() in Events.js) and checking the user's authentication status when the application loads (useEffect in App.js). Communication with the backend API happens through asynchronous axios requests. For protected endpoints, the stored JWT is retrieved from localStorage and included as a Bearer token in the Authorization header. We implement robust error handling within catch blocks to capture API or network failures and display user-friendly error messages, such as setError(errorMsg) in the Login component.

3.3 Backend (Server-Side)

The backend API is a RESTful service built on Node.js, an asynchronous, event-driven runtime, along with the Express.js web framework. The server is set up as an ES Module ("type": "module" in package.json), allowing the use of modern import/export syntax across the codebase. The main

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

server.js file acts as the application's entry point, responsible for initializing the database connection, setting up middleware, and mounting the modular route handlers. The API's structure is highly modular. All routes are namespaced under /api (for instance, /api/auth, /api/events, /api/documents). Each feature's routes are defined in separate files within the routes/ directory (like authRoutes, eventRoutes, documentRoutes) and then imported and applied in server.js using app.use(). This separation keeps the main server file clean and makes the API easier to maintain and extend. A critical aspect of the backend is its middleware pipeline. CORS: The cors middleware is configured with a strict whitelist of allowed origins (for example, http://localhost:3000). This is an essential security measure to ensure only the React frontend can make requests to the API, preventing Cross-Origin Resource Sharing attacks from unauthorized domains. Body Parsing: The express.json() middleware parses incoming JSON payloads from the frontend, making them available on the req.body object for all POST and PUT requests. Static Files: express.static() serves files from the /uploads directory. This middleware makes user- uploaded documents accessible to the frontend via adirect URL (like http://localhost:5000/uploads/document.pdf).File Uploads: The multer library is set up specifically for document uploads. It uses diskStorage to define the destination directory (uploads) and a custom filename generator (Date.now() + '-' + file.originalname) to avoid filename conflicts. This multer middleware [7].

3.4 Database (persistence)

The persistence layer of our application uses MongoDB, a NoSQL, document-oriented database. This choice was intentional, as MongoDB's flexible, BSON (binary JSON) document model directly matches the JavaScript objects used throughout our MERN stack, making data handling simpler. The application connects to the MongoDB instance using the MONGO_URI connection string, securely stored in an .env file rather than hardcoded, which follows security best practices. The server.js file also includes error handling for the initial database connection to ensure the server logs a clear error and does not start if the database is unavailable. We use Mongoose as an Object Data Modeling (ODM) library to communicate with MongoDB. Mongoose provides an effective abstraction layer, allowing us to define strict schemas for our data, ensuring data integrity and offering built-in validation. All data models, including User, Event, Schedule, Meeting, Document, and Notification, are defined in the models/ directory. For example, the User schema defines the role field as an enum restricted to 'admin', 'faculty', or 'student', and specifies the username as required and unique. Mongoose's middleware capabilities are central to our security model. The User schema uses a pre('save') hook to automatically hash a user's password with bcryptjs before it is saved to the database. This guarantees that no plain-text passwords are stored, even in case of a database breach. Additionally, we define a custom instance method on the user schema, compare Password, which includes the bcrypt. compare logic. This method is then used in the login route (/api/auth/login) to securely validate a user's credentials, showing an object-oriented approach to our data modeling [8].

3.5 Role-Based Access Control (RBAC) Framework

Security and authorization are managed using a solid RBAC system.

- **User Roles:** The User model defines three roles: 'admin','faculty,' and 'student. Admin has full control access whereas faculty has few controls itself and students can only view.
- **Authentication:** Users register by hashing their passwords with bcryptjs and authenticate using JWT.
- **Authorization:** A custom authMiddleware protects API endpoints. This middleware decodes the user's JWT, checks their role, and blocks access if they are unauthorized.
- **Example:** The route to post a new document is restricted to 'admin' and 'faculty' roles: router.post('/',authMiddleware(['admin', 'faculty']), ...). The frontend UI also reflects this, hiding the upload form from 'student'

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

users Shown in Figure 1 [9].

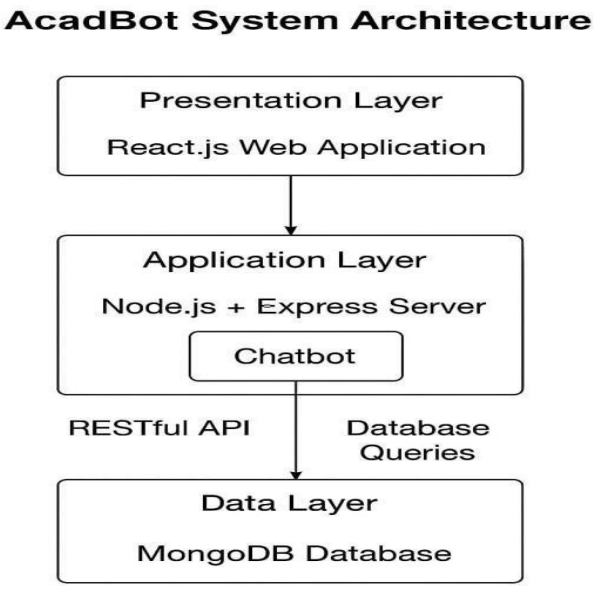

Figure 1 System Architecture

4. Results

The deployment of the Acadbot application produced several important outcomes in both system functionality and the performance of our new retrieval engine.

System Functionality and RBAC Verification the Acadbot MERN-stack application was successfully implemented and operates as a cohesive academic portal. All main components, from user registration to the main dashboard, were fully functional. The Role-Based Access Control (RBAC) system was a key part of our testing and was shown to be effective.Users with 'admin' or 'faculty' roles could successfully access content creation forms. This included posting new events, uploading documents via the multer endpoint, and adding meetings. On the other hand, users with the 'student' role were correctly limited to a view-only interface for these components. This confirms that our security model, enforced at both the API level through authMiddleware and on the client side through conditional rendering, is working as intended. 6.Heuristic-Based Retrieval (HBR) Engine Performance The main results of this study relate to the performance of the Heuristic-Based Retrieval (HBR) engine, which we used instead of a traditional RAG pipeline. The main results of this study relate to the performance of the Heuristic-Based Retrieval (HBR) engine, which we used instead of a traditional RAG pipeline.

Result 1: **Real-Time Data Accuracy:** The biggest advantage of the HBR engine is its 100% data freshness. In testing, any new document, event, or schedule created by an admin was immediately searchable by the AI Chatbot. This is a direct result of querying the live MongoDB collections, which completely resolves the data-staleness problem found in a separate, pre-computed vector index [10].

Result 2: Low Latency and Zero Cost: The retrieval process consists of a single, optimized MongoDB query using case-insensitive regex and token matching, making it extremely fast. This avoids costs from external API calls and network delays linked to embedding generation (for example, to OpenAI) and LLM calls, which were part of the initial RAG plan. The response to the user is nearly instantaneous and incurs no monetary cost per query.

Discussion and Trade-Offs While the HBR approach is successful, it has clear trade-offs. Our discussion should recognize that this engine is excellent for factual, keyword- based retrieval (for example, "meetings today,""exam schedule") but completely lacks semantic understanding. A query like "What should I do if I miss an exam?" will not work because it doesn't include the keywords "event," "schedule," or "meeting" that the engine looks for. Additionally, the response is a formatted data list, not a natural-language, conversational reply. Thus, our HBR engine does not serve as a universal replacement for RAG but rather as a highly specialized tool. It is clearly better for cases where data freshness and low operational cost are more important than conversational depth or semantic search features. The results are summarized in the table below. Feature, Planned RAG (Vector Search), Implemented HBR (Heuristic Search) Data Source, Static FAISS Index, Live MongoDB Database Data Freshness, Stale (requires re-indexing), Real-time Query Type, Semantic (understands meaning), Keyword/Regex (matches text) Overhead, High (FAISS, Embeddings), Low (just a DB query) Cost,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

High (OpenAI API calls), Zero (internal) Response Style, Natural Language (via GPT- 3.5), Formatted Data List.

5. Discussion

The development of AcadBot marks a significant step toward integrating intelligent chat technologies in education. As academic environments grow more complex and the demand for personalized student support increases, automation tools become essential. AcadBot addresses this need through a Node.js backend and MongoDB database, creating a responsive and scalable chatbot that answers institutional queries, provides course details, and assists with administrative tasks. Technically, Node.js supports asynchronous event- driven processing, improving responsiveness and enabling multiple user interactions simultaneously. MongoDB's flexibility as a NoSQL database allows storage of various user inputs, chat histories, and contextual data—making data retrieval quick and efficient. Together, these technologies ensure smooth, real-time communication between students and the system. Testing results showed that AcadBot efficiently handled academic queries related to subjects, schedules, and course structures. The user-friendly interface made it accessible even to non-technical users, and feedback indicated high satisfaction with both speed and accuracy. Its modular architecture also supports easy integration of new features like NLP-based analysis or third-party APIs, ensuring long-term adaptability. AcadBot also highlights the benefits of automation in education. By managing routine queries, it reduces the workload on staff and provides 24/7 academic assistance, especially valuable for remote learners. However, current limitations include reliance on predefined responses, which restricts understanding of complex queries. Future improvements could involve Natural Language Processing (NLP) and Machine Learning (ML) for better contextual comprehension. Ensuring data privacy and security will also be a key focus in upcoming versions.

Conclusion

In this paper, we introduced Acadbot, a complete MERN application aimed at addressing ongoing information management issues in academic institutions. We built a secure, centralized portal for managing dynamic data such as events, schedules, meetings, and documents. The system is safeguarded by a strong Role-Based Access Control (RBAC) framework that customizes application permissions for students, faculty, and administrators. Our main contribution is the design and evaluation of the integrated AI chatbot. We showcased a practical shift from a complex, planned Retrieval-Augmented Generation (RAG) pipeline to a more efficient Heuristic-Based Retrieval (HBR) engine. The results indicate that our HBR engine, which queries the live MongoDB database directly with a multi-stage regex and token-matching algorithm, achieves 100% real-time data accuracy and incurs no operational cost. We demonstrated that for domain-specific applications where data freshness is critical, this lightweight, database-native method is a better and more manageable solution than a demanding vector-search pipeline.

References

[1]. P. Mell and T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication 800-145, 2011.

[2]. J. W. Rittinghouse and J. F. Ransome, Cloud Computing: Implementation, Management, and Security. Boca Raton, FL, USA: CRC Press, 2017.

[3]. MongoDB Inc., "MongoDB Documentation," 2024. [Online]. Available: https://www.mongodb.com/docs

[4]. Meta Platforms Inc., "React – A JavaScript Library for Building User Interfaces," 2024. [Online]. Available: https://react.dev

[5]. Node.js Foundation, "Node.js Documentation," 2024. [Online]. Available: https://nodejs.org/en/docs

[6]. Express.js, "Express Application Framework," 2024. [Online]. Available: https://expressjs.com

[7]. D. F. Ferraiolo, R. Kuhn, and R. Chandramouli, Role-Based Access Control. Norwood, MA, USA: Artech House, 2007.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 01 January 2026
Page No: 198-205
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0027

[8]. R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, “Role-based access control models,” IEEE Computer, vol. 29, no. 2, pp. 38–47, 1996.
[9]. M. McTear, Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots. Cham, Switzerland: Springer, 2020.
[10]. B. A. Shawar and E. Atwell, “Chatbots: Are they really useful?” LDV Forum, vol. 22, no. 1, pp. 29–49, 2007.