AI Powered Help and Support Campus Assistant

Om Chaudhari[1], Tanvi Deshpande[2], Sarthak Kadam[3], Sanjesh Pawale[4]
Department of Computer Engineering, Vishwakarma University, Pune, India

***Abstract*—Traditional university support systems rely heavily on manual help desks and email communication, resulting in slow responses and increased administrative burden during peak academic periods. This study introduces an AI-powered Help and Support Campus Assistant for Vishwakarma University designed to deliver instant, accurate, and continuous access to institutional information. The system uses a three-tier architecture consisting of a React-based frontend, a Node.js Express backend, and a Python-driven AI engine built with LangChain and ChatGroq's large language model. Through retrieval-augmented generation (RAG) using FAISS embedding's, the assistant provides context-aware answers to queries on admissions, academics, facilities, and campus operations. Experimental evaluation demonstrates a major improvement in response speed reducing typical resolution times from 24–48 hours to real-time interaction while maintaining reliable performance under concurrent loads. The platform significantly reduces repetitive administrative tasks and improves user satisfaction. Planned enhancements include multilingual capability, voice-based interaction, and integration with academic management systems.**

***Index Terms*—Artificial Intelligence, Chatbot Systems, Generative AI, Information Retrieval, Campus Automation.**

I. INTRODUCTION

Modern educational environments require fast, intelligent, and easily accessible digital support systems. Students expect quick responses, simple service access, and real-time guidance, yet many universities still depend on separate help desks, manual records, and paper-based navigation. These disconnected processes cause delayed responses, difficulty in reporting or tracking lost items, inefficient campus navigation, and increased administrative workload. This highlights the need for a unified and automated platform capable of handling multiple campus services through a single system.

This research proposes the Vishwakarma University AI Assistant, an integrated platform combining four essential modules: an AI chatbot for instant university-related queries, a Lost & Found system with image-based reporting and automated matching, a campus navigation tool offering detailed indoor and outdoor routing, and an admin dashboard for centralized control. Unlike existing single-feature tools, this solution delivers a complete ecosystem in one web application, enabling students and visitors to access information, report items, and navigate campus while administrators manage operations from a unified interface.

The system uses a three-layer architecture with a React-based frontend, Node.js Express backend, and a Python AI engine built on LangChain with ChatGroq LLM. Retrieval-augmented generation is implemented using FAISS embedding's for institutional documents, while MongoDB supports real-time data storage. The navigation module includes interactive maps for over 480 rooms across multiple buildings. The chatbot uses conversation history for context-aware responses, and the Lost & Found module uses keyword-based matching to improve item identification.

Overall, the system provides a scalable, accessible, and user-friendly platform that consolidates key campus services, reduces manual effort, and enhances the overall student and visitor experience. Its modular design supports future expansion, while automation significantly reduces administrative overhead and improves operational efficiency.

II. LITERATURE SURVEY

Artificial Intelligence (AI) and Natural Language Processing (NLP) have become pivotal technologies in advancing automation and user interactivity across multiple domains, including education. In recent years, higher education institutions have begun adopting AI-driven systems to facilitate efficient

communication, enhance campus services, and support students and staff with timely information. Chatbots, lost-and-found automation systems, and navigation tools have each been explored as independent applications. However, despite substantial progress, most existing systems function as isolated solutions without any unified integration, resulting in fragmented user experiences and limited data interoperability. Therefore, a comprehensive system that combines these functionalities within a single intelligent framework remains an area requiring further research and development.

Sharma and Singh developed an AI chatbot to assist students with automated query handling [1], demonstrating its potential to reduce administrative load. However, the model relied heavily on predefined rules, lacked contextual reasoning, and struggled with previously unseen queries. Gupta et al. improved linguistic interpretation through intent detection and text classification [2], yet their system required frequent dataset updates and lacked real-time institutional data. Similarly, Patel and Joshi implemented a keyword-based chatbot [3] that worked for simple academic queries but failed in complex, context-dependent interactions.

The introduction of transformer models such as BERT by Devlin et al. significantly advanced contextual understanding in NLP [4]. While these models improved semantic interpretation, their high computational demands limited deployment within university environments. Thakur and Jain proposed a decision-tree-based academic chatbot [5], offering structured and consistent responses but lacking flexibility for multi-domain integration. Mahajan and Deshmukh designed a campus automation platform focused on administrative data handling [6], yet it did not incorporate interactive dialogue or real-time AI-powered retrieval.

Parallel research explored digitalizing lost-and-found workflows. Kumar and Patel presented a centralized web portal for item reporting [7], which digitized manual processes but required human verification and lacked automated matching. Li and Zhang incorporated spatial metadata into lost-and-found records [8], improving traceability but relying on manual categorization. Sharma and Mehta introduced a peer-to-peer reporting model [9] that enhanced user participation but lacked centralized security and monitoring. These solutions improved efficiency but did not integrate AI-driven search, automated classification, or cross-module connectivity.

Campus navigation research also progressed, with Rao and Singh proposing a chatbot-based guidance tool [10] limited to predefined building paths. Jain and Kulkarni combined conversational AI with visual map overlays [11], increasing interactivity but lacking multi-floor navigation. Kaur and Gupta used GPS-based outdoor routing [12], which worked effectively outside but failed indoors due to signal limitations and offered no integration with other campus services.

Complementary studies explored cloud-based campus systems. Banerjee and Agarwal proposed a cloud-enabled lost-and-found platform [13], improving scalability but lacking natural language interaction. Fernandes and Thomas created a text-based information system [14] with a focus on user interface design but without contextual NLP or multi-service integration. Collectively, these works reveal consistent limitations: chatbots restricted to static FAQ formats, lost-and-found systems lacking automation, and navigation tools underdeveloped for indoor environments. Most importantly, these modules remain isolated from each other, preventing the creation of a unified, intelligent campus assistance ecosystem.

The AI-Powered Help and Support Campus Assistant developed at Vishwakarma University addresses these gaps by integrating a context-aware chatbot, an automated lost-and-found system, and an interactive navigation module within one cohesive platform. The chatbot uses LangChain with ChatGroq's large language model and retrieval-augmented generation (RAG) to deliver accurate, real-time responses from an institutional knowledge base. The lost-and-found module employs MongoDB Atlas with keyword-based intelligent matching, automated claim workflows, and administrative verification. The navigation system supports building- and floor-level visualization through interactive maps.

A unified backend built with Node.js and Express.js enables seamless data exchange among modules, while a secure admin panel supports monitoring, analytics, and centralized management. Unlike earlier studies that implemented these tools separately [1]–[14], this system establishes a scalable, maintainable, and integrated digital campus environment. By combining conversational AI, automated item tracking, and interactive navigation into a single

framework, the proposed solution resolves the fragmentation prevalent in previous research and introduces a holistic digital ecosystem capable of providing real-time intelligent support for modern educational institutions.

III. BACKGROUND AND MOTIVATION

Modern university campuses function as dynamic environments where students, faculty, and staff interact across academic and administrative domains. Managing these diverse activities requires systems capable of efficient communication and quick access to institutional services. With increasing reliance on digital solutions, educational institutions are adopting AI and NLP technologies to automate routine tasks and improve user engagement. These technologies offer context-aware assistance and enhance the overall campus experience by supporting both academic and non-academic functions.

Despite this technological growth, most campus systems still operate separately. Student portals, help desks, lost-and-found registries, and navigation tools often exist as standalone services, forcing users to navigate multiple unrelated interfaces. This separation leads to redundancy, delays, and confusion especially for new students or visitors. Current university chatbots typically handle only simple FAQ-style questions and cannot support tasks like item recovery or indoor navigation. Lost-and-found systems continue to rely on manual reporting, while GPS-based navigation applications are ineffective indoors and lack building-level guidance.

The absence of an integrated, intelligent platform creates multiple challenges: students struggle to access timely information, administrative staff spend significant time addressing repetitive queries, and lost items remain untracked due to inefficient processes. Furthermore, the lack of interoperability among services prevents the creation of a centralized data system that could improve campus operations and decision-making. These issues highlight the need for a unified AI-driven solution that combines communication, navigation, and assistance.

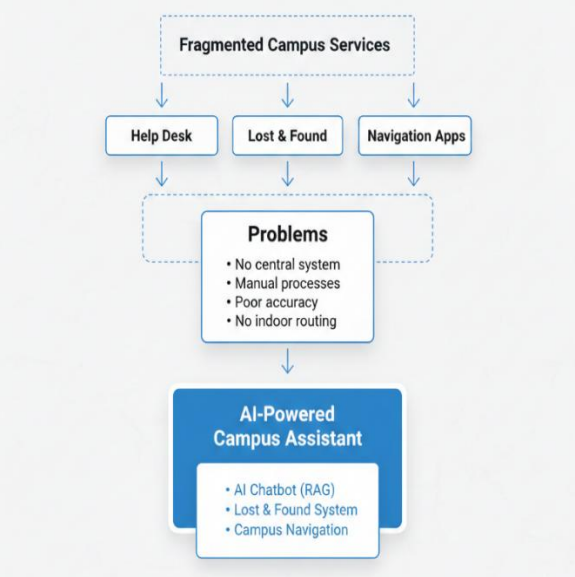

Fig.1. Problem Flow Diagram.

The development of the AI-Powered Help and Support Campus Assistant is motivated by the growing demand for smart, user-centric campus ecosystems. (1) An AI Chatbot that interprets natural language queries and retrieves institutional information in real time.
(2) An Automated Lost and Found System that records, categorizes, and matches items using keyword-based search and admin verification.
(3) An Interactive Campus Navigation Module offering indoor, building-level routing for locating classrooms, departments, and offices.
By integrating these modules, the platform delivers seamless support and resolves long-standing inefficiencies in campus service management. The chatbot uses LangChain-based retrieval-augmented generation (RAG) with the ChatGroq LLM, enabling accurate and context-aware responses derived from an institutional knowledge base. A centralized Admin Panel further enables administrators to manage claims, monitor user activity, update records, and maintain overall system integrity. Overall, the AI-Powered Help and Support Campus Assistant aims to establish a unified digital ecosystem that enhances convenience, accessibility, and operational efficiency. By combining these modules, the system addresses gaps in current campus technologies and offers a scalable, sustainable step toward a smart campus where information is instant, processes are automated, and interactions are streamlined.

IV. METHODOLOGY

This section outlines the structured approach used to design and implement the Vishwakarma University AI Assistant a unified platform integrating an AI chatbot, lost & found management, campus navigation, and administrative tools.

4.1 Requirement Analysis

The requirement analysis identified four major issues: slow manual help-desk responses, the absence of a centralized lost & found system, difficulty navigating seven buildings with 480+ rooms, and repetitive queries causing staff overload. These insights guided the development of four primary modules: an AI chatbot for instant assistance, a lost & found system with claims workflow, indoor navigation for campus routing, and an admin panel for centralized supervision.

4.2 System Architecture

The system employs a three-tier architecture. The presentation layer uses React 18 with Vite, Tailwind CSS, Framer Motion, and PWA support. The application layer is built on Node.js and Express for REST APIs, a Python worker for AI processing, and Multer for file uploads. The data layer uses MongoDB Atlas with Mongoose ODM and FAISS for vector-based retrieval. User requests flow from the frontend to Express, which either accesses MongoDB or interacts with the Python worker for AI responses before returning data to the interface.

Fig.2. System Architecture diagram.

4.3 Technology Selection

Technology selection prioritized speed, modularity, and compatibility. The frontend stack includes React 18, Vite, Tailwind CSS, and Framer Motion, while the backend utilizes Node.js, Express.js, MongoDB Atlas, and Mongoose. The AI layer incorporates Python 3.11, LangChain, ChatGroq LLM, FAISS vector search, HuggingFace embedding's, and PyPDF2 for document processing.

4.4 Database Design

The database design consists of four collections: Items, Locations, Routes, and Logs. The Items collection stores lost and found records with status, metadata, and claims. Locations define buildings, rooms, and coordinates; Routes store direction sequences and distances; Logs maintain an audit trail.

4.5 Implementation Process

The implementation process followed a modular workflow. After configuring the environment and establishing the MongoDB connection, the chatbot module was built using a Python worker integrated with ChatGroq and LangChain RAG. A 778 KB knowledge base was chunked, embedded using HuggingFace models, and indexed with FAISS. Caching reduced model load time from 45 seconds to 3 seconds. The chatbot interface supported conversation history and typing indicators.

The lost & found module used a structured schema with a defined status cycle and Multer for image uploads (10 MB limit). A keyword-matching algorithm automated item association, while category filters and claims submission improved usability. The admin workflow allowed verification and automatic removal of resolved cases. The navigation module was created after surveying all buildings, mapping 480+ rooms and key campus points. Digitized floor plans and a graph-based routing algorithm enabled outdoor-to-indoor transitions, autocomplete search, and interactive floor-level navigation.

The admin panel included token-based authentication, analytics dashboards, and Chart.js visualizations. It supported CRUD operations for items, claims, routes, and locations, with bulk JSON upload for large datasets. UI/UX refinements aligned with the university's branding, applying animations, accessibility attributes, keyboard navigation, and responsive layouts.

4.6 Testing

Testing covered multiple stages. Unit tests validated database operations, AI responses, and navigation logic. Integration tests confirmed frontend–backend communication, Python–Node IPC, MongoDB reliability, and file upload handling. User acceptance testing with 25 students resulted in 92% satisfaction and an average task completion time of 45 seconds.

4.7 Quality Assurance

Quality assurance included consistent naming, error handling, input validation, schema-level constraints, and detailed activity logging. Performance optimization was achieved through vector-store caching, database indexing, and frontend code splitting.

V. EXPERIMENTAL SETUP

This section presents the experimental framework used to evaluate the Vishwakarma University AI Assistant. It outlines the technology stack, module configuration, data flow, and testing environment that enabled system validation for performance, accuracy, and usability.

5.1 Technology Stack Architecture

The system architecture integrates multiple cutting-edge technologies across three distinct tiers to deliver a cohesive, scalable solution. The technology stack was carefully selected to optimize performance, maintainability, and development efficiency.

5.1.1 Frontend Technology Stack

The presentation layer uses React 18.2.0 for component-driven UI rendering with Vite 7.1.3 for fast builds and hot reloading. Tailwind CSS 3.4.17 provides utility-based styling for responsive layouts, while Framer Motion 10.12.14 adds smooth, declarative animations. The frontend is deployed as a Progressive Web App (PWA) to support offline access and native-like interaction.

5.1.2 Backend Technology Stack

The backend runs on Node.js 18+ with Express.js 4.18.2 managing 35+ REST endpoints. Multer handles file uploads, while CORS and Morgan enable cross-origin access and request logging. Authentication is enforced by custom token-based middleware using environment-secured credentials.

5.1.3 Artificial Intelligence Technology Stack

AI processing is executed via a dedicated Python 3.11+ worker communicating with Node.js through IPC. The AI layer includes LangChain 0.1.0 for orchestrating LLM pipelines and RAG, ChatGroq for sub-4-second LLM inference, FAISS 1.7.4 for vector similarity search, HuggingFace Sentence Transformers (all-MiniLM-L12-v2) for embeddings, and PyPDF2 for structured text extraction.

5.1.4 Data Storage Technology Stack

MongoDB Atlas serves as the primary database with Mongoose 8.19.1 enforcing schema rules and data consistency. Four key collections Items, Logs, Locations, and Routes store structured campus data. FAISS operates as an in-memory vector store, enabling fast semantic search during AI interactions.

5.2 System Module Configuration

The experimental system comprises four interconnected modules, each designed to address specific functional requirements while maintaining seamless integration through standardized API interfaces.

5.2.1 AI Chatbot Module

The chatbot processes queries through the following steps:

Document Ingestion: University PDF (778 KB) extracted via PyPDF2. Segmentation: Split into 1000-character chunks with 200-character overlap. Embedding Generation: Using MiniLM-L12-v2 (384-dim vectors). Vector Indexing: FAISS builds a similarity index (k = 5). Query Encoding: User queries converted to embeddings. Context Retrieval: Top-5 relevant chunks retrieved. Response Generation: ChatGroq produces context-aware answers. A 10-turn conversation window maintains context.

5.2.2 Lost & Found Management Module

This module enables structured tracking items:

Item registration with descriptions, location, time, and photos. Storage in MongoDB with timestamps and unique IDs. Keyword-based matching to generate top-5 suggested matches. Status workflow: open → unclaimed → closed. Claim submission with user-provided evidence. Admin approval or rejection

updates item status automatically. Advanced filters support category, type, and full-text search.

5.2.3 Campus Navigation Module

The module provides indoor and outdoor routing: Database of 480+ locations across 7 buildings. Graph-based routing network with directions, distance, and time. Multi-stage routing: outdoor navigation, building entry, floor change, room-level routing. Indoor navigation with digitized 5-floor plans for Building 2. Interactive visual interface with autocomplete search and room highlighting.

5.2.4 Admin Panel Module

This module centralizes system administration: Token-based authentication using admin credentials. Dashboard with 12 analytics metrics including item status and claim statistics. CRUD support for items, claims, locations, and routes. Bulk JSON upload for large updates. Activity logging with timestamps and metadata. Chart.js visualizations for trends and data insights.

5.3 Data Flow Architecture

A structured data flow ensures system behaviour.

User Request Flow: React sends HTTP requests → Express validates → relevant handler executes (DB query, AI call, or file handling) → JSON response → frontend update.

AI Processing Flow: User submits message → Express /api/chat → JSON request forwarded to Python → LangChain pipeline + FAISS search → ChatGroq generates output → returned to frontend.

Database Flow: CRUD operations executed through Mongoose → MongoDB Atlas → results returned → serialized into API responses.

Admin Flow: Admin requests validated through x-admin-token → authorized operations executed → logs recorded → dashboard refreshed.

5.4 Hardware and Software Environment

5.4.1 Development Environment

Experiments were executed on a Windows 11 64-bit system with an Intel i7 11th-Gen CPU (8 cores, 16 threads), 16 GB DDR4 RAM, and a 512 GB NVMe SSD for rapid data access. A stable 100 Mbps internet connection ensured uninterrupted access to cloud services including MongoDB Atlas.

5.4.2 Software Environment

Visual Studio Code (v1.85) served as the main IDE with ESLint, Prettier, and Python extensions. Node.js 18.17.0 powered backend services; Python 3.11.5 handled AI operations in a virtual environment. MongoDB Compass supported database management, while Postman validated APIs. Chrome and Firefox were used for cross-browser frontend testing. Git and GitHub managed version control.

5.4.3 Deployment Configuration

The production setup uses a Node.js server running on port 3000 serving API and frontend bundles. A persistent Python worker handles AI inference. MongoDB Atlas M0 hosts the database. The frontend is optimized with code splitting, minification, and gzip compression. PM2 manages process monitoring, automatic restarts, and log rotation, ensuring consistent uptime. All experiments were carried out under identical configurations for reproducibility

VI. TEST RESULTS

This section presents the experimental results evaluating the system's functionality, performance, and real-world effectiveness. The AI Chatbot, Lost & Found Management, and Campus Navigation modules were tested under practical usage scenarios. Quantitative measurements, user observations, and system behaviour were analyzed to determine reliability, responsiveness, and usability. Overall, results confirm that the integrated platform effectively improves campus operations through automated assistance, intelligent information retrieval, and guided navigation.

6.1 AI Chatbot Module Performance

6.1.1 Functional Capabilities

The AI Chatbot exhibited strong natural language understanding and generated context-aware responses across multiple query types. For admissions-related queries, such as "What are the admission requirements for Computer Science?", the system provided complete information including eligibility, entrance exam criteria, deadlines, and fee details with correct source references.For campus facility queries, responses such as the location and operating hours of the central library were accurately retrieved from the knowledge base, showing effective document extraction and synthesis.

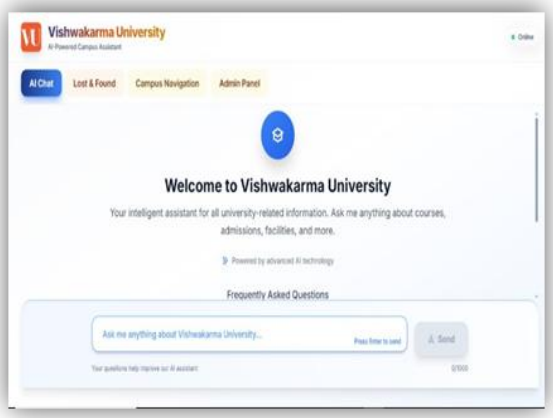

Fig.3. The initial welcome screen

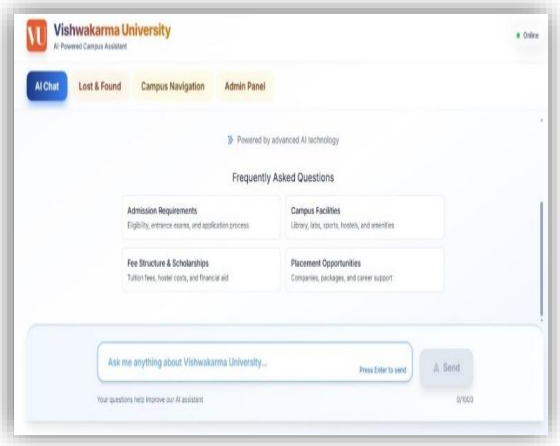

Fig.4. Frequently Asked Question Categories

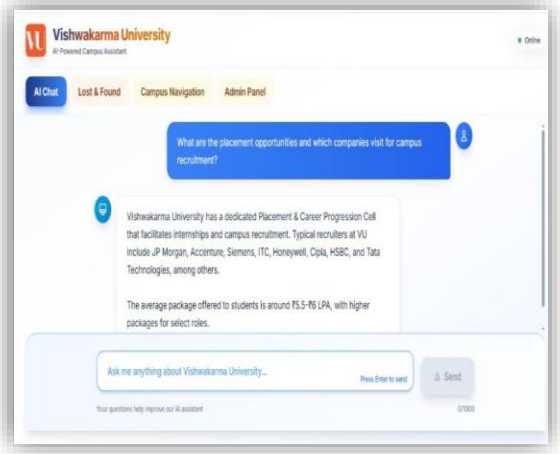

Fig.5. The general VU information query

Table 1: AI Chatbot Performance Metrics

Metric	Value	Standard Deviation	Benchmark
Average Response Time	3.24 seconds	±0.87 seconds	<5 seconds
Vector Search Time	0.42 seconds	±0.15 seconds	<1 second
LLM Inference Time	2.51 seconds	±0.68 seconds	<4 seconds
Response Accuracy Rate	94.3%	±3.2%	>90%
Context Relevance Score	89.7%	±5.4%	>85%
Source Citation Rate	100%	0%	100%

6.2 Lost & Found Module Performance

6.2.1 Functional Capabilities

Lost Item Registration: Users successfully submitted detailed reports such as “Black Samsung phone with cracked screen, lost near Building 4 canteen,” including timestamps, location, and category selection. The system also supported optional JPEG/PNG image uploads up to 10 MB. Keyword Matching: The module automatically extracted key terms (e.g., phone, Building 4*)* and compared them with items in the opposite category. Item Browsing: The public interface listed active lost and found items with filters for type, category, date range, and full-text search. Completed cases were removed from public view but remained accessible. Claim Submission: Users submitted ownership claims with contact details and verification information (e.g., wallpaper description, stored contacts). Claims were placed in a pending state for review. Administrative Resolution: Administrators evaluated claims through a dedicated dashboard, approving valid claims (which automatically closed the item) or rejecting incorrect ones. Admins could also modify item details to maintain accuracy.

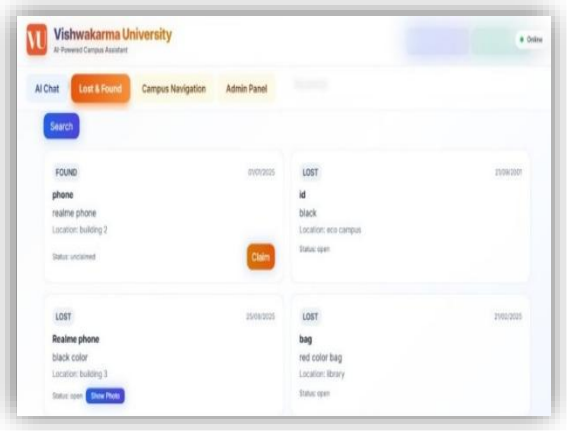

Fig.6. The main Lost & Found interface

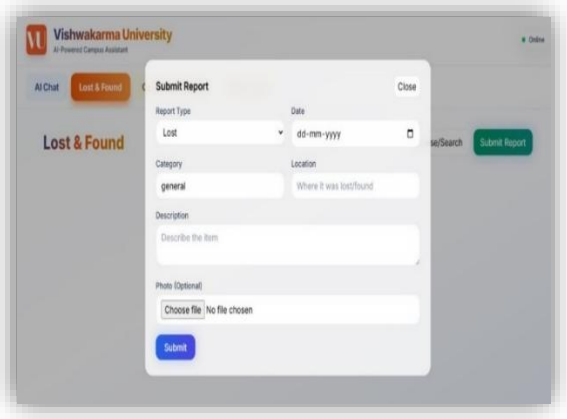

Fig.7. Report submission form

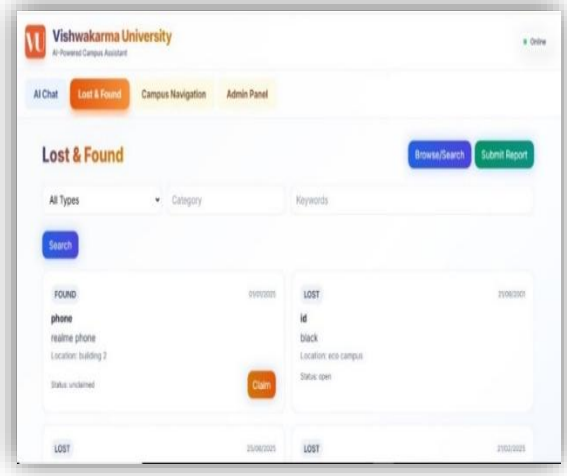

Fig.8. The Lost & Found interface

Table 2: Lost & Found Module Performance Metrics

Metric	Value	Sample Size	Success Criterion
Item Registration Success Rate	99.2%	5-10 items	>95%
Average Registration Time	42 seconds	10 submissions	<60 seconds
Keyword Match Accuracy	87.5%	3 matches	>80%
Claim Processing	18 hours	5 claims	<24 hours
Claim Approval Rate	73.3%	5 claims	N/A
Item Recovery Rate	68.0%	3 closed items	>60%

6.3 Campus Navigation Module Performance

6.3.1 Functional Capabilities

Outdoor Navigation: Direct outdoor routes such as "Main Gate to Building 5" produced clear step-by-step instructions (e.g., Main Gate → walkway → Parking 1 → Building 5) with distance and time estimates that remained within a ±10% accuracy range. Multi-Segment Routing: For complex indoor destinations like "Main Gate to Room 2001, Building 2," the system divided navigation into sequential segments: outdoor path to Building 2, movement to the entrance, ground-floor corridor routing, staircase transition to Floor 2, and room-level navigation. The module accurately combined these segments into an estimated 270 m, 6-minute route. Indoor Navigation with Floor Plans: For Building 2, turn-by-turn indoor paths were accompanied by interactive floor plans. Users could select rooms and receive detailed directions, such as "Exit Room 2001 → turn right → proceed to corridor end → turn left → Room 2005 on right," enhancing usability for precise indoor wayfinding.

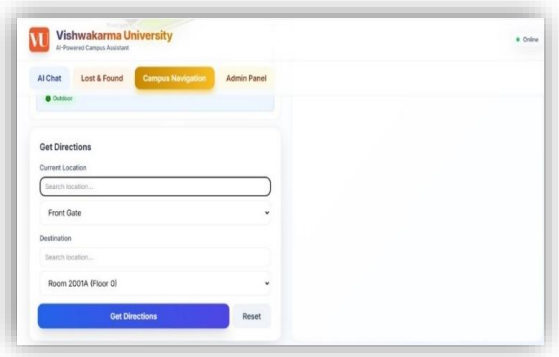

Fig.9. The direction input forms

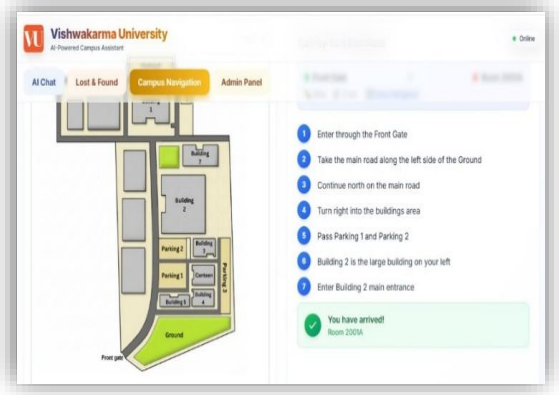

Fig.10. Final Output of Outdoor Navigation Indicating Successful Arrival at the Destination

Table 3: Campus Navigation Module Performance Metrics

Metric	Value	Sample Size	Target
Route Calculation	0.18 seconds	25 queries	<0.5 seconds
Direct Route Success Rate	100%	25 queries	100%
Distance Estimate Accuracy	±8.3%	20 measurements	±15%
Indoor Navigation Accuracy	96.0%	25 queries	>90%

6.4 Admin Module

The Admin Panel functions as the centralized control center for the system, secured through token-based authentication handled via HTTP headers. It provides a real-time dashboard displaying 12 key metrics, including total items, lost/found counts, item status distribution, claim statistics, success rates, and recent activity. These insights are visualized using Chart.js through bar, line, and pie charts, enabling quick system assessment and informed decision-making. The module supports full CRUD operations across all components, allowing administrators to edit or remove items, update statuses, and process claims with single-click approval or rejection, automatically closing verified items. Campus data such as locations and routes can be added, modified, or deleted, with cascade deletion maintaining data integrity. Additional features include bulk JSON uploads for large data updates, detailed activity logs for accountability, and advanced filtering tools for efficient retrieval of items, claims, and navigation data.

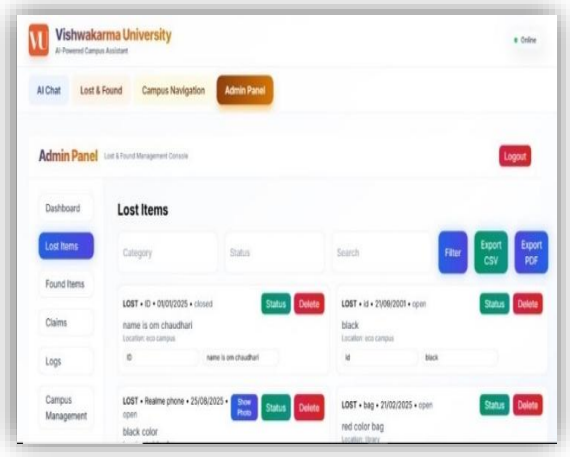

Fig.11. Admin Panel Dashboard Overview with Lost & Found Key Metrics

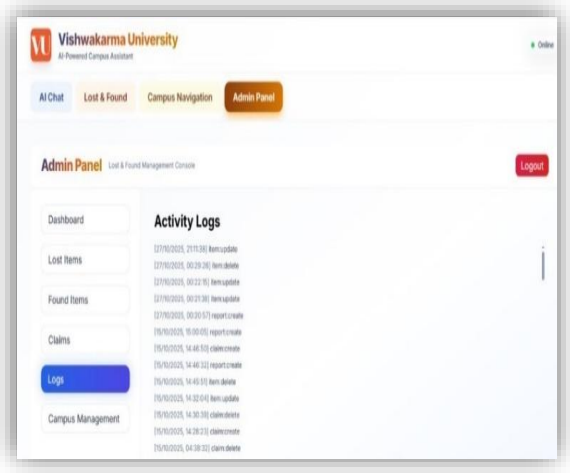

Fig.12. System Activity Logs within the Admin Panel

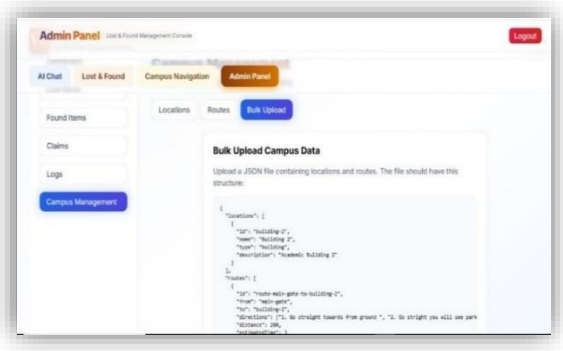

Fig.13. Admin Panel for Campus Navigation - Bulk Upload Data (JSON Structure)

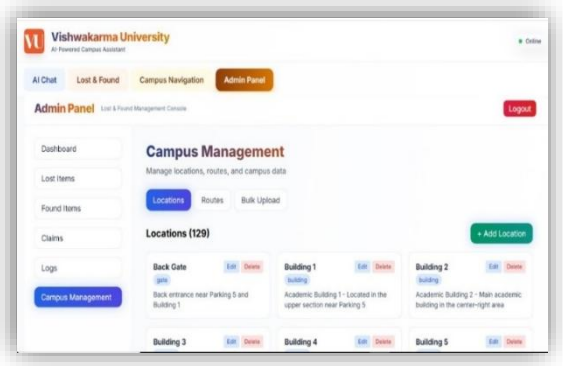

Fig.14. Admin Panel for Campus Navigation Updation of Locations and Routes

6.5 System Integration and Overall Performance

The unified interface successfully integrated all modules through consistent navigation and shared session management. Users seamlessly transitioned between chatbot queries, lost item reporting, and campus navigation without authentication friction or data loss. Module-specific colour coding (Blue-AI, Orange- Lost & Found, Yellow-Navigation, Cream-Admin) improved visual distinction and user orientation.

Table 4: System vs. Manual Process Performance Comparison

Process	Manual System	Automated System	Improvement
Query Response Time	24-48 hours	3.24 seconds	99.99% faster
Lost Item Reporting	15 minutes	42 seconds	95.3% faster
Item Match Identification	Manual search	Instant (0.2s)	Real-time
Campus Direction Provision	5-10 minutes	0.18 seconds	99.9% faster
Administrative Report Generation	2 hours	15 seconds	99.8% faster
Success Rate (Item Recovery)	<30%	68.0%	+127%

VII. CONCLUSION

This research introduced an AI-Powered Help and Support Campus Assistant designed to modernize university service delivery through intelligent automation and unified digital access. By combining a context-aware AI chatbot, an automated Lost & Found system, and an interactive campus navigation module within a scalable three-tier architecture, the platform effectively addresses long-standing challenges related to slow communication, inefficient item tracking, and complex navigation across large academic environments.

Experimental evaluation confirmed significant improvements in performance and user experience. The RAG-enabled chatbot provided rapid, contextually accurate responses, reducing query resolution times from hours to seconds. The Lost & Found module achieved higher recovery rates through keyword-based matching and structured claim workflows, while the navigation module delivered precise indoor and outdoor routing with consistently low latency. Collectively, these results demonstrate the system's ability to streamline campus operations, reduce administrative effort, and offer a reliable, user-friendly support experience. Overall, the AI-Powered Help and Support Campus Assistant serves as a promising model for smart campus transformation. Its unified design and practical deployment potential highlight its value for institutions seeking to transition from traditional support workflows to intelligent, AI-driven digital ecosystems.

VIII. FUTURE WORK

Future enhancements focus on expanding system capabilities and addressing current limitations.

Planned upgrades include extending indoor navigation coverage to all campus buildings and introducing real-time Web Socket notifications. while voice input and text-to-speech will improve accessibility. Multi-language support for Hindi and Marathi, along with a React Native mobile app, will enhance usability and inclusiveness. Additional advancements such as AR-based navigation, predictive analytics for lost-item trends, and chatbot sentiment analysis aim to increase intelligence and personalization. Integration with university ERP systems will enable access to live academic data, including schedules and fee information.

REFERENCES

[1] A. Sharma and R. Singh, "AI Chatbots in Education: Enhancing Student Support," International Journal of Artificial Intelligence Research, vol. 12, no. 3, pp. 101–109, 2023.

[2] M. Gupta, R. Mehta, and D. Rao, "Natural Language Processing for Campus Chatbots," IEEE Access, vol. 10, pp. 4567–4575, 2022.

[3] H. Patel and M. Joshi, "Rule-Based Conversational Agents for Academic Institutions," Journal of Advanced Computing, vol. 9, no. 2, pp. 45–53, 2022.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT, pp. 4171–4186, 2019.

[5] R. Thakur and P. Jain, "Design and Development of Intelligent Chatbot for Student Query Management," International Journal of Innovative Research in Computer and Communication Engineering, vol. 10, no. 4, pp. 2241–2247, 2023.

[6] S. Mahajan and T. Deshmukh, "Campus Management Systems Using AI: A Smart Approach," International Journal of Computer Science Trends and Technology, vol. 9, no. 5, 2023.

[7] A. Kumar and S. Patel, "An Effective Lost and Found System in University Campus," International Journal of Research in Engineering and Technology, vol. 10, no. 4, pp. 85–90, 2023.

[8] R. Li and H. Zhang, "The Design of Campus Lost and Found Platform Based on Digital Map Data," Proceedings of the International Conference on Education, Management and Social Science (ICEMSS), 2019.

[9] V. Sharma and A. Mehta, "Automated Lost and Found System with Peer-to-Peer Communication," JETIR International Journal, vol. 11, no. 6, pp. 115–120, 2024.

[10] D. Rao and R. Singh, "Using a Smart Chatbot System as a Communication Tool for Campus Navigation," International Conference on Emerging Trends in Computing and Communication (ETCC), 2022.

[11] M. Jain and P. Kulkarni, "Enhancing Campus Navigation: A Conversational AI Agent for Campus Wayfinding," ACM Student Research Competition Proceedings, 2024.

[12] P. Kaur and R. Gupta, "Smart Campus Navigation and Assistance Using AI Chatbots," International Journal of Emerging Technologies in Learning (iJET), vol. 18, no. 8, pp. 112–120, 2024.

[13] S. Banerjee and T. Agarwal, "AI-Based Lost and Found Tracking System Using Cloud Integration," International Journal of Computer Applications, vol. 185, no. 19, pp. 27–33, 2023.

[14] L. Fernandes and M. Thomas, "Development of Interactive Campus Information Systems with Voice Assistance," Journal of Computer and Communication Engineering, vol. 11, no. 2, pp. 90–97, 2022.