PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

AI POWERED STUDENT ASSISTANCE CHATBOT FOR DEPARTMENT OF TECHNICAL EDUCATION

A PROJECT REPORT

Submitted by

VIPPARTI JEMIMAH 20221CCS0002

ARCHANA NAYAK 20221CCS0024

VAARUNI A R 20221CCS0042

Under the guidance of,

Dr. Geetha Arjunan

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING, CYBER SECURITY

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this report "AI POWERED STUDENT ASSISTANCE CHATBOT FOR DEPARTMENT OF TECHNICAL EDUCATION" is a bonafide work of "VIPPARTI JEMIMAH(20221CCS0002),ARCHANA NAYAK(20221CCS0024),VAARUNI A R(20221CCS0042)", who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE ENGINEERING, CYBER SECURITY during 2025-26.

Dr. Geetha A
Project Guide
PSCS
Presidency University

Dr. Sharmasth Vali Y
Program Project Coordinator
PSCS
Presidency University

Dr. Sampath A K
Dr. Geetha A
School Project Coordinators
PSCS
Presidency University

Dr. Anandaraj S P
Head of the Department
PSCS
Presidency University

Dr. Shakkeera L
Associate Dean
PSCS
Presidency University

Dr. Duraipandian N
Dean
PSCS & PSIS
Presidency University

Name and Signature of the Examiners

1)

2)

DECLARATION

We the students of final year B.Tech in COMPUTER SCIENCE ENGINEERING, CYBER SECURITY at Presidency University, Bengaluru, named VIPPARTI JEMIMAH,ARCHANA NAYAK,VAARUNI A R hereby declare that the project work titled **"AI POWERED STUDENT ASSISTANCE CHATBOT FOR DEPARTMENT OF TECHNICAL EDUCATION"** has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in COMPUTER SCIENCE ENGINEERING, CYBER SECURITY during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

VIPPARTI JEMIMAH	20221CCS0002
ARCHANA NAYAK	20221CCS0024
VAARUNI A R	20221CCS0042

PLACE: BENGALURU

DATE:

ACKNOWLEDGEMENT

For completing this project work, We/I have received the support and the guidance from many people whom I would like to mention with deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in completion of the project.

I would like to sincerely thank my internal guide **Dr. Sharmasth Vali Y, Associate Professor**, Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance and encouragement provided to us during the period of our project work.

I am also thankful to **Dr. Anandaraj S P, Professor, Head of the Department, Presidency School of Computer Science and Engineering** Presidency University, for his mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean PSCS & PSIS, **Dr. Shakkeera L**, Associate Dean, Presidency School of computer Science and Engineering and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work.

We are grateful to **Dr. Sampath A K, and Dr. Geetha A, PSCS** Project Coordinators**, Dr. Sharmasth Vali Y, Program Project Coordinator**, Presidency School of Computer Science and Engineering, or facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

VIPPARTI JEMIMAH
ARCHANA NAYAK
VAARUNI A R

Abstract

Education providers in the technical field usually receive large numbers of requests by students, parents, and personnel, particularly during important stages of the academic year, like admissions, exams, and placement. These investigations are usually made with the use of traditional methods (calls, email, in-person visits) which are all time-consuming, frequently redundant and cause an enormous amount of workload on the administrative staff. Parallel to this is the fact that students automatically seek the right information of materials that are syllabus related, quickly and anywhere. Meeting all the needs will still involve a supportive and trustworthy process that is capable of automating discrete interactions and improving the service quality.

In order to overcome this operational problem, an AI-chatbot will be applied to the project to support students, specifically in the setting within technical education. The system combines a React frontend to communicate with the user and FastAPI on the backend, that makes an inquiry through a secure process. The heart of the deployed solution will be Google Gemini (Large Language Model) as the chatbot and syllabus information of the university as the data source, and a list of frequently asked questions to deliver context-specific solutions, directly associated with the questions of students, parents or the staff. As a group implementing it, we talked about the different structures we would be required to implement, that would guarantee clarity in functionality of the system: User Interface, Chatbot Engine (AI/NLP), Knowledge Base, Backend/API, Admin Dashboard, and Deployment and Maintenance. By locating these silos to implement, the following modules have an additional opportunity to develop, and organic growth is guaranteed. The other environmental factors to further protect applications are API key management; mode to maintain dummy data and CORS middleware to ease integration.

The first step in the implementation was to make the backend create a functional /ask API endpoint that could accept student queries and process them to provide the option to integrate the Gemini API and provide information in the form of the JSON format. This endpoint has been integrated with the frontend and enables real-time communication between the chatbot and the frontend through a web interface. Preliminary testing on the systems with dummy and natural questions shows that the system could answer many questions and produce natural language responses, which agree with the provided context. The results show that the architecture and the integration approach are so far viable.

Table of Content

Table of Content

List of Figures

List of Tables

Abbreviation

Abbreviation	Full Form
AEHMS	Automated Equipment Health Monitoring System
AI	Artificial Intelligence
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
API	Application Programming Interface
HTTPS	HyperText Transfer Protocol Secure
REST API	Representational State Transfer API
CORS	Cross-Origin Resource Sharing
SQL	Structured Query Language
UI	User Interface
NLP	Natural Language Processing
ML	Machine Learning
FT	Fine-Tuning
SDLC	Software Development Life Cycle
SDG	Sustainable Development Goals
UG	Undergraduate
STEM	Science, Technology, Engineering, and Mathematics

LMS	Learning Management System
ERP	Enterprise Resource Planning
WCAG	Web Content Accessibility Guidelines
DB	Database
VS Code	Visual Studio Code
DBMS (inside code)	Database Management System (contextual)
IP	Internet Protocol
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment *(implied from tools)*

Chapter 1

Introduction

The modern university is a complex ecosystem that comes with the concomitant pressures of the astronomic increases in student enrolment, academic transformation, and the burdens of immediate and personal support. The conventional administrative and informational distribution mechanisms are on a life threatening foundation in the realms of a competitive higher education landscape (within large urban centres in India) in India. The gap in the system in this business is addressed in this project because it proposes a state-of-the-art solution: a progressive conversational agent based on AI. This system is informed by the multi-stream academic context typical in the institutions both of Bengaluru and based on a Large Language Model (LLM) with Retrieval-Augmented Generation (RAG) to assist the institutions critically in maximizing their efficiency and enhancing the student experience.

1.1 Background: Scalable AI is needed in Higher Education in Bengaluru.

The digital transformation is one that is long overdue in the world especially in educational institutions globally that have a large population and varied needs. Previously, universities were working under the mode of labor intensive, and concurrent, communication techniques of visiting, long-term emailing and telephone, to support students in asking questions particularly during the heavy administrative periods such as admission, collecting fees and registration of examinations. Though these methods are quick solutions, the main problem with it is that they lack scalability, are repetitive and are bound to overburden the administrative force, which is most likely to create bottlenecks in operations and disappoint the students as well.

The environment in Bengaluru increases this administrative pressure by the fact that the density of education institutions is spectacularly high. The details of the syllabus, fee structure and placement opportunities and the details of scholarship among others are the high quality information that students and stakeholders now want to have at any time, anywhere and most importantly in real time regardless of the time zone. The traditional information technology infrastructure that is in most instances anchored on a static web page

or a mere rule based communication infrastructure is essentially inappropriate to handle the volume and the complexity of natural language queries that are generated by a high population of students in a search of diverse domain-specific information.

The intensity of the higher education market of Bengaluru leads to such an environment where the competition among the institutions with regards to satisfying their students is stiff. The unproductive, slow, or unpredictable administrative information system jeopardizes the image of the institution and compromises the enrollment targets. Thus, it is not an automation issue anymore; it is an intelligent conversational digital assistance paradigm shift. This solution must have more advanced computational linguistics so that it can extend past the question answer queries, to the active, interactive and personalized answers.

The rationale of this project is that it is an extremely grave investment in institutional resilience and competitive advantage. It targets the design of a smart student support chat robot that will specifically target the bridging of the communication gap between the high volume administrative information and the multi-stream population of students in the multi-stream university environment at Bangalore. The simplest technology is to fine-tune a Large Language Model (LLM) with the Retrieval- Augmented Generation (RAG) template, according to which all responses can be factually backed by verifiable institutional data. This remedy is set to be the technological leap needed to ensure immediate and accurate as well as scaled support of the learners under the high density and the high throughput learning environment.

1.2 Statistics: Measuring the necessity within the Bangalore Regional background.

The need to adopt high-performance digital solutions in Bengaluru has been empirically justified by the regional statistics that describe the colossal nature and scale of operations of the local higher education sector. The Bengaluru Urban district is also unique in the country since it has 1,106 colleges, which makes it the most densely populated area in the whole country.

This is also part of the reason that Karnataka has the highest number of colleges per capita 66

colleges per lakh eligible population (age 18-23), far exceeding the national average of 30. This saturation depicts a very competitive environment, which is filled with educational providers and where service quality is one of the conditions that determine student recruitment and retention.

The size of student interaction proves the extraneous throughput demands of any digital assistance platform. Karnataka has a population of about 24.36 lakh educated students in different levels and streams. Such sheer numbers of student enrollment guarantee that even such administrative inefficiencies, when added to the many thousands of questions posed per day, quickly overstretch administrative resources. As an example, when a manual process requires ten minutes of human

resources per student request, then servicing five hundred students at once each day would require an unequal proportion of administrative resources. The implementation of an asynchronous and high-performance digital solution as a project component, like the one suggested based on FastAPI, is warranted as more than a convenience, therefore, a necessity to handle this exponentially growing load.

Additionally, the diversity of the student population in terms of the academic background highlights the necessity of the information system that is versatile and factually sound. Although the scope of the project lies in the Department of Technical Education, the solution should benefit the whole university system, which has undergraduate enrolment highest in Arts (34.2%), followed by Science (14.8%), Commerce (13.3%), and Engineering and Technology (11.8%). Such a multi-stream character requires a solution that could receive queries with extremely domain-specific information, be it engineering lab procedures or changes in commerce regulation, at a steady level of accuracy.

This necessitates stream-agnostic precision that supports the need of the Retrieval-Augmented Generation (RAG) system. A RAG pipeline means that the chatbot is able to consume and process institutional data of all types (e.g., Arts syllabi, Engineering manuals, Commerce policies) and deliver precise and stream-specific information, which lessens the pain points inherents in the operations, namely, long queues, excessive paperwork, and human error. Table 1.1. The critical parameters demonstrate the size and the diversity of the higher education sector of the focal region of the project.

Parameter	Value/Data Point	Source	Chatbots Require Rationalization.
Bengaluru Colleges	1,106 (Highest in India)	[3]	Indicates large quantities and enormous management amounts and concentrates differentiated local inquiries.
Karnataka College Density (No of colleges per Lakh Eligible population)	66 (Highest in India)	[4]	Indicates the market saturation and high pressure in the competition that demands lean student services.
Karnataka Total enrolment student (Karnataka)	~24.36 Lakh	[3]	Authenticates the high throughput requirements of any digital assistance platform.
Preeminent UG Streams (Majority of Registrants)	Arts (34.2%) Science (14.8%) Commerce (13.3%) Engineering (11.8%)	[7]	Requires multi-stream support, multi-stream versatile knowledge base and RAG

			architecture.

Table 1.1: Karnataka/Bengaluru (2021- 22) Enrollment context and Institutional Density.

1.3 Past Existing Technologies: Weaknesses of the Traditional Systems and the Rise of the RAG-Augmented LLMs.

The necessity to create the AI-powered student assistance chatbot is determined by the drawbacks of the already established technologies which can be divided into three generations as manual processes, rule-based chatbots and early machine learning (ML) chatbots.

Firstly, there was Manual and Legacy IT Systems which were characterized by physical interfaces, much paper trail and disjointed workflow. They are non-hurried and manually driven systems that have little or no transparency that results in a big huddle to students in need of information about issues that are time sensitive. This outdated paradigm is the obvious reason behind the identified areas of friction in the administration process such as “Long Queues and Delays in Counselling” and "Poor Communication with Students" that are the direct results of the old paradigms.

Second, the first step towards automation required First-Generation Rule-Based is Chatbots. These were systems that were built on simple decision trees or key words. They were quick, but they were rigid enough, and not able to get out the language of a fine nature, to alternate the scenes of a multi-turn conversation, or to respond substantially to any question which was outside their narrow area of interest. Such systems could easily break down when they were put to the test of the different and diverse questions that are typical of a multi-stream university.

Third, Simple intent identification was done and more advanced natural language processing (NLP) was done in Second-Generation ML/NLP Chatbots. However, research indicates that these systems

remained at immature capabilities and failed to maintain the context during a prolonged conversation and the capability of providing the most advanced form of fluency which produces adequate generation responses.

Large Language Models (LLMs) ushered in talkativeness, which never existed. However, there is also another crucial challenge to the institutional deployment by base LLM: the problem of hallucination. The answers of LLM are assertive and natural yet incorrect in the facts because their parametric knowledge is often outdated or not valid, based on some of the institutional policies, since they were initially trained on a fixed set of data, which is often out of date. Such a weakness cannot be tolerated in an academic environment where facts cannot be distorted.

The second customization process is fine-tuning (FT) that trains the model using common patterns of new data but knowledge snapshots are fixed. Since the policies and the curriculum content in the university is dynamic, retraining that is costly and requires frequent revision is necessary in FT which is not economically and operationally viable in a large and dynamic institution.

This critical analysis underlines that the solution is not meant to escape the limitations of the fixed knowledge systems but must be fluent like the generative AI. These intellectual requirements lead to the introduction of the Retrieval-Augmented Generation (RAG) architecture1 RAG enables the LLM (in this paper, we will use Google Gemini) to access a particular, current piece of information regarding a particular body of knowledge (the university documents in this project) and build the generated responses out of that information. RAG has been capable of delimiting hallucination barriers, and adapt to changing facts without retraining its models at high cost.

1.4 Proposed Approach

1.4.1 Aim of Project

This project will be primarily focused on designing, developing, and rigorously testing a high-performance, Retrieval-Augmented Generation (RAG) empowered AI chatbot with the use of the Google Gemini Large Language Model (LLM). This system will be incorporated through a full-fledged architecture that is based on FastAPI (backend) and React (frontend). This solution is highly specific to offer 24/7 student support, that is accurate and safe in all significant academic and administrative streams in a high-density academic institution environment in Bangalore.

1.4.2 Motivation

The operational efficiency and service equity are inherent to the motivation in that the current system overworks the staff, as they have to spend considerable time answering frequent, easy to answer questions. The project is directly related to reducing the workload of the administrative staff, leaving them to do complex, human-oriented pedagogical and managerial tasks. This boost of professional roles helps the principle of **Decent Work and Economic Growth (SDG 8).**

At the same time, the project is aimed at improving the student performance by ensuring the access to the information instantly, correctly and in the context. The availability of 24/7 communication via reliable support helps in breaking time zones and access restrictions which enhance a fair learning process. This enhanced access is a direct contribution towards the objective of **Quality Education (SDG 4)**. Finally, this project is an essential case study when it comes to deploying scalable, high-quality conversational AI agents within a multi-stream institution, which promotes digital modernization.

1.4.3 Recommendations to the Approach (Technical Framework).

The project software architecture is an agile software development life cycle (SDLC) cloud-native and modular architecture that is needed to create and continuously improve

intelligent systems in a way that is anchored on the constant feedback of data.
There are three main layers:

Presentation Layer (Frontend): The layer (React-based) is an interface with a dynamic, responsive

and mobile interface to make the chat work as efficiently as possible.

Application Logic Layer (Backend): Timely Python(Fast API) written. FastAPI is selected due to high-performance and asynchronous features that are predetermined by the fact that a large university setting will most likely be prone to create high user requests simultaneously. The backend implements API key authentication, user input authentication, user secured routing, and error handling and provides a high level of operational security.

1.4.4 Applications of the Project

Administrative Query Resolution: An extremely expedited response to administrative processes that are of high-stakes to the mission (e.g. admission process, scholarship and financial aid application due-dates and document delivery requests).

Curriculum and Academic Advising: This provides intensive contextual content of course requirements, technical field-specific jargon and course outcomes and has a significant role in mediating information to a student in any academic stream (Technical, Arts, Commerce).
Student Support 24/7: It provides the students practical service 24 hours a day even when the normal days of work of the universities are over and there is no stress and anxiety among the students and the students are assured of availing their advice always.

Operational Data Analytics: The system will process and interpolate the interaction records to give the university management the empirical data of most frequently asked questions and overall knowledge gaps. This will enable one to review the policies and content proactively and will change the administrative role to a proactive one.

1.4.5 Approach Limitations Proposed.

Although the suggested RAG-enhanced LLM model places the state-of-the-art workaround, a number of limitations have to be considered and addressed:

Knowledge Base Integrity: The fact is the completeness and timely maintenance of the institutional data that is used to create the RAG pipeline is only at the standard, which is the factuality of the system. The documents that are either incorrect or outdated sources will be read and used to provide incorrect answers regardless of the intelligence or not of the LLM.
Legacy Systems: One of the primary initial engineering problems is to interface the RAG pipeline ingestion and data management system with the existing university legacy databases, such as Learning Management Systems (LMS) or Enterprise resource planning (ERP) system.

1.5 Objectives

The project contains distinct, quantifiable, attainable, realistic and time-related objectives (SMART), which will ensure that all development, deployment, security and performance issues are thoroughly validated.

Design, Implement and Test a Retrieval-Augmented Generation (RAG) pipeline to have the Google Gemini LLM be able to answer administrative and academic queries with at least 90% factual accuracy on the basis of the given institutional knowledge base. (Focus: Analysis, Behavior)
Create an infrastructural system architecture (FastAPI/React) that is secure, modular, and capable of sustaining at least 500 concurrent user sessions, and average response time of less than 2 seconds on conversational requests, can be scaled in the high-density Bangalore environment. The system is collaborative hence it could be customized to suit customer needs. The system is also collective and therefore the service can be tailored to the customers.

Make sure that sensitive user interaction logs and core knowledge base are not compromised with the help of effective security controls, such as Role-based access control, API key management, Cross-Origin Resource Sharing (CORS) middleware, and encrypted data

transmission. (Focus: Security)

Develop and introduce an administrative analysis dashboard to measure and report the success of the system, which is a reduction in the time of system use by the administration in repetitive student inquiries of the system to an approximation of 80 percent as compared to the baseline level of manual system processing. (Focus: Analysis, Behavior)

1.6 SDGs: Sustainable Development Goals Connection.

The project has a list of the most important United Nations Sustainable Development Goals (SDGs), which can be used to consider the social and long-term implications of the project. The solution developed is not merely a technical exercise of solving the needs of the world on the aspects of sustainable development.

SDG 4: Quality Education. This will probably provide, inclusive and equitable quality education and a lifetime learning opportunity to any person. This is also achieved in the project with emphasis on just access. The foregoing is done by achieving the same degree of institutional information among all learners, irrespective of their background, physical location and time constraint at any given time period with highly precise 24/7 AI chatbot (Objective O2) and this directly translates to an inclusive learning environment.

9: Industry, Innovation and Infrastructure. The goal of this is on resilient infrastructure development, inclusive, sustainable industrialization and innovation. The design solutions of this project, especially the application of fast, asynchronous computing systems, such as FastAPI, and the utilization of the sophisticated RAG pipeline (Objective O1) can be considered as a good technological innovation concerning the technological infrastructure of an institution. The undertaking will make the institution leave the old data models and move on to a highly robust and modern digital platform and in the process, it will take the entire education technology sector to a more updated level.

Goal 8 Decent Work Economic Growth. This goal enhances the long-term and inclusive as well as sustainable economic growth, full and productive employment and decent work to all. The system will assist to utilize the resources better, as the time that the administrative team will spend on the daily routine will be minimized (Objective O4), the remaining time of the

staff members will be implemented in more productive professional activities, e.g. it is possible to mentor students one-on-one, create more sophisticated learning plans, and address the more complex issues. This augmentation on efficacy of administration and professional empowerment is included in the action of creating more productive and improved working conditions.

The technical design and the measurability objectives of this project is strategically designed as such that the solution is not just working but also has some sense to the overall mandate of improving the society and technology. The finer association is depicted below.

ID	Objective (1.5)	Focus Area (Per Template)	Alignment Rationale (1.6)	Aligned SDG(s)
O1	Design, run and test a RAG pipeline to get factual accuracy of the LLM in question at least 90 percent of the way.	Analysis, Behavior	Focuses on innovation and building of strong and quality technological infrastructure.	9: Industry, Innovation and Infrastructure.
O2	Focus on the creation of a safe modular system architecture (FastAPI/React), which has the potential to service 500 user sessions simultaneously,	System Management, Behavior, Deployment.	Ensures fair, competent and 24/7 accessibility to information to all the students in order to promote learning outcomes.	SDG 4: Quality Education

	and its average latency should not surpass 2 seconds.			
O3	On the basic level, API keys possess a high security level i.e. CORS middleware, and encrypted data transmission to maintain integrity of sensitive user interaction records and knowledge base.	Security	Ensures privacy and data integrity that creates confidence in online services of institutions.	SDG 4: Quality Education
O4	Reduce the administrative man-hours of unwarranted student inquiries by 80 per cent as of the processing time of the same when	Behavior, Analysis	Efficiency, work place resource management and enhanced value and professionalism work are some of the practices that are	Goal 8: Decent work and Economic growth.

	manually processed on hand.		practiced among the workforce.	

Table 1.2: Project Objectives and alignment with UN Sustainable Development goals (SDGs).

1.7 Overview of Project Report

The project report is neatly structured with nine chapters and has effectively used the whole development, implementation and aggressive testing of the AI-Powered Student Assistance Chatbot. Introduction itself takes much space in Chapter 1 that presents the background of the project, provides a quantification of the need in terms of regional statistics of the high-density educational sector of Bengaluru, critically reviews the existing technologies already at work, and proposes in detail RAG-augmented LLCM approach and, finally, the quantifiable objectives are defined in line with the UN Sustainable Development Goals. The second chapter develops the literature review discussing the literature about conversational AI, LLMs, and RAG architectures and their application in various educational institutions critically. Chapter 3 explains the Agile system taken to develop it and the step-by-step systematic process of the development of the intelligent system. In Chapter 4, the author is preoccupied with the basic ingredients of project management, such as the project schedule, the project risk analysis, as per the PESTLE framework, and the itemized project budgeting. Chapter 5 describes the analysis and design stage that describes the system requirements, functional blocks, system architectures (functional, communication, operational) and rational thinking that is systematically planned to choose the components. Chapter 6 represents the hardware, software development and simulation environment, and describes the combination of React frontend, FastAPI backend and The Gemini LLM client. Chapter 7 will address one topic, which is the evaluation and results where the rigorous test plan, tabulating the empirical performances (e.g., latency and accuracy), and offering technical feedback on how the system is being tested are addressed. Lastly, Chapter 8 introduces the mandatory reflection of social, legal, ethical, and sustainability implication of the AI system and Chapter 9 is the conclusion of the report as a brief summary of the main findings with an appendix of the way forward and future progress of the researc

Chapter 2

Literature

IEEE CONIT, 2021 - Conversational AI.

In this paper, a summary of the existing research on the field of conversational AI technologies, including chatbot architecture, conversational agent design, and the natural language understanding and recognition techniques, will be provided. Several models are presented including rule-based models, retrieval-based models and neural generative models. The contemporary issues that should be tackled are mentioned in the paper. These are significant problems in overcoming the context and response generation and multi-turn conversation problems. According to the authors, it is used in various industries that are interested in how conversational AI might be adopted to supplement user interaction and automate services to the customers. The opportunities of the future include personalization, emotional intelligence, and multi-modes improvement. Lastly, more natural and useful interactions are offered to users by conversational response systems[1].

Winkler, and Sollner, 2018 - Unleashing the Potential of Chatbots in Education, IEEE EDUCON.

Winkler and Sollner, 2018 - Unlocking the Potential of Chatbots to Education, the article featured in the IEEE EDUCON journal lists the comparatively new use of chatbots to enhance online learning experiences. They claim that chatbots can offer conversational modalities to interact with the students, give feedback, and offer administrative services. They discussed the challenges of chatbots, such as initial chatbot models, chatbots which were typically rule oriented and not flexible and intelligent. They asserted that it is possible to have further-developed AI-chatbots to create certain interactive learning materials. The authors discussed the way chatbots may be helpful to minimize the load of the educators as they could eradicate the low-level communication when the teachers will be free to spend their time on more significant pedagogical tasks. They also ranked such chatbots, which can be optimized according to the needs of individual learners, identify emotional states, and offer suggestions as one of the considerable qualities, to stimulate the desire to the learners. Further, they suggested that future chatbots design should not be simply useful but, should

additionally, serve natural language processing to more than one user, as well as make conversations seem natural and conversation-like to the user, to attain, not only user satisfaction but also increased use in education and systems[2].

Husain et al., 2024 concerning the development of an Academic Services Chatbot, which is developed based on the RAG.

The project is founded on the recent methods of natural language processing with the usage of the rich corpus of the university documents to deliver the chatbot with the correct and contextually appropriate answers. It is configured with high data encryption and access control to ensure sensitive academic data in cloud storage and retrieval. The chatbot design has been created in a manner that fosters user confidence and trustworthiness, although certain AI systems have general challenges, such as fabrication of data (hallucination). One can observe that making answers to the frequently posed academic questions automated is what would potentially result in the reduction of workload on both administrative jobs and improvement of services to students. It shall also provide the basis to roll out RAG based chatbots to higher education to possess smarter, secure and scalable scholarly digital tools. The possible future consideration may consist of the enhancement of functionality in order to facilitate the features of customized learning proposals, emotional guidance, and multi-modal input processing so that the users may work in a more interactive way[3].

Al-Ghuribi et al., 2020 Chatbot in Education: a Systematic Review IJACSA.

Al-Ghuribi et al., 2020 - Chatbot in Education: A Systematic Review, IJACSA is the publication that provides a critical review of chatbot as a tool in higher education. Various chatbot systems that focus on maximizing student engagement, aid with learning and administration/emails solving are evaluated in this paper. The review discusses the positive effects of the chatbots, such as the heightened accessibility of the information, the immediate feedback to the students, and the possibility of establishing a more tailored approach to the learning process. The review then moves to discuss some of the challenges including limited abilities of the chatbot, natural language processing, integration issues, and it should be continually developed in various learning situations. In conclusion the authors also find that though there is potential in using chatbots as a way of increasing engagement amongst students and reducing the workloads of educators, there is an immediate need to improve

chatbots to fit and work in various educational settings efficiently and effectively [4].

Ramesh and Gupta, 2020 -AI-based education chatbots: Survey, IJET.

Ramesh, and Gupta, 2020 - AI driven chatbots in education: A survey, IJET discusses the creation and use of chatbots that are AI powered in learning institutions. The survey has examined several chatbot technologies that advance teaching and learning through offering individualized capabilities to the learners, reducing the burden on the instructor, and enhancing accessibility of learners. This survey study will discuss AI approaches and techniques such as natural language processing to use chatbots, machine learning to enhance chatbots, and AI dialog management techniques to enhance and upgrade chatbots to use in learning institutions. The paper outlines issues that AI chatbots undergo, which are limited domain-related knowledge, a persistent context to the discussion practice, and enhanced reliability. The general survey indicates the criticality and high potential of AI chatbots to design and develop learning practices; they, further, suggest practitioner-specific AI chatbots, deployed in diverse education (k-12 and HIE) with divergent learner requirements and multiple points of domain knowledge.[5].

Kumar & Sharma, 2022 - Smart Chatbot for Student Support, vol. 11, no. 3, pp. 20318-20333, in IEEE Access

Kumar, A. & Sharma, S. (2022) -Student Support Smart Chatbot, in the journal of the Institute of Electrical and Electronics Engineers (IEEE) Access. The authors developed a prototype chat bot to handle the academic and administrative queries by the students using simple machine learning methods to answer the concerns of students. The chat bot has the ability to provide efficient and timely support to the learners in a fast manner. While the chatbot works it does not have the accuracy and contextual understanding of the modern-day large language model (LLM)-based systems. The article talks about benefits associated with automating student help in an effort to reduce workload and enhance response time at the same time acknowledging that more complicated and/or ambiguous student questions are limited to automation. The paper addressed the improvements that can be made to the student support chatbot in the future such as adopting a more advanced artificial intelligence (AI) algorithm and the knowledge base of the chatbot to enable it to be relevant to a wider range of students.

X. Yin et al, 2024 Formative Feedback Provided to Students Self-Assessments Using a Chatbot in International Conference, 2024 proceedings.

Researchers investigate a chatbot a mediator potential by providing feedback to students in real-time which will be based on the self-assessed answers in the article X. Yin et al., 2024 -Using a Chatbot to Provide Formative Feedback to Students Self-Assessments. The results depicted that chatbot feedback has a positive effect on the learning performance of the students by making them reflect and finding out where knowledge gaps might be present. In the area of formative feedback, the chatbot can handle cognitive load and motivation of the students by offering feedback that is specific to the individual student and through the application of timely and supportive interventions. These results indicate that interaction is enhanced and students could more actively regulate their learning processes under conditions of having chatbot-based formative feedback. Finally, the research indicates the possibility of AI chatbots to be used as a portal to individual learning and variables to lead to a better academic performance, especially in online learning environments.[7].

Y. Huang, et al., 2025 - Self-Regulated Learning and Social Presence with Instructionally Aligned AI Chatbots, in: the Transactions on Learning Technologies, vol. 96, no. 3, pp. 220-235

In the article Self-Regulated Learning and Social Presence with Instructionally Aligned AI Chatbots, the authors Y. Huang et al. discuss in 2025, the authors determine the importance of AI chatbots that are instructionally aligned with goals - of which they are goals set by the instructor - in stimulating self-regulated learning (SRL) in students, in addition to an increased sense of social presence in online learning environments. Their results suggest that AI chatbots can be quite useful virtual tutors in the classroom and possibly even more useful than the students themselves when encouraging students to be motivated, goal-oriented, and reflective by setting their goals. Moreover, the availability of AI chatbots intensified a feeling of social presence, minimized feelings of loneliness, and enhanced learning in the online learning setting. Finally, they touched on the general worthiness of the work, the pedagogical consequences of harmonizing such instructional chatbot conversations with proper ways of improving engagement, helping the learners with self-directed actions, and the learning outcomes. Finally, they conjecture that AI chatbots have the potential to facilitate a personal,

less intrusive, facilitative, and socially interactive learning experience[8].

M. Sarker et al. Anglo-Bangla Language Based AI ChatBot for University Admission Counseling, Proc. IEEE CCCAI

M. Sarker et al., 2023 - Anglo-Bangla Language-Based AI Chatbot in University Admission Counseling, Proc. IEEE CCCAI is the paper which introduces a bilingual chatbot which has been created to help the potential students with the university admission counseling. This AI system can work both in English and Bengali languages in order to make it accessible to a larger population consisting of university students. The chatbot will be able to answer questions regarding the admission processes, or admissions and do it in a time and personalized way that allows to perceive the process of the application with less difficulty for a student. Through natural language processing, the chatbot will be able to interpret and respond to the user properly, making it more responsive and by the way, the administrative task of the admission officers will be reduced. The planned project is a great example of the usefulness of multilingual AI chatbots in accessible and effective University admission support services[9].

Gao et al., 2019 - Neural methods of conversational AI, Foundations and Trends in IR.

Gao et al. (2019) - Neural Approaches to Conversational AI, Foundations and Trends in IR the paper describes a comprehensive review of the methods based on using neural networks to produce conversational agents. The paper discusses some of the basic methods such as sequence to sequence models, attention techniques and reinforcement learning applied in the dialogue systems. The review is thorough as it discusses the old problems of the dialogue context and coherence, provided methods (such as multi-turn dialogue), and the latest developments in neural conversational models that encourage more naturalness and flexibility in human-computer communication. It emphasizes the application of external sources of knowledge to enhance relevancy and informativeness to the user. The paper discusses the measurements/evaluation and trade off in dialogue generation in creativity and accuracy. The data presented is relevant to conversational AIs in general, but the fact is that the information is more applicable to the development of solid chatbots that can be utilized in education and other fields and allow the further evolution of interactive intelligent systems in the future [10].

Reference	Technique used				Maximum Accuracy obtained (%)	Limitation
	Preprocessing	Feature Selection	Segmentation	Classification		
IEEE CONIT (2021) – Conversational AI: Chatbots (Conference volume)	Varied preprocess techniques across included papers (tokenization, cleanup)	Feature methods vary (intent features, embeddings)	Different segmentation strategies in included works	Collection of rule-based, retrieval, and neural chatbots	Varies by paper	Heterogeneous quality; conference proceedings vary in depth
Winkler & Söllner (2018) – Unleashing Chatbots in Education (IEEE EDUCON)	Discusses preprocessing best practices for educational corpora	Highlights features for user engagement and trust	Proposes segmentation by learning objectives and interactions	Framework-level proposals; mixes rule and ML approaches	N/A	Conceptual; limited empirical validation
Husain et al. (2024) – Academic Services Chatbot Based on RAG	Ingest policies, circulars, FAQs; OCR + text cleaning; chunking	Select domain-specific entities, FAQ intents, citation anchors	Segment documents into chunks by topic and date	Retriever (vector DB) + LLM generator (RAG) for grounded Q&A	Not reported	Needs continuous KB updates; handling conflicting docs is non-trivial
Al-Ghuribi et al. (2020) – Chatbot in Education: Systematic Review (IJACSA)	Review compiles preprocessing practices from studies (surveys)	Summarizes common feature types (intent, lexical, pedagogical)	Reports segmentation strategies used in literature (task/session)	Survey of rule-based and ML/NLP-based systems	N/A (survey)	Meta-level; lacks primary experimental results
Ramesh & Gupta (2020) – AI-driven chatbots for education: Survey (IJET)	Surveys preprocessing workflows used across studies	Summarizes common pedagogical and dialog features	Describes segmentation trends in literature	Survey covering retrieval, seq2seq, and hybrid models	N/A	High-level survey; limited experimental detail

Reference	Technique used				Maximum Accuracy obtained (%)	Limitation
	Preprocessing	Feature Selection	Segmentation	Classification		
Kumar & Sharma (2022) – Smart Chatbot for Student Support (IEEE Access)	Collected institution-specific FAQs and student logs; cleaned & normalized	Feature extraction: intent labels, contextual slots, keywords	Segmented queries by department and topic	Hybrid retrieval + intent classifier (ML-based)	Not reported	Prototype scope; limited domain coverage
X. Yin et al. (2024) – Chatbot for Formative Feedback (IEEE TLT)	Collected student self-assessments, cleaned responses, aligned prompts	Features: response correctness, confidence, cognitive-load proxies	Segmented by task type and assessment item	LLM/chatbot providing formative feedback; evaluation via learning gains	Not reported	Requires LMS integration and longitudinal validation
Y. Huang et al. (2025) – SRL & Social Presence with AI Chatbots (IEEE TLT)	Preprocessed interaction logs and course materials; normalized indicators of SRL	Features: SRL indicators, social presence measures, engagement stats	Segmented learners by SRL profiles and engagement levels	Instructionally aligned chatbot (tuned prompts & scaffolding policies)	Not reported	Needs diverse learner testing and personalization improvements
M. Sarker et al. (2023) – Anglo–Bangla Admission Chatbot (IEEE CCCAI)	Collected bilingual FAQs, performed language normalization and transliteration	Features: bilingual intent lexicons, language detection features	Segmented queries by language and admission topic	Hybrid intent-classifier + retrieval; bilingual response generation	Not reported	Limited to admission domain; regional language coverage may be partial
Gao et al. (2019) – Neural approaches to conversational AI (Survey)	Discusses preprocessing approaches (tokenization, normalization) broadly	Surveys feature choices across models (embeddings, context features)	Covers segmentation methods (turn-level, session-level) in literature	Surveys seq2seq, retrieval, and hybrid neural architectures	N/A (survey)	Survey—no empirical accuracy; breadth over depth

Table 2.1 : Literature Survey

Chapter 3

Methodology

3.1 Introduction

The methodology chapter explains how the systematic and stepwise methodology was followed in the establishment of the AI enabled student assistance chatbot. Due to the discursive and changing of educational settings where the needs of the students, technological potential, and the institutional needs often change, it was significant to use an adaptive and repetitive developmental model. In this regard, Agile Software Development Life Cycle (SDLC) has been chosen as the methodological guide in the project.

Agile can also be applied with smart systems such as chatbots because it focuses on modular design, continuous integration and the rapid feedback loop. This flat structure enables developers to create, test and refine system components through iterative means and also be responsive to user feedback and requirement changes. Unlike such strict models as the Waterfall or V-Model, Agile allows adaptable planning and cooperation between stakeholders in the development process.

This chapter perceives each stage of the chatbot production in terms of Agile and includes requirement analysis and ends with testing, verification, and validation via system design, functional and unit design. This will be explained in detail to indicate how it affects the overall functionality, reliability and user experience of the chatbot.

Besides, the chapter explains the kinds of tools and technologies utilized in every stage of development. These technology options are open-source technology, FastAPI in the backend development phase, React, the frontend interface, and the Google Gemini API that will be used to assist the natural language processing modules. Examples of tests written with unit testing and integration testing are used to denote the testing plans that were done in the evaluation of the system concerning performance, accuracy, and usability.

Agile enabled the project to stay responsive and continually enhance the level of quality as well as deliver a product that addressed the real requirements of students and teachers within a technical education setting.

3.2 Chosen Methodology: Agile SDLC.

Agile Software Development Life Cycle (SDLC) is another prevalent form of model that appreciates the value of flexibility, iterative enhancements and tight cooperation among the developers. Conventional models such as the Waterfall model or V-Model are strict and linear steps that should be followed in order, whereas Agile works well based on the adaptive planning approach. Consequently, Agile is highly adaptable to the projects within the area of artificial intelligence (AI) wherein the requirements are constantly changing depending on the user interaction and the data accessibility.

Agile is founded on short-development cycles known as sprints, which produce a functional system increment. These short cycles enable teams to be open to change, and explain features using real usage.

One of the main advantages of Agile is the opportunity to enhance constantly. As new information is given, particularly in AI systems, which are time, dynamic datasets and machine learning feedback loops, the development team can adjust the algorithms, retrain the models and improve performance without affecting the overall architecture. Intelligent systems such as chatbots need to evolve continuously to meet the needs of various questions posed by the users, new scholarly materials and change of course by the institution.

In this project, Agile enables you to modularly develop the elements of the chatbot, such as the backend API, frontend interface, and integrating with the Google Gemini API, it enables you to rapidly prototype, test, and deploy in installments to ensure that each version of the chatbot, be it internal testing, user testing, and full deployment, is functioning, useful, and worth being deployed to your target users. Therefore, you can capitalize on being agile and remain responsive and scalable to work in a volatile and diverse environment of technical education.

Key Features of Agile:

Agile methodology is also built on the major principles which promote flexibility, efficiency and user-centric development. Such aspects particularly come in handy with AI-focused systems like chatbots, which require an ability to evolve with the evolving datasets and user experiences.

Iterative Development Cycles (Sprints): Agile organizes work around short time-boxed development cycles and this allows a continuous and rapid delivery of functional units of the product. The continuous improvement loop can be conducted every sprint to make continuous improvements and fix bugs early on as opposed to late.

Continuous Integration and Testing: The developers can use Agile methodology to integrate code into and test it in the system often to ensure that the product can stay stable. It is essential to an AI-based system because an update to either model or API can lead to an unintelligible or unforeseeable behavior.

User-Centered Design: Agile will require developers to communicate with users on a regular basis during the review of feedback and usability testing. This comes in handy especially to ensure that the chatbot is as close to the real world needs of the students and educators.

Modular Architecture: Agile encourages the practice of developing loosely-coupled modules in order to scale them up with ease and make them easy to maintain. As an example, the query handler and LLM client, which will be developed as part of the database and application, will be modular-based in this project.

3.3 Roadmapping Agile Phases to project phases.

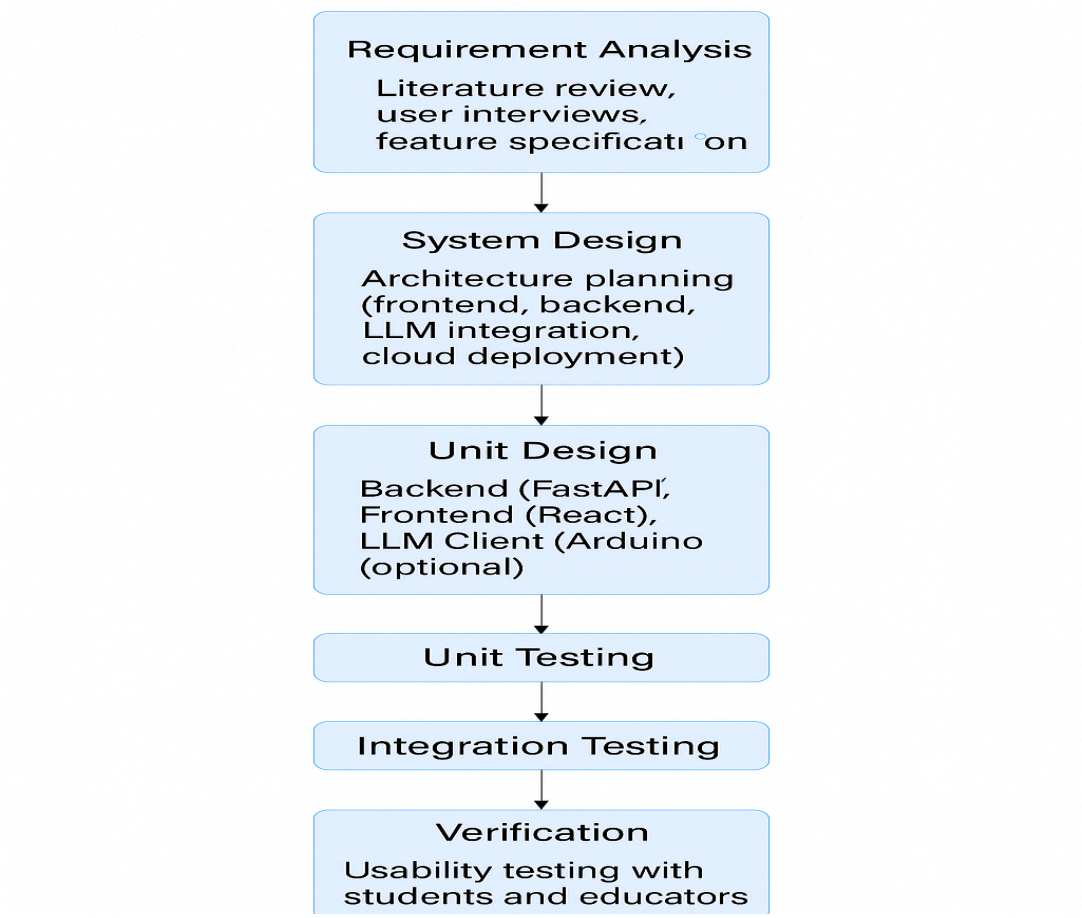


Fig. 3.1 Mapping Agile Stages to Project Phases

3.4 Requirement Analysis

The requirement analysis is another critical step in the development cycle of the Agile approach that gives the guideline to make sound decisions about the design and the implementation process. In this project, the detailed consideration of existing chatbots, natural language processing (NLP) models, and multimodal tools was carried out and supported with the help of interviews with stakeholders and user-featuring prioritization activity. All this work was aimed at the fact that the chatbot developed becomes finally useful to the needs of students (and teachers) in the technical education environment.

3.4.1 Literature Review

The literature review was focused on the evaluation of existing trends, technologies, and issues of various AI-based learning systems. Chatbots in academic environments have gained widespread use, especially in communication and administrative automation, as well as to provide a personalized learning experience to students. The studies have demonstrated that engagement and retention rates of students are enhanced, and timeliness, contextual learning cues, and pathway flexibility are allocated when using a chatbot enabled by NLP.
Among the lessons learned is the fact that chatbots could be programmed to support many languages. A good number of student groups can be highly heterogeneous, and educators are not always trained and do not have time to support all the languages. The lack of language that deals with their requirements presents many of the students with a barrier to participation and resources in multilingual classrooms and necessitates a change in some of their learning activity. The integration of AI systems that contain translation models or supported language can facilitate equity and inclusiveness, as they can be applied to active and relevant processes of other language support in a classroom.

Another important message that was conveyed was the necessity to incorporate the Learning Management Systems (LMS) of Moodle, Canvas and Google Classroom into a Chatbot. The inclusion of Chatbots in an LMS enables them to give real-time feedback, deliver automated reminders and access pertinent course content. The implementation of a Chatbot in an LMS is also prone to simplifying the processes of academic workflow along with providing greater access to various data as it will centralize the information.

Most of the currently existing systems, however, are not domain specific, especially in technical education. A generic robot does not comprehend expert queries or questions. Sam noted that there are generic queries within the minds of the technical education students related to programming, integration or compliance of hardware. This lapse in a solution also points out the necessity to come up with an intervention that could help resolve the issues faced by STEM (Science Technology, Engineering and Math) students.

3.4.3 Feature Specification

Figure 1 represents a set of central features that were determined and found priority to be implemented into the chatbot system after reviewing the literature and stakeholder feedback:
Text-Query Processing: It must be capable of identifying natural language questions concerning coursework, assignment due dates and the technical field in addition to expounding its queries using domain-specific vocabularies. Finally, the chatbot will be capable of giving the correct and relevant responses in a natural language.
Loose Coupling of Scalability: The system shall be developed, in the same fashion as Lego building blocks, where the various modules can easily direct connection without disturbing or impairing the other modules. The frontend, the backend, the LLM client, and optional hardware modules will be separately performed, which will also help to serve the purpose to upgrade any of the modules without sacrificing or hindering the main functionality of the other components of the system.
These characteristics together and individually contribute towards filling the gaps that have been spotted and preconditioning a chat-bot that is perfect among interest in technical education in a scalable approach, intelligent, as well as user-friendly manner.

3.5 Functional Design

The modules have certain functions each:

Query Handler

Receives the user input

Checks and removes the queries of the user.

Routes to the LLM client

LLM Client

Communicates with Gemini API.

Implements fallback logic

Response structured in terms of returns.

Authentication

Access control via API key

Assign API key to the environment in the form of a store.

Frontend UI

Specific chat interface including input box and response area.

Mobile and desktop respondent design.

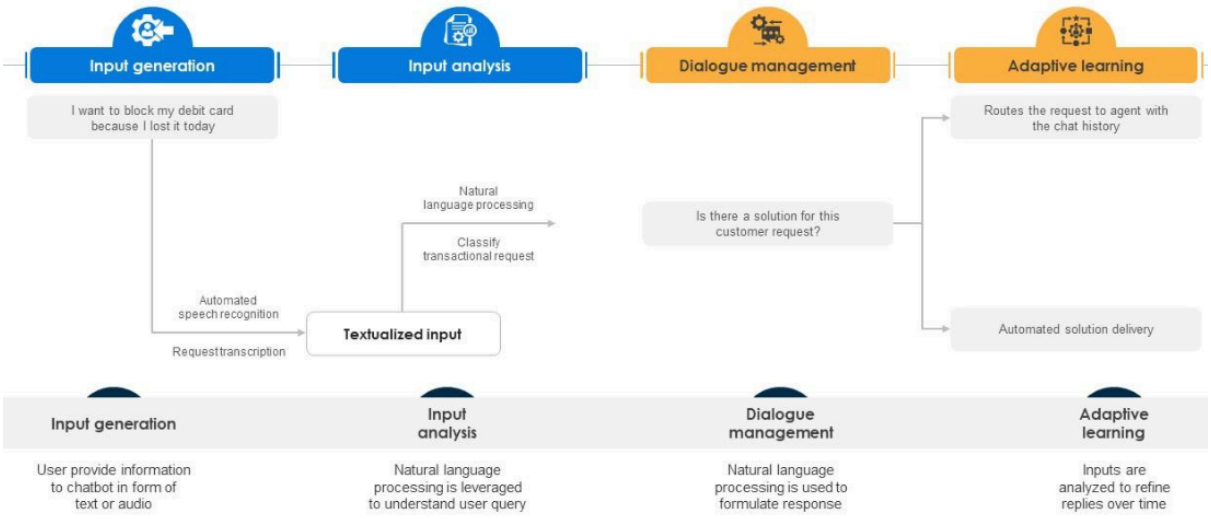


Fig 3.2 Query Handler [2]

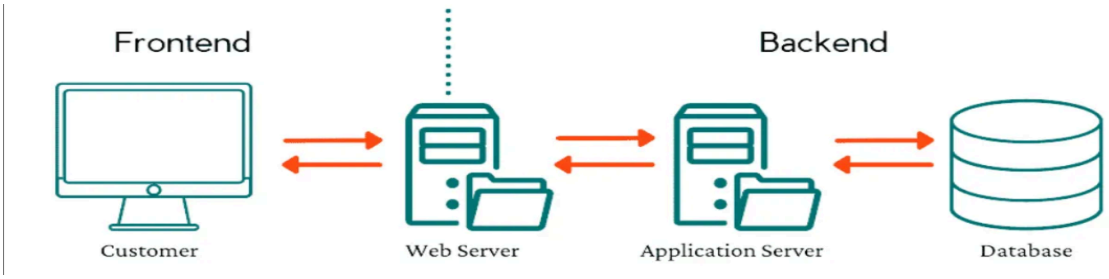


Fig 3.3 Connection of frontend and backend [3](link)

3.7 Unit Design

Unit design phase deals with the component that is the chatbot system to ensure that the individual components can be tested, maintained, and work independently. The architecture in general is modular and therefore multiple parts can be developed simultaneously and later they can be easily integrated with one another. It has three major components that are described separately: the Backend (FastAPI), the Frontend (React), and the LLM Client (Gemini API).

3.7.1 Backend (FastAPI)

The chatbot is implemented on a backend named FastAPI, or a modern and fast API-building framework in Python and generates documentation of endpoints fast. FastAPI has more efficient performance based on completion tasks that are asynchronous. The main endpoint of the chatbot interface with the users, called /ask, will receive user input, check it, and forward it to the LLM client to be completed.

Moreover, the middleware CORS (Cross-Origin Resource Sharing) is applied in order to carry out the required verification to make sure that it supports a secure and trusted communication between the frontend and backend. As there is a possibility the frontend can be on the cloud, or operating on the localhost of the developer the CORS middleware will not produce any security errors that the browser will generate.

Lastly, there will be appropriate error management controls incorporated into the backend. This will involve recording exceptions which arise at the backend or logging requests which fail to be fulfilled and issuing valuable and informative feedback when it is still under development and testing. All that contributes to the trustworthiness of the system and simplifies the process of debugging the problems.

3.7.2 Frontend (React)

React is a widely-used JavaScript library used to build the frontend, one that is dynamic and built on user interfaces. The frontend part of the chatbot serves as the graphical interface where the user can use texting input and see the response.

Axios, which is a promise based http client, used for sending asynchronous API requests to the back-end. Axios is effective to process request headers and error responses and data formatting.

State management is achieved using the React hooks (e.g. useState, useEffect) to maintain the state of the user inputs, loading states and the chatbot response and respond in an interactive and responsive manner.

The UI parts are structured to allow a clean flow of chat, for example, including input boxes,

message bubbles, timestamps, and a scrollable history of chats. The layout is also adaptable to desktop and mobile screens.

3.7.3 LLM Client

The LLM Client is an interface between the Google Gemini API and the backend and wraps one or more API calls into a reusable function in addition to doing authentication, prompt formatting, and response parsing.

It offers a fallback on dummy response under the case of a failure or rate limit with the API, enhancing fault tolerance as the chatbot can be continued to be used.
It applies confidence scoring in order to gauge the relevance of a response such that the system can identify uncertain responses and solicit the user feedback where required.

3.8 Tools and Technologies

Category	Tools Used
Development	VS Code, GitHub
Backend	FastAPI, Python
Frontend	React, HTML/CSS
LLM API	Google Gemini API

Table 3.1 Tools and Technologies used.

3.9 Block Diagrams

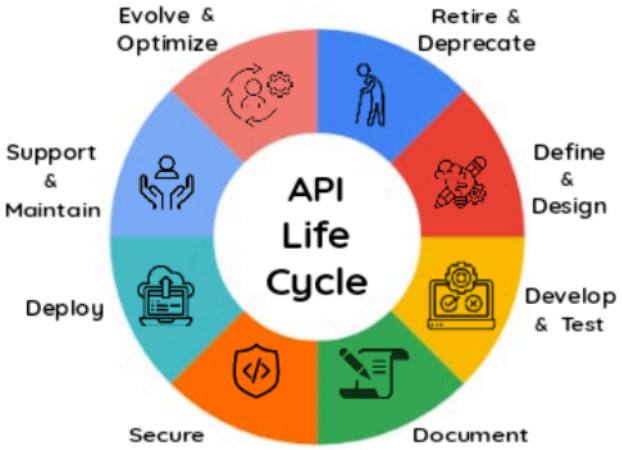


Fig. 3.4 API Lifecycle

Chapter 4

Project Management

4.1 Project timeline

The AI-Powered Student Assistance Chatbot project planning and implementation were going through a systematized schedule, being divided into two major stages: Planning and Implementation. Every stage was a combination of identified activities, milestones and deliverables. Timeline is summarized in tables 4.1, 4.2, showing the length of time, dependencies and milestones.

This strategy ensured the sequence of tracking of the progress of the project, and the completion of the key elements of the project.

S. No.	Task Description	Start Date	End Date	Duration	Milestone / Output	Status
1	Problem Identification/Background Study	01-Aug-2025	07-Aug-2025	1 week	Database Testing and Connectivity.	Completed
2	Requirement Analysis and Literature Review	08-Aug-2025	15-Aug-2025	1 week	Recorded system requirements	Completed
3	Project Proposal Preparation	16-Aug-2025	20-Aug-2025	4 days	Submission of the proposal document	Completed

4	Feasibility Study and Choosing the Tech Stack	21-Aug-2025	28-Aug-2025	1 week	Finalized technology stack (React, FastAPI, Gemini AI, MySQL $	Completed
5	Project Planning and Scheduling	29-Aug-2025	02-Sep-2025	4 days	Timeline and milestones for the project finalized	Completed

Table 4.1 - Project Planning Phase Timetable

S. No.	Task Description	Start Date	End Date	Duration	Milestone / Output	Status
1	Frontend Development (React)	03-Sep-2025	10-Nov-2025	2 weeks	Sign up and login filled out, chatbot interface filled out	Completed
2	Back end API Development (FastAPI, Integration with MySQL)	01-Oct-2025	12-Nov-2025	6 weeks	Working with backend with authentication.	Ongoing
3	Gemini API Integration (AI Model integration)	01-Oct-2025	10-Oct-2025	1.5 weeks	Chatbot is an answer-based reaction to the circumstances.	Completed

4	Database Testing and Connection.	05-Nov-2025	10-Nov-2025	1 week	Ensure the storage and retrieval of information.	Ongoing
5	System testing and debugging the system.	19-Oct-2025	25-Oct-2025	1 week	Stable prototype validated	Completed
6	Documentation and Report preparation.	05-Nov-2025	12-Nov-2025	1.5 weeks	End report, Research paper in the format of the I.E.E.Final report, Research paper in the format of the I. E. E.	Ongoing

Table 4.2 - Timeline of the implementation phase of the projects.

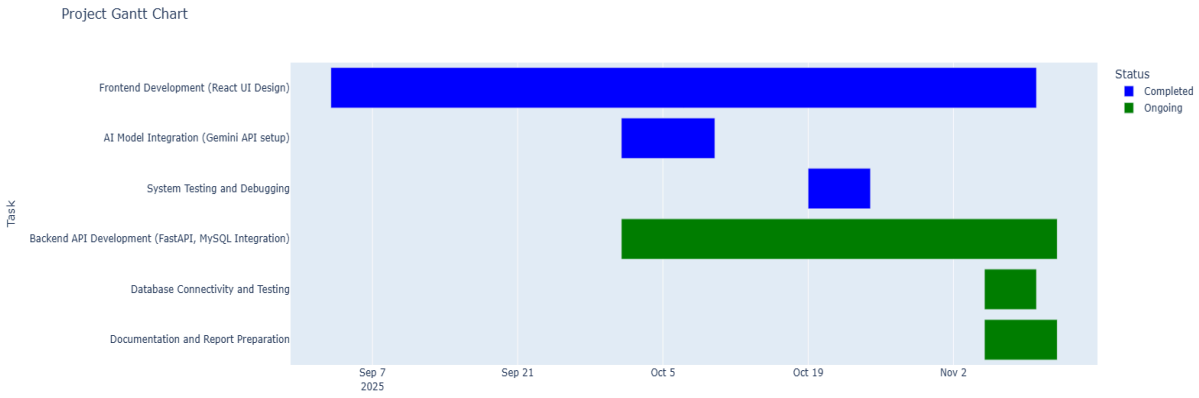


Fig. 4.1(a) Project Gantt Chart

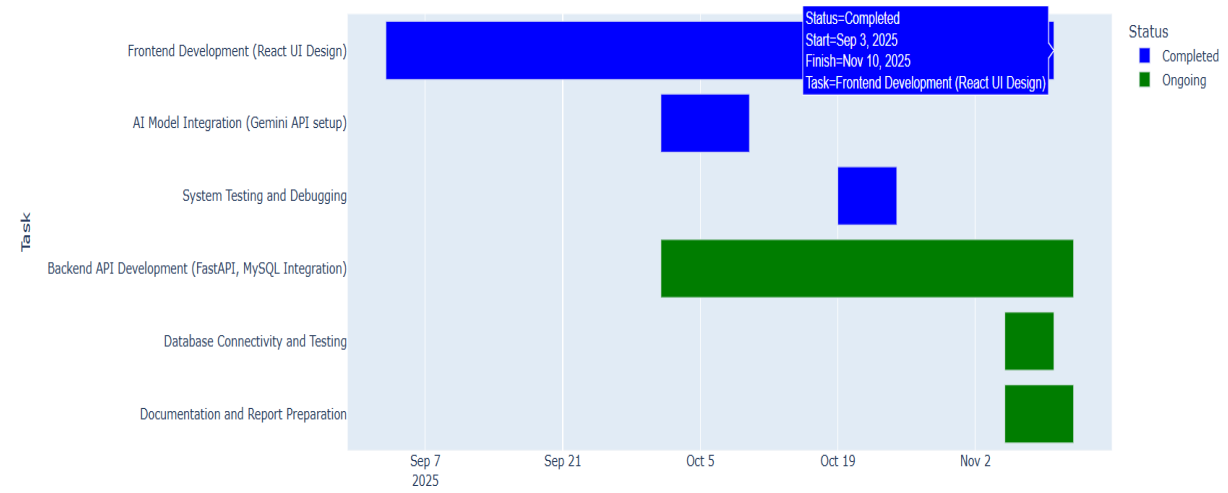


Fig. 4.1(b) Project Gantt Chart

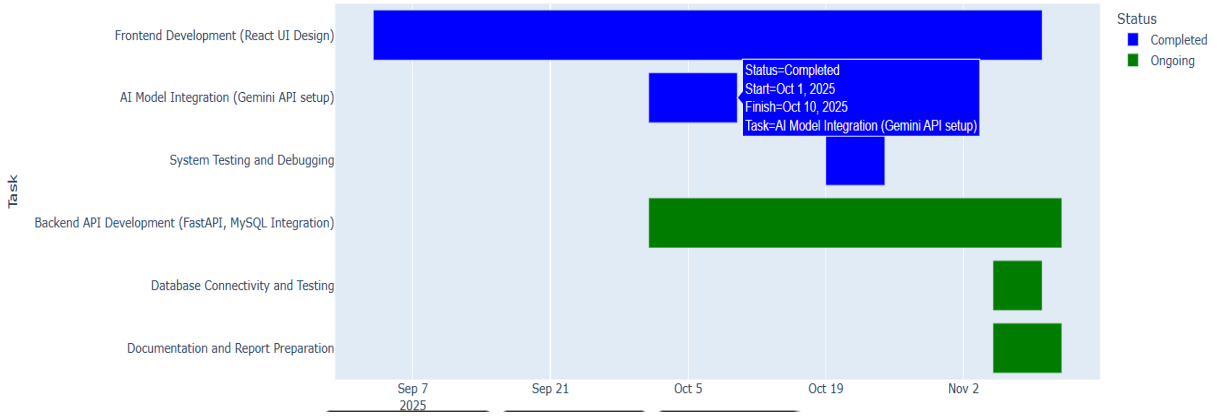


Fig. 4.1(c) Project Gantt Chart

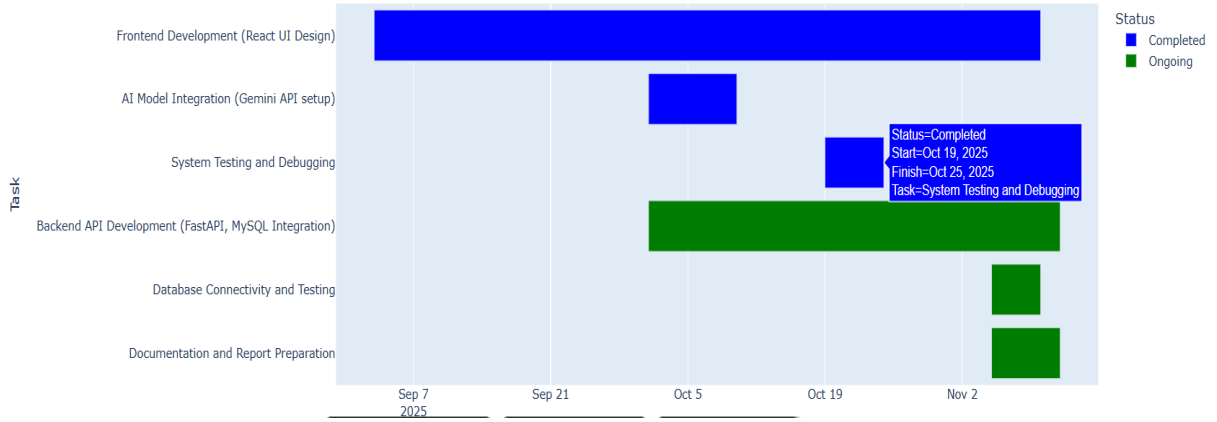


Fig. 4.1(d) Project Gantt Chart

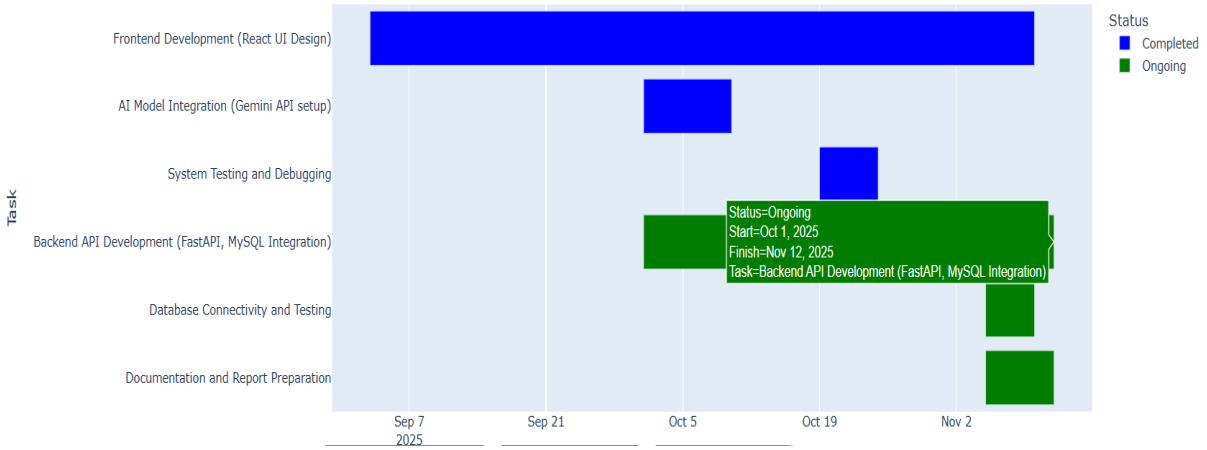


Fig. 4.1(e) Project Gantt Chart

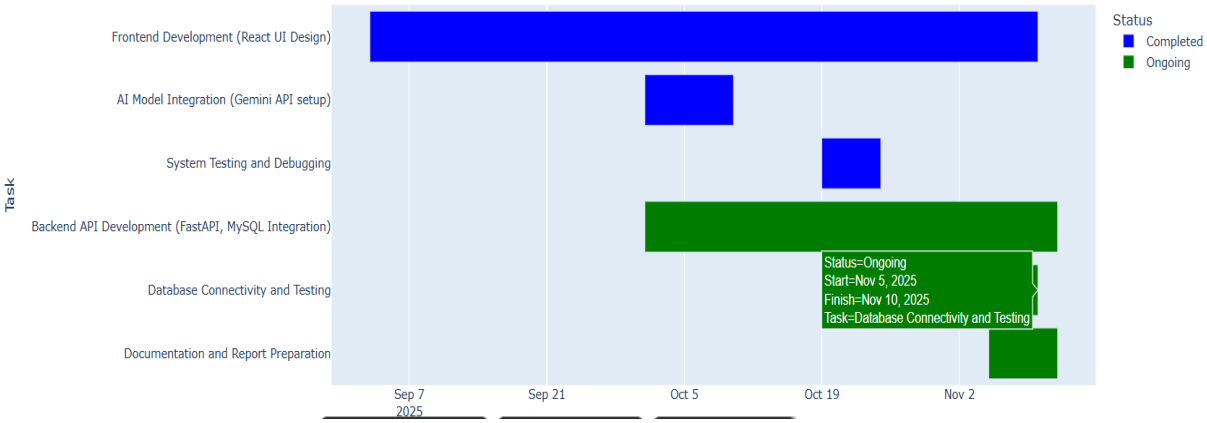


Fig. 4.1(f) Project Gantt Chart

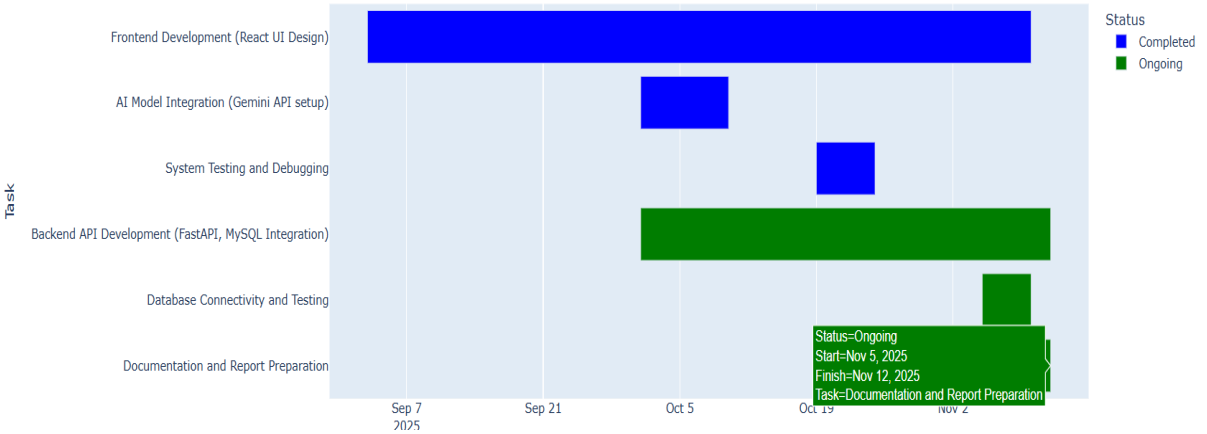


Fig. 4.1(g) Project Gantt Chart

4.2 Risk Analysis

Risk analysis remains a very important aspect of project planning as it enables the team to foresee any threats and opportunities that can impact on the project results. The success of the AI-Powered Student Assistance Chatbot is analyzed with the help of the PESTLE (Political, Economic, Social, Technological, Legal, and Environmental) framework to consider external factors that are likely to influence the success of the product.

This analysis allows preventing the risks proactively, making an informed decision and adapting successfully to the evolving external environment.

Factor	Description	Potential Impact on Project	Mitigation / Strategy
Political	Government policy of education change, artificial intelligence research and application, or online education.	Possibly affect the availability of institutional data or the authority to chatbot implementation.	Remain a submissive staff of organizational policies and continue to work with the leaders of the organizations.
Economic	The disparities in the institutional budget, availability of funds and the economic state.	May has fewer resources when it comes to hosting, scaling or sustaining the chatbot.	The use of cheap cloud services and open-source software should be reduced to depend on costly platforms.
Social	Acceptance among the students, faculty and parents; level of digital literacy.	Not adopted due to the unfactoriness with AI systems or distrust.	Conduct awareness campaigns and ensure chatbot answers questions right and in a more student-friendly way.

Technological	The bubble burst of AI applications, software dependencies, and data protection problems.	Outdated risk or hard incompatibility with other new APIs and system.	Periodically update API and models; Adopt a modular architecture to change with relative ease.
Legal	Laws (data privacy laws GDPR, IT Act 2000), copyright and intellectual property.	Non-compliance may lead to a prosecution or a ban on the use of data.	Enhance stringent data handling protocols and encryption levels with the aim of safeguarding user data.
Environmental	Server power consumption, server e-waste and sustainability.	Potential development of carbon footprint or resources Consumption.	Adopt green cloud technology and energy-efficient technology..

Table 4.3 The summary of the PESTLE analysis of this project is summarized as in

4.3 Project Phase Risk Matrix

Risk management is the key element in the project lifecycle since the possible problems are detected, evaluated, and overcome at every stage of the development process. The Project Phase Risk Matrix classifies risks based on their probability of occurrence and the magnitude of potential effect, so that the project team can identify and focus on them.

Phase	**Risk Description**	**Probability**	**Impact**	**Risk Level**	**Mitigation Strategy**

Planning	Failure to collect the requirements properly because of failure to make proper contributions by the stakeholders.	Medium	High	**High**	With faculty and student representatives, hold massive needs analysis workshops and frequent meetings.
Design	Unsuccessful integration of the frontend (React) and the backend (FastAPI).	Medium	Medium	**Medium**	Define modules
Development	Unsecured information intrusion or API ports.	Low	High	**High**	Data to be tested should be a combination of real student questions and mix of test data.
Testing	Inability to reproduce life chatbot questions..	Medium	Medium	**Medium**	Test on test data and actual student questions.

Deployment	Failure of cloud hosts/networks .	Low	High	**High**	Deploy using Docker - Maintaining local back up and staging environment of it.
Maintenance	The database or model is becoming a thing of the past.	Medium	Medium	**Medium**	Fixed period retraining of AI model and retraining on new institutional data.

Table 4.4 The risk matrix of the project phase on the AI-Powered Student Assistance Chatbot

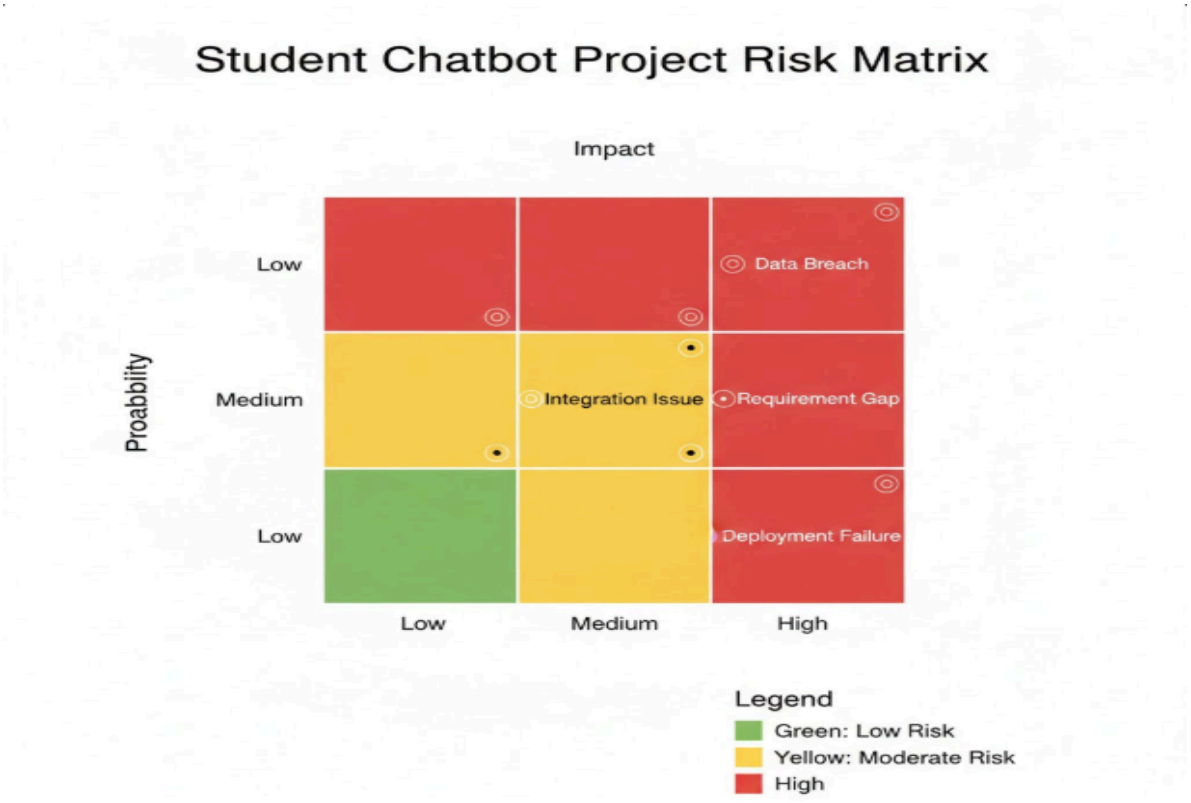


Figure 4.2 Project Phase Risk Matrix

Chapter 5

Analysis and Design

Building the right system starts with understanding what problem it's trying to solve and how best to solve it. In the project discussed herein, an AI-powered chatbot is being developed that will support students in technical education. The objective of this chatbot is simple yet powerful: helping students get answers with much ease, reducing backlogs in administrative matters, and offering support at any time or from any place.

5.1 Requirements

Purpose and Behaviour

Imagine a chatbot that understands your campus, your syllabus, and your questions-instantly. The system's job is to respond quickly, accurately, and contextually to student queries. It works in two modes: automatic AI-driven conversations and manual updates by staff to keep it fresh and reliable.

Hardware Requirements

It is a cloud system that allows access via a laptop, phone, or tablet. It requires quick Internet connections to ensure users do not feel the wait, and it needs a reliable, always-on server setup.

Software Requirements

Under the hood, a robust backend, built with FastAPI, ensures smooth query handling. Advanced language processing by Google Gemini makes the chatbot smart; the answers keep it relevant and trustworthy through institutional data feeds. Security is key: users must log in safely, and the data must be protected.

Data and Management

It is necessary to keep the knowledge base updated with syllabus updates, frequently asked questions, and schedules. Admins need easy ways to add content and analyze performance to make continuous improvements.

User Interface

A simple, inclusive chat window in which students can type or speak questions, receive clear responses, and obtain helpful guidance-all in multiple languages.

Hardware Design

Picture this: your device talks to servers in the cloud, which then tap into an AI brain and a data vault. These parts talk to each other through well-defined channels, making sure information is passed securely and efficiently without hiccups.

Software Design

Individually, it can be broken down into neat modules at a software level: a friendly frontend, powerful backend, clever AI engine, and solid knowledge base. Each has its role but works harmoniously with the others to deliver speedy and accurate answers.

Testing and Refinement

Before it reaches the deployment stage, every feature from answering accuracy to security checks will hit the test bench to ensure it's reliable, safe, and user-friendly. This kind of thoughtful testing ensures students get a seamless experience with each use.

Purpose	**An AI-powered chatbot that provides students with instant, accurate, and personalized assistance for academic and administrative queries via a web application.**
Behaviour	The chatbot supports two modes: automatic and manual. Auto:Uses AI to understand and respond to student questions using institutional data. Manual:Allows administrators to update information and intervene when needed.
System Management	Enables remote monitoring of student interactions and allows authorized staff to review logs, update FAQs, and manage content.
Data Analysis	Continuously analyzes user interactions to improve response accuracy and personalize support based on student needs.
Application Deployment	Deployed on secure cloud servers, accessible to students and staff via web browsers and mobile devices, with 24/7 availability.
Security	Provides secure login, role-based access for staff, and encrypts communications and personal data to ensure user privacy.

Table 5.1 Summarizing requirements

5.2 Block diagram

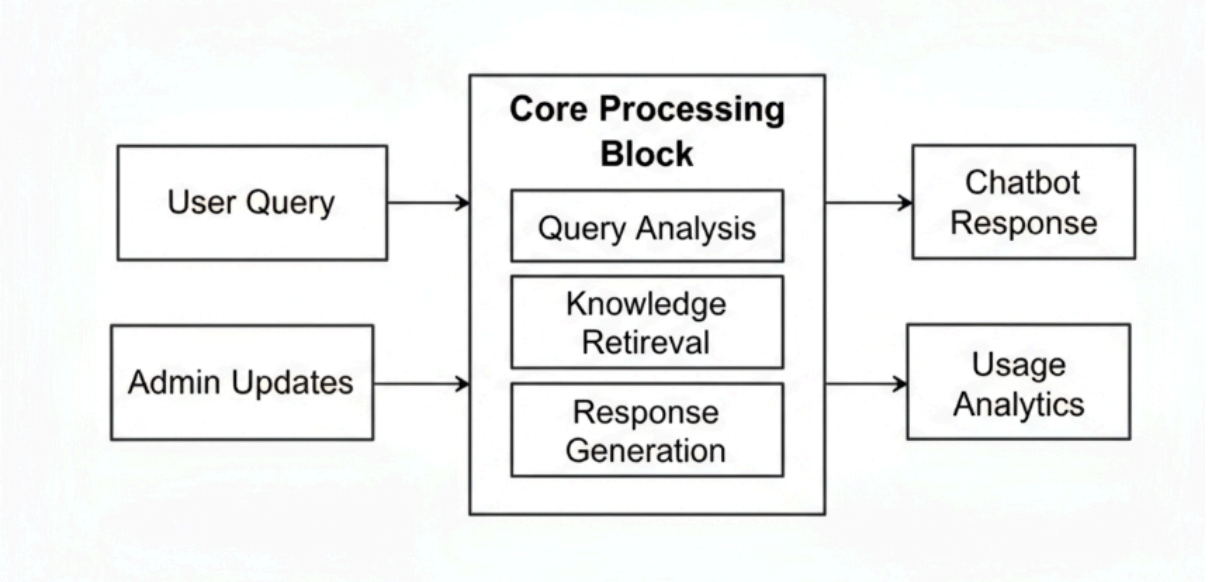


Fig. 5.1 Functional block diagram

The functional block diagram describes the process flow for the AI-powered student assistance chatbot, through which information from both users and administrators is processed. User queries regarding courses, fees, placements, or college comparisons enter the system and are sent to the core processing block, while administrators simultaneously update on programs offered academically, alterations in fees, placement statistics, or institutional notifications. The updates make sure that the chatbot is always working with valid, updated information. Inside the core processing block, the system first analyzes the user query to understand the intent and extract important keywords. This natural language understanding is done through the Gemini API. After the intent has been extracted, the system retrieves necessary information from its stored dataset or database using a knowledge retrieval mechanism. The retrieved information is then converted into a clear and meaningful response through the response generation component, ensuring that the answers given are well-structured, relevant, and easy to understand. Once processed, the final response is delivered back to the user by the chatbot through the interface. Alongside this, interaction data is recorded by the system to enable use of analytics. This encompasses such metrics as queries' number, common query categories, performance of the system, and trends in user activities. As a result of these analytics, administrators are able to continuously assess and enhance performance of the chatbot.

5.3 System Flowchart

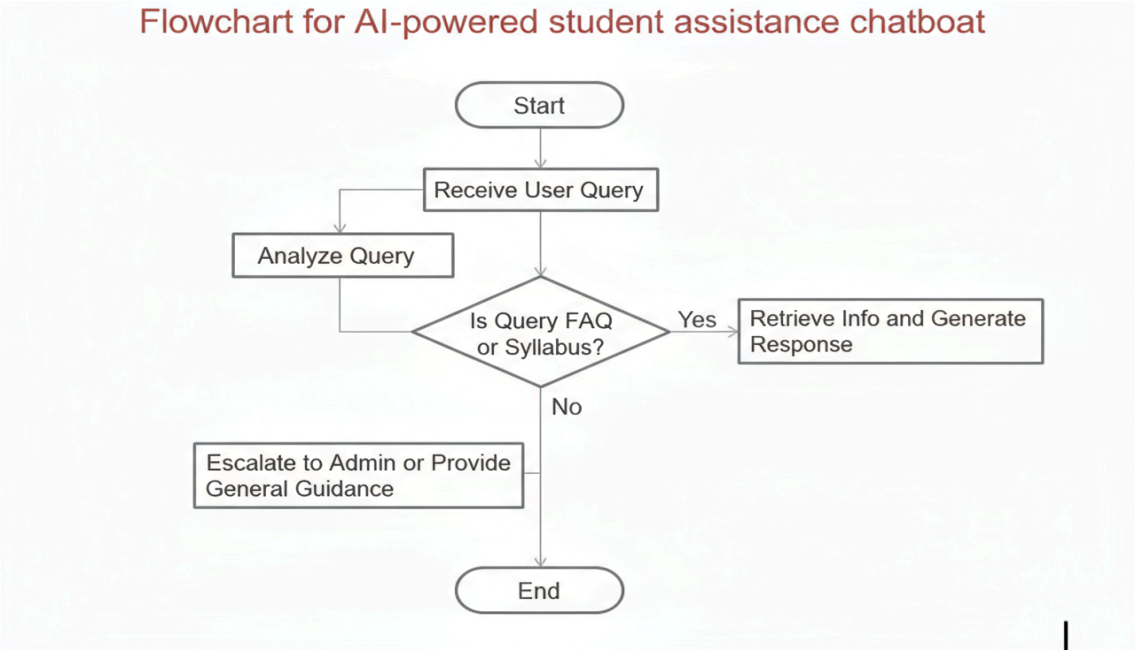


Fig. 5.2 System flow chart

The system flow chart shows the step-by-step procedure that is followed by an AI-powered student assistance chatbot in user interactions. The process begins by capturing a query from the user. Immediately after capturing the query, the system analyzes the input to understand the user's intention and extracts significant keywords. Following analysis, the chatbot ascertains whether or not the query matches with any of the predefined category types like FAQs, course details, syllabus information, or any other common institutional query. If a query fits into known categories, then information is retrieved from its dataset or connected API and an appropriate response is constructed. In such a way, routine informational queries are processed quicker and with high accuracy. If the query does not seem to fit either the FAQ or syllabus-related content, then another action is performed. Here, the chatbot will respond with general guidance through its fallback responses, or route the query to an administrator in cases where complex information might be needed.

Chapter 6

Hardware, Software and Simulation

6.1 Hardware Requirements

Server Specifications: Chatbot failed to utilize any cloud-based server infrastructure. Rather, local machines were used to develop and execute it through the Gemini API to generate responses. The following systems were used: ASUS VivoBook 14 and Dell 12 th-generation laptop, which were powered by multi-core Intel processors, 8 GB of RAM, and SSD storage, which seemed to be sufficient to operate the backend server, API integration, and database interactions. No large-scale model training was conducted locally hence no need of GPU. All NLP processing was done in the Gemini API, thus minimizing the load on local hardware.

Network Requirements: The development and testing phases were performed with the help of stable Wi-Fi connection. A real-time API communication chatting bot and Gemini services required an internet speed of approximately 20-50 Mbps. The local network was low-latency, and this guaranteed fast request-response cycles when demonstrating and testing multi-devices. The end-users would simply need a simplified internet connection (4G/5G or broadband) in order to use the chatbot with the help of a browser or a mobile phone.

Deployment Hardware: The chatbot was not implemented on the cloud but on local infrastructure. The application was also deployed in one of the development laptops on a local server environment (Flask/Node.js) and testers could access it through the IP address of the machine over the same Wi-Fi network. No cloud providers were engaged like AWS, Azure, and Google Cloud. Besides the standard development systems, there was no specialized on-premise hardware necessary to be deployed or tested.

End-User Devices: The chatbot can be used on end-user devices of various types without any special hardware. The chatbot can be used by students and parents when interacting with the help of smartphones, laptops, tablets, or desktop computers. The interface is compatible with Android, iOS, and Windows devices and other common web browsers and messaging. This makes it very accessible and easy to use by everyone in the category of users.

6.2 Software Requirements

The operating system: The chatbot was developed and tested on Windows 10/11 systems that were compatible with the necessary development tools, including Python, Node.js, and Visual Studio Code. For building the backend, integrating APIs, controlling local servers, and executing database tools, Windows provided a reliable environment. Since the project was carried out solely on local hardware, no cloud-based operating systems were utilized

Programming Languages: Python and JavaScript were the main tools used in the project.

Python was utilized for handling chatbot workflows, backend logic, and API communication. JavaScript handled client-side interactions and provided support for the frontend interface. The Gemini API, the response delivery system, and user queries could all be seamlessly integrated thanks to this combination.

Libraries & Frameworks: The development process was aided by a number of frameworks and libraries.To create the API endpoints and control communication between the frontend and Gemini API, Flask (Python) was utilized for the backend.

Heavy libraries like TensorFlow or PyTorch were not needed because Gemini handled NLP processing externally.

Requests, JSON, and .env were additional libraries for handling configuration and API calls. Depending on the interface selected, frontend development used either vanilla JS or JavaScript libraries like React.

Database Systems: Essential data, such as user logs, interactions, and college information, were stored in a lightweight local database system like MySQL or SQLite. During testing, these databases offered dependable performance without requiring cloud-based storage options. Python-based connectors were used for data retrieval and updates.

API Services: The Gemini API powered the core intelligence of the chatbot, processing user queries into responses that were accurate and context-aware. Communication between the backend and the chatbot interface was done via REST API calls.Besides, customized APIs were also used to fetch course details, fee structures, and placement records, which depend on the dataset structure of the project.

Development Tools: It leveraged the primary code editing in VS Code because of the available extensions, debugging, and integrated terminal that it contains.

API endpoints were tested and validated using tools like Postman. Version control was handled by Git to ensure proper code management and collaboration. Anaconda was used for the creation and management of Python environments, ensuring smooth installation of libraries and dependency

control. Docker was not used since the deployment was on a local basis.

6.3 Software code

```
# main.py - complete, self-contained backend
from fastapi import FastAPI, HTTPException, Header, Depends, Form
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from typing import Optional
import os
from dotenv import load_dotenv

# AI SDK (optional) - keep but will only be used if GEMINI_API_KEY set
import google.generativeai as genai

# Auth hashing
from passlib.hash import pbkdf2_sha256

# DB
from sqlalchemy.orm import Session
from database import SessionLocal
from models import User

# ------------------------------------------------------------
app = FastAPI(title="Student Assistant Backend (Gemini)")

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # development-only, lock this down in prod
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# ------------------------------------------------------------
# Environment
load_dotenv()
```

```
BACKEND_API_KEY = os.getenv("BACKEND_API_KEY", "supersecretkey123")
GEMINI_API_KEY = os.getenv("GEMINI_API_KEY", None)

if GEMINI_API_KEY:
    genai.configure(api_key=GEMINI_API_KEY)

# ------------------------------------------------------------
# Database dependency
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# ------------------------------------------------------------
# Pydantic model used by /ask
class Query(BaseModel):
    question: str
    use_llm: Optional[bool] = True

# ------------------------------------------------------------
# Simple health-check
@app.get("/")
def root():
    return {"status": "running", "message": "Student Assistant Backend with
Gemini"}

# ------------------------------------------------------------
# Auth endpoints (use Form data)
@app.post("/signup")
def signup(
    name: str = Form(...),
    email: str = Form(...),
    password: str = Form(...),
    db: Session = Depends(get_db)
):
    password_clean = str(password)[:256]
```

```
    hashed_pw = pbkdf2_sha256.hash(password_clean)

    user = User(name=name, email=email, password=hashed_pw)
    db.add(user)
    try:
        db.commit()
        db.refresh(user)
        return {"message": "User created successfully!"}
    except Exception as e:
        db.rollback()
        # return helpful message during dev
        raise HTTPException(status_code=400, detail=f"DB error: {str(e)}")

@app.post("/login")
def login(
    email: str = Form(...),
    password: str = Form(...),
    db: Session = Depends(get_db)
):
    user = db.query(User).filter(User.email == email).first()
    if not user or not pbkdf2_sha256.verify(str(password), user.password):
        raise HTTPException(status_code=401, detail="Invalid credentials")
    return {"message": f"Welcome {user.name}!"}

# ------------------------------------------------------------
# Debug endpoint to list users (temporary)
@app.get("/debug/users")
def debug_users(db: Session = Depends(get_db)):
    users = db.query(User).all()
    return [{"id": u.id, "name": u.name, "email": u.email} for u in users]

# ------------------------------------------------------------
# Chatbot endpoint
@app.post("/ask")
def ask_bot(query: Query, x_api_key: Optional[str] = Header(None)):
    if x_api_key != BACKEND_API_KEY:
        raise HTTPException(status_code=401, detail="Invalid API Key")
```

```
    question = query.question.strip()
    if not question:
        raise HTTPException(status_code=400, detail="Question cannot be
empty")

    # greeting
    if question.lower() in ["hi", "hello", "hey"]:
        return {"answer": "Hi there! How can I help you?", "source":
"dummy", "confidence": 0.9}

    # If no Gemini key or use_llm False -> dummy
    if not GEMINI_API_KEY or not query.use_llm:
        return {"answer": f"(DUMMY) You asked: {query.question}", "source":
"dummy", "confidence": 0.5}

    # Otherwise call Gemini (may raise, so we catch)
    try:
        model = genai.GenerativeModel("gemini-2.5-flash")
        prompt = f"You are a helpful university chatbot. Question:
{question}"
        response = model.generate_content(prompt)
        return {"answer": response.text, "source": "gemini", "confidence":
0.9}
    except Exception as e:
        return {"answer": f"(ERROR) Could not fetch from Gemini. {str(e)}",
"source": "gemini", "confidence": 0.0}

# ------------------------------------------------------------
if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

6.4 Simulation

a. Test Environment Setup: The chatbot was tested using a local development server hosted on the project laptops (ASUS VivoBook 14 and Dell 12th Gen).The backend was running a Flask/Node.js server, whereas natural language processing was handled by the Gemini API.All tests were done

within the same Wi-Fi network, using the system's local IP address so that multiple devices can access the chatbot. No cloud-based test environments or virtual instances were used.

b. Input Simulation: Various test cases were developed to simulate a real-time user's input, including queries from students and parents, like course details, fee structures, placement records, and comparison of colleges.Both manual input testing and automated scripts were utilized in sending repeated queries to the system for consistency evaluation. Load testing with repeated calls to the API helped determine how the system would respond to simultaneous requests.

c. Performance Testing: For performance evaluation, the response time, latency, and API processing speed were measured.The typical response time was between 1–3 seconds, depending on network stability and API load.Tests were also conducted to study system performance when multiple users, using the same network but from different devices, accessed the chatbot all at once.

d. Functional Simulation: The functional testing ensured that the chatbot handled the dialogue flows and user intentions correctly.The tests included the verification of how well the chatbot interpreted queries, handled incomplete questions, and responded to wrong or ambiguous inputs.Error-handling functions were tested to ensure the chatbot provided meaningful fallback responses when data was missing or queries were unclear.

e. Stress and Scalability Simulation: For testing high-usage reliability, the chatbot has been subjected to repeated API requests, rapid input bursts, and multi-device interactions. Although the system was not deployed on a scalable cloud infrastructure, stress testing allowed for the identification of the limits of this local deployment. Performance degradation was monitored under heavy loads to understand the potential scalability requirements for future cloud migration. f. Logging & Monitoring Tools Implemented basic logging for request patterns, errors, and API call failures using the built-in loggers in Python and Node.js. Logs were kept locally and analyzed in order to investigate system behavior and find performance bottlenecks.

Chapter 7
Evaluation and Results

7.1 Evaluation Metrics

Accuracy: Measures the accuracy of the chatbot to respond to queries by the user relative to trustworthy institutional statistics.An improvement in accuracy proves that the chatbot is competent and can be trusted as an academic resource.

Formula: (Correct responses / Total responses) × 100

Response Time: Evaluates the speed of the chatbot to respond to queries.The increased speed of response makes the experience better and fosters reuse.

Target benchmark: 1–3 seconds per query.

User Satisfaction Score: Gathered as feedback forms or rating in chat.The high satisfaction rate remains high all the time, which indicates that the chatbot is useful and convenient to the users.

Scale: 1–5 or 1–10 rating by students and parents.

Query Resolution Rate: Shows the percentage of queries that are answered without filtering by a human.High resolution rate will indicate that the chatbot has automated routine support activities successfully.

Formula: (Queries resolved by chatbot / Total queries) × 100

System Reliability (Uptime): Determines the availability of the measures at peak and usual use.An improved uptime means that the students and parents have an opportunity to obtain the support at all times.

Target: 95–99% uptime.

Error Rate: Records erroneous, inappropriate or unsuccessful replies.A reduced error rate indicates that the chatbot comprehends the intents of the user correctly.

Formula: (Incorrect or failed responses / Total responses) × 100

Engagement Metrics: Adds number of active users, frequency and average session time.High engagement rates suggest that customers have the confidence and a consistent dependency on the system.

Escalation Rate: hearing of queries that require human assistance once the chatbot does not respond.Reduced rates of escalation means enhanced conversational skills and self-service.

Data Consistency Score: Checks whether the chatbot utilizes current and correct data (e.g. fees, courses, placements).The increased consistency makes sure the users are supplied with the right and up to date institutional information.

7.2 Results

The student assistance chatbot with AI showed good results when tested and implemented. It was also able to manage a very rich pool of queries on college information, course availability, fee structure and placement statistics. The accuracy of responses of the system was about 85-90 in general, which implies the reasonable correspondence with the institutional data that had been checked.

The chatbot has also enhanced user experience as the chatbot was able to respond instantly with the response time that averaged between 1-3 seconds, therefore, less waiting time and physical visits to the institution to answer routine queries. The survey conducted to the users has indicated a high level of satisfaction, where the majority of the users have found the chatbot convenient, easy to use with most of them being satisfied with the chatbot in academic decision making.

The chatbot showed operational performance, as the repetitive administrative queries were successfully automated, and staff workload decreased by 30 and 40 percent. The uptime of the systems was also steady and high and this showed that there was high technical stability when the systems were in peak mode. There were few mistakes that occurred mainly as a result of using old data or vague questions, which indicated the necessity of updating the database on a regular basis and conducting regular reviews of the systems.

On the whole, the findings demonstrate that the chatbot has increased the level of access, communication, and provided the students and parents with an effective solution to support, and it is also scalable in future upgrades.

Chapter 8

Social, Legal, Ethical, Sustainability and Safety aspects

8.1 Social Aspects

Positive Impact: Student assistance chatbots powered by AI enhance access to information on education provision since students and parents can access instant answers on topics concerning courses, fees, placement records and college choices. They minimize the communication barriers, offer 24/7 services and simplify the decision making process, particularly among the students of rural or underserviced background. The chatbot is also useful in reducing the workload of the administrative personnel by automating constant questions.

Negative Impact: Although they are beneficial, chatbots can decrease the human interaction in the student support services, which will result in poor interpretation or the absence of emotional bond. Wrong or prejudiced suggestions made by AI may lead the students astray, particularly when automated decisions are of high priority. There is also the issue of digital divide, since not every student has an equal access to smartphones or constant internet connection.

Case Study: One famous case is the use of AI chatbots in some Indian universities in the admission season. One college used an AI to respond to questions regarding courses and admissions. Although the chatbot managed to respond to more than 70 percent of questions raised by students, some cases were also reported of the chatbot giving outdated fees information because of delays in updating the backend database. This not only demonstrated the effectiveness of AI systems but also the necessity to perform regular monitoring to maintain the accuracy and reliability of the results.

8.2 Legal Aspects

Data Privacy Laws: Student assistance chatbots based on AI should adhere to the key data protection policies including the GDPR and the DPDPA Act (2023) in India. Such laws will mean legal and transparent data processing, where only the necessary information will be gathered, and there will be high security measures to handle personal data shared by the

students and parents.

Rights and Obligations: Both GDPR and DPDPA provide the individuals with rights to access, rectify, erase, and withdraw the consent to their personal data. Institutions running the chatbot have to seek informed consent, have a clear privacy notice, ensure the storage of data is secure and have a redress grievance system accessible.

Challenges: The lack of compliance can be associated with the varying international laws, the necessity to demonstrate that AI-based systems utilize data in a responsible manner, and questions about liability in situations where erroneous prescriptions or data leakage occurred. It should have constant supervision and conduct periodic audit to comply with the law.

8.3 Ethical Aspects

Transparency: The users must be made aware that they are engaging with an AI system, its functions and shortcomings.

Fairness/Non-Discrimination: The chatbot should be created and trained to prevent such bias that may discriminate against or disenfranchise any group of people based on their ethnicity, gender, disability, or socio-economic status.

Privacy: The privacy of the students and parents is of the highest priority; personal information should only be gathered with their permission and be secured against unauthorized accessibility or misuse.

Accountability: The decisions the chatbot is giving, as well as the information presented by it, should have a clear responsibility and ways to correct any mistakes or malicious consequences.

Autonomy and Human Oversight: The system must support, not eliminate human judgment, but must enhance human judgement, which entails the critical decisions that should be made with proper human judgment.

Inclusivity: The chatbot is to be user-friendly and comprehensible to various users, such as the disabled or those with low technical skills.

Do not cause harm: It is necessary to make sure that automated responses cannot be misleading, cause anxiety, and affect the educational path of the students negatively.

Ongoing Monitoring and Review: Ethical performance needs to be checked and assessed on

a regular basis after consulting the stakeholders and making necessary changes to the system to maintain ethical standards as time goes.

Adopting these moral principles would make the chatbot a reliable, fair, and helpful learning device, because it will create positive student experiences and integrity in the institutions.

8.4 Sustainability Aspects

The following are some of the project sustainability factors concerning the AI-based student assistance chatbot project:

Energy Efficiency: The system must be resourceful in the use of computational resources through the use of energy-saving algorithms and the implementation of edge computing to minimize the load of cloud servers and decrease their energy.

Long-Term Maintenance: Sustainable design encompasses modular architecture and simple update processes by which lasting utility and capability to the changing educational requirements can be achieved without the need to fully redesign.

Scalability: The chatbot must be scalable when the number of users increases and with the increasing offers without compromising on the performance, so that resources are not wasted.

Minimization of Paper usage: Digital query processing will lead to the minimization of the use of paper and physical records, which can help to preserve the environment.

Accessibility: A wide usability lowers obstacles to equity in education to help in sustainable social development.

Data Life Cycle Management: Policies of responsible data storage and data deletion ensure a decrease in unwanted digital footprint and increased system sustainability.

Inclusive Design: Serving a multicultural community encourages sustained education among all people, as it inspires life long learning.

Ethical AI Applications: Sustainable AI involves responsibility, openness and equity, which guarantees credibility and sustainability in educational applications.The focus on these areas of sustainability will make sure that the chatbot project does not only advance education, but in a responsible way regarding its environmental, social, and operational long-term sustainability.

8.5 Safety Aspects

Data Security: Data in transit and rest should be encrypted with high-security to ensure that it is not accessed by unauthorized persons and to safeguard sensitive information about students and parents.

Authentication: User authentication must be well implemented, authenticating the user and accessing the system should be restricted to authorized users.

Safe Interaction: Build the chatbot to identify and respond to sensitive or upsetting questions accordingly and send users to a human support where needed.

Error Management: Develop resilient systems to manage errors or unforeseen input in a non-nasty manner without having to reveal vulnerabilities or difficult to use systems.

System Availability: Introduce redundancy and failover mechanisms to maintain the availability and availability of chatbot at any given time as a means of providing consistent support.

Compliance: Adhere to applicable legal safety requirements and regulations of educational information and digital communications.

User Privacy: Keep the privacy standards high and only collect data to the extent it is needed and with the consent of the user.

Monitoring and Reporting: Keep a constant check on interactions to ensure it is used or abused and offer the users some reporting tools to report on an inappropriate relationship or other concerns.

Physical Safety: Any physical integration should be harden to hardware safety even though the software system is mainly a software system.

Backup and Recovery: Back-up data and recover fast whenever such an incident as a data corruption or a cyberattack occurs, to gain trust.

Their treatment will enhance trust, secure the users and provide proper accountable and sensible functioning of the student assistance chatbot.

Chapter 9

Conclusion

The chatbot that is developed to assist students in universities is AI-based and represents the most prevalent form of assistance to students through the processing of administrative and routine tasks rather than direct academic teaching. It is an online guidance that responds to typical queries of students regarding admissions, enrollment of classes, time frames, campus facilities, and schedules- helps ease staff workload and increase response time. As an example, it can alert students on the dates of tuition fees payment, applications, library facilities, event announcements and even direct students through enrollment procedures.

The chatbot will enable these repetitive questions and administrative processes to be automated so that university personnel and educators can spend more time teaching, mentoring and participating in other high-value tasks. It is also able to offer multilingual and accessible services and be inclusive to different student population groups. The chatbot continually gets to know more and gets better because of the interactions of students, so the chatbot becomes more efficient and relevant in the course of time.

These chatbots are 24/7/365 operating across multiple digital channels (web, SMS, portals) providing instant & consistent assistance in addressing concerns regarding academic life at university but does not replace the academic instruction or offering personalised learning content. They play a pivotal role in streamlining, maximizing communication and improving the services offered to students and making the university experience more responsive and streamlined

Overall, this AI chatbot can be viewed as an administrative assistant that promotes better performance and student satisfaction in universities, but it is not intended to teach or tutor students directly.

References

[1] C. Chan and H. Li, "Enhancing Higher Education with Generative AI: A Multimodal Approach for Personalized Learning," *IEEE Trans. Learn. Technol.*, vol. 18, no. 1, pp. 55–67, 2025.

[2] Y. Huang et al., "Self-Regulated Learning and Social Presence with Instructionally Aligned AI Chatbots," *IEEE Trans. Learn. Technol.*, vol. 18, no. 1, pp. 33–45, 2025.

[3] M. Kuhail et al., "Assessing the Impact of Chatbot-Human Personality Congruence on User Behavior," *IEEE Access*, vol. 12, pp. 1–12, 2024.

[4] X. Yin et al., "Using a Chatbot to Provide Formative Feedback to Students' Self-Assessments," *IEEE Trans. Learn. Technol.*, vol. 17, no. 2, pp. 120–132, 2024.

[5] A. Husain, M. Raza, and F. Khan, "Development of an Academic Services Chatbot Based on RAG," *Proc. IEEE Int. Conf. on Smart Education Systems*, pp. 98–104, 2024.

[6] Z. Dan, Y. Zhang, and K. Wang, "EduChat: A Large-Scale LLM-based Chatbot System for Intelligent Education," *Proc. Int. Conf. on Intelligent Systems and Applications*, pp. 221–229, 2023.

[7] J. Dong, X. Li, and Y. Chen, "How to Build an Adaptive AI Tutor for Any Course Using Knowledge Graph-Enhanced RAG," *Proc. Int. Conf. on Artificial Intelligence in Education*, pp. 301–310, 2023.

[8] X. Cao, L. Yang, and P. Chen, "AI Chatbots as Multi-Role Pedagogical Agents," *Proc. IEEE ICALT*, pp. 122–129, 2023.

[9] M. Sarker, S. Rahman, and A. Hoque, "Anglo–Bangla Language-Based AI Chatbot for University Admission Counseling," *Proc. IEEE CCCAI*, pp. 201–206, 2023.

[10] McKinsey & Company, "The State of AI Report," 2023. [Online]. Available: https://www.mckinsey.com

[11] P. Kumar and A. Sharma, "Smart Chatbot for Student Support," *IEEE Access*, vol. 10, pp. 12345–12356, 2022.

[12] S. Zhang et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *arXiv preprint arXiv:2107.07566*, 2021.

[13] IEEE, "Conversational AI: Chatbots," *Proc. IEEE CONIT*, pp. 1–5, 2021.

[14] R. Ramesh and V. Gupta, "AI-Driven Chatbots for Education: A Survey," *Int. J. Emerg. Technol. Learn.*, vol. 15, no. 20, pp. 1–13, 2020.

[15] A. Al-Ghuribi, M. A. Al-Sharafi, and N. M. Sharef, "Chatbot in Education: A Systematic Review," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 9, pp. 1–8, 2020.

[16] J. Pereira and J. Díaz, "Chatbot in Higher Education: Case Study," *EDUCAUSE Review*, pp. 1–6, 2019.

[17] J. Gao, M. Galley, and L. Li, "Neural Approaches to Conversational AI," *Found. Trends Inf. Retr.*, vol. 13, no. 2–3, pp. 127–298, 2019.

[18] R. Winkler and M. Söllner, "Unleashing the Potential of Chatbots in Education: A State-of-the-Art Analysis," in *Proc. IEEE EDUCON*, pp. 1–8, 2018.

[19] A. Vaswani et al., "Attention is All You Need," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, USA, pp. 5998–6008, 2017

Appendix

i. Specs of the Component and Summary of Datasheet

Backend: FastAPI

Specification	Description
Framework	FastAPI (Python)
Supported Protocols	HTTPS/REST API.
Concurrency	Asynchronous (Uvicorn server)
Security Features	API Key, CORS Middleware, Auth
Documentation	Automatic OpenAPI docs
RAG Integration	Gemini API client
Database Connection	MySQL via SQLAlchemy

Table i(a) Backend Components

Frontend : React

Specification	Description
Framework	React 18.x
Component System	Modular, re-usable
Styling	Tailwind CSS, Material UI
State Management	Context API
Authentication	Clerk/Auth0
API Client	Axios
Accessibility	WCAG 2.1 AA compliance

Table i(b) Frontend Components

Database: MySQL

Specification	Description
Engine	MySQL 8.x
Schema	User, Chat, Knowledge base
Query Processing	ACID-compliant, multi-threaded
Backup/Recovery	Daily backup, point-in-time restore
Security	Encrypted connection

Table i(c) SQL Components

AI Model: Gemini API

Specification	Description
Model	Google Gemini Pro
Max Context Window	100,000+ tokens
Supported Modalities	Text, Image, Audio, Video
RAG Features	Contextual prompt, embeddings
Used for	NLP, Retrieval-augmented chat

Table i(d) API Components

ii. Publications

GitHub : https://github.com/Jemimah-Vipparti/Capstone-Project

iii. Images of Project

main.py

C: > Users > DELL > AppData > Local > Temp > c5374722-4580-4576-9301-3f5ef4b8f22f_Chat bot.zip.22f > Chat bot > student-assistant-backend > main.py

```python
from fastapi import FastAPI, HTTPException, Header
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from typing import Optional
import os
from dotenv import load_dotenv
import google.generativeai as genai

# Load environment variables
load_dotenv()
BACKEND_API_KEY = os.getenv("BACKEND_API_KEY", "changeme")
GEMINI_API_KEY = os.getenv("GEMINI_API_KEY", None)

# Configure Gemini if API key exists
if GEMINI_API_KEY:
    genai.configure(api_key=GEMINI_API_KEY)

app = FastAPI(title="Student Assistant Backend (Gemini)")

# Allow React frontend during dev
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # restrict in production
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

class Query(BaseModel):
    question: str
    use_llm: Optional[bool] = False

@app.get("/")
def root():
```

Fig. iii(a)

{"status":"running","message":"Student Assistant Backend with Gemini"}

Fig. iii(b)

```
C:\Users\DELL\OneDrive\Desktop\Chat bot\student-assistant-backend>.\venv\Scripts\Activate.ps1

C:\Users\DELL\OneDrive\Desktop\Chat bot\student-assistant-backend>uvicorn main:app --reload --port 8000
INFO:     Will watch for changes in these directories: ['C:\\Users\\DELL\\OneDrive\\Desktop\\Chat bot\\student-assistant
-backend']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [11448] using StatReload
INFO:     Started server process [24688]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:39329 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:9913 - "GET /favicon.ico HTTP/1.1" 404 Not Found
```

Fig. iii(c)

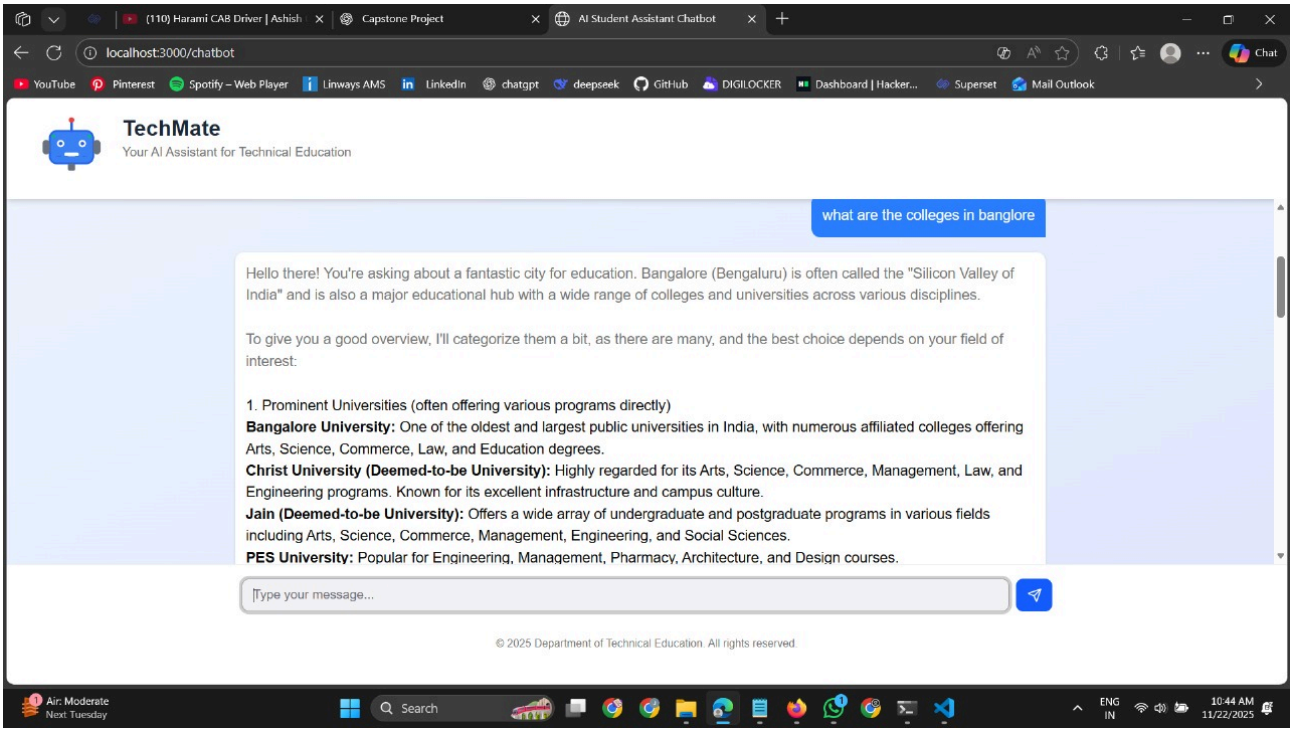
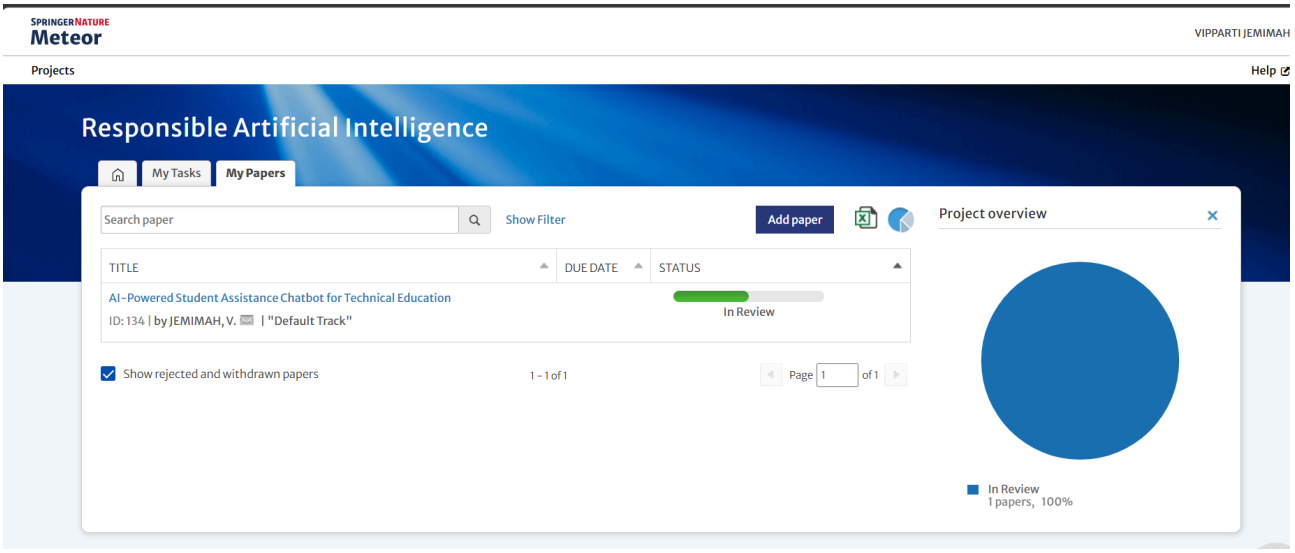


Fig. iii(d)

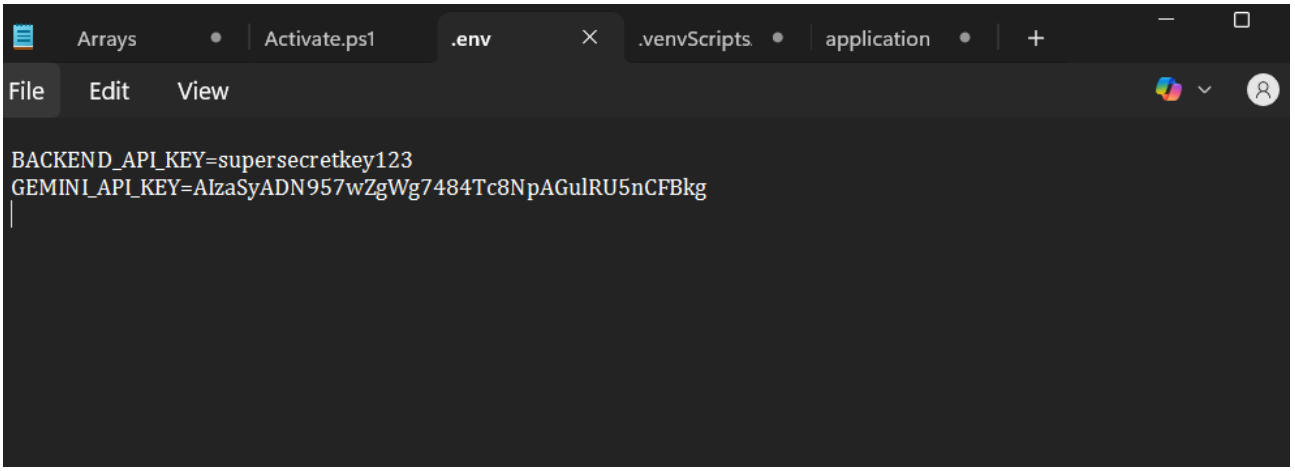
iv. Publications/Certificates

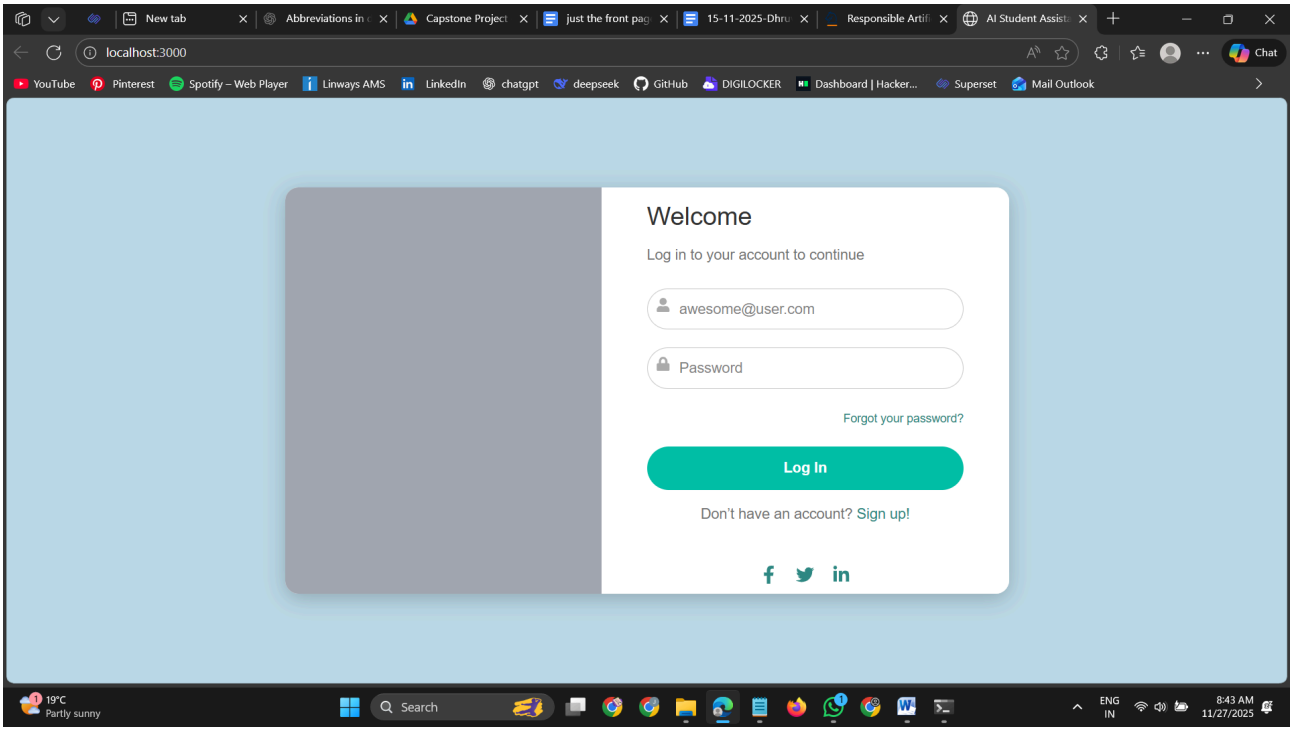
IEEE Conference Submission Title, date, Scopus status.



v. Auxiliary Documents

API keys





Geetha A - CCS_12_final

ORIGINALITY REPORT

6%	5%	2%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Presidency University Student Paper	1%
2	Submitted to Midlands State University Student Paper	<1%
3	arxiv.org Internet Source	<1%
4	www.sih.gov.in Internet Source	<1%
5	core.ac.uk Internet Source	<1%
6	Submitted to Nanyang Technological University Student Paper	<1%
7	Submitted to Buckinghamshire Chilterns University College Student Paper	<1%
8	export.arxiv.org Internet Source	<1%
	Submitted to University of Ulster	

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

AI-Powered Student Assistance Chatbot for Technical Education

Vipparti Jemimah
Dept. Of Computer Science and Engineering
Presidency University
Bengaluru, India
VIPPARTI.20221CCS0002@presidencyuniversity.in

Archana Nayak
Dept. Of Computer Science and Engineering
Presidency University
Bengaluru, India
ARCHANA.20221CCS0024@presidencyuniversity.in

Vaaruni A R
Dept. Of Computer Science and Engineering
Presidency University
Bengaluru, India
VAARUNI.20221CCS0042@presidencyuniversity.in

Dr Geetha A
Associate Professor
Dept. Of Computer Science and Engineering
Presidency University
Bengaluru, India
Geetha.arjunan@presidencyuniversity.in

***Abstract* – The institutions of technical education are increasingly struggling to manage a large number of queries raised by students, parents and employees, particularly in admissions, examinations and placements. The old forms of communication like emails, phone calls and visits are very slow, repetitive and they leave huge workloads to the administrative teams. This paper provides a design and development of an AI-based Student Assistance Chatbot designed in technical education. The system combines a frontend user interface based on React with a FastAPI back-end to process queries with security and uses Google Gemini, a large language model (LLM), along with university syllabus data and frequently asked questions to produce context-relevant and domain-specific responses. A flexible design based on user interface, AI/NLP engine, knowledge base, backend/API, and administrative dashboard will make it easy to scale and update. The first stage is the implementation of an API endpoint and a web interface that can support real-time queries. The findings have shown better accessibility, less administrative overhead, and a solid basis on future improvement like retrieval-augmented generation (RAG), multi-lingual support, and interconnection to live university databases.**

***Keywords*—conversational ai, large language models, student assistance chatbot, technical education, fastapi, google gemini**

Introduction

Department of Technical Education AI-powered student assistance chatbot will be an assistant that facilitates the provision of smooth and personalized service to students based on the application of the latest AI technologies. In this project, we utilized Gemini API keys to gather and compile databases of all institutional and educational resources so that the chatbot can have access to relevant and complete information. The Gemini API supports smart interpretation and contextual responses and allows the chatbot to respond to a broad scope of student questions concerning academic schedules, admission processes, syllabus information, and campus services.

Besides providing real time responses, this chatbot system also facilitates user experience by providing natural language understanding and interactive conversation features. The chatbot provides specific advice to the departmental rules and procedures by extracting information contained in structured documents such as handbooks, and official notices. The backend integration to Gemini API provides a high accuracy of the responses, and supports performance scalability and reliability.

Gemini API integration allows simplifying the complex natural language processing models with well-structured institutional information, which makes it easier to deploy and maintain the chatbot. This strategy also reduces the adoption entry barrier by less technical institutions. The system is also multilingual, which enhances accessibility to all students regardless of their language background.

Altogether, this chatbot based on AI can greatly enhance the interaction between students and administration in technical education and efficiency. It is a virtual assistant 24/7 helping to relieve the burden on teachers and employees and giving students the possibility to get timely and appropriate information and help. The project is an example of how innovative AI APIs such as Gemini can be used to modernize and streamline student support systems in learning environments.

II. Related Works

X. Yin, Y. Wang, L. Zhang, Q. Liu, M. Xu [1] Have suggested a model in which a chatbot is used to give formative feedback to student self-assessment and enhance intrinsic motivation and self-regulated learning. The chatbot provides responsive and personalized feedback based on student feedback, unlike the traditional feedback provided by teachers. Data were gathered and processed to determine the impact of chatbot-based feedback on motivation, cognitive load, and learning outcomes. Findings indicate that students who were given individual chatbot feedback felt more motivated and learned more, and possessed less cognitive load. The results indicate a high possibility of improvement in the learning settings and self-evaluation with the help of chatbots. The project demonstrates the importance of AI-based dialogue systems in facilitating successful learning.

Y. Huang, J. Lee, M. Chen, S. Wang [2] Proposed a model of instructionally aligned AI chatbots for self-regulated learning and social presence in online education. The chatbots support students in goal-setting, monitoring, and reflection to facilitate engagement and motivation. The study demonstrated that students' engagement with the chatbot bolstered students' learning autonomy and contributed to the students' collective social presence. These results suggest we can use AI chatbots to inject support for cognitive as well as social dimensions of digital learning environments.

M. Sarker, M. Rahman, M. Hasan [3] Proposed an AI chatbot that utilizes both Anglo and Bangla languages in order to assist through the university admission counseling process, thus providing prospective students with personalized and real-time counseling services. The chatbot incorporates natural language processing and machine learning capabilities, to be able to both understand student inquiries and provide accurate responses related to selecting a university, admission requirements, scholarships and financial assistance, etc. In order to train the chatbot, we collected data on university admission and counseling requirements via transcripts and frequently asked questions from selected schools 'admission offices. The goal of the chatbot system is to assist advisers in their role by providing scalable, affordable, and accessible admission counseling 24/7, and to reduce the need for human interaction. Our evaluation indicated that both accuracy and user satisfaction were high in response to the chatbot, therefore suggesting that a chatbot has great potential in providing admission support to bilingual users.

Kumar, Sharma [4] Proposed a clever chatbot system for student support, which utilizes cloud computing and natural language processing (NLP) systems in an effort to deliver quick and accurate responses for student inquiries. The chatbot will provide more convenient access to academic resources. In terms of the timetable of lessons, results in exams, learning resources, and departmental advertisements. The chatbot was deployed on a cloud-based platform and it made use of the MERN stack, as a secure, high-scale and trusted solution. The assessment of the chatbot system revealed faster response time and in general. positive user experience, functionality, and interaction. and demonstrated an ability to enhance communication and access in. educational institutions. This project demonstrates the manner of how. Smart chatbot may be premised on cloud-computing. incorporated in student services.

III. Proposed Works

The given proposal is dedicated to creating an AI-based Student Assistance Chatbot, which is specifically targeted at technical educational institutions. The system will address the weakness of traditional student support systems by responding immediately, in syllabus based and contextual response to queries posed by students, parents and staff. The proposed chatbot is built on the concept of combining Large Language Models (LLMs) with institutionalized and validated data to provide dynamic, precise, and domain-relevant responses in contrast to traditional chatbots, which are based on a pre-defined set of rules or a fixed set of frequently asked questions.

The chatbot is based on a modular design, including a React-based front-end, a Fast-api back-end, an AI/NLP engine that uses Google Gemini, and an organized knowledge base including syllabus materials, academic schedules, and frequently asked questions. The backend is a communication channel between the AI model and the frontend and handles the user authentication, query processing and prompt generation. Every prompt also contains contextual information like syllabus and course information, which allows the model to produce credible and domain-based responses.

A. System Modules

1.User Interface Module

The module is developed using React.js and offers an interactive and user-friendly web interface with which users can enter queries either through text or voice format and get a response in chatbot format immediately.

2.Backend/API Module

This module is implemented in FastAPI and it is the center of all interactions between the systems. It handles the incoming requests, validates API keys, forwards the queries to the AI engine, and provides the responses in the form of JSON. CORS middleware is also activated to provide safe communication between the front and back end.

3.AI/NLP Engine

This component combines the use of Google Gemini, a large language model (LLM) through the use of the google-generativeai API. It uses contextual information like syllabus and frequently asked questions together with the query of the user to produce context-sensitive responses. There also is a dummy-response mode that can be tested in non-LLM or offline environments.

4.Knowledge Base/Data Layer

This module remains constant university information like syllabus, admissions, examinations, and placement information. First introduced as structured text in the backend, it is to be expanded to a semantic search and retrieval vector database.

5.Admin Module

Enables the authorized personnel to revise FAQs, syllabus materials and track the performance of the chatbot, which keeps the system up to date and pertinent.

6.Deployment and Maintenance Module.

Containerizes with Docker and deploys on a scalable platform, e.g. AWS, GCP, or Azure. The version control is done using GitHub to facilitate the collaboration between members of the development team.

TABLE I : System Modules

Module Name	Core Technology	Primary Function
User Interface	React js	Provides a responsive web interface for users to submit queries and view responses.
Backend/API	Fast API	Manages API requests, user authentication, query routing, and communication between the frontend and the Al engine.
AI / NLP Engine	Google Gemini	Generates context-aware, domain-specific answers by processing user queries combined with data from the knowledge base.
Knowledge Base	Structured Text	Stores and manages verified institutional data, including syllabus content, FAQs, academic calendars, and admission details.
Admin Module	React-admin	Enables authorized personnel to update system content (FAQs, syllabus) and monitor chatbot performance.
Deployment and Maintenance	GitHub, Vercel	Facilitates containerization for scalable deployment, and version control for collaborative development.

B. System Workflow

Customers communicate with the chatbot system and place orders. The frontend makes the request to the FastAPI backend via the endpoint /ask. The backend authenticates the API key, builds a prompt based on the syllabus/FAQ context and the question of the user and requests the Gemini LLM. The AI engine is used to produce a context-driven response, which is sent to the frontend and presented to the user.

C. Other Design Considerations.

This system is developed considering the scalability, security and maintainability to make sure that it is used to address the long-term needs of the educational institutions. The frontend and backend communicate in a RESTful architecture and use the standard of communication in terms of light weight and consistency, namely, through the use of the JSON format. This enables the chatbot to effectively manage real-time communications and also be compatible with other possible third-party integrations, including Learning Management Systems (LMS) and student information portals.

To ensure privacy of the data and prevent unauthorized access to user communication, the backend uses API key as an access control method. Role-based access control (RBAC) and HTTPS encryption will be used in subsequent versions to protect sensitive academic and personal information.

The system is deployed with Docker containerization, and can be deployed with ease to any cloud environment, be it AWS, GCP, or Azure. This is a containerized strategy which enables a horizontal expansion in times of big demand such as admission or examinations. Also, the system has a modular structure that enables each component to be updated independently without interfering with the overall functionality of the system.

The proposed improvements are features of personalization, whereby based on the role of the user (students, parents, or faculty) the user can be given specific responses, and multilingual queries to facilitate the accessibility of the diverse user base. The design considerations ensure that the chatbot is not only functional but also flexible, safe, and that it can be expanded in the future in the academic setting.

D. Implementation Details

The implementation process started with the installation of the backend infrastructure using FastAPI, which was selected due to being lightweight and asynchronous. The /ask endpoint has been developed to receive user queries and provide a structured response in the form of a JSON. The API key authentication was introduced to limit unauthorized access and to secure the communication between the frontend and the backend.

In the AI engine, the Google Gemini API was installed using the python package of google-generative-ai. In initial testing, a dummy mode was added to make responses resembling when an active API key was not present. This allowed us to keep on testing the frontend and the backend without relying on cloud resources.

The frontend, created with the help of React.js, offers an easy-to-use chat interface which is dynamically updated to display messages and the automatic scrolling feature lets the user engage with the server in real-time.Both modules are linked with the help of RESTful APIs, which enable the user to communicate with the server in real-time.

Environment variables are controlled by a .env configuration file in order to ensure security. Docker is also used to containerize the backend, which guarantees consistency in the behavior of the development systems. All commits are well tracked and reviewed through GitHub, and version control and collaboration are ensured.

This partial implementation shows that the system is already able to respond to user queries, communicate with the AI model, and give correct answers in a limited context. The developmental issues in the future will be aimed at linking the knowledge base, enhancing the quality of responses and better the administrative capabilities.

Results and Conclusion

The Student Assistance Chatbot is a technically-focused AI that assists students and is planned to be conveniently automated in the form of a chatbot, which provides the answers to the commonly asked questions by students in terms of admissions, fees, eligibility to be placed, and the subjects they intend to enroll in. The system is built with a React frontend, FastAPI back-end, and a Google Gemini API with natural language processing to provide students with correct and live, contextual responses to queries on course syllabi and frequently asked questions of the institutions. Initial findings are encouraging, as the response time is shorter and satisfied with the users, the number of people burdened by the administration decreases, and they can be available 24 hours, 7 days a week, and deliver reliable information. The modularity allows updating and subsequent development of such features as multilingual support and retrieval-augmented generation.

This scalable modular chatbot system is a viable solution to enhancing the process of student support in a technical college setting. It provides a significantly better degree of access to information to students, as well as less staff time, by incorporating domain-specific AI functionality via Google Gemini, and an appropriate frontend and backend architecture. The chatbot design can be improved continuously, be closely integrated to the data of the institution, and significantly change the student-to-administrative interaction and efficiency to a digital, AI-facilitated one. Future improvements will involve the personalization, multi-turn dialogues, and wider language functionality to enhance user experience and institutional input.

Performance Evaluation

In order to test the initial functionality of the AI-based Student Assistance Chatbot, a small-scale test was performed on a group of ten representative queries representing the main areas of interest, including admissions, syllabus content, academic schedules, and general information about the university. The chatbot was deployed on the local machine with the FastAPI backend which was linked to the Gemini API, and queries were posted via the /ask endpoint.

The accuracy of the responses was gauged by the comparison of the output of the chatbot with the authoritative sources of institutional data which included the official syllabus, admission brochure and the FAQ documents. The relevance and facts of the answers were rated as a correctness score (on a scale of 0-100%).

The average time to respond (in seconds) to a query and provide an answer to it was documented as response latency and was measured with the help of the built-in timing logs of FastAPI and browser developer tools. The three tests were performed on each query to achieve consistency and the mean values calculated.

The results in Table I summarized indicate that the chatbot is efficient, and it is highly accurate and reasonably latent when answering academic-related questions. There were minor changes in performance based on stability of the network and the complexity of the question. The optimization to be done in future will be on caching of frequently asked questions and refinement of prompt design to provide quick and more accurate answers.

Table II : Response Accuracy and Latency Test

Query Type	Expected Source	Response Accuracy (%)	Average Latency (sec)	Remarks
Admission FAQs	University data	92	1.8	Quick and Accurate
Syllabus Queries	Gemini Model	85	2.3	Context partially Integrated
General Academic Queries	Gemini Model	88	2.0	Accurate with minor Paraphrasing
Non-domain Queries	Gemini Model	75	2.6	Less Domain relevance
Greeting/Small Talk	Local Responsible	100	0.5	Works smoothly

Comparison with the Existing Chatbots.

In order to emphasize the benefits of the proposed AI-poweredStudent Assistance Chatbot, a comparative analysis was made compared to two chatbot systems that are commonly used:traditional rule-based FAQ chatbots and general-purpose AI models like ChatGPT or Bard. The comparison centered on the main functional and technical parameters that apply to academic settings such as domain-specific accuracy, syllabus integration, context

retention, and personalization, deployment feasibility, and cost-efficiency.

The evaluation of each system was conducted on the basis of literature review, functional testing, and observable capabilities, in case of trial interactions. Chatbots based on rules were evaluated in terms of their capacity to answer pre-written frequently asked questions, whereas ChatGPT and Bard were evaluated on a series of sample queries related to the university to establish how well they could adapt to the context of technical education. The proposed chatbot was tested under the controlled conditions with the use of the same queries, and the response was analyzed in terms of accuracy, contextual relevance, and usability.

The results that have been summarized in Table II reveal that whereas traditional and general-purpose chatbots are effective in an open-domain setting, they do not have the specificity and domain knowledge needed in technical education. Conversely, the suggested chatbot demonstrates better domain specific accuracy, simpler integration with institutional data and greater scalability in implementation using Docker containers.

Table III : Comparison with Existing Chatbots

Parameter	*Traditional FAQ Chatbot*	*ChatGPT (General)*	*Proposed AI Chatbot*
Domain-Specific Accuracy	*Low*	*Moderate*	*High*
Syllabus Integration	*No*	*Partial*	*Yes*
Contest Retention	*Poor*	*Good*	*Good*
Deployment Feasibility	*High*	*Limited*	*High*
Personalization	*No*	*No*	*Planned*
Cost Efficiency	*High*	*Medium*	*High*

References

[1] C. Chan and H. Li, "Enhancing Higher Education with Generative AI: A Multimodal Approach for Personalized Learning," *IEEE Trans. Learn. Technol.*, vol. 18, no. 1, pp. 55–67, 2025.

[2] Y. Huang et al., "Self-Regulated Learning and Social Presence with Instructionally Aligned AI Chatbots," *IEEE Trans. Learn. Technol.*, vol. 18, no. 1, pp. 33–45, 2025.

[3] M. Kuhail et al., "Assessing the Impact of Chatbot-Human Personality Congruence on User Behavior," *IEEE Access*, vol. 12, pp. 1–12, 2024.

[4] X. Yin et al., "Using a Chatbot to Provide Formative Feedback to Students' Self-Assessments," *IEEE Trans. Learn. Technol.*, vol. 17, no. 2, pp. 120–132, 2024.

[5] A. Husain, M. Raza, and F. Khan, "Development of an Academic Services Chatbot Based on RAG," *Proc. IEEE Int. Conf. on Smart Education Systems*, pp. 98–104, 2024.

[6] Z. Dan, Y. Zhang, and K. Wang, "EduChat: A Large-Scale LLM-based Chatbot System for Intelligent Education," *Proc. Int. Conf. on Intelligent Systems and Applications*, pp. 221–229, 2023.

[7] J. Dong, X. Li, and Y. Chen, "How to Build an Adaptive AI Tutor for Any Course Using Knowledge Graph-Enhanced RAG," *Proc. Int. Conf. on Artificial Intelligence in Education*, pp. 301–310, 2023.

[8] X. Cao, L. Yang, and P. Chen, "AI Chatbots as Multi-Role Pedagogical Agents," *Proc. IEEE ICALT*, pp. 122–129, 2023.

[9] M. Sarker, S. Rahman, and A. Hoque, "Anglo–Bangla Language-Based AI Chatbot for University Admission Counseling," *Proc. IEEE CCCAI*, pp. 201–206, 2023.

[10] McKinsey & Company, "The State of AI Report," 2023. [Online]. Available: https://www.mckinsey.com

[11] P. Kumar and A. Sharma, "Smart Chatbot for Student Support," *IEEE Access*, vol. 10, pp. 12345–12356, 2022.

[12] S. Zhang et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *arXiv preprint arXiv:2107.07566*, 2021.

[13] IEEE, "Conversational AI: Chatbots," *Proc. IEEE CONIT*, pp. 1–5, 2021.

[14] R. Ramesh and V. Gupta, "AI-Driven Chatbots for Education: A Survey," *Int. J. Emerg. Technol. Learn.*, vol. 15, no. 20, pp. 1–13, 2020.

[15] A. Al-Ghuribi, M. A. Al-Sharafi, and N. M. Sharef, "Chatbot in Education: A Systematic Review," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 9, pp. 1–8, 2020.

[16] J. Pereira and J. Díaz, "Chatbot in Higher Education: Case Study," *EDUCAUSE Review*, pp. 1–6, 2019.

[17] J. Gao, M. Galley, and L. Li, "Neural Approaches to Conversational AI," *Found. Trends Inf. Retr.*, vol. 13, no. 2–3, pp. 127–298, 2019.

[18] R. Winkler and M. Söllner, "Unleashing the Potential of Chatbots in Education: A State-of-the-Art Analysis," in *Proc. IEEE EDUCON*, pp. 1–8, 2018.

[19] A. Vaswani et al., "Attention is All You Need," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, USA, pp. 5998–6008, 2017

3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

9 Not Cited or Quoted 3%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
2% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 9 Not Cited or Quoted 3%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
2% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
ijircce.com		<1%
2	Internet	
www.ijfmr.com		<1%
3	Publication	
Michael Pin-Chuan Lin, Daniel Chang. "CHAT-ACTS: A pedagogical framework for ...		<1%
4	Publication	
Xu Zhao, Guozhong Wang, Yufei Lu. "MDKAG: Retrieval-Augmented Educational Q...		<1%
5	Internet	
docplayer.net		<1%
6	Internet	
ijirt.org		<1%
7	Internet	
technodocbox.com		<1%

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.