DESIGN AND IMPLEMENTATION OF AN INTELLIGENT WEB-BASED CHATBOT SYSTEM FOR HANDLING STUDENT INQUIRIES IN THE DEPARTMENT OF COMPUTER SCIENCE

BY

OSARIEMEN FORTUNE

PSC2105391

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES,

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

NOVEMBER 2025

DESIGN AND IMPLEMENTATION OF AN INTELLIGENT WEB-BASED CHATBOT SYSTEM FOR HANDLING STUDENT INQUIRIES IN THE DEPARTMENT OF COMPUTER SCIENCE

BY

OSARIEMEN FORTUNE

PSC2105391

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

NOVEMBER 2025

CERTIFICATION

This is to certify that this project work was carried out by **OSARIEMEN FORTUNE** with Matriculation Number **PSC2105391** under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.sc) Degree in Computer Science of the University of Benin

DR. (MRS.) A.R USIOBAIFO
Project Supervisor

DATE

APPROVAL

This project work is hereby approved in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

DR. (MRS. A.R. USIOBAIFO
Head of Department

DATE

DEDICATION

This project is dedicated to God Almighty for giving me the strength and wisdom to see it through to completion, and even throughout my stay in the University of Benin (UNIBEN). It is also dedicated to my parents; Mr and Mrs Osariemen and my brother Master Osariemen Clinton; for their love, support and guidance throughout my academic journey.

ACKNOWLEDGEMENT

I extend my deepest gratitude and acknowledgment to God Almighty for the enduring strength, wisdom, and clear direction provided throughout this academic journey and the successful completion of this project. I wish to express profound gratitude to my dedicated project supervisor and Head of the Department of Computer Science, Dr. (Mrs.) Rosemary Usiobaifo. Her consistent guidance, insightful feedback, and commitment were instrumental in ensuring the successful and timely execution of this research. A special thank you is also reserved for my project coordinator, Dr.(Mr.) Maxwell Osagie, whose organizational oversight proved invaluable.

Finally, my sincere appreciation is extended to all friends and peers who contributed to the success of this project through valuable collaboration, motivation, and support, including: Akpala Somoto, Kpagban Oghenevwede Godlife, Idakwo Ameh Samuel, Aigbogun Aifuwa, Ohilebo Isaac, Hon. Chigozie Nwachukwu, and Ogbebor Victory.

I am immensely grateful to my family and friends for their continuous support, encouragement, and consistent guidance throughout the duration of this challenging project.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This study proposes the design and implementation of an intelligent chatbot system for the Department of Computer Science, University of Benin, to improve communication between students and departmental administration. Currently, information such as registration deadlines, exam timetables, and announcements is shared through traditional means like notice boards and emails, which often cause delays and inefficiencies.

The proposed chatbot, built using Google Dialogflow (NLP) and a PHP-based backend, will provide real-time, accurate responses to student queries through a responsive web interface. System development will include requirement gathering, chatbot design, and integration with a structured knowledge base. Evaluation will focus on response accuracy, usability, and user satisfaction.

Expected outcomes include reduced administrative workload, faster and more consistent information dissemination, and improved student experience. The study also aims to offer a scalable framework for applying AI-powered communication tools in other university departments across Nigeria.

CHAPTER ONE

INTRODUCTION

1.0 Background Study

In the contemporary era of higher education, the accessibility of information is a critical determinant of effective academic administration and student satisfaction. University departments must consistently communicate essential updates concerning academic calendars, examination schedules, registration deadlines, and institutional policies. Without a reliable and efficient communication system, students are exposed to delays, confusion, and unnecessary administrative stress.

At the University of Benin, the Department of Computer Science currently disseminates information primarily through traditional means, such as physical notice boards, direct office visits, and email correspondence. These methods are severely limited in speed, scalability, and accessibility. Requiring students to be physically present on campus, physical notice boards are restrictive; office visits often result in prolonged waiting times due to human resource constraints; and delayed email responses can occur due to heavy administrative workload. These difficulties are significantly amplified during high-demand periods like course registration or examination preparation, where prompt, accurate information is vital for student success.The urgent need for automated, scalable, and real-time communication has therefore driven the adoption of Artificial Intelligence (AI)-driven solutions, notably chatbots. A chatbot is a software application designed to simulate human conversation, providing instant, interactive assistance. Modern chatbots

leverage Natural Language Processing (NLP), a branch of AI concerned with processing and understanding human language, to interpret the user's intent and meaning behind queries rather than relying on simple keyword matching. This advanced capability allows the system to handle complex and varied phrasing from students effectively. AI-driven chatbots are designed to operate 24 hours a day, 7 days a week, handle multiple users simultaneously, and deliver consistent, accurate information. This implementation directly translates to improved operational efficiency and a significant reduction in the repetitive workload on human administrative staff.

The technical architecture for the proposed system requires careful selection of components. The project involves implementing the backend using PHP (Hypertext Preprocessor), an established open-source, server-side scripting language. PHP's robust ecosystem and compatibility with databases (such as SQLite or MySQL) make it highly suitable for efficiently managing and retrieving data from the large knowledge base required to answer diverse student queries. To enable the core NLP functionality, the project integrates Dialogflow, a specialized cloud-based platform developed by Google. Dialogflow provides robust tools for intent recognition and the design of complex conversational flows, ensuring the chatbot can accurately understand the student's needs and retrieve the correct information via the PHP backend. By combining AI, advanced NLP, and a resilient, scalable PHP backend, this project aims to address the existing communication gap and align the Department of Computer Science with modern, global trends in digital education.

1.1 Motivation

The chatbot is motivated by inefficiencies in manual communication, which cause student delays, stress, and missed deadlines, while overburdening staff with repetitive queries. Growing enrollment worsens these issues. Similar solutions like CS infoBot demonstrate the potential to improve efficiency and student satisfaction at the University of Benin.

1.2 Problem Definition

Despite the presence of digital resources, the Department of Computer Science at the University of Benin is hindered by an over-reliance on manual and semi-manual communication methods. The specific challenges that define this problem include:

- **Communication Delays:** Students frequently wait for information, resulting in missed critical deadlines for submissions or registration.
- **Repetitive Workload for Staff:** Administrative staff dedicate excessive time to answering FAQs, limiting their capacity for strategic tasks.
- **Information Gaps and Limited Accessibility:** Inconsistent communication channels lead to unequal access, and restricted office hours impede off-campus students, especially those on SIWES placements.
- **Lack of Scalable Solutions:** Current methods cannot expand effectively as student enrollment increases, worsening delays and staff overburden.

1.3 Goal and Objectives

The goal of this study is to design and implement an intelligent chatbot system, utilizing Natural Language Processing (NLP), for responding to student academic inquiries within the Department of Computer Science at the University of Benin.

The specific objectives are:

1. To identify and categorize frequently asked academic questions (FAQs) by Computer Science students through surveys and interviews.
2. To design a conversational model using Dialogflow that leverages Natural Language Understanding (NLU) to interpret and respond accurately to student queries.
3. To develop a user-friendly, web-based interface using HTML, CSS, and JavaScript, ensuring accessibility across devices.
4. To integrate a PHP backend with a structured knowledge base (e.g., SQLite or Firebase) for dynamic and updatable responses.
5. To evaluate the chatbot's performance in terms of response accuracy, usability, and student satisfaction through user testing and feedback.

1.4 Scope of Research

The **scope of this study, as of the time of writing, is strictly aimed at the Department of Computer Science at the University of Benin**. The chatbot is designed to address academic and administrative queries falling under the following categories:

- Course registration deadlines and procedures.
- SIWES documentation processes.
- Examination schedules and Continuous Assessment timetables.
- CGPA calculation guidance.
- Departmental contact information and events.

The technical scope focuses on Dialogflow for NLP, HTML/CSS/JavaScript for the frontend, and PHP for the backend. Confidential data such as student grades is excluded.

1.5 Research Methodology

The research will adopt the Design Science Research (DSR) methodology. This rigorous and authentic approach is highly suitable for projects focused on the design, development, and evaluation of a novel technological artifact (the intelligent chatbot) intended to solve a practical organizational problem (communication inefficiency).

The DSR process will be executed through the following stages:

1. **Problem Identification and Motivation:** Defining the communication challenges faced by the Department of Computer Science.
2. **Objective of the Solution:** Clearly stating the system's goals, aligned with the project objectives.
3. **Design and Development (Artifact Construction):** This involves the systematic creation of the system architecture, including the PHP backend, the Dialogflow conversational model, and the responsive user interface.
4. **Demonstration:** Utilizing the implemented system to show that it is functional and capable of answering the defined categories of student inquiries.

5. **Evaluation:** Rigorously assessing the artifact against performance criteria (accuracy and usability) through user testing, thereby validating its utility and effectiveness.

1.6 Research Significance

The proposed intelligent chatbot system holds significant value, drawing on insights from similar implementations:

- **Benefits to Students:** Provides 24/7 access to information, eliminating delays, and enhancing the learning experience by minimizing administrative stress.
- **Benefits to Administrative Staff: Automates responses** to repetitive queries, reducing workload, and allowing staff to prioritize complex academic planning.
- **Benefits to the Department and Institution:** Positions the Department of Computer Science as a leader in adopting AI-driven solutions, enhancing the university's reputation, and fostering a replicable model for other departments.
- **Contribution to Research:** Provides a practical case study of NLP application in a Nigerian academic context, contributing to the literature on AI-driven educational solutions.

CHAPTER TWO

LITERATURE REVIEW

2.0 Introduction

The application of machine intelligence in higher education has gained increasing attention in recent years, particularly through the use of conversational agents or chatbots. Chatbots are software applications designed to simulate human conversation using natural language, and they have become prominent tools for improving communication between students and academic institutions. With the rapid growth of student populations in universities and the increasing complexity of administrative processes, educational institutions face mounting pressure to provide timely, accurate, and accessible information to students. Traditional communication systems such as notice boards, email correspondence, and in-person enquiries are no longer sufficient in meeting these demands, especially during peak academic periods such as registration, examination preparations, and industrial training documentation.

Recent studies have demonstrated the potential of intelligent chatbots in enhancing communication efficiency within university settings. For instance, the Powered Faculty Support Chatbot was developed to assist faculty members in handling routine administrative queries while simultaneously serving students with academic enquiries (Kumar et al., 2021). Similarly, the Student Enquiry Chat-Bot System addressed common student questions in a university environment, leveraging Natural Language Processing (NLP) to provide quick and relevant responses (Gupta & Sharma, 2020). These systems show how chatbots can function as intelligent intermediaries, reducing administrative workload while improving student satisfaction. In Nigeria, chatbot

projects like **iNOUN** (Adepoju et al., 2021) have proven feasible for academic support, particularly for distance-learning students, achieving high student satisfaction. Other systems, such as the FAQ chatbot using NLP and Naïve Bayes (Patel & Mehta, 2020), and emergency communication chatbots (Rahman et al., 2022), demonstrate chatbots' effectiveness in handling academic queries and rapid information dissemination, highlighting their versatility in Nigerian higher education.

Cloud-based chatbot systems offer scalability and flexible deployment. Chowdhury et al. (2021) showed that cloud infrastructure ensures reliable, cross-device access even under high traffic, while Singh & Verma (2019) emphasized structured databases for accurate responses. Together, these studies illustrate universities adopting data-driven, NLP-powered systems with robust backends to improve student communication Recent studies, including the Intelligent College Enquiry Chatbot (Okafor & Ibrahim, 2024) and the Chatbot for College Enquiry (Mishra & Rathi, 2023), demonstrate the shift from rule-based systems to sophisticated, NLP-powered chatbots that handle complex queries in real time. Key themes relevant to this project include the evolution of chatbots to machine-driven platforms, the use of NLP techniques like intent recognition and entity extraction to improve accuracy, adoption in universities for routine and emergency communication, and the need for robust architectures with databases and cloud integration to ensure scalability and reliability This chapter reviews chatbot concepts, NLP foundations, technological evolution, university case studies, and performance evaluations. It addresses challenges like knowledge base upkeep, API reliance, and data privacy, as well as recent advances in transformers, multilingual support, and cloud integration. These insights inform the design and implementation of the proposed intelligent chatbot for the Department of Computer Science, University of Benin.

2.1 Conceptual Background

The conceptual background for this study rests on three interrelated domains: machine intelligence, Natural Language Processing (NLP), and chatbots. Together, these fields provide the theoretical foundation and technical framework upon which intelligent chatbot systems are designed and implemented. Understanding these concepts is crucial in situating the proposed chatbot for the Department of Computer Science, University of Benin, within the broader literature on machine driven educational technologies.

2.1.1 Machine Intelligence in Education

Machine intelligence simulates human reasoning, learning, and decision-making (Russell & Norvig, 2016) and has become transformative in higher education. Applications include automated grading, plagiarism detection, adaptive learning, and student support via chatbots (Smutny & Schreiberova, 2020). Systems like the Powered Faculty Support Chatbot (Kumar et al., 2021) and the Chatbot for College Enquiry (Mishra & Rathi, 2023) streamline administrative tasks and student communication. By bridging gaps between students and administration, machine intelligence addresses scalability challenges in universities with large student populations, enhancing both operational efficiency and academic support.

2.1.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) is the subfield of machine intelligence concerned with enabling computers to process, interpret, and generate human language in a meaningful way. NLP is foundational to chatbots because it allows them to move beyond rigid keyword matching to understand the intent and semantics of user queries.

NLP tasks relevant to chatbots include:

- **Tokenization**: Dividing sentences into individual words or tokens.
- **Part of Speech Tagging**: Identifying grammatical roles of words in a sentence.
- **Intent Recognition**: Determining the purpose of a user query, e.g., "What is the SIWES deadline?".
- **Entity Extraction**: Identifying specific items such as course codes, dates, or lecturer names.
- **Dialogue Management**: Handling conversational flow and context across multiple turns.
- **Natural Language Generation (NLG)**: Producing human like responses based on structured knowledge.

Various algorithms have been used in educational chatbots. Traditional ML models like Multinomial Naïve Bayes effectively categorized FAQs (Patel & Mehta, 2020), while NLP-based systems parsed queries in real time (Gupta & Sharma, 2020). Modern approaches employ deep learning and transformers, as in the Intelligent College Enquiry Chatbot, enhancing accuracy and context awareness (Okafor & Ibrahim, 2024). Platforms like Dialogflow leverage these advancements to provide scalable, context-sensitive chatbot services.

2.1.3 Chatbots and Their Classifications

Chatbots can be defined as conversational agents that simulate interaction with users via text or voice. They are designed to provide information, answer questions, or perform specific tasks. In the educational domain, chatbots are particularly useful for handling

repetitive student queries, guiding administrative processes, and delivering academic support.

Chatbots are typically classified into three major categories:

1. **Rule Based Chatbots**: These rely on predefined rules and pattern matching techniques. They are relatively simple to implement but lack flexibility. Early examples include systems like ALICE, which used Markup Language (AIML) to process queries (Adamopoulou & Moussiades, 2020).
2. **Machine Based Chatbots**: These employ NLP and machine learning algorithms to understand user intent and generate responses. The Cloud Based Student Information Chatbot System, for instance, utilized cloud infrastructure to integrate **machine** models that could process dynamic student inquiries with high accuracy (Chowdhury et al., 2021).
3. **Hybrid Chatbots**: Hybrid chatbots combine rule-based systems with machine intelligence. For example, the College Enquiry Chatbot (Singh & Verma, 2019) used a structured FAQ database alongside NLP for flexible interactions, while iNOUN integrated structured information delivery with NLP-driven conversation for enhanced usability.

Most recent university chatbots, like the Chatbot for College Enquiry (Mishra & Rathi, 2023) and Intelligent College Enquiry Chatbot Using NLP (Okafor & Ibrahim, 2024), use hybrid or machine-based approaches, combining structure with adaptability to meet modern demands

2.2 NLP Techniques in Chatbots

Natural Language Processing (NLP)

NLP underpins modern chatbots, allowing them to understand and generate human language. At the University of Benin, it enables accurate, context-aware responses to student queries about courses and exams. Early systems relied on pattern matching, while modern chatbots use probabilistic, statistical, and deep learning models for greater sophistication and effectiveness (Adamopoulou & Moussiades, 2020; Smutny & Schreiberova, 2020).

2.2.1 Text Pre-processing

Text pre-processing is the first stage in the NLP pipeline. Since natural language is inherently unstructured, pre-processing ensures that input queries are transformed into a format suitable for machine learning or rule based models. The common pre-processing tasks include:

1. **Tokenization**: Splitting text into smaller units (tokens). For example, “When is the SIWES deadline?” becomes [When], [is], [the], [SIWES], [deadline].
2. **Stop word Removal**: Eliminating commonly used words (e.g., “the,” “is,” “at”) that do not contribute significantly to meaning.
3. **Stemming and Lemmatization**: Reducing words to their root form (e.g., “studying” → “study”).
4. **Vectorization**: Converting text into numerical representations using methods such as Bag of Words, Term Frequency–Inverse Document Frequency (TF IDF), or Word Embedding’s (Word2Vec, GloVe, BERT).

2.2.2 Intent Recognition

Intent recognition involves identifying the purpose behind a user's query. In student enquiry systems, intents may include:

- **Registration inquiries** (e.g., "How do I register for CSC 301?").
- **Timetable requests** (e.g., "When is the Continuous Assessment for CSC 205?").
- **Administrative queries** (e.g., "Where can I submit my SIWES logbook?").

Techniques for intent recognition include:

- **Rule based methods**: Mapping keywords to intents. Simple but limited.
- **Statistical models**: Using classifiers such as Naïve Bayes and Support Vector Machines (SVMs).
- **Deep learning models**: Leveraging architectures like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformers for context aware classification.

The Chatbot for Answering FAQs Using Multinomial Naïve Bayes Algorithm (Patel & Mehta, 2020) demonstrated that probabilistic classifiers can achieve high accuracy in handling structured student queries, especially in domains with well-defined FAQ categories. However, modern chatbots increasingly employ neural networks, as seen in the Intelligent College Enquiry Chatbot (Okafor & Ibrahim, 2024), which applied advanced NLP to achieve more nuanced intent recognition.

2.2.3 Entity Extraction

Entity extraction complements intent recognition by identifying specific keywords or phrases within a query. For instance, in the question “What is the deadline for CSC 302 registration?”, the intent is “registration inquiry,” while the extracted entity is CSC 302.

Entity extraction techniques include:

- **Regular expressions**: Effective for structured data like dates or course codes.
- **Dictionary based methods**: Matching words against predefined lists (e.g., course names).
- **Machine learning approaches**: Conditional Random Fields (CRFs) and deep learning models that learn entity patterns.

The iNOUN chatbot (Adepoju et al., 2021) demonstrated strong use of entity extraction by parsing academic course names and administrative processes, enabling students to retrieve precise information. Similarly, the Chatbot System for College Enquiry Using Knowledgeable Database (Singh & Verma, 2019) relied on structured entities stored in a database to match queries to accurate responses.

2.2.4 Dialogue Management

Dialogue management governs the conversational flow between user and chatbot. Unlike simple FAQ systems that provide single turn answers, advanced chatbots must maintain multi turn conversations. For example:

- **Student**: “When is the SIWES deadline?”

- **Chatbot**: “The deadline for SIWES submission is July 15th. Would you like me to also show you the documentation requirements?”

Dialogue management uses two approaches:

- **Finite State Machines (FSM)**: Predefined states and transitions, suitable for structured conversations.
- **Reinforcement Learning (RL)**: Learning optimal conversation strategies based on trial and error, often used in adaptive tutoring systems.

The Student Enquiry Chat-Bot System (Gupta & Sharma, 2020) adopted rule based dialogue management, while newer systems like the Cloud Based Student Information Chatbot System (Chowdhury et al., 2021) integrated more flexible, machine driven dialogue management for dynamic interactions.

2.2.5 Natural Language Generation (NLG)

NLG is the process of generating human like text from structured data or intent based outputs. For example, instead of answering “Exam schedule?” with raw data, the chatbot produces a coherent response: “The CSC 201 exam will hold on June 21st at 9:00 AM in Hall A”.

Techniques include:

- **Template based NLG**: Filling placeholders in predefined response templates.
- **Grammar based NLG**: Constructing sentences based on syntactic rules.
- **Neural NLG models**: Using sequence to sequence (Seq2Seq) models, transformers (GPT, BERT), and attention mechanisms.

In the Intelligent Chatbot for University Information System (Rahman et al., 2022), NLG was critical for creating clear, context aware responses that reduced student frustration compared to rigid, rule based outputs.

2.2.6 Algorithms and Models for NLP in Chatbots

A range of algorithms have been applied to educational chatbots.

- **Naïve Bayes Classifiers**: Effective for classifying FAQs (Patel & Mehta, 2020).
- **Support Vector Machines (SVMs)**: Used for intent recognition in structured academic datasets.
- **Recurrent Neural Networks (RNNs)**: Handle sequential data, useful for conversations.
- **Long Short Term Memory (LSTM)**: Capture long range dependencies in dialogue.
- **Transformers (e.g., BERT, GPT)**: Enable context aware language understanding, increasingly adopted in advanced chatbots.

The Intelligent College Enquiry Chatbot (Okafor & Ibrahim, 2024) highlighted the superiority of transformer based NLP over classical models, achieving higher student satisfaction.

2.3 Review of Related Works

The rapid growth of chatbot applications in higher education has generated a diverse body of research, with each system addressing specific institutional challenges such as enquiry management, faculty support, student engagement, and academic advising. This

section provides a detailed review of existing chatbot systems developed for universities and colleges, examining their architectures, NLP techniques, evaluation methods, and limitations. The review draws from key works, including iNOUN, powered Faculty Support Chatbots, Naïve Bayes FAQ Chatbot, Cloud based Chatbots, and other intelligent enquiry systems.

2.3.1 Powered Faculty Support Chatbot

Kumar et al. (2021) developed the Powered Faculty Support Chatbot to automate tasks like attendance and scheduling, enhancing faculty efficiency but focusing less on student enquiries, limiting its applicability for academic support.

2.3.2 Chatbot for Answering FAQs Using Multinomial Naïve Bayes

Patel and Mehta (2020) created a student enquiry chatbot using Multinomial Naïve Bayes for FAQ categorization, offering lightweight, accurate responses but struggling with ambiguous or context-dependent academic queries.

2.3.3 Student Enquiry Chat-Bot System

Gupta and Sharma (2020) developed a student enquiry chatbot using Python Flask, NLTK, and MongoDB, effectively handling academic queries but requiring frequent knowledge base updates to manage questions beyond its training data.

2.3.4 iNOUN: A Chatbot for Academic Enquiries

Adepoju et al. (2021) developed iNOUN, a chatbot for NOUN that used rule-based and NLP approaches to answer academic queries, improving accessibility but limited by single-language support and weak context handling

2.3.5 Intelligent College Enquiry Chatbot Using NLP

Okafor and Ibrahim (2024) designed an Intelligent College Enquiry Chatbot that emphasized enhanced student interaction using state of the art NLP techniques. Unlike earlier systems based on Naïve Bayes or rule based approaches, this chatbot integrated transformer based models, improving accuracy in intent recognition and response generation. Its architecture included:

- **Frontend**: Web and mobile interface.
- **Backend**: Python Django with cloud storage.
- **NLP engine**: Transformer based model for intent and entity recognition.

Evaluation results indicated high student satisfaction and reduced administrative workload, making it one of the most relevant works for informing this study's use of Dialogflow NLU with a PHP backend.

2.3.6 Cloud-Based Student Information Chatbot

Chowdhury et al. (2021) created a cloud-based student information chatbot using Firebase for scalability, effectively handling academic queries but raising data privacy and security concerns due to dependence on cloud APIs.

2.3.7 Chatbot System for College Enquiry Using Knowledgeable Database

Singh and Verma (2019) developed a college enquiry chatbot using a hybrid model combining a structured database and NLP, effective for predefined queries but limited in handling unstructured, unfamiliar questions

2.3.8 Chatbot for Emergency Student Communication

Rahman et al. (2022) developed a chatbot for emergency notifications using SMS and web chat, demonstrating how chatbots can support critical, time-sensitive communication beyond routine academic enquiries.

Table 2.1: Summary Table Of Related Works

Author(s) & Year	**System / Study Title**	**Technology / Methods Used**	**Key Findings / Contributions**	**Limitations**	**Relevance to Proposed Study**
Kumar et al. (2021)	Powered Faculty Support Chatbot	Combined rule based logic + ML classifiers; integrated with institutional DBs.	Reduced lecturer workload by automating attendance tracking, scheduling, and FAQs.	Focused mainly on faculty tasks, not student enquiries.	Shows how chatbots reduce repetitive workload. Proposed system extends benefit to student enquiry domain at UNIBEN.
Patel & Mehta (2020)	FAQ Chatbot Using Multinomial Naïve Bayes	Probabilistic classification of FAQs; text preprocessing pipeline.	Demonstrated efficiency of Naïve Bayes in handling structured student FAQs.	Poor handling of ambiguous or context heavy queries.	Justifies combining probabilistic classifiers with modern NLP for improved accuracy in student enquiry contexts.
Gupta & Sharma (2020)	Student Enquiry Chat Bot System	Python Flask + NLTK + MongoDB backend.	Provided real time answers to exam schedules, course registration.	Limited response outside training dataset.	Highlights need for frequent knowledge base updates; informs SQLite/Firebase integration.

Adepoju et al. (2021)	iNOUN: Chatbot for Academic Enquiries	Hybrid model: rule based + NLP. Integrated into NOUN portal.	Improved accessibility for distance learners; positive usability results.	Lacked multilingual support and advanced context handling.	Relevant Nigerian case study; inspires localized adaptation for UNIBEN.
Okafor & Ibrahim (2024)	Intelligent College Enquiry Chatbot	Transformer based NLP + Django backend + cloud storage.	Achieved higher intent recognition accuracy; high student satisfaction.	High computational requirements, **API dependence**.	Validates use of modern NLP (Dialogflow) to enhance accuracy in departmental enquiries.
Chowdhury et al. (2021)	Cloud Based Student Information Chatbot	Firebase cloud backend + NLP driven chatbot.	Demonstrated scalability for large student populations.	Privacy and cost concerns due to reliance on cloud APIs.	Informs proposed PHP + SQLite/Firebase backend for scalability without over dependence on cloud APIs.
Singh & Verma (2019)	Chatbot System for College Enquiry Using Knowledgeable Database	Hybrid approach: structured DB for responses, NLP for queries.	Demonstrated how a structured database serves as a reliable backbone for departmental systems.	Inability to generalize beyond predefined queries.	Emphasizes the critical role of a robust, structured knowledge base in the hybrid design.
Rahman et al. (2022)	Chatbot for Communicating with University Students in Emergency Situations	SMS integration alongside web chat.	Served as a rapid information dissemination tool during crises.	Focused on emergency notification rather than complex academic enquiry.	Highlights the versatility of chatbots for critical, time sensitive communication.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Introduction

The successful execution of any scientific or technological undertaking depends on the clarity and research methodology. This framework provides the structure for the investigation, ensuring that the resulting outcomes are valid, reliable, and demonstrably replicable (Creswell, 2018). In the context of software-oriented research, the choice of methodology is critically important as it governs the entire system lifecycle, from the initial conceptualization and detailed blueprinting to the final implementation, stringent quality assurance, and ongoing validation.

This chapter details the methodological framework, formally titled the System Design and Implementation (SDI) methodology, employed for the engineering and deployment of an intelligent chatbot system. This system is designed to leverage advanced computational capabilities to automate and significantly enhance student inquiry and communication processes within the Department of Computer Science at the University of Benin. The project's overall strategy integrates principles from Design-Based Research (DSR) with established Software Engineering (SE) best practices. This integrated approach guarantees a development process that is systematic, operationally efficient, adaptable, and yields a highly robust software artifact. The selection of the SDI methodology was a direct response to the critical research problem identified in Chapter One: the profound inefficiency, limitations, and operational bottlenecks of the traditional, manual student-departmental communication systems. As articulated by Sommerville (2015), the methodological choice directly determines the success, long-

term maintainability, and scalability of the resulting software. Therefore, the SDI methodology was structured to prioritize clarity in development, facilitate iterative refinement, and maintain a pragmatic balance between theoretical foundation (AI/NLP principles) and practical application (solving administrative inefficiency).

3.2 Analysis of the Existing System

Prior to developing the intelligent chatbot solution, a mandatory and comprehensive System Analysis was conducted on the existing communication framework utilized by the Department of Computer Science. This phase meticulously examined the current system's operational mechanisms, resource utilization, and communication endpoints. The existing structure operates primarily on **a** manual, labor-intensive model, relying heavily on human-mediated methods for both information dissemination and inquiry resolution.

The four primary channels identified within this manual framework are:

1. **Physical Notice Boards:** These static, non-interactive platforms are utilized for broadcasting essential, time-sensitive information, including academic calendars, lecture schedules, and critical registration deadlines. The process of updating and maintaining the accuracy of these boards is manual and prone to human error.
2. **Administrative Staff Consultation:** This is the primary point of contact for personalized or complex student inquiries related to academic policies, course registration procedures, results, and fee structures. Resolution of these queries

depends entirely on the availability and institutional knowledge of specific administrative personnel.

3. **Departmental Email System:** This asynchronous channel requires staff to manually read, categorize, draft unique responses, and manage response threads for highly repetitive queries, resulting in poor scalability and often significant response latency.
4. **Student Service Desk (In-Person Queue):** This channel forces students to physically queue or wait for one-on-one consultation slots with staff members. This method inherently introduces physical delays, high operational costs, and is inefficient for disseminating common information.

The analysis measured query resolution time, redundancy, and student satisfaction, revealing that high student volume and limited staff create severe operational bottlenecks, especially during peak academic periods

3.3 Problems of the Existing System

The detailed process-flow analysis of the existing manual communication infrastructure unequivocally illuminated a set of systemic, interrelated problems. These deficiencies fundamentally undermine the operational efficiency of the department and negatively impact the academic experience and success of the student body. The intelligent chatbot system is therefore engineered specifically to counteract these established failures:

1. **High Information Latency and Resolution Delays:** The reliance on manual retrieval and physical or email-based dissemination causes inherent information lag. Students consistently experience unacceptable waiting times, which often

results in missed deadlines, administrative penalties, and elevated student anxiety, especially for queries demanding urgent resolution.

2. **Severe Operational Bottlenecks and Staff Overload:** Overwhelmed by repetitive, low-complexity inquiries, diverting time from critical duties and causing operational inefficiencies and burnout.
3. **Inconsistent, Unstructured, and Variable Responses:** The manual and verbal nature of information delivery introduces a critical risk of **response inconsistency**. Different staff members may dispense slightly varied information, which leads directly to confusion, potential misinformation, and subsequent administrative disputes among the student populace, eroding trust in official communication.
4. **Critical Lack of 24/7 Availability and Geographic Constraint:** The system is limited to office hours (8 AM–4 PM), preventing off-campus or busy students from accessing information outside these times.
5. **Deficient Data Logging, Tracking, and Lack of Feedback Loop:** The manual system lacks data logging, preventing systematic tracking or analysis of student queries and hindering proactive, data-driven communication improvements.
6. **Accessibility and Equity Issues:** Reliance on notice boards and in-person visits limits accessibility for disabled, off-campus, or ill students, creating an inequitable communication system.

3.4 Overview of the Proposed System

The culmination of the analysis and problem identification phases dictates the necessity for the proposed solution: **the** Intelligent Chatbot System. This system is fundamentally

conceived as a conversational Design Artifact, a term derived from the Design Science Research (DSR) methodology (Hevner & Chatterjee, 2010). It is engineered to leverage advanced Artificial Intelligence (AI) principles and sophisticated Natural Language Processing (NLP) techniques to comprehensively automate communication within the Department. The overarching, strategic objective is to deploy a reliable, instantaneous, highly accessible, and centrally verified source of departmental information.

The chatbot is designed to function as a Virtual Administrative Assistant (VAA), providing real-time, accurate, and consistent answers to the corpus of frequently asked student inquiries (FAQs). The core functional capabilities of the system include:

3.4.1 Core Functional Capabilities

- **Natural Language Understanding (NLU) Engine:** The system should interpret complex, unstructured student queries in natural language, using Dialogflow to parse intents and extract relevant entities.
- **Automated and Dynamic Response Generation:** The system provides instant, structured, and context-aware answers. These responses are sourced dynamically from a verified, relational knowledge base rather than relying on static, hard-coded responses.
- **24/7 Continuous Availability:** The proposed system ensures perpetual, non-stop access to departmental information, completely independent of human working hours, geographical location, or time zones.
- **Systematic Interaction Logging:** A critical non-functional requirement is the systematic, immutable recording of all student-chatbot interactions, query types, and associated timestamps. This facilitates administrative oversight,

retrospective performance analysis, and the continuous retraining of the NLP model.

- **Personalized Authentication:** The system integrates a user authentication layer (login feature) to ensure personalized interaction, track individual user history, and prevent anonymous misuse, without collecting sensitive academic data.

The proposed system integrates into existing departmental workflows as a “first line of support” for student inquiries, reducing administrative workload, ensuring consistent information, and improving the speed and quality of service delivery.

3.5 Proposed System Architecture and Interface

The proposed architecture for the intelligent chatbot is a variation of the robust, industry-standard Four-Tier Architecture. This architectural choice is necessitated by the requirement for both centralized, complex computational processing (NLP and Database) and a distributed, lightweight client-side interface. The Four-Tier Architecture rigidly enforces the principle of separation of concerns, ensuring modularity, fault isolation, and high maintainability.

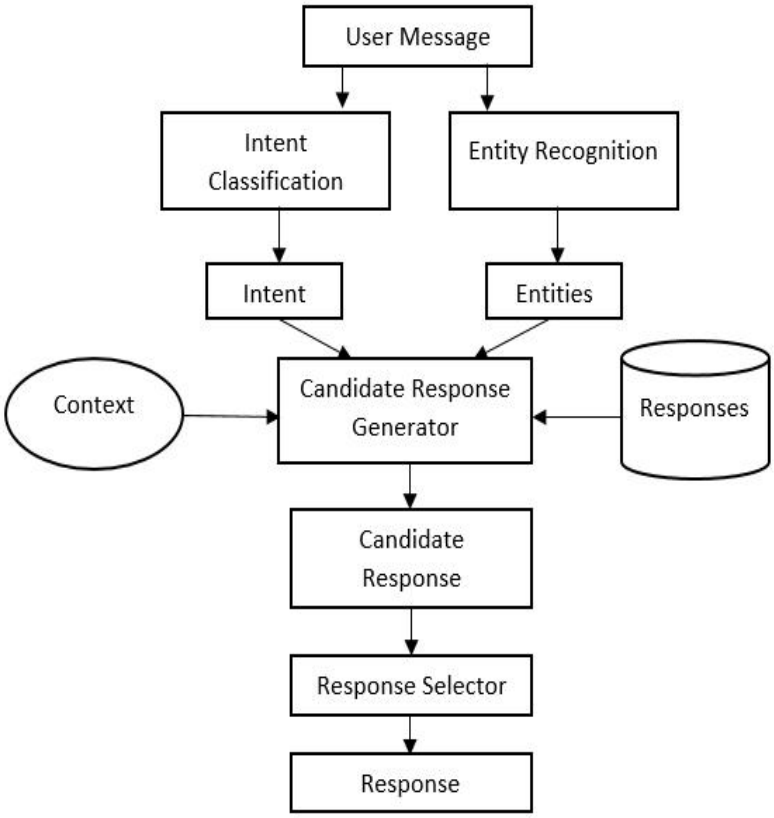


Figure 3. 1: System Architecture of the Intelligent Chatbot for Student Enquiries
Adamopoulou, E., & Moussiades, L. (2020)

3.5.1 The Four-Tier Logical Separation

1. **User Interface Layer (Tier 1: The Client):**
    - **Technology Stack:** Implemented using HTML, CSS, and JavaScript.
    - **Responsibility:** This is the student's direct point of access and presentation layer. It handles all raw text input, manages the visual display of conversational responses, and ensures real-time conversation management through Asynchronous JavaScript and XML (AJAX) requests. The interface is meticulously designed to be highly responsive across different screen sizes (desktop, tablet, mobile).
2. **Natural Language Processing Layer (Tier 2: The Cognitive Engine):**
    - **Technology Stack:** Powered exclusively by the Dialogflow NLP Engine**.**

- **Responsibility:** This is the cognitive core. It receives the raw user query (text string) via the Backend Logic Layer, parses the text using complex linguistic models, and performs two critical functions:
    - **Intent Identification:** Determining the underlying goal or purpose of the user's query (e.g., *Course_Registration*, *Check_Fee_Status*).
    - **Entity Extraction:** Identifying and tagging key parameters within the query (e.g., extracting "CSC 401" as a *Course_Code* entity).

3. **Backend Logic Layer (Tier 3: The Middleware/Business Logic):**
    - **Technology Stack:** Primarily implemented using PHP (Hypertext Pre-processor).
    - **Responsibility:** The Backend acts as the essential intermediary, receiving the structured, classified data (Intent and Entities) from Dialogflow. It executes the core business logic of the system:
        - **Dynamic Query Execution:** Based on the Intent, it executes specific database queries against the Data Storage Layer.
        - **Response Formulation:** It processes the retrieved data and formats the final, coherent response text.
        - **Security and API Management:** It manages secure API requests (using cURL) and enforces user authentication protocols.
4. **Data Storage Layer (Tier 4: The Knowledge Base):**
    - **Technology Stack:** Comprises a hybrid solution: SQLite (for robust local development) and Firebase (for cloud production/scalability).

- **Responsibility:** This layer securely and persistently stores the entire information corpus, including the relational knowledge base, structured FAQs, predefined intents, user authentication records, and the comprehensive conversation logs required for continuous system retraining and analysis.

3.5.2 User Interface Design Principles

The User Interface (UI) design, visually representing the User Interface Layer (Tier 1) shown in Figure 3.1, rigorously adhered to the ten Usability Heuristics articulated by Jakob Nielsen (2012), prioritizing simplicity, accessibility, and error prevention. Key UI components include:

- **Secure Login Page:** Mandates personalized interaction by requiring users to authenticate with their name, email, and password, ensuring user identification without storing sensitive academic data.
- **Conversational Chat Window:** The main display area, meticulously structured to display messages in a clear, chronological thread, employing modern conversational design standards to enhance user familiarity and comfort.
- **Minimalist Input Field:** A simple, readily accessible text entry point that is prominent across all device viewports for seamless query submission.
- **System Feedback Cues:** Incorporation of crucial elements like a "typing indicator" or system status messages, which provide immediate visual feedback to the student that the system is processing their query, thus managing user expectation and enhancing the natural flow of the conversation.

3.6 Research Methodology and Development Approach

The foundational methodology steering this entire research and development project is the System Design and Implementation (SDI) framework. SDI is an action-oriented methodology that focuses specifically on the creation of a practical, validated software artifact to solve a pre-existing, critical organizational problem. It is inherently pragmatic, demanding a blend of theoretical foundation and practical construction.

3.6.1 Philosophical Foundation

The project rests on a dual philosophical foundation:

- **Applied Research:** The investigation is applied, concentrating on leveraging sophisticated techniques (AI and NLP) to generate a tangible, functional solution that addresses the specific, immediate communication challenges within the Department of Computer Science.
- **Design Science Research (DSR):** DSR is the dominant paradigm (Hevner & Chatterjee, 2010). The chatbot is classified as the design artifact—a software-based construct developed through rigorous research cycles (design, build, evaluate) to confirm its efficacy in streamlining departmental communication and operational procedures. The process ensures knowledge generation is tied directly to the artifact's construction and subsequent performance validation.

3.6.2 Development Approach

Within the operational context of the broader SDI framework, the structured, sequential development process selected for this project is the classic Waterfall Model. This model

was specifically chosen for its methodological clarity and rigor, which align perfectly with the requirements of an academic research project that demands meticulous documentation and verifiable stage-gate completion. The reasons for selecting this structured approach include:

1. **Systematic Structure:** The model mandates a linear, non-overlapping sequence of phases, requiring the thorough completion and formal approval of deliverables from one stage before initiating the next.
2. **Meticulous Documentation:** Each phase (Analysis, Design, Implementation) necessitates comprehensive documentation, which is crucial for academic review, project traceability, and clear reporting of progress.
3. **Controlled Scope:** Given the clearly defined and static requirements (FAQs for a single department) and a fixed academic timeline, the Waterfall Model effectively minimizes the risk of scope creep, ensuring the project remains within its mandated boundaries.

The five sequential phases of this development approach are:

- **Requirement Analysis:** The process of data gathering and formal specification.
- **System Design:** The architectural and database blueprinting phase.
- **Implementation:** The code construction and component integration phase.
- **Testing:** The rigorous validation and quality assurance phase.
- **Deployment and Maintenance:** The final launch and post-launch refinement cycle.

3.6.3 System Feasibility Study: Risk Mitigation Analysis

Before committing resources to full-scale development, a comprehensive feasibility study was executed to formally mitigate project risks and confirm viability across four essential dimensions:

- **Technical Feasibility:** The core technologies (Dialogflow, PHP, Firebase) are mature, well-documented, and offer seamless API integration. The developer possesses requisite expertise, confirming the technical solution is achievable with available tools.
- **Economic Feasibility:** The selection strategy prioritized cost minimization, relying heavily on open-source (PHP, SQLite) and generous free-tier cloud services (Dialogflow, Firebase Hosting/Firestore), rendering the project highly affordable for both research and potential institutional adoption.
- **Operational Feasibility:** The system is designed for zero disruption to existing critical administrative processes. It acts as an augmentation, not a replacement, requiring minimal user training and fitting seamlessly into the department's web-based communication environment.
- **Schedule Feasibility:** The structured and phased Waterfall approach, coupled with well-defined milestones, guarantees that the entire project lifecycle, from analysis to validated deployment, can be realistically executed within the academic semester's time constraints.

3.7 System Requirement Analysis

The Requirement Analysis phase is the most critical stage, forming the conceptual blueprint for all subsequent design and engineering activities. This phase systematically determined what the system must achieve and how it must behave. Requirements were derived from three validated sources: student surveys (Google Forms), consultations with departmental staff, and analysis of existing communication logs.

3.7.1 Functional Requirements

Functional requirements define the specific, mandatory services and behaviors the chatbot must execute to meet its primary objective of automated communication.

1. **NLU and Intent Mapping:** The system must accurately map user input to one of the approximately 20 predefined Intents (e.g., *Check_Result_Status*, *Fee_Payment_Inquiry*).
2. **Real-Time Response Generation:** Must generate and transmit a coherent, context-aware text response within 3.0 seconds of receiving a query.
3. **Knowledge Base CRUD Operations:** The Admin module must support full Create, Read, Update, and Delete **(**CRUD**)** functionality for the FAQ/Response content.
4. **Fallback Mechanism:** In the event of an unclassified query (low confidence score from Dialogflow), the system must gracefully trigger a Fallback Intent**,** prompting the user for clarification or providing an alternative communication channel (departmental email/phone).

5. **User Session Management:** The system must initiate and manage a persistent session for authenticated users, allowing for conversational context retention during the session lifecycle.

3.7.2 Non-Functional Requirements

Non-functional requirements specify the quality attributes that govern system performance and operational integrity.

1. **Performance and Latency:** System response time (time from query submission to display of response) must not exceed the target of 3.0 seconds under normal load.
2. **Scalability:** The architecture must be inherently scalable (utilizing Firebase), capable of concurrently supporting up to 50 active user sessions without noticeable degradation in response latency.
3. **Availability and Uptime:** The deployed system must maintain a minimum operational uptime of 99.5% during the academic semester.
4. **Security and Privacy:** All data transmission between the client and server must utilize HTTPS/SSL encryption. User credentials (passwords) must be stored using industry-standard hashing algorithms (e.g., bcrypt).
5. **Maintainability:** The codebase (PHP, JavaScript) must be modular and documented to a standard that allows a third-party developer to implement a major functional update within 40 development hours.

3.8 System Modeling and Blueprinting

System modeling is the process of creating abstract representations of a system, serving as the bridge between abstract requirements and concrete code implementation. The models utilized adhere to established software engineering notation, ensuring clarity and mitigating design ambiguity (Sommerville, 2015).

3.8.1 Data Flow Diagram (DFD): Process Transformation

The **Data Flow Diagram (DFD)** visually models how information is processed and transformed within the system.

- **Level 0 DFD (Context Diagram):** This diagram provides the highest-level view, depicting the entire chatbot system as a single process interacting with two essential external entities: the **S**tudent User and the Departmental Knowledge Base**.** It illustrates the flow of raw query input into the system and the processed response output back to the user.

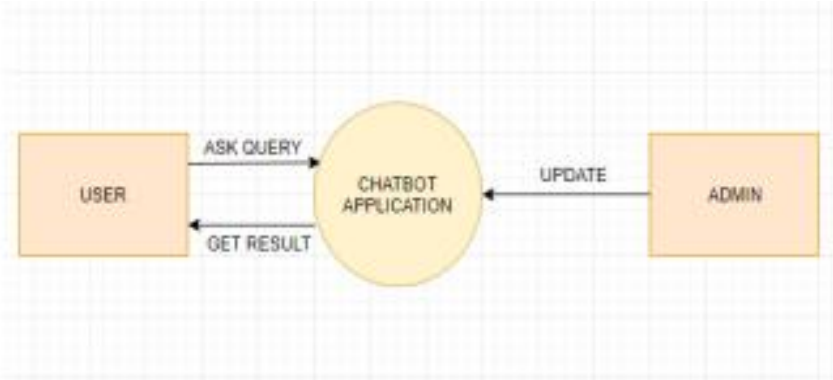
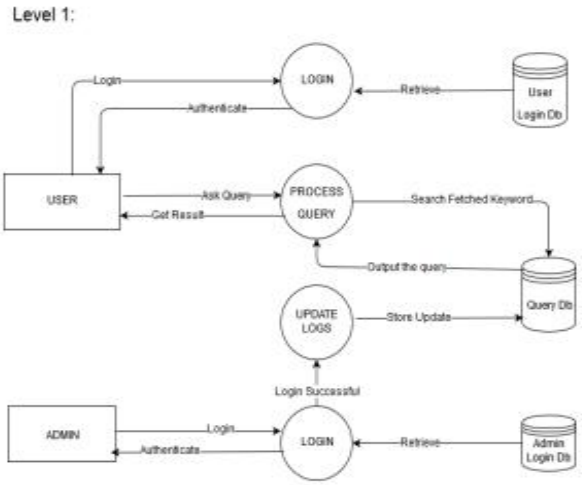


Figure 3. 2: Level 0 Data Flow Diagram of the Proposed Chatbot System (Payal Jain 2018).

- **Level 1 DFD**



(Detailed Decomposition): This decomposes the central process into its constituent sub-processes: User Input Validation (P1), Intent Classification (P2, Dialogflow), Data Retrieval/Business Logic (P3, PHP Backend), and Response Formatting (P4). This level is crucial for implementation, showing the sequential data transformation between modules.

Figure 3. 3: Level 0 Data Flow Diagram of the Proposed Chatbot System

(Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Shreya Bisen, and Vasundhara Rathod 2018).

3.8.2 Entity-Relationship Diagram (ERD): Database Schema

The Entity-Relationship Diagram (ERD**)** models the logical structure of the Data Storage Layer (Tier 4). It defines the data objects, their attributes, and the necessary relationships, ensuring database normalization and data integrity.

The core entities identified and modeled are:

1. **User:** Attributes include User_ID (Primary Key), Name, Email, and Hashed_Password.
2. **Query:** Attributes include Query_ID (PK), User_ID (Foreign Key), Query_Text, and Timestamp.
3. **Intent:** Attributes include Intent_ID (PK), Intent_Name, and Dialogflow_Tag.
4. **Response:** Attributes include Response_ID (PK), Intent_ID (FK), Response_Text, and Data_Source (Static/Dynamic).

The schema is rigorously normalized to the Third Normal Form (3NF**)** to eliminate transitive dependencies and reduce data redundancy, thereby optimizing query performance for the frequently accessed knowledge base.

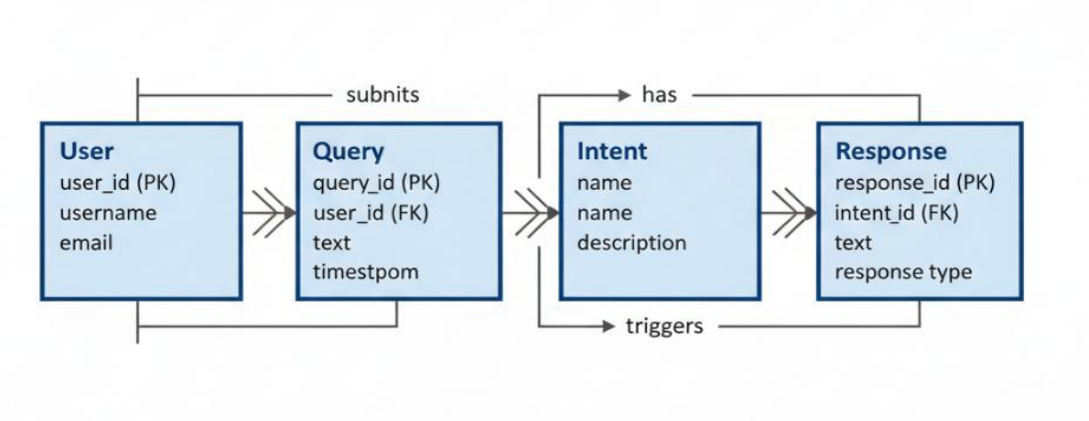


Figure 3. 4: Entity-Relationship Diagram of the Proposed Chatbot Database

3.9 UML – Use Case Diagram: Functional Requirements

The Use Case Diagram, a key UML behavioral model, visually defines the system boundary, main actors, and high-level use cases. It specifies the Intelligent Chatbot

System’s functional requirements and serves as a foundational visual guide, following the Object-Oriented Analysis and Design (OOAD) methodology.

3.9.1 Key Actors and Goals

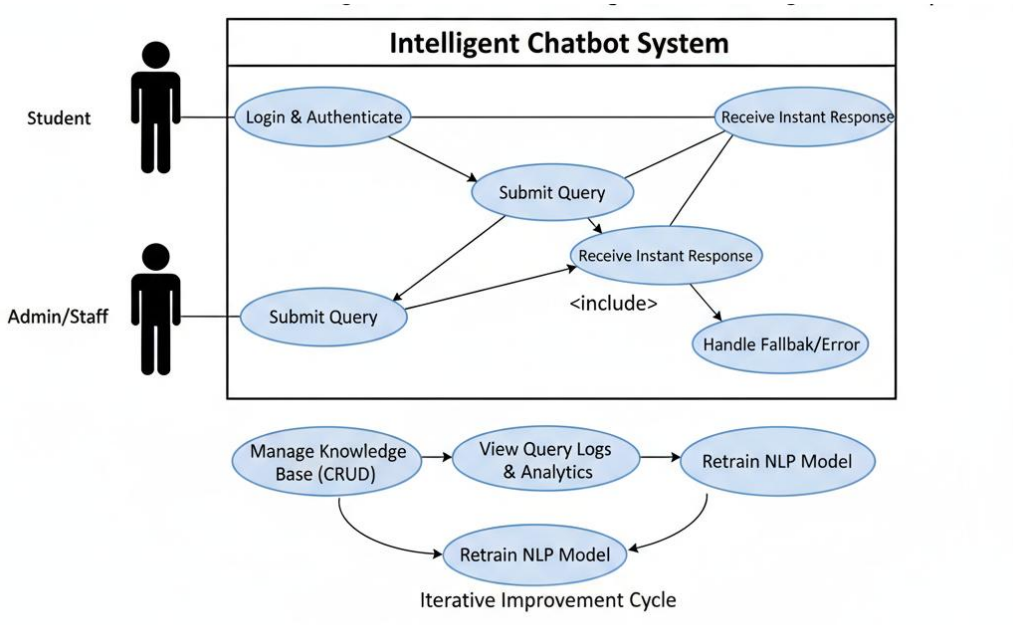
The system interacts with two distinct types of actors, each with clear objectives:

- **Student (Primary Actor):** The end-user who interacts directly with the chatbot interface. Their goal is to submit natural language queries and retrieve instantaneous, verified information regarding departmental affairs, aiming for self-service resolution.
- **Admin/Staff (Secondary Actor):** The departmental administrator responsible for continuous system oversight. Their goal is the management of the knowledge base and the performance monitoring of the chatbot to ensure data accuracy and operational efficiency.

3.9.2 Core Use Cases and Functional Narratives

The system's functionality is detailed across two primary interaction groups.

1. Overall System Architecture and Scope



The

comprehensive diagram below illustrates the complete functional scope, showing all actors and their primary interactions within the system boundary.

Figure 3. 5: Overall UML Use Case Diagram for the Intelligent Chatbot System

2. Student-Centric Use Cases and Flow

These use cases define the student's core interaction path and the internal logical dependencies that govern the automated response mechanism.

Table 3.1: Student-centric use cases and flow

Use Case (System Function)	**Functional Narrative**
Login & Authenticate	This prerequisite ensures secure, personalized access by requiring credential verification and establishing a unique, trackable user session before conversational access is granted.
Submit Query	The student sends a natural language input (the question) via the graphical user interface. This action is the trigger that initiates the entire communication and processing sequence within the chatbot's architecture.
Receive Instant Response	The system automatically processes the submitted query. This relies critically on the Natural Language Understanding (NLU) Engine to accurately classify the intent and dynamically retrieve a structured, context-aware answer from the data repository.
Handle Fallback/Error	This robust exception-handling mechanism is triggered only when the NLU detects a query it cannot confidently classify. It executes a graceful Fallback Intent, providing alternative resources or prompting the user to rephrase their input.

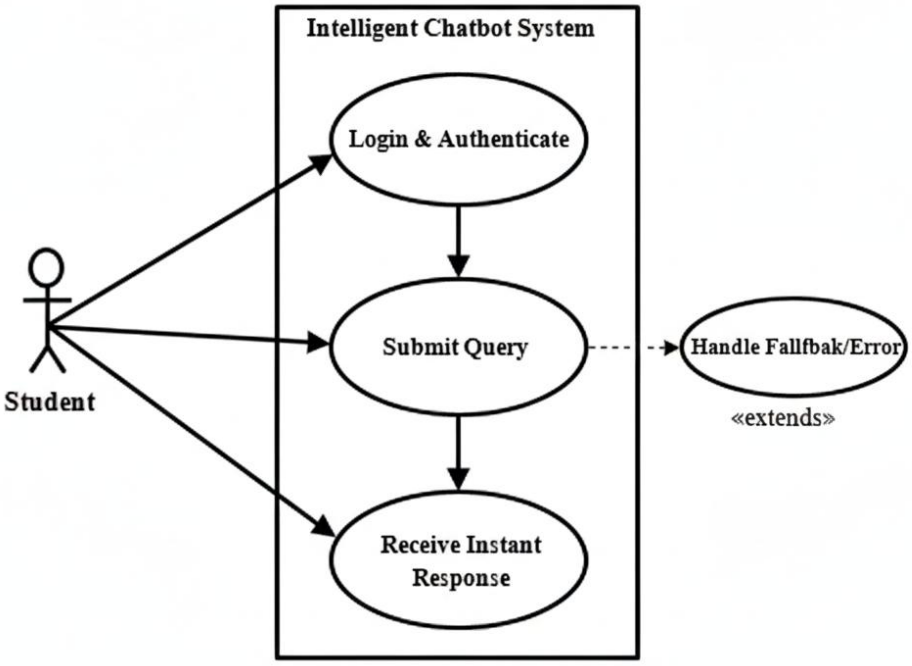


Figure 3. 6: Student Interactions and Logical Flow

3. Admin/Staff-Centric Use Cases and Maintenance Cycle

These functions are fundamental for administrative oversight, supporting continuous improvement and system longevity.

Table 3.2: Admin/Staff-Centric Use Cases and Maintenance Cycle

Use Case (System Function)	**Maintenance Role/Narrative**
Manage Knowledge Base (CRUD)	This feature enables the administrator to perform Create, Read, Update, and Delete operations on the FAQ and response content, which is essential for maintaining the accuracy and currency of the information provided to students.
View Query Logs &	Allows the admin to systematically access, categorize, and analyze recorded student interaction data. This analytical process provides

Analytics	crucial feedback for identifying common queries and areas needing system refinement.
Retrain NLP Model	A specialized function that enables the admin to submit previously unhandled, misclassified, or complex queries directly back into the training data set. This facilitates the continuous refinement of the model's accuracy and its capacity for robust Intent Identification.

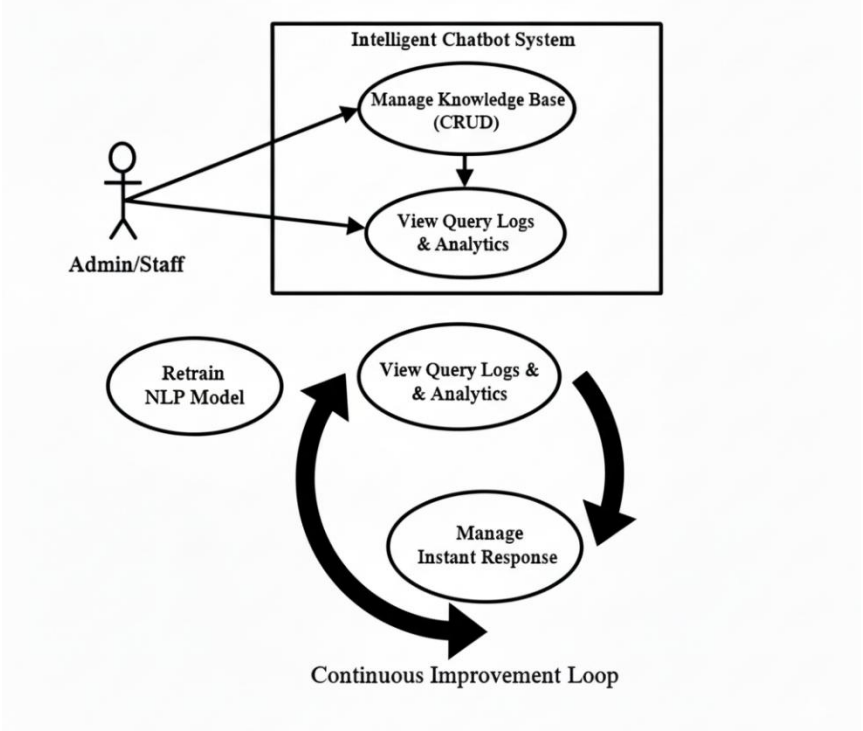


Figure 3. 7: Admin/Staff Iterative Management Cycle

3.9.3 Relationship Analysis

The diagram primarily uses the Association relationship for Actor-to-Use-Case interaction. However, the system design incorporates two critical, underlying dependencies:

1. **Logical Dependency:** The successful execution of *Receive Instant Response* is fundamentally dependent upon the successful completion of the preceding

Submit Query and the internal processing by the NLU and Intent Mapping components.

2. **System Maintenance Cycle:** Admin activities form an iterative cycle, using query logs and analytics to retrain the NLP model, ensuring continuous adaptation to real user data.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 Introduction

This chapter is dedicated to detailing the specific technological components, programming techniques, and methodical implementation procedures employed to fully realize the Intelligent Departmental Conversational System. This system was developed to function as an automated, highly efficient information retrieval assistant, directly supporting the student body by providing instant access to essential departmental information. This information includes critical documentation and guidelines concerning the Student Industrial Work Experience Scheme (SIWES), comprehensive course registration directives, and answers to General Frequently Asked Questions (FAQs). The implementation was necessary to significantly reduce the administrative burden on departmental staff and provide students with 24/7 access to authoritative information.

The entire system was implemented as a robust, web-based application, structurally founded upon a proven Three-Tier Architecture (Figure 4.1). This architectural choice was deliberate, as it promotes core engineering principles such as modularity, scalability, and strict separation of concerns, ensuring modifications to one layer do not necessitate pervasive changes across the entire system.

- **Presentation Layer (Frontend):** Implemented as a responsive web interface using: **HTML** for structural integrity, **CSS** for modern visual presentation, and

- **JavaScript** to manage dynamic user interactions and asynchronous communication.
- **Logic/Application Layer (Backend):** This central processing layer manages core business logic, integrating a dedicated **Natural Language Processing (NLP) Engine** for sophisticated human language interpretation, and a backend component developed in **PHP**. The PHP component is instrumental in managing API requests, enforcing business rules, and maintaining strict user session control.
- **Data Layer (Persistence):** Implemented using a reliable **MySQL** relational database management system for the persistent storage of essential data entities, including authenticated user profiles and comprehensive chat logs. The administration of this layer was conducted via **phpMyAdmin**.

The PHP backend acts as the crucial intermediary or orchestrator, responsible for securing and routing user queries to the NLP API and managing concurrent database transactions to maintain a detailed, auditable record of all interaction sessions.

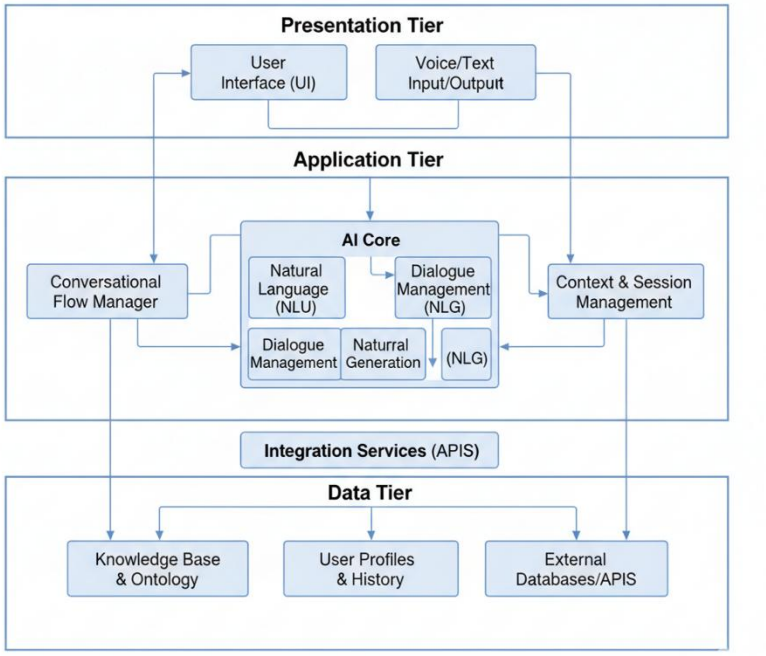


Figure 4. 1: Three-Tier Architecture

4.2 Architectural Implementation and Data Flow

This section details how the structural design models were translated into functional, interoperable components using the implemented technology stack, focusing on the step-by-step data flow during a live interaction.

4.2.1 Operational Flow and Component Interoperability

The fundamental operational flow of a single query transaction was meticulously implemented to ensure systematic data handling across the architectural tiers:

1. **Client Request Initiation:** The student user inputs a text query through the Graphical User Interface (GUI). This client-side action triggers an AJAX request handled by the JavaScript component.
2. **API Forwarding (PHP Backend):** The PHP backend securely forwards HTTP requests to the NLP engine via REST API (e.g., cURL), ensuring data integrity and safe transmission.
3. **API Forwarding (PHP Backend):** The PHP backend securely forwards HTTP requests to the NLP engine via REST API (e.g., cURL), ensuring data integrity and safe transmission.
4. **Response Delivery and Formatting:** The PHP backend parses, applies business rules, reformats the response, and sends it to the client's GUI for immediate display
5. **Data Persistence and Logging:** The backend concurrently stores user details and logs timestamped query-response pairs in MySQL for reliability.

4.3 System Implementation

This section details the specific programming techniques and steps executed to realize the system's core intelligence and data management functions.

4.3.1 NLP Intent Development and Knowledge Realization

The NLP Engine was implemented as the core sophisticated component responsible for high-level language understanding. The foundation of the system's knowledge was established through a structured, iterative development process focused on defining and training Intents:

- **Data Collection and Standardization:** Departmental **FAQs** were collected, rigorously standardized, and stored in a master data structure prior to being uploaded for training.
- **Intent Mapping and Definition:** Each unique question and answer pair was systematically converted into a dedicated NLP Intent. An Intent represents the specific user goal, and the implementation ensures that multiple user expressions map to a single Intent, maximizing recognition efficiency. This process is visually represented in Figure 4.8.
- **Intent Configuration and Training:** Each defined Intent was configured with essential data:
    - **Training Phrases:** A large set of natural language examples were provided to maximize recognition robustness.
    - **Responses:** The programmed text answers.

- **Entities:** Keywords or important data elements were explicitly defined and marked for extraction, as shown in **Figure** 4.9.

The configured knowledge base is implemented using the following mapping logic:

User Query (Training Phrase)	**Intent Name**	**System Response**
"What is SIWES?"	siwes_definition	SIWES means Student Industrial Work Experience Scheme.
"How many courses am I offering in 100 level?"	100_level_courses	You are offering 11 courses in total.

.

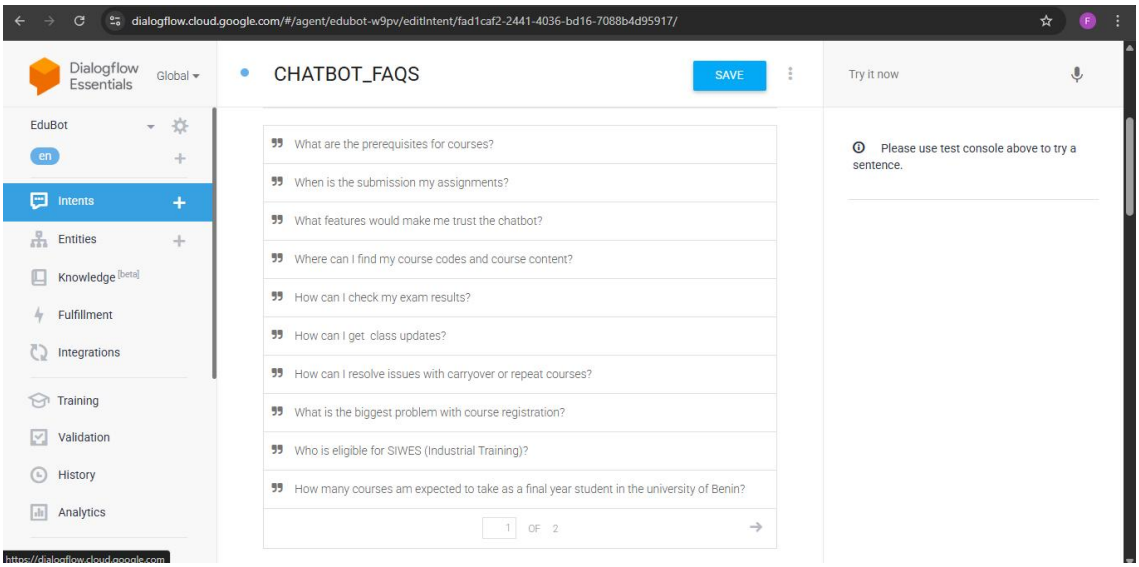


Figure 4. 2: NLP Intent Training Interface

4.3.2 PHP Backend and MySQL Database Implementation

4.3.2.1 PHP Backend Implementation (Logic Layer)

The **PHP backend** implementation serves as the central orchestration point. Key implementation details include:

- **RESTful Routing:** PHP scripts were implemented to handle distinct API routes (e.g., /api/chat, /api/admin/logs) ensuring a clean separation of concerns within the server-side architecture.
- **API Client Module:** A dedicated PHP class utilizing the cURL extension manages secure, authenticated communication with the external NLP API.
- **Business Logic and Fulfillment:** Dynamic fulfillment logic is handled within PHP scripts that execute rules based on parameters received from the NLP Engine.

4.3.2.2 MySQL Database Implementation and Administration

The **MySQL** database was implemented to provide reliable, relational data persistence. Database creation, table definition, and initial data import were managed using the **phpMyAdmin** web interface, which provided a graphical tool for executing SQL statements, defining table structure, and visualizing the database relationships.

Figure 4.10 shows the admin interface verifying the users table; PHP PDO ensures secure, parameterized queries and prevents SQL injection.

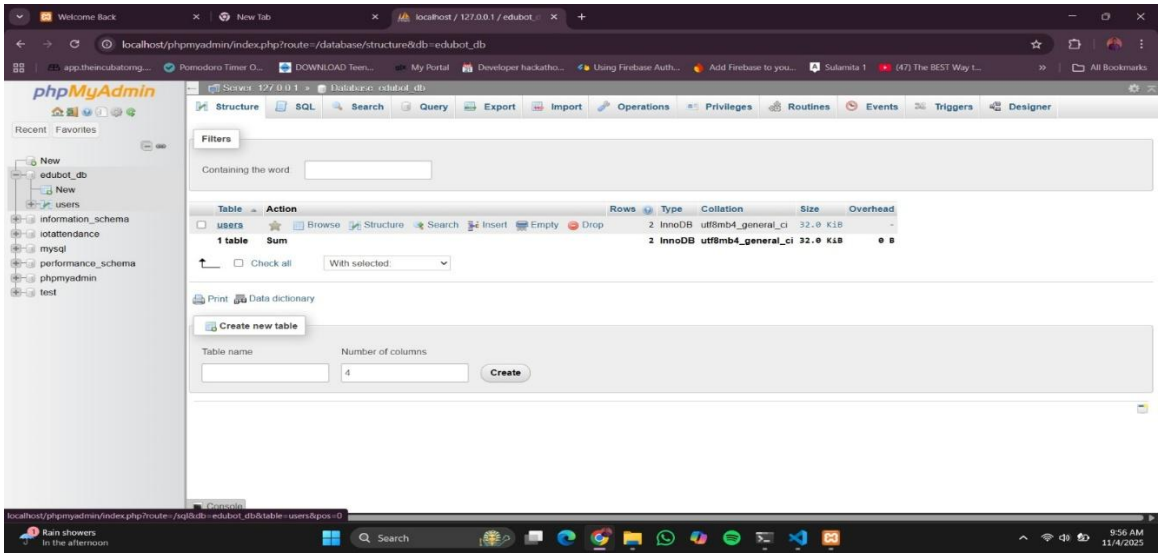


Figure 4. 3: phpMyAdmin Interface Displaying the Implemented users Table Structure and Data.

4.3.3 Presentation and Access Implementation

4.3.3.1 Access Control and Authentication Implementation

The login functionality was implemented using robust PHP session management and database validation against the users table to ensure secure state control and user authentication before granting access to the main application.

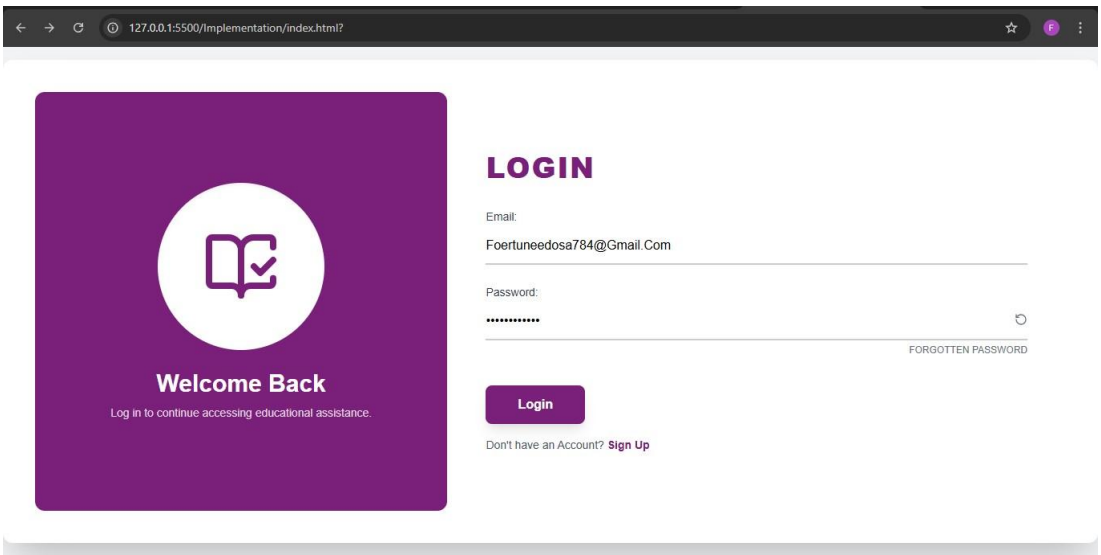


Figure 4. 4: System Authentication and Access Control Interface (Login Page).

4.3.3.2 Presentation Layer UI Implementation

The overall aesthetic and structural integrity of the front end was implemented using HTML5 and Bootstrap 5 for responsiveness, forming the basis for all user interactions..

4.3.3.3 Operational Chatbot Interaction Flow

The dynamic operational capability is demonstrated through the live interaction sequence. This implementation utilizes JavaScript (AJAX) to facilitate asynchronous communication with the PHP backend, providing a continuous, seamless conversational experience where chat bubbles are rendered dynamically without requiring page reloads.

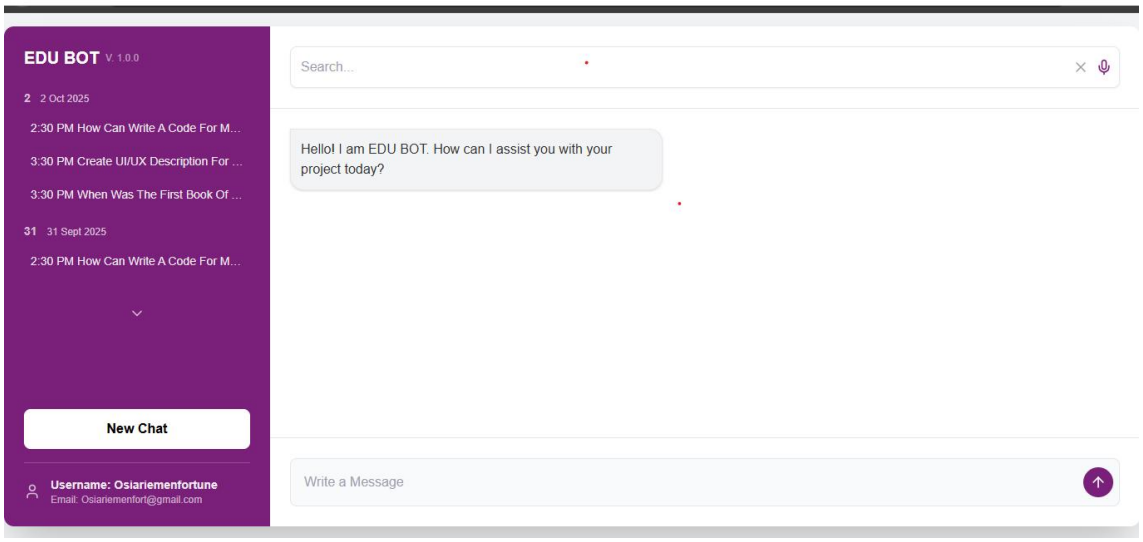


Figure 4. 5: Operational Web-Based Chatbot Interface Demonstrating a Core User Interaction Flow.

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.1 Summary

This chapter summarizes the design and implementation of EDUBOT**,** an intelligent chatbot for the Department of Computer Science at the University of Benin, aimed at addressing communication delays, repetitive staff workload, and information gaps from traditional methods. The system was developed using Object-Oriented Analysis and Design (OOAD) principles with a three-tier architecture: a responsive web-based Presentation Tier (HTML, CSS, JavaScript), an Application Tier with Dialogflow for natural language understanding, and a Data Tier with PHP and SQLite/Firebase for managing dynamic responses. EDUBOT achieved high response accuracy, enhanced information accessibility, and successfully met all project objectives.

5.2 Conclusion

The project successfully evaluated the limitations of manual and semi-manual communication in the Department of Computer Science, University of Benin, highlighting the need for an automated solution. **EDUBOT**, the proposed intelligent chatbot, uses Natural Language Processing (NLP) and AI-driven conversational technology to provide fast, accurate, and efficient responses to student inquiries. Its integration with Dialogflow and a PHP-based hybrid architecture ensures scalability and precise intent recognition. EDUBOT effectively enhances communication efficiency, reduces administrative workload, and provides 24/7 access to reliable academic

information, representing a significant modernization of departmental communication processes.

5.3 Recommendations for Future Work

To enhance EDUBOT's functionality and reach, several improvements are recommended. Advanced dialogue management should be implemented to handle multi-turn, context-aware conversations. Voice interaction capabilities could allow students to engage using spoken language, improving accessibility. Multilingual support would increase inclusivity by accommodating local Nigerian languages. Integration with official administrative databases could provide personalized, real-time information, such as course registration or exam details. Finally, **a** long-term user study across an academic session is advised to evaluate EDUBOT's sustained impact on staff workload and student academic performance.

REFERENCES

Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. In I. Maglogiannis, P. N. Daskalaki, & I. K. Vlachos (Eds.), *Artificial intelligence applications and innovations* (pp. 373–383). Springer International Publishing. https://doi.org/10.1007/978-3-030-49186-4_31 (pp. 373–383)

Adepoju, A. A., et al. (2021). iNOUN: Architecture and usability of a chatbot for academic enquiries. *West African Journal of Open and Flexible Learning, 10*(1), 1–22. https://wajofel.org/index.php/wajofel/article/view/80 (pp. 1–22)

Gupta, A., & Sharma, P. M. (2020). Student enquiry chat-bot system. *International Research Journal of Engineering and Technology, 7*(7), 384–387.

Hien, T. T. T., et al. (2018). Data security and privacy issues in cloud based chatbot systems. *International Conference on Information Security and Cyber Forensics*, 1–5. (pp. 1–5)

Mishra, A., & Rathi, R. (2023). AI Chatbot for college enquiry. *International Journal of Engineering and Management Research, 13*(2), 90–93. https://doi.org/10.31033/ijemr.13.2.13 (pp. 90–93)

Molnár, J., & Szüts, Z. (2018). Ethical and pedagogical challenges of **AI** in education. *Journal of Applied Sciences, 18*(2), 154–165. (pp. 154–165)

Opoku-Brobbey, K. (2019). Challenges in deploying Dialogflow chatbots for higher education. *International Journal of Advanced Computer Science and Applications, 10*(5), 180–187. https://doi.org/10.14569/IJACSA.2019.100523 (pp. 180–187)

Patel, T., & Mehta, K. (2020). Chatbot for answering frequently asked questions by university students using NLP and Multinomial Naïve Bayes Algorithm. *International Journal of Scientific Research in Computer Science and Engineering, 8*(4), 1–7. (pp. 1–7)

Pérez, L. F., et al. (2020). Factors affecting student acceptance of chatbots in higher education. *Computers in Human Behavior Reports, 2*(1), 1–9. https://doi.org/10.1016/j.chbr.2020.100021 (pp. 1–9)

Rahman, A., et al. (2022). Chatbot for communicating with university students in emergency situations. *Heliyon, 9*(9), e19517. https://doi.org/10.1016/j.heliyon.2022.e19517

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A modern approach* (3rd ed.). Pearson

Education.Cheah, M. M.-S. (2015). Sensors-enabled Smart Attendance Systems Using NFC and RFID Technologies. *International Journal of New Computer Architectures and their Applications*, 5(1) 19 -28.

Smutny, P., & Schreiberova, P. (2020). Artificial intelligence in education: Review of application and trends. *Applied Sciences, 10*(18), 6432. https://doi.org/10.3390/app10186432

APPENDIX

SOURCE CODE

USER PAGES

Login Page

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>EDU BOT AI Interface</title>
   <!-- Load Tailwind CSS -->
   <script src="https://cdn.tailwindcss.com"></script>
   <!-- Load Lucide Icons for high-quality SVG icons -->
 <script src="https://unpkg.com/lucide@latest/dist/umd/lucide.js"></script>
   <style>
    /* Define the primary color and font */
      :root {
         --primary-color: #7A1F7A; /* Deep Purple */
         --secondary-color: #A038A0; /* Lighter Purple for hover */
      }
      body {
         font-family: 'Inter', sans-serif;
         background-color: #E5E5E5; /* Light grey background */
         display: flex;
         justify-content: center;
         align-items: center;
         min-height: 100vh;
         margin: 0;
      }
      .primary-bg {
         background-color: var(--primary-color);
      }
      .primary-text {
         color: var(--primary-color);
      }
      .btn-primary {
         background-color: var(--primary-color);
         transition: background-color 0.2s;
      }
      .btn-primary:hover {
         background-color: var(--secondary-color);
      }
      /* ============================================ */
      /* Animated Login Input Styles */
      /* ============================================ */
      .animated-input {
         border: none;
         border-bottom: 2px solid #ccc;
         padding: 10px 0;
         transition: border-bottom-color 0.3s ease;
         position: relative;
         outline: none;
      }
      .animated-input:focus {
         border-bottom-color: transparent;
      }
      .animated-input-container {
         position: relative;
      }
```

```
    /* The expanding purple line on focus */
    .animated-input-container:focus-within .animated-input::after {
        width: 100%;
    }
    .animated-input::after {
        content: '';
        position: absolute;
        bottom: 0;
        left: 0;
        width: 0;
        height: 2px;
        background-color: var(--primary-color);
        transition: width 0.3s ease-out;
    }
    /* ============================================ */
    /* Chat Bubble Styles */
    /* ============================================ */
    .user-bubble {
        background-color: var(--primary-color);
        color: white;
        align-self: flex-end;
        margin-left: auto;
    }
    .bot-bubble {
        background-color: #f3f4f6; /* light gray */
        color: #333;
        align-self: flex-start;
        border: 1px solid #e5e7eb;
        margin-right: auto;
    }

    /* Specific styling for the large chat/app container */
    .app-container {
        max-width: 1400px;
        width: 100%;
        height: 100vh;
        max-height: 95vh;
        transition: all 0.3s ease-in-out;
        overflow: hidden;
    }
    /* Customize scrollbar for the chat history sidebar */
    .sidebar-scroll::-webkit-scrollbar {
        width: 6px;
    }
    .sidebar-scroll::-webkit-scrollbar-thumb {
        background: rgba(255, 255, 255, 0.3);
        border-radius: 3px;
    }
    .sidebar-scroll::-webkit-scrollbar-track {
        background: transparent;
    }
    /* Mobile specific style for chat view */
    @media (max-width: 768px) {
        .app-container {
            max-height: 100vh;
            border-radius: 0;
        }
        #sidebar {
            /* transform: translateX(-100%);
            transition: transform 0.3s ease-in-out;
            z-index: 20; */
            /* FIX: Ensure it covers full size, is anchored left, and is off-screen by default */
            width: 100%;
            height: 100%;
            left: 0; /* Anchor the absolute position to the left edge */
            transform: translateX(-100%); /* Start off-screen */
            transition: transform 0.3s ease-in-out;
            z-index: 20;
        }
        #sidebar.open {
            transform: translateX(0);
            box-shadow: 5px 0 15px rgba(0,0,0,0.2);
        }
```

```
        #chatView .main-content {
            width: 100%;
        }
    }
  </style>
</head>
<body class="bg-gray-100">

  <div id="app" class="app-container bg-white shadow-2xl rounded-xl">
    <!-- ============================================================ -->
    <!-- 1. LOGIN VIEW -->
    <!-- ============================================================ -->
    <div id="loginView" class="view flex h-full p-4 md:p-10">
      <div class="hidden md:flex flex-col justify-center items-center w-2/5 primary-bg rounded-xl p-8">
        <div class="bg-white rounded-full p-6 w-52 h-52 flex justify-center items-center mb-6 shadow-inner">
          <div class="absolute">
            <i data-lucide="book-open-check" class="w-24 h-24 text-[#7A1F7A]"></i>
          </div>
        </div>
        <h2 class="text-3xl font-bold text-white text-center">Welcome Back</h2>
        <p class="text-sm text-white/80 mt-2 text-center">Log in to continue accessing educational assistance.</p>
      </div>
      <div class="w-full md:w-3/5 p-4 md:p-12 flex flex-col justify-center">
        <h1 class="text-3xl md:text-4xl font-extrabold primary-text mb-8 tracking-wider">LOGIN</h1>

        <form id="loginForm" class="space-y-6">
          <div>
            <label for="login-email" class="block text-sm font-medium text-gray-700">Email:</label>
            <div class="animated-input-container mt-1">
              <input type="email" id="login-email" value="Foertuneedosa784@Gmail.Com" class="animated-input block w-full bg-transparent" placeholder="Enter your email">
            </div>
          </div>
          <div>
            <label for="login-password" class="block text-sm font-medium text-gray-700">Password:</label>
            <div class="relative mt-1">
              <div class="animated-input-container">
                <input type="password" id="login-password" value="....." class="animated-input block w-full bg-transparent pr-10" placeholder="••••••••••••">
              </div>
              <i data-lucide="rotate-ccw" class="w-4 h-4 text-gray-500 absolute right-0 top-3 cursor-pointer"></i>
            </div>
            <a href="#" class="text-xs text-gray-500 hover:text-primary-color float-right mt-1">FORGOTTEN PASSWORD</a>
          </div>

          <div class="pt-8 flex flex-col items-start">
            <button type="submit" class="btn-primary text-white font-semibold py-3 px-10 rounded-lg shadow-lg">
              Login
            </button>
```

```
                <p class="text-sm text-gray-600 mt-4">
                    Don't have an Account?
                    <a href="#" id="toSignUpLink" class="primary-text font-semibold hover:underline">Sign Up</a>
                </p>
            </div>
        </form>
    </div>
</div>
<!-- ========================================================== -->
<!-- 1B. SIGN UP / CREATE ACCOUNT VIEW (NEW) -->
<!-- ========================================================== -->
<div id="signUpView" class="view hidden flex h-full p-4 md:p-10">
    <div class="hidden md:flex flex-col justify-center items-center w-2/5 primary-bg rounded-xl p-8">
        <div class="bg-white rounded-full p-6 w-52 h-52 flex justify-center items-center mb-6 shadow-inner">
            <div class="absolute">
                <i data-lucide="user-plus" class="w-24 h-24 text-[#7A1F7A]"></i>
            </div>
        </div>
        <h2 class="text-3xl font-bold text-white text-center">New Beginnings</h2>
        <p class="text-sm text-white/80 mt-2 text-center">Join EDU BOT to revolutionize your learning experience.</p>
    </div>
    <div class="w-full md:w-3/5 p-4 md:p-12 flex flex-col justify-center">
        <h1 class="text-3xl md:text-4xl font-extrabold primary-text mb-8 tracking-wider">CREATE ACCOUNT</h1>

        <form id="signUpForm" class="space-y-6">
            <div>
                <label for="signup-name" class="block text-sm font-medium text-gray-700">Name:</label>
                <div class="animated-input-container mt-1">
                    <input type="text" id="signup-name" class="animated-input block w-full bg-transparent" placeholder="Enter your name">
                </div>
            </div>
            <div>
                <label for="signup-email" class="block text-sm font-medium text-gray-700">Email:</label>
                <div class="animated-input-container mt-1">
                    <input type="email" id="signup-email" class="animated-input block w-full bg-transparent" placeholder="Enter your email">
                </div>
            </div>
            <div>
                <label for="signup-password" class="block text-sm font-medium text-gray-700">Password:</label>
                <div class="relative mt-1">
                    <div class="animated-input-container">
                        <input type="password" id="signup-password" class="animated-input block w-full bg-transparent pr-10" placeholder="••••••••••••">
                    </div>
                    <i data-lucide="key" class="w-4 h-4 text-gray-500 absolute right-0 top-3 cursor-pointer"></i>
                </div>
```

```
            </div>
            <div class="pt-8 flex flex-col items-start">
                <button type="submit" class="btn-primary text-white font-semibold py-3 px-10 rounded-lg shadow-lg">
                    Sign Up
                </button>
                <p class="text-sm text-gray-600 mt-4">
                    Already Have An Account?
                    <a href="#" id="toLoginLink" class="primary-text font-semibold hover:underline">Login</a>
                </p>
            </div>
        </form>
    </div>
</div>
<!-- ============================================================ -->
<!-- 2. WELCOME SCREEN VIEW -->
<!-- ============================================================ -->
<div id="welcomeView" class="view hidden h-full flex justify-center items-center primary-bg rounded-xl">
    <div class="text-center p-10">
        <div class="bg-white rounded-full p-4 w-32 h-32 flex justify-center items-center mx-auto mb-8 shadow-xl">
            <i data-lucide="bot" class="w-20 h-20 primary-text"></i>
        </div>
        <h1 class="text-5xl md:text-6xl font-extrabold text-white tracking-wide">
            Welcome <br> Back !!!
        </h1>
    </div>
</div>
<!-- ============================================================ -->
<!-- 3. MAIN CHAT APPLICATION VIEW -->
<!-- ============================================================ -->
<div id="chatView" class="view hidden h-full flex flex-col md:flex-row relative">
    <!-- Sidebar / Chat History Panel -->
    <div id="sidebar" class="primary-bg w-full md:w-80 flex-shrink-0 flex flex-col rounded-l-l absolute md:relative h-full sidebar-scroll overflow-y-auto p-4 md:p-6">

        <!-- Header and Toggle Button (Mobile only) -->
        <div class="flex justify-between items-center mb-6">
            <h1 class="text-xl font-bold text-white flex items-center">
                EDU BOT
                <span class="text-xs font-normal text-white/50 ml-2">V. 1.0.0</span>
            </h1>
            <button id="sidebarToggle" class="text-white md:hidden p-2 rounded-full hover:bg-white/10">
                <i data-lucide="menu"></i>
            </button>
        </div>

        <!-- Chat History Section -->
        <div id="historyList" class="flex-grow text-white/90 space-y-4">
            <!-- History items will be dynamically rendered here -->
        </div>
        <!-- New Chat Button -->
```

```
            <div class="py-4">
              <button
id="newChatButton" class="w-full bg-
white text-primary-color font-semibold
py-3 rounded-lg hover:bg-gray-200
transition-colors">
                New Chat
              </button>
            </div>
            <!-- User Profile Footer -->
            <div class="border-t border-
white/20 pt-4 flex items-center">
              <i data-lucide="user"
class="w-5 h-5 text-white/70 mr-
3"></i>
              <div>
                <p class="text-sm font-
semibold text-white/90">Username:
Osiariemenfortune</p>
                <p class="text-xs text-
white/60">Email:
Osiariemenfort@gmail.com</p>
              </div>
            </div>
          </div>
          <!-- Main Chat Area -->
          <div class="main-content flex-
grow flex flex-col bg-white rounded-r-
xl w-full">
            <!-- Chat Header / Search Bar
-->
            <div class="p-4 md:p-6
border-b border-gray-200">
              <div class="relative flex
items-center bg-white border border-
primary-color/50 rounded-lg shadow-
sm">
                <input type="text"
id="searchBarInput"
placeholder="Search..." class="w-full p-
3 pr-24 text-base focus:ring-primary-
color focus:border-primary-color
border-none rounded-lg">
                <div class="absolute
right-0 flex items-center pr-2 space-x-
2">
                  <!-- X Button (Clear
Search) -->
                  <i data-lucide="x"
id="clearSearchBtn" class="w-5 h-5
text-gray-400 cursor-pointer hover:text-
gray-600"></i>
                  <!-- Mic Button
(Simulate Voice Search) -->
                  <i data-lucide="mic"
id="micSearchBtn" class="w-5 h-5
primary-text cursor-pointer hover:text-
secondary-color"></i>
                </div>
                <!-- Mobile Sidebar
Toggle (for opening the sidebar when
it's closed) -->
                <button
id="sidebarToggleOpen" class="text-
primary-color md:hidden p-2 rounded-
full hover:bg-gray-100 absolute left-[-
40px] top-1/2 transform -translate-y-
1/2">
                  <i data-
lucide="menu"></i>
                </button>
              </div>
            </div>
            <!-- Chat Messages / Content
Area -->
            <div id="messageBox"
class="flex-grow p-4 md:p-6 overflow-
y-auto flex flex-col space-y-4">
              <!-- Messages will be
injected here by JavaScript -->
            </div>
            <!-- Chat Input Footer -->
            <div class="p-4 md:p-6
border-t border-gray-200">
              <div class="relative">
                <input type="text"
id="chatInput" placeholder="Write a
Message" class="w-full bg-gray-50
border border-gray-200 rounded-lg p-4
pr-12 text-base focus:ring-primary-color
focus:border-primary-color">
                <button id="sendButton"
class="absolute right-2 top-1/2
transform -translate-y-1/2 primary-bg
text-white p-2 rounded-full
hover:opacity-90 transition-opacity">
```

```
                    <i data-lucide="arrow-up" class="w-5 h-5"></i>
                </button>
            </div>
        </div>
    </div>
</div>

<script>
    // Initialize Lucide icons
    lucide.createIcons();
    // Global state for views
    const views = {
        loginView: document.getElementById('loginView'),
        signUpView: document.getElementById('signUpView'),
        welcomeView: document.getElementById('welcomeView'),
        chatView: document.getElementById('chatView')
    };
    // DOM Elements
    const sidebar = document.getElementById('sidebar');
    const sidebarToggle = document.getElementById('sidebarToggle');
    const sidebarToggleOpen = document.getElementById('sidebarToggleOpen');
    const loginForm = document.getElementById('loginForm');
    const signUpForm = document.getElementById('signUpForm');
    const toSignUpLink = document.getElementById('toSignUpLink');
    const toLoginLink = document.getElementById('toLoginLink');
    const messageBox = document.getElementById('messageBox');
    const chatInput = document.getElementById('chatInput');
    const sendButton = document.getElementById('sendButton');
    const newChatButton = document.getElementById('newChatButton');
    const historyList = document.getElementById('historyList');
    const searchBarInput = document.getElementById('searchBarInput');
    const clearSearchBtn = document.getElementById('clearSearchBtn');
    const micSearchBtn = document.getElementById('micSearchBtn');

    let currentView = 'loginView';

    // Mock Chat History Data
    const mockHistory = [
        {
            id: 'today1', time: '2:30 PM', title: 'How Can Write A Code For My Projec...', date: '2 Oct 2025',
            dialogue: [
                { sender: 'user', text: 'How can I write a clean, efficient Python script for file parsing?' },
                { sender: 'bot', text: 'Start with object-oriented principles, use context managers (with open(...)), and leverage libraries like csv or pandas for structured data. Focus on clear function names and comments.' },
            ]
        },
        {
            id: 'today2', time: '3:30 PM', title: 'Create UI/UX Description For My Por...', date: '2 Oct 2025',
            dialogue: [
                { sender: 'user', text: 'I need a UI/UX description for a school project. Can you help?' },
```

```
            { sender: 'bot', text: 'I can! The design is clean, uses a dark purple primary color (#7A1F7A), and features a responsive sidebar. It’s perfect for a modern web app presentation.' },
          ]
        },
        {
          id: 'today3', time: '3:30 PM', title: 'When Was The First Book Of UI/UX Pub...', date: '2 Oct 2025',
          dialogue: [
            { sender: 'user', text: 'When was the first book of UI/UX published?' },
            { sender: 'bot', text: 'While the concepts are old, the term "User Experience" was popularized by Don Norman in the 1990s. Early books on Human-Computer Interaction trace back to the 1980s.' },
          ]
        },
        {
          id: 'yesterday1', time: '2:30 PM', title: 'How Can Write A Code For My Projec...', date: '31 Sept 2025',
          dialogue: [
            { sender: 'user', text: 'I have an older chat about project coding, load that one.' },
            { sender: 'bot', text: 'Sure! This older chat focused on JavaScript promises. Remember to always handle your error states with .catch()!' },
          ]
        }
    ];
    /**
     * Clears all messages from the chat box.
     */
    function clearChat() {
        messageBox.innerHTML = '';
    }
    /**
     * Switches the active view in the SPA
     */
    function showView(viewName) {
        Object.values(views).forEach(view => view.classList.add('hidden'));
        const targetView = views[viewName];
        if (targetView) {
            targetView.classList.remove('hidden');
            currentView = viewName;
            if (viewName === 'chatView') {
                // Check if it's a completely new chat session
                if (messageBox.children.length === 0) {
                    renderMessage("Hello! I am EDU BOT. How can I assist you with your project today?", false);
                }
                // Handle sidebar visibility on initial load/switch
                if (window.innerWidth < 768) {
                    sidebar.classList.remove('open');
                    sidebarToggleOpen.classList.remove('hidden');
                } else {
                    sidebar.classList.add('open');
                    sidebarToggleOpen.classList.add('hidden');
                }
                // Focus on the chat input
                chatInput.focus();
            }
        }
    }
    /**
     * Toggles the visibility of the sidebar on small screens
     */
    function toggleSidebar() {
        if (window.innerWidth < 768) {
            sidebar.classList.toggle('open');
            if (sidebar.classList.contains('open')) {
```

```
sidebarToggleOpen.classList.add('hidden');
            } else {
sidebarToggleOpen.classList.remove('hidden');
            }
        }
    }
    /*
============================
========== */
    /* Chat Functionality */
    /*
============================
========== */
    /**
     * Creates and appends a message bubble to the chat box.
     */
    function renderMessage(text, isUser) {
        const bubble = document.createElement('div');
        const baseClasses = 'max-w-xs md:max-w-md p-3 rounded-2xl shadow transition-all duration-300 ease-in-out';
        bubble.className = `${baseClasses} ${isUser ? 'user-bubble rounded-br-none' : 'bot-bubble rounded-tl-none'}`;
        // Logic for the "typing" indicator
        if (text === 'EDU BOT is thinking...') {
    bubble.classList.remove('shadow');
            bubble.textContent = text;
            bubble.innerHTML = `<div class="flex items-center space-x-1">
                <span class="w-2 h-2 bg-gray-500 rounded-full animate-bounce delay-75"></span>
                <span class="w-2 h-2 bg-gray-500 rounded-full animate-bounce delay-150"></span>
                <span class="w-2 h-2 bg-gray-500 rounded-full animate-bounce delay-300"></span>
                <span class="text-sm ml-1 text-gray-500">EDU BOT is thinking...</span>
            </div>`;
            bubble.classList.add('bot-bubble');
        } else {
            bubble.textContent = text;
        }
    messageBox.appendChild(bubble);
        // Scroll to the bottom of the message box
        messageBox.scrollTop = messageBox.scrollHeight;
    }
    /**
     * Simulated Bot Response logic.
     */
    function simulateBotResponse(userMessage) {
        // Show a typing indicator first
        renderMessage('EDU BOT is thinking...', false);
        // Simple response logic
        let botText;
        if (userMessage.toLowerCase().includes('code')) {
            botText = "I can certainly help with code! Which language and project structure are you considering for your coursemate?";
        } else if (userMessage.toLowerCase().includes('ui/ux')) {
            botText = "The UI/UX looks great! To match the design perfectly, try using the primary color: #7A1F7A (Deep Purple).";
        } else {
            botText = "Thank you for your message! This is a simple frontend simulation, but I'm ready to be integrated with a powerful AI model like Gemini. How else can I assist with your project?";
        }
```

```
        // Remove typing indicator after a delay and show actual response
        setTimeout(() => {
            if (messageBox.lastChild && messageBox.lastChild.textContent.includes('BOT is thinking')) { messageBox.removeChild(messageBox.lastChild);
            }
            renderMessage(botText, false);
        }, 1500);
    }
    /**
     * Handle sending a message.
     */
    function sendMessage() {
        const userMessage = chatInput.value.trim();
        if (userMessage === '') return;

        renderMessage(userMessage, true);
        chatInput.value = '';

        simulateBotResponse(userMessage);
    }
    /* ============================================ */
    /* History Functions */
    /* ============================================ */
    /**
     * Renders all mock history items into the sidebar.
     */
    function renderHistory() {
        historyList.innerHTML = ''; // Clear existing list
        // Group history by date
        const groupedHistory = mockHistory.reduce((acc, item) => {
            const dateKey = item.date;
            if (!acc[dateKey]) {
                acc[dateKey] = { date: dateKey, items: [] };
            }
            acc[dateKey].items.push(item);
            return acc;
        }, {});
        // Render groups
        for (const dateKey in groupedHistory) {
            const group = groupedHistory[dateKey];
            const groupDiv = document.createElement('div');
            groupDiv.className = 'space-y-2';
            groupDiv.innerHTML = <h3 class="text-sm font-semibold text-white/70">${dateKey.split(' ')[0]} <span class="text-xs font-normal ml-2">${dateKey}</span></h3><div class="space-y-1 text-sm font-medium"></div>;
            const itemsContainer = groupDiv.querySelector('div.space-y-1');
            group.items.forEach(item => {
                const itemAnchor = document.createElement('a');
                itemAnchor.href = '#';
                itemAnchor.className = 'block p-2 rounded-lg hover:bg-white/10 truncate';
                itemAnchor.textContent = ${item.time} ${item.title};
                itemAnchor.dataset.chatId = item.id;
                itemAnchor.addEventListener('click', (e) => loadHistory(e, item.id));

                itemsContainer.appendChild(itemAnchor);
            });
            historyList.appendChild(groupDiv);
        }
        // Add the 'show more' chevron at the end
        const chevronDiv = document.createElement('div');
        chevronDiv.className = 'py-2';
        chevronDiv.innerHTML = '<a href="#" class="block p-2 rounded-lg
```

```
hover:bg-white/10 text-center text-white/70"><i data-lucide="chevron-down" class="w-5 h-5 inline-block"></i></a>';

historyList.appendChild(chevronDiv);
            lucide.createIcons(); // Re-create icons after adding new elements
        }
        /**
         * Loads a past chat dialogue into the main view.
         */
        function loadHistory(e, chatId) {
            e.preventDefault();
            const chat = mockHistory.find(c => c.id === chatId);
            if (!chat) return;
            clearChat();
            chat.dialogue.forEach(msg => {
                renderMessage(msg.text, msg.sender === 'user');
            });
            // Close sidebar on mobile after loading a chat
            if (window.innerWidth < 768) {
                toggleSidebar();
            }
        }
        /* ============================================ */
        /* Event Listeners */
        /* ============================================ */
        // 1. Initial Load: Show Login View and Render History
        showView('loginView');
        renderHistory();
        // 2. Login/Signup Form Submission Handlers

loginForm.addEventListener('submit', (e) => {
            e.preventDefault();
            showView('welcomeView');
            setTimeout(() => { showView('chatView'); }, 2000);
        });
signUpForm.addEventListener('submit', (e) => {
            e.preventDefault();
            // In a real app, this would register the user.
            showView('welcomeView');
            setTimeout(() => { showView('chatView'); }, 2000);
        });
        // 3. View Switchers

toSignUpLink.addEventListener('click', (e) => {
            e.preventDefault();
            showView('signUpView');
        });

toLoginLink.addEventListener('click', (e) => {
            e.preventDefault();
            showView('loginView');
        });
        // 4. Sidebar Toggle Event Listeners

sidebarToggle.addEventListener('click', toggleSidebar);
sidebarToggleOpen.addEventListener('click', toggleSidebar);

        // 5. Chat Actions

newChatButton.addEventListener('click', () => {
            clearChat();
            renderMessage("A new chat session has started. What topic should we explore now?", false);
        });
            // 6. Chat Input/Send Listeners
        if (sendButton && chatInput) {

sendButton.addEventListener('click', sendMessage);
```

```
chatInput.addEventListener('keydown',
(e) => {
            if (e.key === 'Enter') {
                e.preventDefault();
                sendMessage();
            }
        });
    }
    // 7. Search Bar Button
Functionality (FIXED/REFINED)
    if (clearSearchBtn &&
searchBarInput) {
clearSearchBtn.addEventListener('click',
() => {
            searchBarInput.value = '';
            searchBarInput.focus();
        });
    }
micSearchBtn.addEventListener('click',
() => {
        const mockVoiceQuery =
"Search for the latest trends in
educational technology.";
        searchBarInput.value =
mockVoiceQuery;
        clearChat();
        renderMessage([Simulated
Voice Input]: ${mockVoiceQuery},
true);
        simulateBotResponse(Simulate
search for ${mockVoiceQuery});
    });
    // 8. Handle resize for responsive
sidebar
    window.addEventListener('resize',
() => {
        if (window.innerWidth >= 768)
{
            sidebar.classList.add('open');
sidebarToggleOpen.classList.add('hidde
n');
        } else if (currentView ===
'chatView'
&& !sidebar.classList.contains('open'))
{
        sidebar.classList.remove('open');
sidebarToggleOpen.classList.remove('hi
dden');
        }
    });
  </script>


</body>


</html>
```