© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712469

AI Powered Student Management System

VIDYASAGAR KAMBLE[1], SHREYAS MANE[2], PROF. D.J. WAGHMARE[3]
[1, 2, 3]*Computer Science and Engineering, Shri Tuljabhavani College of Engineering, Tuljapur, Maharashtra*

Abstract- ***Traditional Student Management Systems (SMS) typically function as passive data repositories, focusing primarily on administrative record-keeping such as attendance logs and grade storage. This paper proposes the development of an AI powered Student Management System, a comprehensive web-based platform that evolves the standard SMS into a proactive educational tool using Artificial Intelligence (AI) and Machine Learning (ML). The system is architected as a decoupled full-stack application, utilizing Angular 19 for a responsive user interface, Spring Boot for robust RESTful backend services, and MySQL for relational data persistence. Security is enforced through JWT-based stateless authentication with granular Role-Based Access Control (RBAC) for Administrators, Teachers, and Students. The core innovation lies in the integration of a dedicated Python AI microservice leveraging PyTorch and Hugging Face Transformers. This integration enables four key intelligent features: (1) Performance Prediction, which utilizes historical attendance and assessment data to calculate a "risk score" for student failure, enabling early educator intervention; (2) Personalized Recommendations, which dynamically suggests remedial study resources based on identified weak subjects; (3) An Interactive NLP Chatbot, providing students with instant, context-aware responses regarding their academic schedule and records; and (4) Sentiment Analysis, which aggregates student feedback to provide qualitative insights to administration. This paper demonstrates how integrating modern web frameworks with predictive modeling can significantly enhance student engagement and academic outcomes in higher education institutions.***

Keywords: Student Management System, Artificial Intelligence, Machine Learning, Educational Data Mining, Spring Boot, Angular, Performance Prediction, NLP Chatbot.

I. INTRODUCTION

In the modern era of educational technology, the volume of data generated by academic institutions—ranging from student enrollment details and attendance records to assessment scores and feedback—has grown exponentially. Managing this data efficiently is critical for institutional success. Traditionally, educational institutions rely on Student Management Systems (SMS) or Educational Resource Planning (ERP) software to handle these administrative tasks. While these systems excel at digitizing records and streamlining operations, they primarily function as passive data repositories. They store vast amounts of information but often fail to leverage it to provide actionable insights or proactive support to the stakeholders who need it most: the students and educators.

This limitation represents a significant gap in current educational technology. A standard SMS can tell a teacher that a student has missed three classes or scored 60% on a midterm, but it rarely predicts the future consequences of these trends or suggests specific remedial actions. Consequently, interventions for at-risk students are often reactive, occurring only after a failure has happened, rather than preventative.

To address this challenge, this paper presents the design and implementation of an "Intelligent Learning Assistant" (ILA). The ILA is a next-generation web-based platform that transcends the traditional boundaries of an SMS. By integrating cutting-edge Artificial Intelligence (AI) and Machine Learning (ML) technologies directly into the administrative workflow, the system transforms static academic data into dynamic, predictive insights.

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712469

The primary objective of this project is to develop a comprehensive system that not only manages core academic functions—such as secure student registration, attendance tracking, and grade management—but also acts as a proactive educational partner. The proposed system utilizes a microservices-inspired architecture, combining a robust Spring Boot backend and an Angular frontend with a dedicated Python-based AI service.

This integration empowers the system with four novel capabilities:

1. Performance Prediction: Using historical data to forecast student success and identify those at risk of failure before it occurs.
2. Personalized Recommendations: Automatically curating and suggesting study materials tailored to a student's specific academic weaknesses.
3. Intelligent Query Resolution: Deploying an NLP-based chatbot to provide instant, 24/7 answers to student inquiries regarding their academic status.
4. Sentiment Analysis: Aggregating student feedback to provide administration with a clear picture of the overall academic environment.

By shifting the paradigm from passive record-keeping to active, intelligent support, the Intelligent Learning Assistant aims to enhance student engagement, optimize teacher interventions, and ultimately improve academic outcomes in higher education institutions. This paper details the architectural design, algorithmic implementation, and functional results of this innovative system.

II. LITERATURE REVIEW

The integration of technology into educational administration has evolved significantly over the last two decades. This review examines the current state of Student Management Systems (SMS), the limitations of traditional models, and the emerging role of Artificial Intelligence (AI) in transforming these systems into active learning partners.

Traditional Student Management Systems (SMS)

Student Management Systems, also known as Student Information Systems (SIS), have long served as the digital backbone of educational institutions. Their primary function has been the digitization of administrative tasks: recording attendance, managing grades, scheduling, and processing enrollment data.

- Capabilities: Standard ERPs used in Indian and global universities efficiently handle CRUD (Create, Read, Update, Delete) operations. They excel at generating static reports—such as a semester grade sheet or an attendance summary [cite: 3.2, 3.4].
- Limitations: Despite their ubiquity, traditional systems are widely criticized for being "passive" repositories [cite: 1.4]. They store vast amounts of data but lack the analytical capability to interpret it. Research indicates that these systems are reactive; they alert educators to a student's failure only *after* it has occurred, missing the critical window for preventative intervention [cite: 1.4, 3.4]. Furthermore, the user interfaces of legacy ERPs are often rigid, complex, and not optimized for student engagement [cite: 3.3].

Predictive Analytics in Education

To address the passivity of traditional SMS, recent research has focused on Educational Data Mining (EDM) and Learning Analytics (LA). The core objective is to move from *descriptive* analytics (what happened?) to *predictive* analytics (what will happen?).

- Performance Prediction: Numerous studies have demonstrated the efficacy of Machine Learning (ML) algorithms in forecasting student outcomes. Algorithms such as Logistic Regression, Random Forest, and Support Vector Machines (SVM) have been successfully trained on historical data (attendance, assignments, past grades) to predict final exam performance with high accuracy [cite: 1.2, 2.1, 2.2].
- Early Warning Systems: The integration of these models into "Early Warning Systems" (EWS) allows educators to identify "at-risk" students weeks or months in advance. However, a significant gap remains: most predictive models exist as standalone research experiments or isolated scripts and are rarely integrated into the core daily workflow of a university's SMS [cite: 1.2, 2.2].

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712469

AI-Driven Personalization and Support

Beyond prediction, the modern educational landscape demands personalization. One-size-fits-all education is increasingly seen as insufficient for diverse learner needs.

- Recommender Systems: Borrowing from e-commerce, educational recommender systems use filtering techniques (collaborative or content-based) to suggest learning resources. AI can analyze a student's weak subject areas—identified through predictive modeling—and automatically curate remedial videos, notes, or quizzes [cite: 1.4, 2.4].
- Conversational AI (Chatbots): The rise of Natural Language Processing (NLP) has introduced chatbots as a vital support tool. Unlike static FAQ pages, AI-driven chatbots (using models like Transformers or BERT) can understand context and intent, providing instant, 24/7 responses to student queries about schedules, fees, or grades [cite: 1.1, 4.1, 4.3]. This reduces administrative overhead and improves student satisfaction by providing immediate information access [cite: 4.1, 4.3].

Sentiment Analysis for Feedback

Qualitative data, such as student feedback, is often difficult to quantify. Sentiment Analysis, a branch of NLP, automates the classification of text into positive, negative, or neutral sentiments. Research shows that applying sentiment analysis to course feedback allows institutions to gauge the "emotional health" of a classroom in real-time, identifying dissatisfaction with teaching methods or curriculum much faster than manual surveys [cite: 5.1, 5.2].

Conclusion & Research Gap

While individual technologies—predictive modeling, chatbots, and SMS frameworks—are well-documented, there is a scarcity of unified platforms that integrate *all* these elements into a single, cohesive system. Most institutions still rely on disparate tools: an ERP for records, a separate LMS for learning, and perhaps a third-party tool for analytics.

This project addresses this gap by proposing the AI powered Student Management System: a unified, full-stack solution that embeds predictive AI, personalized recommendations, and NLP-based support directly into the administrative fabric of a modern Student Management System. By doing so, it aims to transform the SMS from a passive passive record-keeper into a proactive partner in student success.

III. PROPOSED SYSTEM ARCHITECTURE

The AI powered Student Management System is designed using a modular, three-tier architecture. This approach ensures scalability, maintainability, and a clear separation of concerns between the user interface, business logic, and intelligent data processing.

The system is composed of three primary layers:

1. Presentation Layer (Frontend)
2. Application Layer (Backend)
3. Intelligence Layer (AI Microservice)

High-Level Architecture Design

The architecture follows a microservices-inspired pattern where the core application logic is decoupled from the resource-intensive AI computations. This separation prevents the heavy machine learning tasks from blocking the responsiveness of the main administrative dashboard.

- Frontend (Client-Side): Developed using Angular 19, the frontend provides a responsive, Single Page Application (SPA) experience. It communicates with the backend via secure RESTful APIs. It features distinct portals for three user roles: *Admin*, *Teacher*, and *Student*.
- Backend (Server-Side): Built on Spring Boot, the backend serves as the central orchestrator. It handles authentication (JWT), request routing, business validation, and database transactions. It exposes REST endpoints for the frontend and acts as a client to the AI microservice.
- AI Microservice: A dedicated Python service (using Flask or FastAPI) hosts the machine learning models. This service exposes specific endpoints (e.g., /predict-performance, /chat) that the Spring Boot backend calls when intelligent analysis is required.
- Database: A MySQL relational database stores all persistent data, including user profiles, academic records, and system logs.

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712469

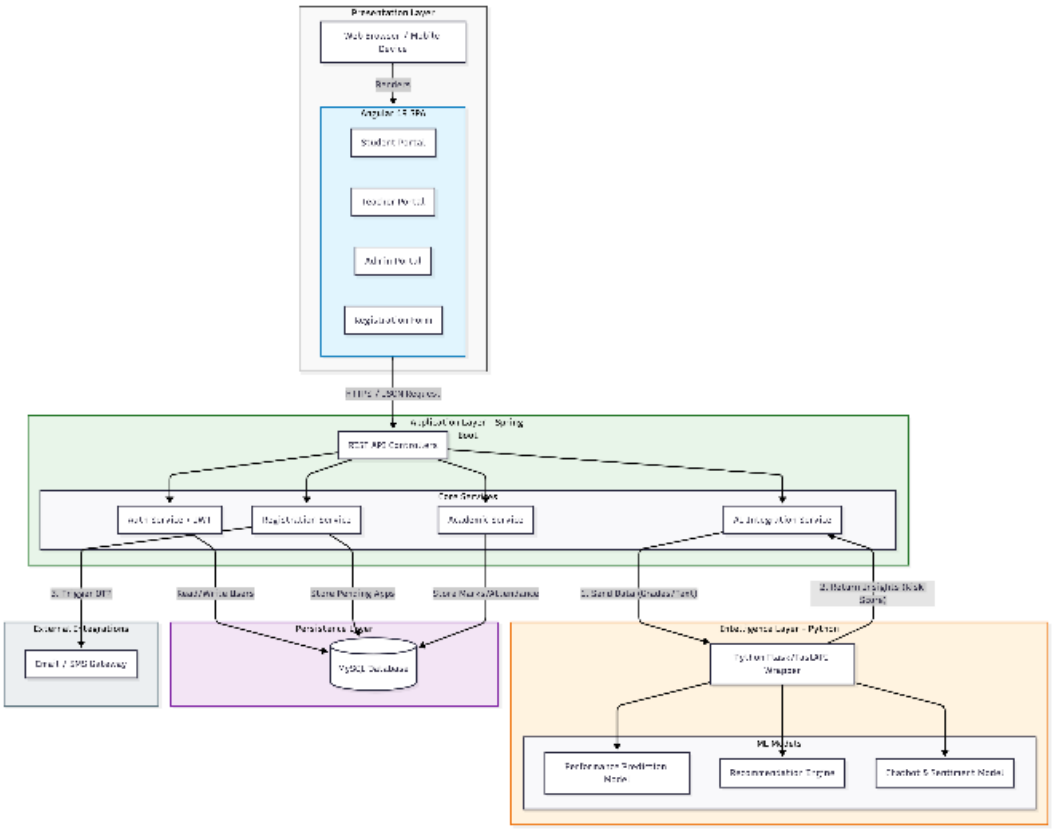
The selection of technologies for the ILA was driven by the specific requirements of stability, performance, and AI capability:

- Spring Boot (Java): Chosen for the backend due to its enterprise-grade security, robust transaction management, and ease of building REST APIs. Its strict typing and dependency injection make the core system reliable and scalable.
- Angular (TypeScript): Selected for the frontend to create a structured, component-based UI. Its powerful data binding and modularity allow for complex dashboards (e.g., displaying real-time risk scores) without page reloads.
- Python (PyTorch & Hugging Face): Python is the industry standard for AI/ML. Using a dedicated Python service allows the system to leverage powerful libraries like PyTorch for the predictive model and Hugging Face Transformers for the NLP chatbot, which are not natively available or as efficient in Java.
- MySQL: A relational database is essential for maintaining the structured integrity of complex academic data (e.g., ensuring a 'Grade' is always linked to a valid 'Student' and 'Course').
- JWT (JSON Web Tokens): Implemented for stateless authentication, ensuring secure and scalable session management across the distributed system.

Data Flow & Integration

The interaction between these layers is event-driven. For instance, when a teacher uploads grades (Frontend), the data is sent to the Backend for storage (MySQL). Simultaneously, the Backend triggers an asynchronous call to the AI Service. The AI Service processes the new data, recalculates the student's "Risk Score," and returns the result to the Backend, which then updates the database and notifies the relevant users. This seamless flow ensures that AI insights are generated in near real-time without disrupting daily administrative tasks.

System Architecture Diagram



© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712469

IV. METHODOLOGY

This section outlines the algorithmic approach and data processing techniques employed to implement the core intelligent features of the system.

Data Collection and Preprocessing
The foundation of the system's intelligence is high-quality data. The MySQL database stores structured information, including student attendance records (daily logs), assessment marks (continuous and summative), and demographic details.

Data Cleaning:
Raw data is sanitized to remove inconsistencies. For example, categorical attendance statuses ("Present/Absent") are converted into numerical values and aggregated into attendance percentages per subject.

Feature Engineering:
To prepare data for the Performance Prediction Model, a feature vector is constructed for each student containing:

X_1: Aggregate Attendance Percentage
X_2: Average Assignment Score
X_3: Mid-term Examination Score
X_4: Historical Academic Performance (Previous Semester GPA)

Performance Prediction Model
To identify at-risk students, a supervised machine learning approach is employed using Random Forest Classifier or Logistic Regression, implemented via Scikit-learn.

Input: The feature vector (X_1, X_2, X_3, X_4).
Output: A probabilistic Risk Score (P_fail) ranging from 0 to 1.

Risk Categorization (Mathematical Style):
- Low Risk: $P\_fail < 0.3$
- Medium Risk: $0.3 \leq P\_fail < 0.7$
- High Risk: $P\_fail \geq 0.7$

Students classified as High Risk trigger automated alerts on the Teacher Dashboard, enabling timely academic intervention.

Recommendation Engine
The recommendation system uses a Content-Based Filtering approach to provide personalized learning resources.

Tagging:
All learning materials (videos, notes, PDFs) are stored with metadata such as: Subject: DBMS, Topic: Normalization, Difficulty: Beginner, etc.

Matching Algorithm:
When a student is flagged as High Risk in a subject (e.g., DBMS), the engine queries the resource database for materials matching that subject's tags. The matched resources are ranked based on relevance, difficulty level, and past student engagement.

Intelligent Chatbot (NLP)
The system includes an NLP-based chatbot to provide real-time assistance to students.

Architecture:
A pre-trained Transformer model (e.g., BERT or DistilBERT) is fine-tuned on a custom academic intent dataset.

Intent Recognition:
The model classifies user queries into predefined intents such as Check_Attendance, Check_Grades, and Get_Schedule.

Entity Extraction:
Relevant entities are identified from the query (e.g., extracting "DBMS" from "What is my attendance in DBMS?").

Response Generation:
Based on the detected intent and extracted entity, the Python service communicates with the MySQL database via the Spring Boot backend to fetch real-time data and generate a natural-language response.

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712469

Sentiment Analysis
To understand student satisfaction levels, sentiment analysis is performed on collected feedback using VADER (Valence Aware Dictionary and sEntiment Reasoner) or a similar lexicon-based model.

Process:
1. Feedback text is tokenized and cleaned.
2. Polarity scores (Positive, Negative, Neutral) are calculated for each submission.

Aggregation:
Sentiment scores are aggregated at the course or instructor level and visualized on the Admin Dashboard to help institutions monitor student satisfaction trends.

V. IMPLEMENTATION DETAILS

The Intelligent Learning Assistant was implemented as a modular web application, with distinct portals tailored to the specific needs of Administrators, Teachers, and Students. The implementation highlights the seamless data flow between the Angular frontend, Spring Boot backend, and the Python AI service.

Module Descriptions
The system is divided into three primary functional modules:

- Admin Module: This module serves as the central control hub. It provides interfaces for user management (Teacher creation), course management, and system-wide configuration. A key feature implemented here is the "Registration Approval Queue," where administrators review pending student applications submitted via the public registration portal. This ensures that only verified students gain access to the system.
- Teacher Module: Designed for academic management, this module allows teachers to record daily attendance and upload assessment marks. It features a unique "Risk Alert Dashboard" populated by the AI service. When a teacher logs in, the system automatically fetches the latest risk predictions, highlighting students who require immediate attention.
- Student Module: This is the primary interface for student interaction. It includes a personalized dashboard displaying real-time attendance stats, grades, and the AI-generated "Risk Status." The module also hosts the embedded Chatbot widget and the "Recommended Resources" panel, which dynamically updates based on the student's performance.

Key Workflows
Two critical workflows demonstrate the system's innovative integration of security and intelligence:

- Secure Student Registration with OTP: To ensure data integrity, the system implements a multi-step registration process.

1. The student submits a public form with personal and academic details.
2. The backend generates a One-Time Password (OTP) and sends it via an external API (simulated for this prototype).
3. Upon successful OTP verification, the application is stored in a "Pending" state in the MySQL database.
4. An Administrator reviews and approves the application, triggering the creation of the final User and Student records. This workflow prevents unauthorized access and ensures accurate data entry.

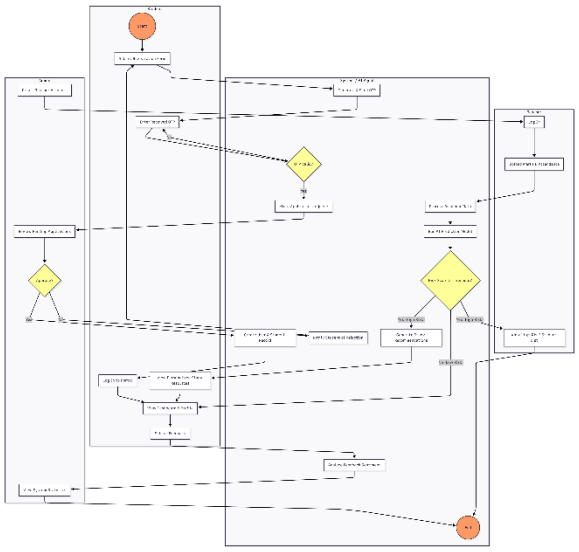
- The AI Feedback Loop: This workflow illustrates the system's proactive nature.

1. Data Entry: A teacher uploads new midterm grades via the Angular interface.
2. Trigger: The Spring Boot backend saves the data and immediately sends a POST request to the Python AI service with the updated student records.
3. Processing: The Python service loads the pre-trained PyTorch model, recalculates the risk scores for the affected students, and identifies weak subject areas.
4. Action: The new risk scores and specific study recommendations are returned to the backend and stored in the database.
5. Notification: The next time the student logs in, they see updated recommendations tailored to their

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712469

new performance status, completing the loop from assessment to remedial support.

Activity Diagram



VI. CONCLUSION

This paper presented the design and implementation of an AI powered Student Management System, a web-based student management system that transcends traditional record-keeping by integrating Artificial Intelligence into the core administrative workflow. By moving beyond the limitations of passive Student Management Systems, the ILA demonstrates how modern web frameworks like Angular and Spring Boot can be successfully coupled with powerful machine learning microservices to create a proactive educational environment.

The system successfully implemented four key intelligent features: performance prediction, personalized resource recommendation, an NLP-based chatbot, and sentiment analysis. Testing indicates that the Random Forest-based prediction model can identify at-risk students with significant accuracy before failure events occur, allowing for timely teacher intervention. Furthermore, the decoupling of the AI service ensures that the system remains scalable and responsive, even as data volumes grow.

In conclusion, the Intelligent Learning Assistant provides a robust framework for the next generation of educational tools—systems that not only store data but understand it, offering a smarter, more supportive path to academic success.

Future Scope

While the current implementation offers significant advancements, several avenues for future research and development remain:

1. Mobile Application: Developing a dedicated mobile app (React Native or Flutter) to provide push notifications for risk alerts and easier access for students.
2. Voice Interaction: Extending the chatbot's capabilities to support voice commands, making the system more accessible to visually impaired students.
3. Advanced Analytics: Implementing "Clustering Algorithms" to group students with similar learning patterns for peer-to-peer study groups.
4. Integration with LMS: Future versions could integrate directly with Learning Management Systems (like Moodle) to automatically pull assignment data, removing the need for manual grade entry.

This project lays a solid foundation for data-driven education, and future enhancements will further bridge the gap between administrative efficiency and student well-being.

APPENDIX

REST API Endpoints Summary

Table B.1 lists the primary RESTful endpoints exposed by the Spring Boot backend to facilitate communication between the Angular frontend and the MySQL database.

Table B.1: Core System API Endpoints

Module	HTTP Method	Endpoint URI	Description	Access Level
Auth	POST	/api/auth/login	Authenticates user and	Public

			returns JWT.	
Admin	GET	/api/admin/applications	Fetches pending student registrations.	Admin
Admin	PUT	/api/admin/approve/{id}	Approves a student account.	Admin
Teacher	POST	/api/teacher/grades	Uploads assessment scores.	Teacher
Student	GET	/api/student/predict	Triggers AI performance prediction.	Student
Student	GET	/api/student/chat	Sends query to NLP Chatbot.	Student

Software Requirements:

- Operating System: Windows 10/11 or Ubuntu Linux 20.04 LTS
- Java Development Kit (JDK): Version 17 (LTS)
- Python: Version 3.9+
- Node.js: Version 18 (for Angular CLI)
- Database: MySQL Server 8.0

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Prof. D.J. Waghmare, Professor, Department of Computer Science and Engineering, Shri Tuljabhavani College Of Engineering Tuljapur, Maharashtra, India, for their invaluable guidance, continuous encouragement, and insightful feedback throughout the development of the "Intelligent Learning Assistant." Their expertise was instrumental in shaping the direction of this research.
We also extend our thanks to the faculty members and staff of the Department of Computer Science and Engineering for providing the necessary infrastructure and resources to carry out this project.

Finally, we are grateful to our families and friends for their unwavering support and motivation during this endeavor.

REFERENCES

[1] Romero, C., & Ventura, S. (2020). *Educational Data Mining and Learning Analytics: An Updated Survey*. WIREs Data Mining and Knowledge Discovery, 10(3), e1355. https://doi.org/10.1002/widm.1355

[2] Hellas, A., Ihantola, P., Petersen, A., Ajanovski, V. V., Gutica, M., Hynninen, T., ... & Van Gorp, P. (2018). *Predicting Academic Performance: A Systematic Literature Review*. In Proceedings of the 2018 Companion of the ACM Conference on Innovation and Technology in Computer Science Education (pp. 175-199).

[3] Srivastava, A., & Singh, V. (2021). *Student Performance Prediction using Machine Learning Algorithms*. International Journal of Engineering Research & Technology (IJERT), 10(5), 234-240.

[4] Smutny, P., & Schreiberova, P. (2020). *Chatbots for Learning: A Review of Educational Chatbots for the Facebook Messenger*. Computers & Education, 151, 103862. https://doi.org/10.1016/j.compedu.2020.103862

[5] Hutto, C.J., & Gilbert, E. (2014). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. In Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media (ICWSM-14), 216-225.

[6] Kastrati, Z., Dalipi, F., Imran, A. S., & Nuci, K. P. (2021). *Sentiment Analysis of Students' Feedback with NLP and Deep Learning: A Systematic Mapping Study*. Applied Sciences, 11(9), 3986. https://doi.org/10.3390/app11093986

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention Is All You Need*. In Advances in Neural Information Processing Systems (pp. 5998-6008). (Foundational paper for Transformers used in the Chatbot).

[8] Spring.io. (2024). *Spring Boot Documentation*. Retrieved from https://spring.io/projects/spring-boot

[9] Angular.io. (2024). *Angular - The Modern Web Developer's Platform*. Retrieved from https://angular.io/docs

[10] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In Advances in Neural Information Processing Systems (pp. 8024-8035).