TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

SAGARMATHA ENGINEERING COLLEGE

A

PROGRESS REPORT

ON

CHAT BASED COLLEGE RECOMMENDATION SYSTEM

BY

BISHWA KANDEL SEC078BCT009

RABI CHANDRA ARYAL SEC078BCT023

SHASHANK KATUWAL SEC078BCT032

SUMESH DHONJU SEC078BCT034

A PROPOSAL REPORT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

SANEPA, LALITPUR, NEPAL

August, 2025

ABSTRACT

The increasing volume of higher education institutions and the complexity of admission criteria make it challenging for students to identify the most suitable colleges based on their preferences and qualifications. This research proposes a College Recommendation Chatbot System to overcome these limitations. The system is designed to function on websites with the use of Artificial Intelligence (AI) aimed to enhance the user service and improving operational efficiency.

Developed using Python, FastAPI, MongoDB, and React, the chatbot leverages Natural Language Processing (NLP) techniques to understand user queries and provide personalized college suggestions based on parameters such as academic scores, location, stream of interest, and budget. The backend is powered by a scalable API that integrates MongoDB for efficient data management, while the frontend ensures an intuitive and responsive user experience.

By employing machine learning models and NLP, the system can comprehend a wide range of user inputs and dynamically adjust its responses. The proposed chatbot not only simplifies the college search process but also enhances user engagement by offering real-time, accurate, and context-aware recommendations. This approach significantly contributes to the digitalization of the college counseling process and promotes equitable access to academic opportunities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
DCG	Definite Clause Grammar
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IRF	Improved Random Forest
JS	JavaScript
JSON	JavaScript Object Notation
ML	Machine Learning
MHT-CET	Maharashtra Health and Technical Common Entrance Test
NLP	Natural Language Processing
NER	Named Entity Recognition
NoSQL	Not Only SQL
RAG	Retrieval-Augmented Generation
REST	Representational State Transfer
RDBMS	Relational Database Management System
UI	User Interface
UX	User Experience
URL	Uniform Resource Locator

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, pursuing education has become an important need in people's lives. Thus,a wide range of disciplines are offered by numerous colleges where students often find it overwhelming to make informed decisions about which institution and course to choose.

The decision-making process is influenced by several factors including location, affordability, entrance exam results, specialization interests, accreditation, faculty quality, and infrastructure. However, many students and parents lack access to centralized, reliable, and easy-to-understand information about these colleges and their offerings. Existing resources are often scattered, outdated, or too technical for the average student to use efficiently.

To address this gap, the development of a chatbot for college recommendation presents a promising solution. Chatbots, powered by Natural Language Processing (NLP), offer interactive and user-friendly interfaces that can simulate human-like conversation. They can efficiently provide personalized guidance based on user preferences and academic profiles.

1.2 Problem Definition

Choosing the right college is a critical decision for students after completing their secondary or higher secondary education. With numerous colleges offering diverse programs under different universities, process of selecting the most suitable institution can be confusing and overwhelming.

Students often struggle to access consolidated and accurate information regarding college rankings, program availability, entrance requirements, location, tuition fees, and facilities. The lack of centralized, easy-to-navigate platforms forces students to rely on informal sources, word of mouth, or scattered online data, which may not always be reliable or up to date. Additionally, many students and parents are not tech-savvy enough to extract relevant information from university websites or government portals.

Traditional career counseling services are limited, expensive, or inaccessible to students in rural and semi-urban areas. As a result, students may end up choosing colleges or programs that do not align with their interests, qualifications, or long-term goals.

1.3 Objectives

The objectives of this project are:

- To develop a conversational system capable of automating admission-related query handling through natural language understanding and intelligent response generation.

1.4 Features

The features of our project are as follows:

i. College Exploration: It allows users to browse and explore all the available collges using the Web and chatbot UI interface.

ii. Multilingual support: The system can interact with the users using multiple language offering a clear explanation.

iii. Graphical and Visual Representation : The Web UI displays different graphs of the available data .

1.5 Feasibility

1.5.1 Technical Feasibility

The proposed project is technically feasible as it leverages existing technologies and tools such as NLP techniques. The chatbot interface is implemented using React for web platforms, ensuring seamless user interaction through real-time communication with the backend powered by FastAPI.

1.5.2 Operational Feasibility

The proposed project is operationally feasible as it addresses the challenges and requirements faced by traditional methods for choosing a college. It automates the process through the chatbot system and aims to enhance the operational efficiency, reduce errors and save a lot of time and energy while providing a seamless and convenient experience to the users.

1.6 System Requirements

The system requirements of our project are:

1.6.1 Software Requirements

The software requirements for our project are as follows:

i. Natural Language Processing (NLP) Libraries: Libraries for NLP tasks like entity recognition, and intent classification.

ii. Backend Technologies : Python, Fast API for backend and managing requests/responses. MSSQL as the database for storing data, user queries, etc.

iii. Frontend Technologies: HTML , CSS and Javascript for creating a responsive, dynamic, and user-friendly interface.

iv. Version Control: Utilizing a version control system like Git for collaboration, tracking code changes, and ensuring code integrity.

v. Libraries of Python:The Python libraries required for various tasks include:NumPy: For numerical computations and array operations. PyTorch: For data visualization and plotting. Panda: For data manipulation and analysis.

1.6.2 Hardware Requirements

The hardware requirements for our project are as follows:

i. To run the application:
PC with minimum specifications such as a dual-core processor (Intel Core i3 equivalent), 4GB of RAM, and a storage capacity sufficient to accommodate the application and related files.

ii. To extract content from chatbot, handling various formats and structures.

1.6.3 Functional Requirements

i. College Exploration: The chatbot should allow users to explore colleges based on stream, rank, location, and other filters.

ii. Personalized Recommendations: Provide college suggestions based on student preferences, past choices, entrance exam scores, or budget.

iii. Information Retrieval: Display college details like cutoffs, fees, reviews, and courses offered.

iv. User Queries: Allow natural language queries such as "Best engineering colleges in Kathmandu" and return relevant results.

1.6.4 Non-functional Requirements

i. Usability and User Interface: The chatbot should have an interface that is visually appealing, easy to navigate, and responsive across different devices.

ii. Speed and Performance: The chatbot system should respond promptly to customer requests, ensuring minimal waiting time and efficient order processing.

iii. Reliability and Availability: The chatbot system should be reliable and available at all times, with a robust infrastructure to handle failures or high traffic situations.

iv. Security and Privacy: The chatbot system should prioritize the security and privacy of customer information, implementing encryption, authentication, and compliance with data protection regulations.

v. Integration: The chatbot system should seamlessly integrate with kitchen management, inventory management to ensure smooth information flow and coordination.

vi. Scalability: The chatbot system should be designed to handle future growth, allowing for increased user demand, menu expansions, and system upgrades without any significant impact on performance or functionality.

CHAPTER 2

LITERATURE REVIEW

The paper ‘Admission Fortune Using Machine Learning Designing an Interactive Chatbot for Educational Assistance’ is a chatbot used to assist students in making well- informed decisions about their college applications. It is a model that uses machine learning algorithms to estimate the probability of a student’s admission to different colleges based on their academic performance and the admission criteria of the colleges. This system helps students to strategize their college applications and increase their possibility of acceptance to the college that aligns with their preferences.[1]

The literature review of the research paper ‘The Educational Recommendation System with Artificial Intelligence Chatbot: A Case Study in Thailand’ is an educational recommendation system capable of providing educational recommendations and valuable information about the engineering degree program. The main objectives of this study are to develop the capacity to analyze educational issues and provide useful and accurate information for educational recommendations, as well as to explore the possibilities for enhancing this system. [2]

The research article ‘Enhancement in Selection Process of Institute by CET Score using Cutoff Prediction and Recommendation System with Integrated Chatbot’ proposes a system which predicts the future cutoffs for engineering colleges using machine learning models based on CET exam scores. The system allows tailored college recommendations based on individual preferences and performance, thus simplifying the decision-making process for the students. An integral part of this would be a chatbot built using Natural Language Processing (NLP) to assist the students by answering questions, explaining processes, and providing real-time information on available college options.[3]

The research article ‘EduChat: An AI-Based Chatbot for University-Related Information Using a Hybrid Approach’ focuses on a chatbot system for university related questions. It is an effective artificial intelligence application designed by combining rule-based methods, an innovative improved random machine learning approach, and ChatGPT to automatically answer common questions related to universities, academic programs, admission procedures, student life, and other related topics. It provides quick and easy information to users, thereby reducing the time spent searching for information directly from source documents or contacting support staff. [4]

The paper ‘EduAssist: Chatbot for Student Admission Queries’ presents an AI-powered chatbot using Retrieval-Augmented Generation (RAG) and NLP offers a solution by providing 24/7 instant, accurate, and personalized responses in multiple languages. It automates answers to common questions, enhancing access to key information, easing staff workload, and improving the overall user experience covering topics like admissions processes, eligibility, college details, fees, curriculums, scholarships, and placements. [5].

CHAPTER 3

RELATED THEORY

3.1 Chatbot

A chatbot is a software application used to conduct an online chat conversation via text or text-to-speech, in place of providing direct contact with a live human agent. Chatbots are generally designed to convincingly simulate human conversational behavior. They are used in dialog systems for various purposes, including customer service, request routing, or information gathering.

Some chatbot applications use extensive word classification and natural language processing (NLP), while others scan for general keywords and generate responses using common phrases stored in libraries or databases.

Benefits of Chatbots:

i. Improve lead generation

ii. Faster resolution

iii. Around-the-clock support

iv. Reduce customer service costs

v. Boost customer engagement

vi. Reduce human error

vii. Give customers what they want

viii. Always getting smarter

ix. Endless patience

3.2 HTML

HTML provides the foundational structure of web pages through a system of elements and tags. It defines the content of a page, such as headings, paragraphs, links, and images—using a standardized markup language. HTML forms the backbone of any webpage, ensuring proper organization of content within the browser.

HTML is not a programming language but a markup language. It uses a hierarchical structure of nested elements, often styled and manipulated through CSS and JavaScript respectively. Tags like `<div>`, `<p>`, and `<a>` are used to organize and link content semantically.

Among these, the `<body>` tag is critical. It contains all the visible content rendered on the page. All structural elements reside within it, forming the visible document tree that browsers interpret and render for the user.

3.3 CSS

CSS controls the visual presentation of HTML elements using selectors and declarations. It allows developers to separate style from structure, enabling consistent design across multiple pages. CSS can be applied inline, via internal style sheets, or through external files for better modularity.

CSS works by targeting HTML elements and applying styles such as color, layout, and typography. It uses a cascading system to determine which styles take precedence when multiple rules apply. Classes, IDs, and element selectors provide fine-grained control over styling.

3.4 JavaScript

JavaScript is a dynamic scripting language that enables interactivity and logic in web pages. It runs in the browser and can manipulate the DOM, respond to user input, and fetch data from servers asynchronously. JavaScript transforms static pages into rich, interactive experiences.

JavaScript operates by interacting with the browser's DOM and BOM (Browser Object Model). It can add, remove, or modify elements in real time, as well as handle events like clicks and form submissions. JavaScript supports functions, objects, and event-driven programming.

3.5 ASP .NET CORE

ASP.NET Core is a cross-platform, high-performance web framework developed by Microsoft for building modern, cloud-enabled, and internet-connected applications. It is an open-source framework that supports the development of web applications, APIs, and microservices.

ASP.NET Core follows the Model-View-Controller (MVC) architectural pattern, which separates application logic, user interface, and data handling. This separation enhances maintainability, scalability, and testability. It also supports Razor Pages and minimal APIs for more lightweight or page-focused development needs.

3.6 FastAPI

FastAPI is a modern, high-performance web framework for building APIs with Python 3.7+ based on standard Python type hints. It is designed to be fast to run and fast to code, making it ideal for creating RESTful APIs quickly. FastAPI automatically generates interactive API documentation using Swagger UI and ReDoc. It ensures data validation, serialization, and type checking through Pydantic models. The framework supports asynchronous programming, allowing for efficient handling of many requests. FastAPI is widely used for microservices and backend services due to its simplicity, performance, and developer-friendly features.

3.7 MS SQL

MSSQL is a relational database management system developed by Microsoft, designed to store, retrieve, and manage data efficiently. It uses Transact-SQL (T-SQL), an extension of SQL, to query and manipulate structured data across tables and relationships.

MSSQL is based on a client-server architecture, where the server processes and serves data to connected clients. Databases in MSSQL consist of tables, views, stored procedures, and functions that help organize and manage large-scale datasets securely and efficiently.

3.8 Artificial Intelligence

Artificial Intelligence (AI) is the foundation of chat-based recommendation systems. It enables machines to simulate human intelligence, including learning, reasoning, and decision-making. AI-driven systems interpret user intent, retrieve relevant data, and deliver personalized recommendations through conversational interfaces. This transforms static interactions into dynamic, tailored user experiences, enhancing decision accuracy and system efficiency.

3.8.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) that focuses on enabling machines to understand, interpret, generate, and respond to human language in a meaningful way. It bridges the gap between human communication and computer understanding. NLP techniques are widely used in applications such as chatbots, machine translation, sentiment analysis, speech recognition, and text summarization.

Natural Language Processing (NLP) enables computers to understand and interpret human language. The process involves multiple stages, each responsible for analyzing a specific aspect of the text. The steps of NLP are:

i. Lexical Analysis
   This is the first phase in NLP, where the input text is divided into tokens such as words and punctuation. It involves identifying meaningful elements in the text and eliminating invalid characters or meaningless symbols.

ii. Syntactic Analysis
   Also known as parsing, this step checks the text for grammatical structure. It verifies whether the sentence follows the rules of the language and creates a parse

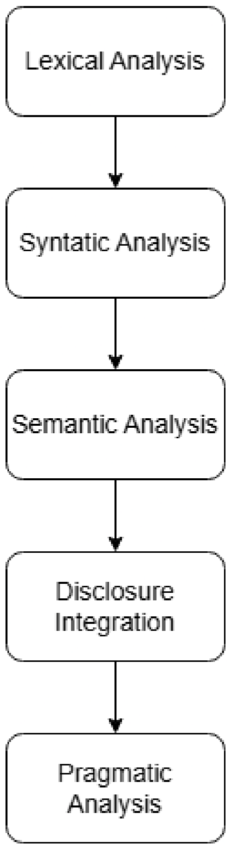


Figure 3.1: Block diagram for steps of NLP

tree that represents the syntactic structure of the sentence.

iii. Semantic Analysis

In this stage, the system derives the literal meaning of the sentence. It maps the syntactic structures into representations that capture the meanings of words and their relationships.

iv. Discourse Integration

This step ensures that the meaning of a sentence is coherent with the context of surrounding sentences. It integrates the current sentence with the previous discourse to maintain continuity and relevance.

v. Pragmatic Analysis

The final step involves interpreting the sentence in its real-world context. It looks beyond the literal meaning to understand the intended effect of the sentence, considering factors like speaker intention and context.

Each of these stages plays a vital role in enabling machines to understand and process human language accurately and effectively.

3.8.2 Named Entity Recognition(NER)

Named Entity Recognition (NER) is a subtask of Natural Language Processing (NLP) that focuses on identifying and classifying key elements or entities in unstructured text into predefined categories. The goal of NER is to extract structured information from text, enabling downstream applications such as information retrieval, question answering, recommendation systems, and data analytics. NER plays a crucial role in transforming raw text into structured, machine-readable data.

spaCy is a popular open-source library for advanced Natural Language Processing (NLP) in Python. One of its most powerful features is Named Entity Recognition (NER), which is the process of locating and classifying named entities in text into predefined categories. spaCy provides a pre-trained statistical model capable of recognizing named entities out of the box. The model uses deep learning techniques to make predictions based on the context of words. spaCy also allows users to customize NER by adding new entity types or training from scratch.

We selected spaCy instead of NLTK for this project because of its ease to train and fine tune NER models for custom datasets. It is designed for fast, efficient processing and offers a modular pipeline which can work together seamlessly.

3.8.3 TF-IDF vectorizer

A TF-IDF vectorizer is a powerful tool in natural language processing that transforms raw text into numerical feature vectors suitable for machine learning models. TF-IDF stands for **Term Frequency-Inverse Document Frequency**, which is a statistical measure used to evaluate how important a word is to a document in a collection. The vectorizer first calculates the term frequency (TF), which measures how often a word appears in a document, and the inverse document frequency (IDF), which reduces the weight of words that are common across many documents. By multiplying these two values, the TF-IDF score highlights words that are frequent in a specific document but rare across the entire dataset.

This approach is especially useful for tasks like intent recognition, where distinguishing between different types of user queries is important. The TF-IDF vectorizer creates a matrix where each row represents a document (or query) and each column represents a word, with the values indicating the importance of each word in each document. This representation allows machine learning models, such as Random Forest classifiers, to learn patterns in the data and make accurate predictions. By focusing on the most relevant words, TF-IDF helps improve the performance of text classification systems and reduces the influence of common, less informative words.

3.8.4 Random Forest

Random Forest is an ensemble learning method used for classification and regression tasks. It constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) of the individual trees. This approach reduces overfitting by averaging multiple trees, thereby improving accuracy and robustness. Random Forest uses random feature selection and bootstrap sampling to build diverse trees.

Given: Training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{1, 2, \ldots, C\}$.

1. Set hyperparameters:
    a) Number of trees: n_{trees}
    b) Number of features to sample at each split: m_{features}
    c) Minimum samples to split a node: s_{min}
    d) Maximum depth: d_{max}
2. For each tree T_j where $j = 1, 2, \ldots, n_{\text{trees}}$:
    a) Draw a bootstrap sample D_j by sampling N instances from D with replacement.
    b) Grow a decision tree T_j on D_j:
        i. At each internal node:
            - Randomly select m_{features} features from total d features.

- For each feature f, find the threshold θ_f that minimizes Gini impurity:

$$\text{Gini}(S) = 1 - \sum_{c=1}^{C} p_c^2, \quad \text{Gini}_{\text{split}} = \frac{|S_{\text{left}}|}{|S|}\text{Gini}(S_{\text{left}}) + \frac{|S_{\text{right}}|}{|S|}\text{Gini}(S_{\text{right}})$$

- Choose the split with minimum $\text{Gini}_{\text{split}}$

ii. Repeat until stopping criteria met: (e.g., $|S| < s_{\min}$ or depth $\geq d_{\max}$)

3. Prediction for a new input x_{new}:

   a) For each tree T_j, compute prediction $\hat{y}_j = T_j(x_{\text{new}})$

   b) Compute majority vote:

$$\hat{y} = \arg\max_c \sum_{j=1}^{n_{\text{trees}}} \nVdash[\hat{y}_j = c]$$

3.9 Existing Ideas

i. The existing system predicts future engineering college cutoffs using machine learning models trained on MHT-CET exam data. It offers personalized college recommendations based on students' scores and preferences. Additionally, an integrated NLP-based chatbot assists students by answering queries, explaining admission procedures, and providing real-time college information. This combination of predictive analytics and conversational AI simplifies the college selection process and enhances decision-making for applicants.

ii. The existing system of EduChat employs a hybrid approach that combines rule-based parsing, machine learning classification, and generative AI to provide accurate and contextually appropriate responses. The system begins by preprocessing and encoding the user's input to ensure consistency. It then attempts to interpret the input using Definite Clause Grammar (DCG) rules via a Prolog instance. If rule-based parsing fails, an Improved Random Forest (IRF) model predicts the user's intent. If the prediction's confidence exceeds a predefined threshold, the system accepts this classification. Upon identifying a task, the system retrieves a relevant response from its database. If no task is determined, it forwards the conversation to ChatGPT for a generative response. This multi-layered approach ensures that users receive accurate and helpful information, enhancing their experience when seeking university-related assistance.

iii. One system provides tailored guidance on engineering programs which offers real-time, location-independent access to educational information via mobile devices and integrates with popular social media platforms. The system accurately analyzes user queries to deliver relevant recommendations. While effective, further research is needed to expand its application and understand the skills influencing its use in diverse educational contexts.

iv. Another existing system is EduAssist which is an AI-powered chatbot using Retrieval-Augmented Generation (RAG) and NLP to handle student and parent inquiries during engineering admissions. It provides instant, accurate, and personalized 24/7 responses in multiple languages, reducing staff workload and improving access to information on admissions, fees, courses, scholarships, and placements. This addresses inefficiencies in traditional communication methods like calls, emails, and visits.

v. One of the existing systems referenced is a rank-based college selection prediction platform that uses machine learning to estimate a student's admission chances based on academic performance and college criteria. It helps students make informed decisions, strategize applications, and target colleges that align with their goals and preferences.

CHAPTER 4

METHODOLOGY

4.1 System Block Diagram

The block diagram of our system is as follows:

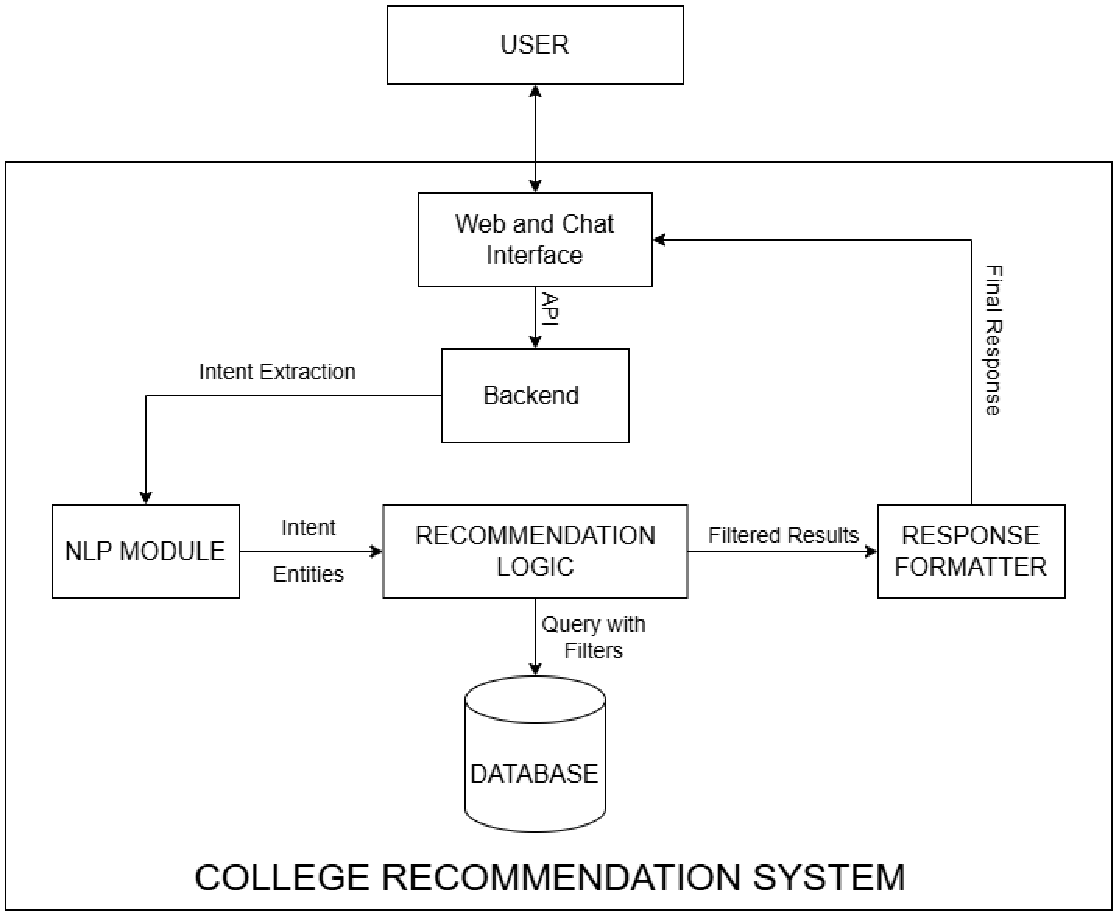


Figure 4.1: System Block diagram for College Recommendation System

The Chat-Based College Recommendation System follows a modular architecture for scalability and maintainability where users interact via a Chat UI. Queries are processed by a FastAPI backend and passed to an NLP Engine for intent and entity extraction. The Recommendation Logic uses this data to query a College Database. Results are formatted and returned to the user. Each module is independently deployable for easy scaling and updates.

4.2 Workflow for College Recommendation System

The workflow diagram of our system is as follows:

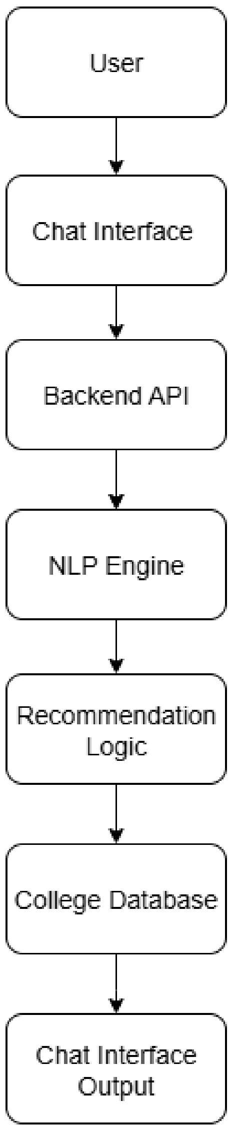


Figure 4.2: Workflow for College Recommendation System

The system workflow for the Chat-Based College Recommendation System is designed with modular components to ensure flexibility, scalability, and maintainability. At the top of the system, the User (typically a student or parent) interacts with the system through a Chat UI. This frontend interface captures user input in natural language and sends it to the backend system for processing.

The Backend API Layer, developed using FastAPI, acts as the central manager. Upon receiving the user query from the frontend, the backend forwards it to the NLP Engine for interpretation. This engine is responsible for Intent Extraction and Entity Recognition. It processes the raw text to extract structured information such as the intent (e.g., “college recommendation”) and parameters (e.g., course, location, budget).

Once the NLP engine returns the structured data, it is passed to the Recommendation Logic module. This component applies rule-based logic to generate a list of relevant colleges. To do so, it queries the College Database, which is implemented using a MSSQL solution. The database stores essential information including college names, course offerings, fee structures, locations, and other relevant attributes.

Each module in this architecture is independently deployable and replaceable, ensuring that improvements in one area (e.g., switching to a better NLP engine) do not disrupt the functionality of the others. This modular and event-driven design facilitates robust performance and easy future scaling.

4.3 Working of ChatBot

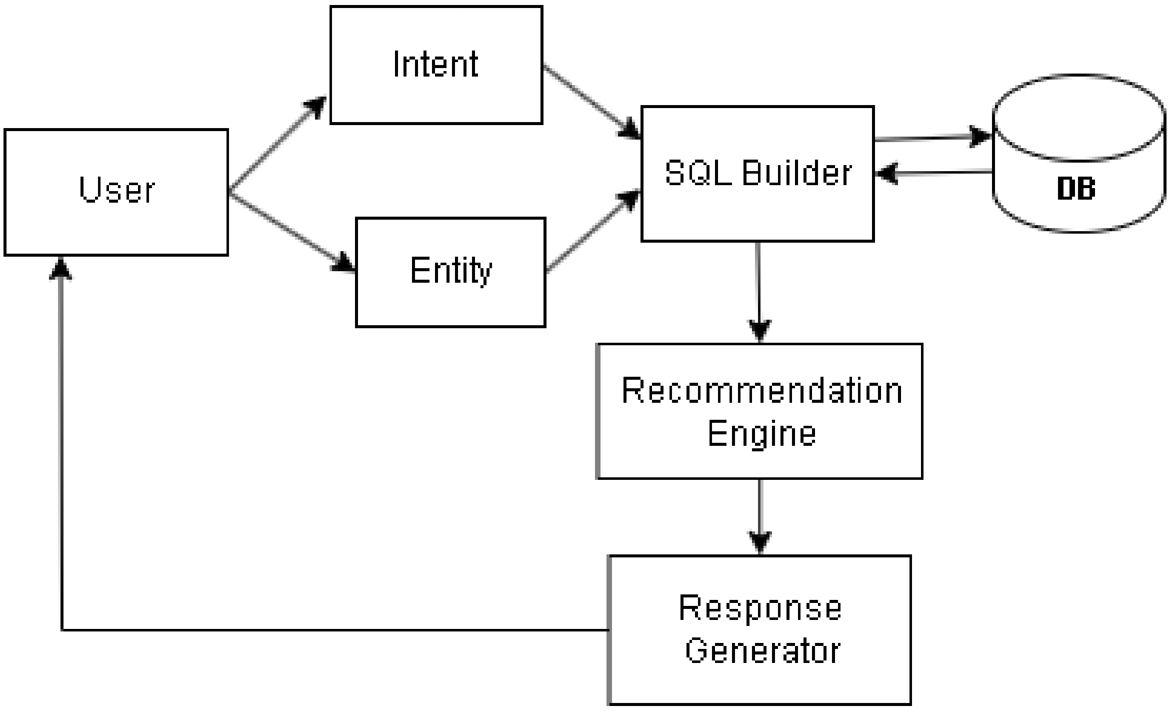


Figure 4.3: Working of ChatBot

The working of the chatbot, as shown in the figure, starts when the user sends a query through the interface. This query is first processed by the Intent Recognition module, which determines the purpose of the request, such as finding courses, retrieving department details, or suggesting resources. At the same time, the Entity Extraction module identifies important keywords or phrases from the user's input, such as subject names, department names, or other specific requirements. Both the detected intent and entities are then passed to the SQL Builder, which constructs a structured database query to fetch the relevant data from the Database (DB). Once the data is retrieved, it is sent to the Recommendation Engine, which filters, ranks, and personalizes the results based on user preferences, historical interactions, and other relevance criteria. The refined output is then passed to the Response Generator, which formats the information into a clear, user-friendly reply, possibly adding summaries, links, or related suggestions. Finally, the generated response is delivered back to the user, completing the interaction loop and enabling them to take further actions or refine their queries.

4.4 Use case Diagram

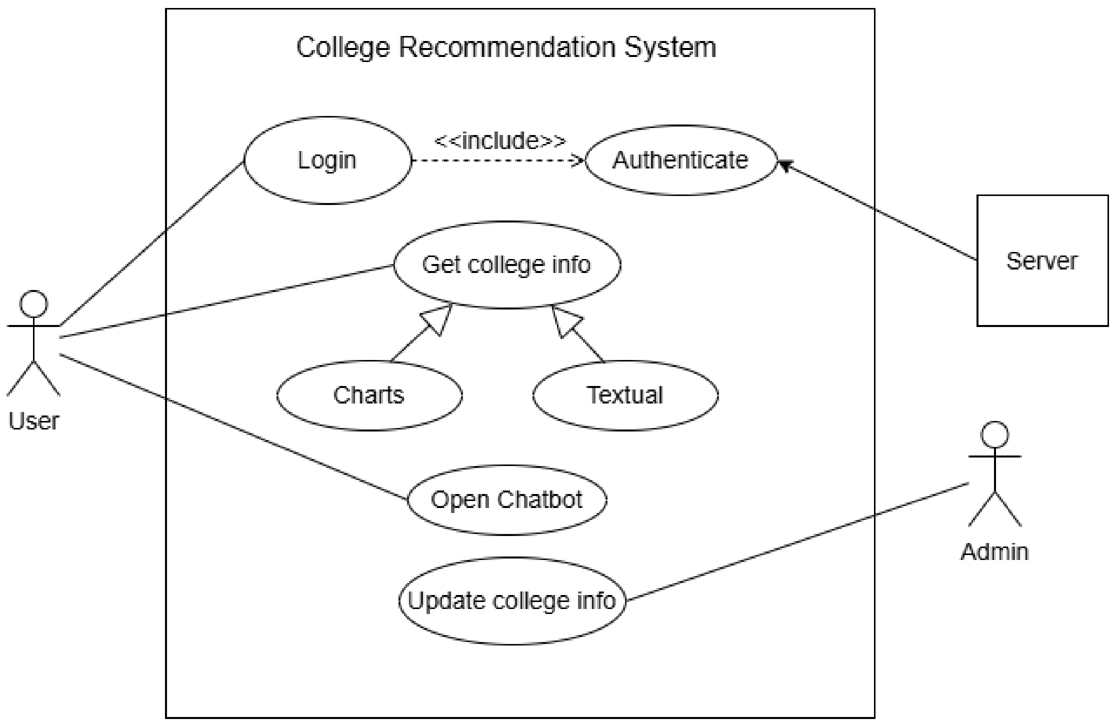


Figure 4.4: Use case diagram for College Recommendation System

The use case diagram for the College Recommendation System provides a high-level overview of the system architecture and key functionalities. The system is primarily designed for two types of users:students and administrators. Students interact with the system through a chat bot interface, which allows them to log in, view college information in both graphical and tabular formats, and open the chat bot for personalized guidance. The login process includes an authentication mechanism, which securely verifies user credentials by communicating with a centralized server. Administrators have exclusive access to the update college info feature, ensuring that the database remains current and relevant. This modular structure supports a dynamic and user-friendly interface, enabling students to explore and compare colleges efficiently while maintaining system integrity and data accuracy through admin-level controls.

4.5 ER Diagram

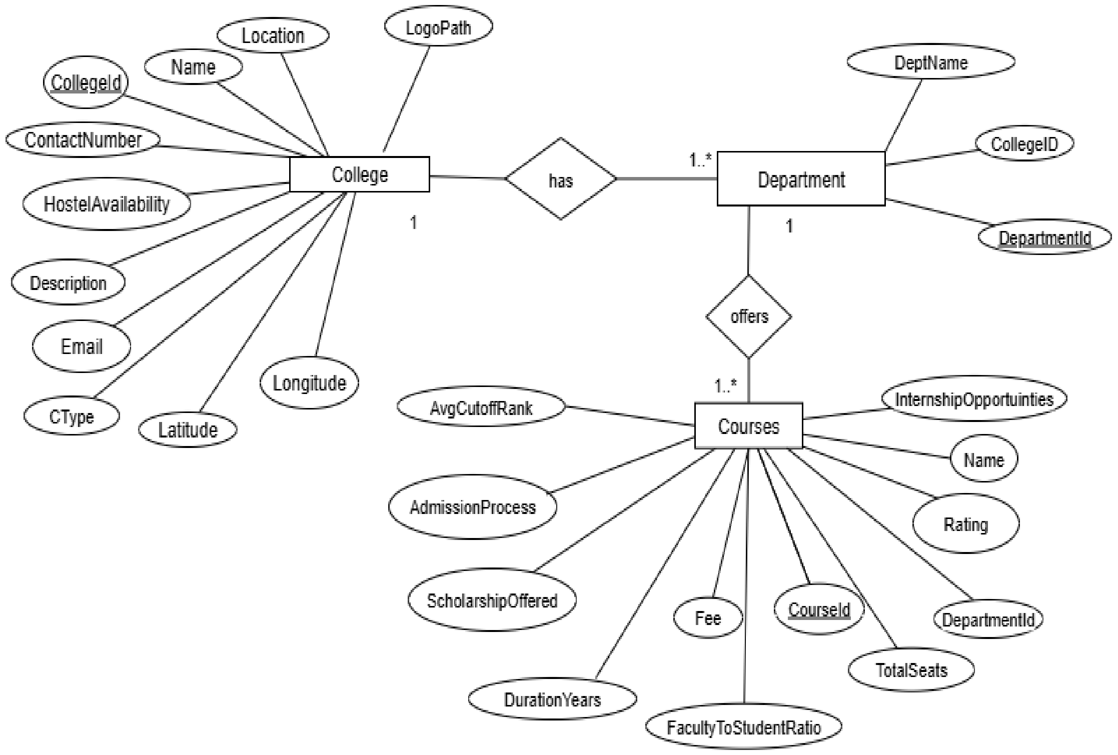


Figure 4.5: ER Diagram

This ER diagram represents the relationship between Colleges, Departments, and Courses within an educational system. A College entity includes attributes such as CollegeId, Name, Location, LogoPath, ContactNumber, HostelAvailability, Description, Email, CType, Latitude, and Longitude. Each College can have one or more Departments, where each Department has attributes like DeptName, CollegeID, and DepartmentID. Departments offer one or more Courses, which are detailed by attributes including CourseId, Name, DepartmentID, Rating, Admission Process, Average Cutoff Rank, Total Seats, Faculty to Student Ratio, Fee, Duration Years, and Scholarship Offered. The diagram illustrates a one-to-many relationship from College to Department and from Department to Courses, emphasizing that each college houses multiple departments, and each department provides multiple courses, including opportunities for internships

4.6 Dataset

4.6.1 Preparation of Dataset

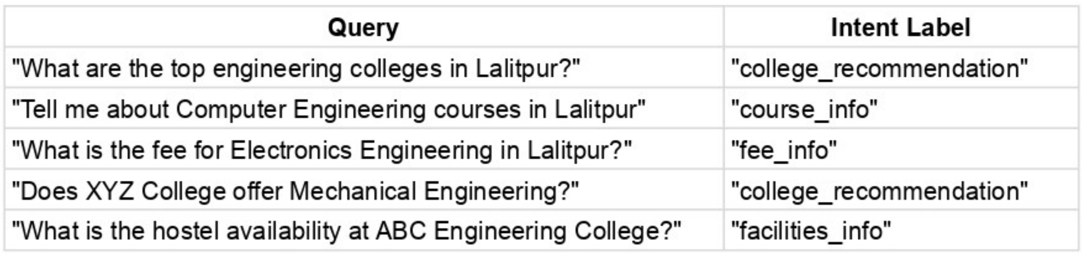
This dataset provides key details about various engineering colleges and their respective courses. It is designed to help students choose the best fit for their educational and career goals. The dataset therefore includes various columns that provide key details about engineering colleges and their courses with the detailed description of the columns given below.

The college table consists of College Name column represents the name of the engineering college or institution. The Location column indicates the geographic location of the college, including the city or town. Type indicates if the college is a private institution or a public institution. Also, the details of college such as ContactNumber, email, location are specified in this table.

The Department table identifies the academic department under which the course is categorized, such as Electronics and Computer. The Course table specifies the engineering course offered, such as Bachelors in Computer Engineering. The Average Cutoff Rank refers to the average rank required in an entrance exam for admission over the past years. The Fee (in NPR) column represents the annual fee for the course, stated in Nepalese Rupees (NPR). The Total Seats column shows the total number of seats available for the specific course. The Faculty to Student Ratio is the ratio of faculty members to students in the department or course. The Internship Opportunities column indicates whether the college offers internships for students. The Industry Collaboration column describes the companies or industries with which the college collaborates for research, internships, or placements. The Hostel Availability column indicates whether the college provides hostel accommodation for students. The Scholarships Offered column specifies whether the college offers merit-based or need-based scholarships. The Duration (years) column shows the duration of the course, which is typically 4 years for B.Tech programs. The Admission Process column details the entrance exam required for admission, such as the IOE Entrance Exam. Finally, the Reviews (Rating) column represents student or alumni reviews of the college, usually in the form of a rating out of 5.

4.6.2 Sample Dataset for Intent Classification

The following is a sample of the intent and it's respective label dataset:



Query	Intent Label
"What are the top engineering colleges in Lalitpur?"	"college_recommendation"
"Tell me about Computer Engineering courses in Lalitpur"	"course_info"
"What is the fee for Electronics Engineering in Lalitpur?"	"fee_info"
"Does XYZ College offer Mechanical Engineering?"	"college_recommendation"
"What is the hostel availability at ABC Engineering College?"	"facilities_info"

Figure 4.6: Snapshot of a Sample intent and it's respective label dataset

The Intent Classification Dataset consists of a collection of user queries paired with corresponding intent labels representing the user's underlying goal in the conversation. Each data point includes:

i. Input Utterance (User Query): A text string capturing the user's message, which can be multilingual. Examples include "What is the fee for Computer Science?" or "Show me colleges in Kathmandu."

ii. Intent Label: A predefined category indicating the specific intent behind the query. Common intent labels may include:

   a. fee_inquiry

   b. location_search

   c. course_availability

   d. admission_process

   e. hostel_info

   f. scholarship_query

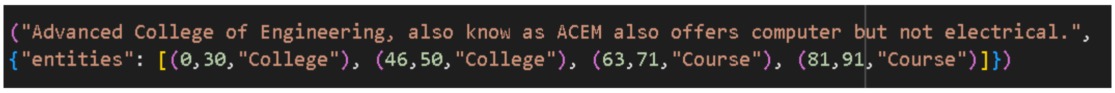
   g. fallback (for unrecognized queries)

4.6.3 Named Entity Recognition using spaCy

In this project, we utilized spaCy, a powerful NLP library in Python, to develop a custom Named Entity Recognition (NER) model. The methodology involved several key steps to prepare and structure the training data. First, a blank English NLP pipeline was initialized using spacy.blank("en"), and the NER component was added to the pipeline to enable entity recognition functionality.

The training dataset was structured as pairs of text samples and their corresponding entity annotations. Each annotation specified the start and end character positions of the entity in the text, along with the associated label (e.g., "COLLEGE", "COURSE"). For each training example, the raw text was converted into a Doc object using spaCy's tokenizer. Then, entity spans were created using the doc.char_span() method with alignment mode set to "contract" to handle any discrepancies between character-level annotations and token boundaries.

Only valid spans were included in the final list of entities for each document. These annotated Doc objects were collected into a DocBin object—a spaCy class designed for efficient serialization of documents. Finally, the compiled data was saved in .spacy binary format using the to_disk() method. This prepared dataset was later used to train a custom NER model capable of accurately identifying domain-specific entities in text.

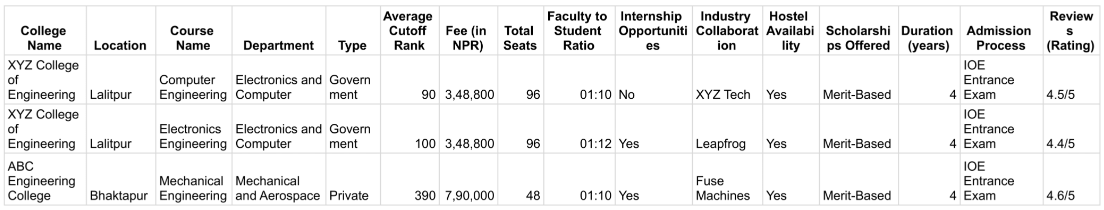


```
("Advanced College of Engineering, also know as ACEM also offers computer but not electrical.",
{"entities": [(0,30,"College"), (46,50,"College"), (63,71,"Course"), (81,91,"Course")]})
```

Figure 4.7: Training Model for NER using spaCy

4.6.4 Sample Dataset for College Information

The following is a sample of the College Information dataset, showing details for three colleges offering engineering courses:



College Name	Location	Course Name	Department	Type	Average Cutoff Rank	Fee (in NPR)	Total Seats	Faculty to Student Ratio	Internship Opportunities	Industry Collaboration	Hostel Availability	Scholarships Offered	Duration (years)	Admission Process	Reviews (Rating)
XYZ College of Engineering	Lalitpur	Computer Engineering	Electronics and Computer	Government	90	3,48,800	96	01:10	No	XYZ Tech	Yes	Merit-Based	4	IOE Entrance Exam	4.5/5
XYZ College of Engineering	Lalitpur	Electronics Engineering	Electronics and Computer	Government	100	3,48,800	96	01:12	Yes	Leapfrog	Yes	Merit-Based	4	IOE Entrance Exam	4.4/5
ABC Engineering College	Bhaktapur	Mechanical Engineering	Mechanical and Aerospace	Private	390	7,90,000	48	01:10	Yes	Fuse Machines	Yes	Merit-Based	4	IOE Entrance Exam	4.6/5

Figure 4.8: Snapshot of a Sample College Information dataset

Explanation of the Sample Dataset

The dataset provided above includes the following sample records:

1. XYZ College of Engineering (Lalitpur):

   i. Offers courses like Computer Engineering and Electronics Engineering.

   ii. Government institution with an average cutoff rank of 90 (for Computer Engineering).

   iii. Course fee is 3,48,800 NPR per year.

   iv. Hostel available for students and industry collaboration with companies like XYZ Tech and Leapfrog.

   v. Rated 4.5/5 by students.

2. ABC Engineering College (Bhaktapur):

   i. Offers *Mechanical Engineering*.

   ii. Private institution with a higher cutoff rank of 390.

   iii. Course fee is 7,90,000 NPR per year, with 48 seats.

   iv. Rated 4.6/5 by students, with internship opportunities and industry collaborations like Fuse Machines.

4.6.5 Ranking Criteria and Weight Assignment

To recommend colleges effectively, we assign a weight to each relevant feature based on its importance in the decision-making process.Here,Weights are assigned manually based on expert judgment and domain knowledge to reflect the relative importance of each feature in the recommendation process. This approach is suitable when historical data is unavailable or when the system must adhere to specific business priorities or policy guidelines. In this simplified example, we use only two features:

- Location (Distance in km) — A negative feature where a smaller value is better.
- Pass Rate (%) — A positive feature where a larger value is better.

The assigned weights are as follows:

Feature	Weight	Description
Location	0.40	Reflects the importance of geographical proximity (closer is better).
Pass Rate	0.60	Indicates a high priority for academic success and quality.

Table 4.1: Weights for features in this example

Example Dataset: We consider three colleges (A, B, C) with the following raw values:

College	Distance (km)	Pass Rate (%)
A	10	85
B	30	70
C	20	90

Table 4.2: Raw feature values for three example colleges

Normalization: We normalize values to the interval $[0, 1]$ so that all features are comparable.

For positive features (higher is better, e.g., Pass Rate):

$$s_{\text{pos}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

For negative features (lower is better, e.g., Distance):

$$s_{\text{neg}} = \frac{x_{\text{max}} - x}{x_{\text{max}} - x_{\text{min}}}$$

First, we find:

$$\text{Distance: } x_{\min} = 10, \quad x_{\max} = 30$$

$$\text{Pass Rate: } x_{\min} = 70, \quad x_{\max} = 90$$

Step 1: Normalize Location (negative feature)

$$\begin{aligned}
s_{\text{loc}}(A) &= \frac{30-10}{30-10} = \frac{20}{20} = 1.00 \\
s_{\text{loc}}(B) &= \frac{30-30}{20} = 0.00 \\
s_{\text{loc}}(C) &= \frac{30-20}{20} = \frac{10}{20} = 0.50
\end{aligned}$$

Step 2: Normalize Pass Rate (positive feature)

$$\begin{aligned}
s_{\text{pass}}(A) &= \frac{85-70}{20} = 0.75 \\
s_{\text{pass}}(B) &= \frac{70-70}{20} = 0.00 \\
s_{\text{pass}}(C) &= \frac{90-70}{20} = 1.00
\end{aligned}$$

Final Score Calculation The overall score for each college is:

$$S_r = w_{\text{location}} \cdot s_{\text{loc}} + w_{\text{pass\_rate}} \cdot s_{\text{pass}}$$

College A:

$$S_A = 0.40 \cdot 1.00 + 0.60 \cdot 0.75 = 0.40 + 0.45 = 0.85$$

College B:

$$S_B = 0.40 \cdot 0.00 + 0.60 \cdot 0.00 = 0.00$$

College C:

$$S_C = 0.40 \cdot 0.50 + 0.60 \cdot 1.00 = 0.20 + 0.60 = 0.80$$

Ranking Result:

Ranking the colleges in descending order of S_r:

$$A\,(0.85)\ >\ C\,(0.80)\ >\ B\,(0.00)$$

This method ensures that both proximity and academic performance are fairly compared and combined according to the specified weights.

4.7 SQL Builder

The SQL builder is a core component that translates user requirements expressed as intent and entities into SQL queries for retrieving relevant data from the college database. Its main responsibilities are:

a) Mapping User Needs to SQL Syntax:

   i) The system receives an intent (what the user wants) and entities (specific criteria like location, type, course, etc.).

   ii) It uses these to construct a SQL query string, adding appropriate WHERE conditions for each entity.

b) Dynamic Query Construction:

   i) The SQL builder starts with a base query that joins the necessary tables (College, Department, Courses).

   ii) For each entity, it appends a condition to the WHERE clause.

   iii) The intent may also influence the ORDER BY clause (e.g., sorting by fee for affordability).

c) Query Execution and Data Handling:

   i) The generated SQL query and its parameters are stored in local variables within the code.

   ii) These are passed to the database connector, which executes the query and fetches results.

iii) The results are converted into structured objects for further processing (e.g., ranking in the recommendation engine).

Example:

a) User Input

   Suppose the user asks:

   `Show me private engineering colleges in Lalitpur with hostel.`

b) Intent and Entities Extraction

   i) Intent: `find_affordable_college`

   ii) Entities:

      a) LOCATION: [Lalitpur]

      b) TYPE: [PRIVATE]

      c) HOSTEL: [YES]

c) SQL Query Generation

   The SQL builder constructs the following query:

```
SELECT ... FROM College c
LEFT JOIN Department d ON c.CollegeId = d.CollegeId
LEFT JOIN Courses co ON d.DepartmentId = co.DepartmentId
WHERE co.CourseId IS NOT NULL
  AND c.Location LIKE '%Lalitpur%'
  AND c.Type = 'PRIVATE'
  AND c.HostelAvailability = TRUE
ORDER BY co.Fee ASC
```

   Each entity adds a condition to the WHERE clause. The intent determines the sorting (ORDER BY fee for affordability).

d) Query Storage and Execution

i) The query string and its parameters are stored in local variables (`sql_query`, `params`) inside the function.

ii) They are not saved to a file or database; they exist only in memory during execution.

iii) The query is passed to the SQL builder execution method:

```
sql_results = self.db_extractor.get_colleges_by_filters(
sql_query, parameters)
```

iv) The results are then converted to objects and sent to the recommendation engine.

The SQL builder dynamically generates and executes SQL queries based on user intent and entities, storing the query in memory and passing the results directly to the recommendation engine for ranking and final output.

4.8 Algorithms

4.8.1 Intent Generation using the Random Forest model and TF-IDF:

a) Collect and label data: Gather user queries and assign each query an intent class label.

b) Text vectorization: Convert the text queries into numerical feature vectors using TF-IDF.

c) Data splitting: Divide the dataset into training and validation sets.

d) Model initialization: Set up the Random Forest classifier with chosen hyperparameters.

e) Model training: Train the Random Forest model on the training data.

f) Model evaluation: Evaluate model accuracy and other metrics on the validation set.

g) Save model and vectorizer: Store the trained Random Forest model and TF-IDF vectorizer for future use.

h) Predict intent for new queries: Transform new user queries using the saved TF-IDF vectorizer.

i) Intent prediction: Use the trained Random Forest model to predict the intent class of the transformed query.

j) Return result: Send the predicted intent to the chatbot or application for further processing.

4.8.2 Algorithm for Named Entity Recognization using spaCy

i. Import: Import the spacy library.

ii. Load: Load a pre-trained spaCy language model.

iii. Input: Provide the input text for entity recognition.

iv. Process:

   a. Process the text with the loaded NLP model to create a Doc object.

   b. Iterate over the doc.ents to retrieve entity text and labels.

v. Output: Store or display the extracted entities and their corresponding labels.

vi. Fine-tune: Train or fine-tune the model with custom entity labels and annotated data.

4.8.3 Workflow for Named Entity Recognization using spaCy

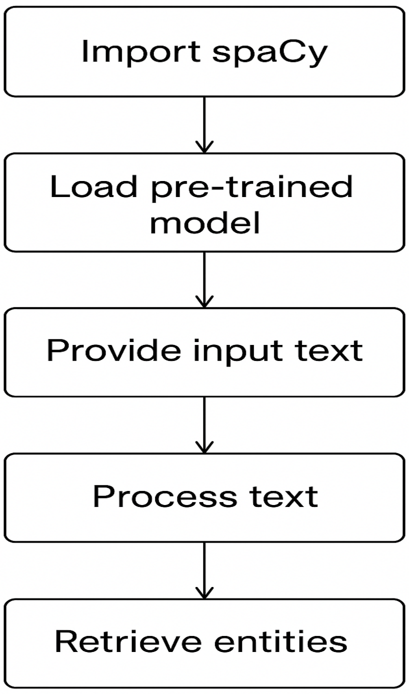


Figure 4.9: Workflow for Named Entity Recognization using spaCy

4.8.4 Recommendation Logic

The recommendation logic determines the ranking of colleges based on extracted user intent, entities, and preferences. This process integrates filtering, normalization, and weighted scoring to produce personalized results.

a) Intent and Entity Extraction: Let the detected intent be I (e.g., `find_college`), extracted entities be E (e.g., $\{$location: "Kathmandu", max_fee $: 500000\}$), and user preference be P (e.g., compare by location or overall recommendation). If $I \neq$ `find_college`, the system returns "Recommendation logic not applicable".

b) Query Construction: Build an SQL query Q using E as filters. For example:

   $Q =$ `SELECT * FROM College WHERE Location = 'Kathmandu' AND Fee <= 500000;`

   Execute Q to retrieve the dataset D.

c) Recommendation Modes:

   i. *Single-feature comparison:* If the user specifies a single comparison criterion

f_j:

- If f_j is a "positive" feature (higher is better), sort D in descending order of f_j.
- If f_j is a "negative" feature (lower is better), sort D in ascending order of f_j.

ii. *Overall recommendation:* Let the set of features be $F = \{f_1, f_2, \ldots, f_n\}$, with corresponding manually assigned weights:

$$w_1, w_2, \ldots, w_n, \quad \text{such that} \quad \sum_{j=1}^{n} w_j = 1.$$

Normalization with Direction Handling: For each feature f_j:

- If f_j is positive (higher is better):

$$s_{f_j}(r) = \frac{x_{r,j} - x_{j,\min}}{x_{j,\max} - x_{j,\min}}$$

- If f_j is negative (lower is better):

$$s_{f_j}(r) = \frac{x_{j,\max} - x_{r,j}}{x_{j,\max} - x_{j,\min}}$$

where $x_{r,j}$ is the raw value of feature f_j for record r, and $x_{j,\min}$, $x_{j,\max}$ are the minimum and maximum values of f_j in D.

Score Calculation: For each college $r \in D$:

$$S_r = \sum_{j=1}^{n} \left(s_{f_j}(r) \cdot w_j \right)$$

Append S_r to each record and sort D in descending order of S_r.

d) Top-N Selection: Choose the top N colleges as R_{top} and pass them to the response generator.

e) Output: Present R_{top} as a user-friendly ranked list.

Example Workflow:

a) Extracted entities: `{"location": "Kathmandu", "max_fee": 500000}`

b) SQL Query: `SELECT * FROM College WHERE Location = 'Kathmandu' AND Fee <= 500000;`

c) Dataset D: Colleges matching the filters.

d) Normalize features (*location similarity*, *pass rate*, *fee*) to $[0, 1]$ scale, accounting for whether higher or lower is better.

e) Apply weights and compute total scores S_r.

f) Sort by S_r and select top N.

g) Return the ranked list to the user.

4.8.5 Algorithm for Fallback Detection with Softmax Confidence Thresholding

i. Input: User query in text form.

ii. Preprocess: Tokenize and encode using the model's tokenizer.

iii. Predict: Pass the encoded input to the classifier to get raw logits.

iv. Apply Softmax: Convert logits to probabilities using

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}.$$

v. Check Confidence:

   a. Get the highest probability from the softmax output.

   b. If confidence $\geq$ threshold (e.g., 0.7), return the predicted intent.

   c. Else, return "unknown" or trigger a fallback response.

vi. Output: Intent label or fallback.

4.8.6 Flowchart for Fallback Detection with Softmax Confidence Thresholding

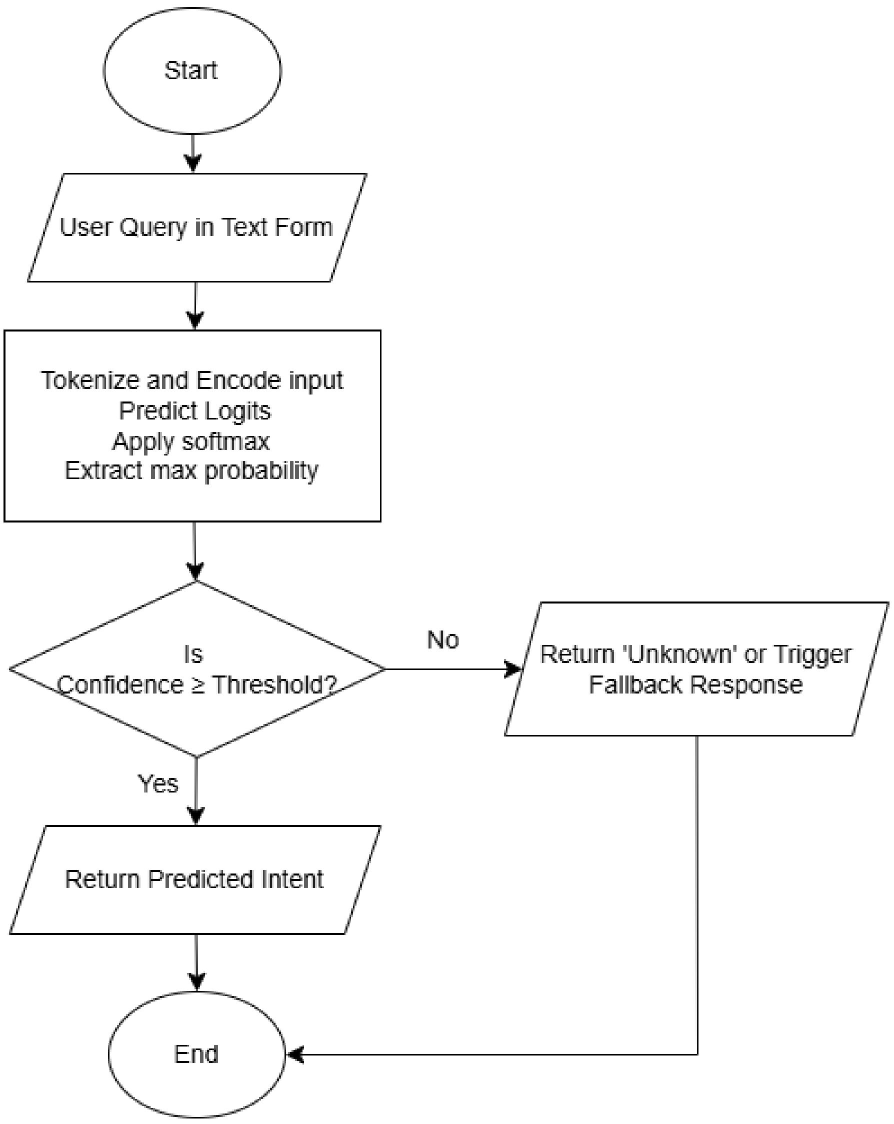


Figure 4.10: Flowchart for Fallback Detection with Softmax Confidence Thresholding

4.9 Evaluation Metrics

I. Accuracy

Measures the overall proportion of correct predictions out of all predictions.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

II. Precision

Measures how many of the predicted intent classes were actually correct.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

III. Recall (Sensitivity)

Indicates how many of the actual intent classes were correctly identified.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

IV. F1 Score

The harmonic mean of Precision and Recall, useful for imbalanced datasets.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

V. Confusion Matrix

A tabular representation showing true versus predicted classes, helpful for analyzing where the model makes mistakes.

4.10 Software Development Life Cycle

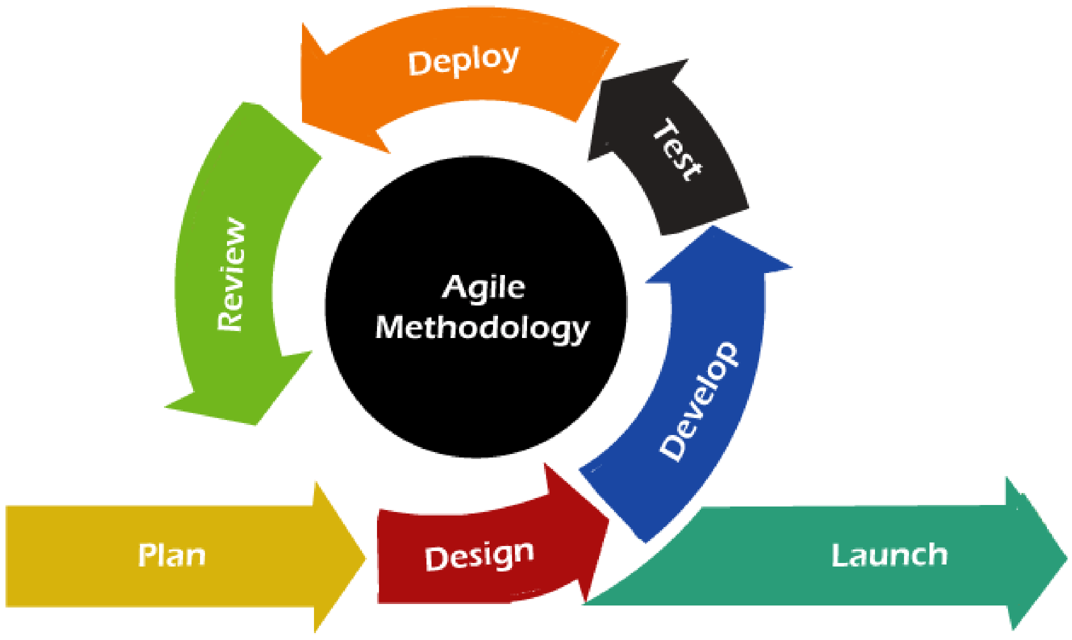


Figure 4.11: Agile Model

The Agile Model is a flexible and iterative approach to software development that focuses on delivering small, working increments of a project over short development cycles known as sprints. Instead of trying to build the entire system from start to finish in a single stretch, Agile divides the work into manageable chunks that are planned, developed, and tested continuously. This allows teams to adapt quickly to changes in requirements, user needs, or technology, and to incorporate feedback early and often throughout the development process.

Agile is particularly well-suited for the development of the Chat-Based College Recommendation System. Since the system involves natural language processing, multilingual understanding, and user interaction, it is highly dynamic and prone to change as more data is collected and user behaviors are better understood. For example, the types of questions users ask, the way they phrase them in different languages, and their preferences when choosing a college can vary widely. Agile allows the development team to respond to these evolving needs quickly by iteratively refining the chatbot's NLP engine, training data, and recommendation logic.

Furthermore, the project's modular design , consisting of components like the frontend chat UI, backend API, NLP engine, recommendation logic, and database—aligns well with Agile's incremental development strategy. Each module can be built, tested, and improved independently within separate sprints, allowing parallel development and faster progress.

CHAPTER 5

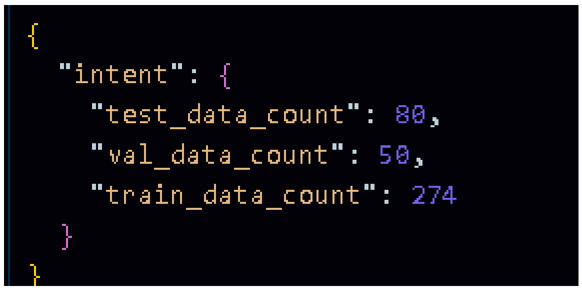
WORK PROGRESS

5.1 Work Breakdown Hierarchy

5.1.1 Work Accomplished

i. Data Collection for Entity and Intent Training

For both entity recognition and intent classification, we collected a diverse set of conversational data relevant to our chatbot's domain. This involved gathering user queries and annotating them with both intent labels (such as booking, inquiry, or cancellation) and entity tags (like names, dates, and locations). By ensuring the dataset included a wide range of examples, variations in phrasing, and edge cases, we created a robust foundation for training models that accurately identify user intent and extract key entities from conversations. This comprehensive data collection process is essential for building a chatbot that understands user needs and responds effectively.

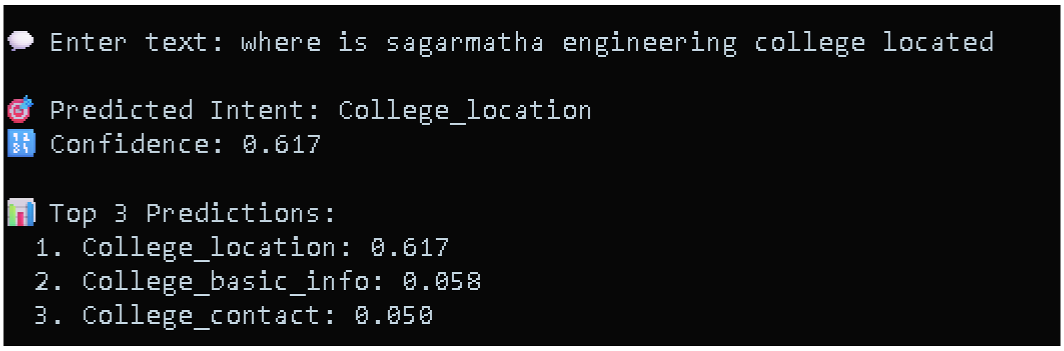


```
{
  "intent": {
    "test_data_count": 80,
    "val_data_count": 50,
    "train_data_count": 274
  }
}
```

Figure 5.1: Count od Data used on the Intent Training

ii. Intent Recognition

Here, the intent recognition system is built using machine learning models trained on user query data. Text inputs are vectorized and processed by the classifier, which predicts the intent label. The workflow includes data preparation, model training, and prediction scripts, enabling automated understanding of user requests within the chatbot framework. We can observe the output from the NER as

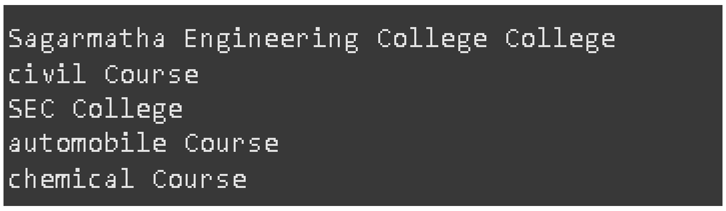


```
💬 Enter text: where is sagarmatha engineering college located

🎯 Predicted Intent: College_location
🔢 Confidence: 0.617

📊 Top 3 Predictions:
  1. College_location: 0.617
  2. College_basic_info: 0.058
  3. College_contact: 0.050
```

Figure 5.2: Output of Intent

iii. Named Entity Recognition

From NER using spaCy, we made a model to detect two entities which are 'course' and 'college' from a given text. For example, in the text 'Sagarmatha Engineering College has civil, computer and electronics. But SEC does not have automobile and chemical.' , we can observe the output from the NER as



```
Sagarmatha Engineering College College
civil Course
SEC College
automobile Course
chemical Course
```

Figure 5.3: Output of NER

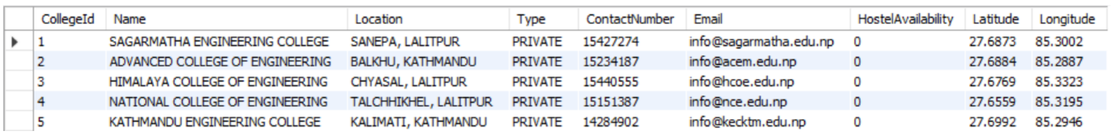
iv. UI and Frontend Design

Implemented the frontend of the application using ASP.NET Core MVC, incorporating HTML, CSS, and JavaScript to build a responsive and user-friendly interface. Utilized Razor views to integrate dynamic content rendering, and ensured a clean, structured layout aligned with the overall design requirements. The UI components were designed to support future integration with backend logic and database-driven content.

v. Data Collection and Database Creation

The database that stores the information about various colleges, their respective departments with the course offered by them were created as:

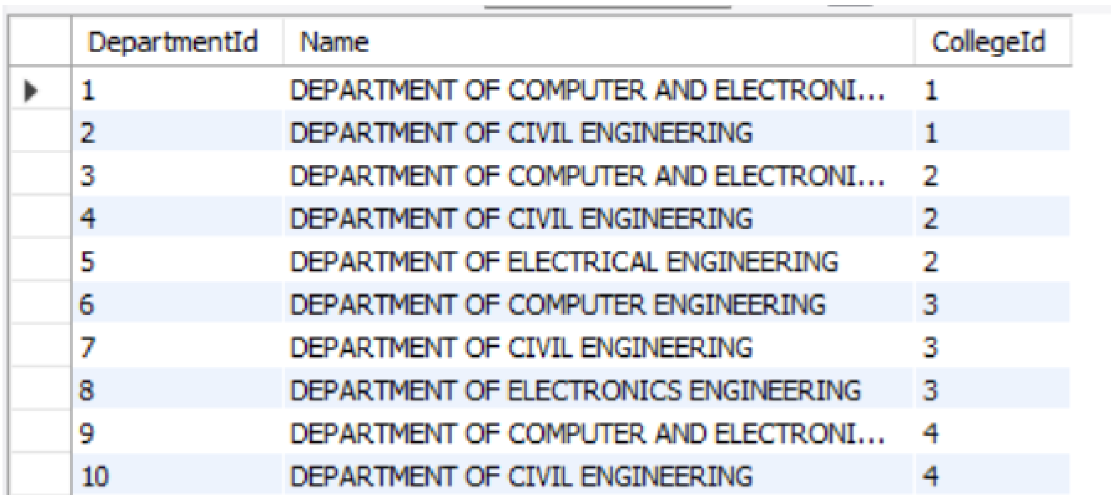
The table at *Figure 5.4* shows the college database,stores information about each college, including its name, location, type (public/private), contact details, and



CollegeId	Name	Location	Type	ContactNumber	Email	HostelAvailability	Latitude	Longitude
1	SAGARMATHA ENGINEERING COLLEGE	SANEPA, LALITPUR	PRIVATE	15427274	info@sagarmatha.edu.np	0	27.6873	85.3002
2	ADVANCED COLLEGE OF ENGINEERING	BALKHU, KATHMANDU	PRIVATE	15234187	info@acem.edu.np	0	27.6884	85.2887
3	HIMALAYA COLLEGE OF ENGINEERING	CHYASAL, LALITPUR	PRIVATE	15440555	info@hcoe.edu.np	0	27.6769	85.3323
4	NATIONAL COLLEGE OF ENGINEERING	TALCHHIKHEL, LALITPUR	PRIVATE	15151387	info@nce.edu.np	0	27.6559	85.3195
5	KATHMANDU ENGINEERING COLLEGE	KALIMATI, KATHMANDU	PRIVATE	14284902	info@kecktm.edu.np	0	27.6992	85.2946

Figure 5.4: College database

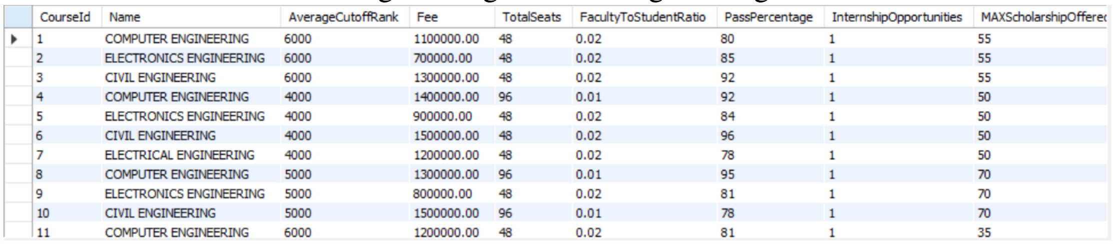
geographic coordinates. Each college has a unique CollegeId as the primary key. Hostel availability is indicated as a boolean value. This table serves as the foundation for linking departments and courses.



DepartmentId	Name	CollegeId
1	DEPARTMENT OF COMPUTER AND ELECTRONI...	1
2	DEPARTMENT OF CIVIL ENGINEERING	1
3	DEPARTMENT OF COMPUTER AND ELECTRONI...	2
4	DEPARTMENT OF CIVIL ENGINEERING	2
5	DEPARTMENT OF ELECTRICAL ENGINEERING	2
6	DEPARTMENT OF COMPUTER ENGINEERING	3
7	DEPARTMENT OF CIVIL ENGINEERING	3
8	DEPARTMENT OF ELECTRONICS ENGINEERING	3
9	DEPARTMENT OF COMPUTER AND ELECTRONI...	4
10	DEPARTMENT OF CIVIL ENGINEERING	4

Figure 5.5: Department database

The *Figure 5.5*,Contains details of departments within each college, such as department name and associated CollegeId. DepartmentId is the primary key and links each department to a specific college. Supports cascading updates and deletes to maintain referential integrity. Departments represent academic divisions like Computer Engineering or Civil Engineering.



CourseId	Name	AverageCutoffRank	Fee	TotalSeats	FacultyToStudentRatio	PassPercentage	InternshipOpportunities	MAXScholarshipOfferec
1	COMPUTER ENGINEERING	6000	1100000.00	48	0.02	80	1	55
2	ELECTRONICS ENGINEERING	6000	700000.00	48	0.02	85	1	55
3	CIVIL ENGINEERING	6000	1300000.00	48	0.02	92	1	55
4	COMPUTER ENGINEERING	4000	1400000.00	96	0.01	92	1	50
5	ELECTRONICS ENGINEERING	4000	900000.00	48	0.02	84	1	50
6	CIVIL ENGINEERING	4000	1500000.00	48	0.02	96	1	50
7	ELECTRICAL ENGINEERING	4000	1200000.00	48	0.02	78	1	50
8	COMPUTER ENGINEERING	5000	1300000.00	96	0.01	95	1	70
9	ELECTRONICS ENGINEERING	5000	800000.00	48	0.02	81	1	70
10	CIVIL ENGINEERING	5000	1500000.00	96	0.01	78	1	70
11	COMPUTER ENGINEERING	6000	1200000.00	48	0.02	81	1	35

Figure 5.6: Course database

The *Figure 5.6*, lists all courses offered by departments, including course name, cutoff rank, fee, seats, and other academic metrics. CourseId is the primary key, and each course is linked to a department via DepartmentId. Includes details like duration, admission process, rating, and scholarship opportunities. Captures essential information for students evaluating academic programs.

5.1.2 Work On Progress

i. Named Entity Recognition
For NER, we are training the model to detect other entities like location, entrance score, category of the college (private, government or community) and some other useful entities on the model.

ii. Integration of Frontend and Backend
Currently, the website is entirely static, meaning that all content is hard-coded and not dynamically retrieved from a backend system. The next phase of development involves integrating the necessary backend functionality to fetch and display data from a database.

iii. Pipeline Development
We are currently working on creating a processing pipeline for the chatbot. This involves designing a sequence of steps that automatically handle user queries, starting from text preprocessing, followed by entity extraction, and then intent classification. Each component is being developed to work together efficiently, allowing the chatbot to understand and respond to user inputs more accurately. The pipeline is structured to be modular, so improvements and new features can be added easily as the project evolves.

5.1.3 Work Remaining

i. Intent Handling

   The remaining work involves integrating an intent classification model into chatbot pipeline, mapping detected intents to predefined actions and designing the appropriate response generation logic. It will ensure that the chatbot not only recognizes important entities in user queries but also understands the underlying intention, enabling it to deliver accurate, context-aware, and task-oriented responses.

ii. Recommendation Engine

   A recommendation engine will help the chatbot suggest courses, departments, or resources based on the user's needs. It will use user preferences, available college and course data, and machine learning. Regular updates based on user feedback will make the suggestions more accurate.

iii. Pipeline Improvement

   The chatbot's process from receiving a question to giving an answer should be faster and more accurate. This includes improving steps like cleaning data, finding key details, and understanding user intent. Breaking the process into smaller parts and testing them automatically will make it easier to find and fix issues. Adding new features and checking performance regularly will keep the chatbot efficient and reliable.

iv. Response Generator

   The response generator, which will convert the recommendation engine's results into clear and user-friendly replies, is still to be implemented. This module will ensure that the chatbot's answers are not only correct but also well-structured, readable, and tailored to the user's query. To achieve this, we plan to design a set of dynamic templates for different response types (e.g., course suggestions, department information, resource links) and integrate them with the chatbot's output. The system will automatically fill these templates with data from the recommendation engine, while also adjusting tone and detail based on the query context. This approach will provide consistency, reduce errors, and make future updates easier to manage.

CHAPTER 6

EXPECTED OUTPUT

The chatbot is expected to provide accurate and personalized college recommendations based on user input such as academic performance, preferred location, stream, and budget. It should interact smoothly with users through a responsive and intuitive web interface. Upon entering relevant details, users will receive acurate list of colleges with information like eligibility criteria, fees,location , hostel availability and many more. The system will also suggest alternatives based on similar profiles. Additionally, the chatbot should handle multiple queries, provide real-time multilingual responses, and adapt to varying user intents using NLP, ensuring an efficient and user-friendly college exploration experience.

REFERENCES

[1] Patil Prathmesh Oswal Sujal P. R. Shahane, Pedgulwar Sumit. Admission fortune using machine learning designing an interactive chatbot for educational assistance. 2024.

[2] Pinanta Chatwattana, Piyada Yangthisarn, and Areeya Tabubpha. The educational recommendation system with artificial intelligence chatbot: A case study in thailand. *International Journal of Engineering Pedagogy (iJEP)*, 14(5):pp. 51–64, Jun. 2024.

[3] *African Journal of Biomedical Research*, 27(4S):5757–5767, Dec. 2024.

[4] Hoa Dinh and Thien Khai Tran. Educhat: An ai-based chatbot for university-related information using a hybrid approach. *Applied Sciences*, 13(22), 2023.

[5] Lokesh Deshmukh Nishica Kothawade Manisha Mali Sahil Baviskar, Kshitija Deshmukh. Eduassist: Chatbot for student admission queries. 2024.

APPENDIX A

APPENDIX

A.1 Gantt Chart

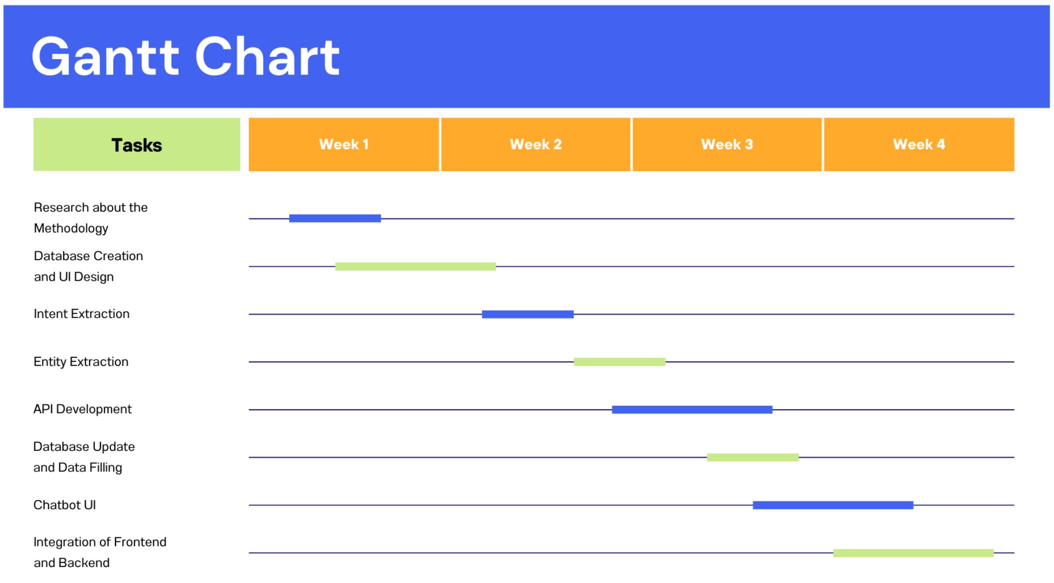


Figure A.1: Gantt Chart for College Recommendation System