

# **JusLab - AI-Powered PGDIT Assistant (Custom chatbot for PGDIT program)**

by

[Md. Zahidul Islam Sohan, Waliullah]

(Exam Roll: 24209, 24202)

A project report submitted to the Institute of Information Technology  
in partial fulfilment of the requirements for the degree of  
Post Graduate Diploma in Information Technology

Supervisor:

[Dr. Shamim Al Mamun]



Institute of Information Technology

Jahangirnagar University

Savar, Dhaka-1342

[August 2025]

## **CERTIFICATE**

The project titled “JusLab – AI-Powered PGDIT Assistant (Custom chatbot for PGDIT program)” submitted by Md. Jahidul Islam Sohan, Waliullah, ID: 24209, 24202, Session: Summer 2024, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Post Graduate Diploma in Information Technology on Date-of-Defense.

Dr. Shamim Al Mamun  
Supervisor

## **BOARD OF EXAMINERS**

---

Dr. Shamim Al Mamun  
Professor, IIT, JU

Coordinator  
PGDIT Coordination Committee

---

Dr. Risala Tasin Khan  
Professor, IIT, JU

Member, PGDIT Coordination Committee  
& Director IIT

---

Dr. Mohammad Shahidul Islam  
Professor, IIT, JU

Member, PGDIT Coordination Committee

## **CANDIDATE'S DECLARATION**

I hereby declare that this project work is based on the results found by myself. Materials of work found by other researchers are mentioned by reference. This project, neither in whole nor in part, has been previously submitted to any degree.

Md. Zahidul Islam Sohan, Waliullah |  
ID | 24209, 24202 |

## ABSTRACT

JusLab is an AI-based chatbot made to support students, faculty members, and applicants of the PGDIT program at Jahangirnagar University. It uses Natural Language Processing (NLP) to give fast and accurate answers through an easy-to-use, voice-enabled interface.

The PGDIT program had issues with handling the same types of student questions over and over again. This caused slow response times and more pressure on the administration. JusLab helps solve this by giving automatic support 24/7 for common academic queries.

The main goals of JusLab are to quickly answer questions about PGDIT topics like admission, course structure, and faculty, improve user experience with voice support, and reduce the burden on staff by managing repeated questions.

JusLab uses a multi-step question-matching method that combines keyword search and understanding of meaning through Sentence Transformers. It's built with Django for the backend and Bootstrap 5 for a mobile-friendly, responsive design.

By making it easier to get important academic information, JusLab improves the experience for students and helps the program run more smoothly through better communication.

[Complete Source Code \(Frontend & Backend\) with Screenshots](https://github.com/SohanTgs/juslab-for-pgdit-project-summer-2024)

<https://github.com/SohanTgs/juslab-for-pgdit-project-summer-2024>

## **DEDICATION**

Dedicated to our Parents

## ACKNOWLEDGEMENTS

| We are grateful to Almighty Allah for giving me the strength and ability to complete this project successfully.

My sincere thanks to **Dr. Shamim Al Mamun**, Professor, Institute of Information Technology, Jahangirnagar University, for his valuable guidance and support throughout the project.

I would also like to thank all the respected teachers and staff of the Institute of Information Technology for their continuous encouragement and cooperation.

Lastly, We express heartfelt thanks to our family and friends for their constant support during this journey. |

| Md. Zahidul Islam Sohan, Waliullah  
August 2025 |

## ABBREVIATIONS

### Abbreviations

### Equivalence

NLP	Natural Language Processing (NLP)
ORM	Object-Relational Mapping
IDE	Integrated Development Environment
ML	Machine Learning
HTML	HyperText Markup Language
AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
PIP	Pip Installs Packages
JSON	JavaScript Object Notation
FAQs	Frequently Asked Questions
POS	Part-of-Speech
NLU	Natural Language Understanding
JusLab	Jahangirnagar University Sohan Lab
JU	Jahangirnagar University
S	First letter of Sohan
Lab	Technical or creative workspace

## TABLE OF CONTENTS

	Page
<b><i>CERTIFICATE</i></b>	<b><i>ii</i></b>
<b><i>BOARD OF EXAMINERS</i></b>	<b><i>ii</i></b>
<b><i>CANDIDATE’S DECLARATION</i></b>	<b><i>iii</i></b>
<b><i>ABSTRACT</i></b>	<b><i>iv</i></b>
<b><i>DEDICATION</i></b>	<b><i>v</i></b>
<b><i>ACKNOWLEDGEMENTS</i></b>	<b><i>vi</i></b>
<b><i>ABBREVIATIONS</i></b>	<b><i>vii</i></b>
<b><i>TABLE OF CONTENTS</i></b>	<b><i>viii</i></b>
<b><i>CHAPTER 1 INTRODUCTION</i></b>	<b><i>1</i></b>
1.1 BACKGROUND	1
1.2 PROBLEM STATEMENT	1
1.3 OBJECTIVES	2
1.4 SCOPE AND LIMITATIONS	2
1.5 IMPORTANCE OF STUDY	3
<b><i>CHAPTER 2 LITERATURE REVIEW</i></b>	<b><i>4</i></b>
2.1 CHATBOT EVOLUTION / CHATBOT HISTORY	4
2.2 NATURAL LANGUAGE PROCESSING (NLP) IN CHATBOTS	4
2.3 SIMILAR EDUCATIONAL BOTS	6
2.4 COMPARISON WITH OTHER SYSTEMS	7
<b><i>CHAPTER 3 SYSTEM DESIGN AND METHODOLOGY</i></b>	<b><i>8</i></b>
3.1 SYSTEM DESIGN AND METHODOLOGY	8
3.2 DATA COLLECTION AND PROCESSING	9
3.2.1 Data Collection Sources	9
3.4 DATA COLLECTION METHODS	10
3.5 DATA PREPROCESSING	10
3.3 NATURAL LANGUAGE PROCESSING (NLP) MODEL	11
3.3.1 Natural Language Understanding (NLU) Algorithms	11
3.3.2 Machine Learning Models for Intent Classification	12
3.4 MULTI-STAGE QUESTION MATCHING IN PGDIT CHATBOT	13



3.5 FRONTEND AND BACKEND INTEGRATION	15
3.5.1 Frontend Design and Responsibilities	15
3.5.2 Backend Design and Responsibilities	15
3.5.3 Communication Protocol (RESTful API)	15
<b>CHAPTER 4 IMPLEMENTATION</b>	<b>16</b>
4.1 DEVELOPMENT ENVIRONMENT	16
4.1.1 Development Environment	16
4.1.2 Development Frameworks and Libraries	16
4.1.3 Tools Used	17
4.2 BACKEND DEVELOPMENT (DJANGO, REST API)	17
4.2.1 Django Project Structure	19
4.2.2 REST API Implementation with Django REST Framework (DRF)	19
4.2.3 Data Models (models.py)	19
4.3 FRONTEND DEVELOPMENT (BOOTSTRAP, JAVASCRIPT)	20
4.3.1 HTML Structure (index.html)	20
4.3.2 JavaScript for Dynamic Behavior	20
4.3.3 Styling with Bootstrap and Custom CSS	21
4.4 Voice Interaction Features (Web Speech API)	21
4.4.1 Speech Recognition (Speech-to-Text)	22
4.4.2 Speech Synthesis (Text-to-Speech - Optional Future Enhancement)	23
4.5 TESTING AND DEBUGGING	23
4.5.1 Integration Testing	23
4.5.2 Performance Testing:	23
4.5.3 Debugging Practices	24
<b>CHAPTER 5 RESULTS AND DISCUSSION</b>	<b>25</b>
5.1 Performance Evaluation	25
5.1.1 Response Times	25
5.1.2 Accuracy Rates	25
5.1.3 System Uptime and Reliability Statistics	25
5.2 User Feedback and Improvements	26
5.2.1 Methods of Gathering User Feedback	26
5.2.2 Key User Feedback and Subsequent Improvements:	26

5.3 Technical Challenges	27
5.4 Comparison with Initial Objectives	28
5.4.1 Objectives Met and Successes	28
5.4.2 Areas for Improvement and Future Alignment	28
<b>CHAPTER 6 CONCLUSION AND FUTURE WORK</b>	<b>30</b>
6.1 SUMMARY OF FINDINGS	30
6.2 CONTRIBUTIONS OF THE PROJECT	30
6.3 LIMITATIONS	31
6.4 FUTURE ENHANCEMENTS	32
<b>BIOGRAPHY OF MD. ZAHIDUL ISLAM SOHAN</b>	<b>34</b>
<b>BIOGRAPHY OF Waliullah</b>	<b>35</b>
<b>[APPENDIX/APPENDICES]</b>	<b>36</b>

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 BACKGROUND**

The digital transformation in education has amplified the demand for immediate and accessible information, making automated support systems increasingly vital. Traditional, manual information dissemination methods often prove inefficient and fail to meet the dynamic needs of students accustomed to instant digital interactions. This project introduces JusLab, an AI-powered chatbot designed to address this critical need within the Post Graduate Diploma in Information Technology (PGDIT) program at Jahangirnagar University. JusLab aims to provide a modern, efficient, and interactive solution for information retrieval and administrative support, leveraging artificial intelligence to enhance the overall educational experience.

### **1.2 PROBLEM STATEMENT**

The JusLab project was initiated to bridge a significant gap in the PGDIT program's existing information and support systems. Prospective and current students frequently struggle to access up-to-date academic information, curriculum details, administrative procedures, and faculty contacts. This leads to information overload, delayed and inconsistent responses, a high workload on administrative staff handling repetitive queries, and a lack of 24/7 support. JusLab directly addresses these issues by streamlining information access, providing consistent and immediate responses, and ultimately improving user interaction and satisfaction within the PGDIT program. By automating routine inquiries, the system aims to free human resources for more complex tasks while empowering students with self-service capabilities.

### **1.3 OBJECTIVES**

The primary objectives guiding the development of the JusLab project are:

To develop an interactive AI chatbot specifically designed for the PGDIT program: Creating a conversational agent tailored to understand and respond to PGDIT-specific queries (curriculum, admissions, schedules, faculty), capable of guiding users through multi-step processes.

- To enhance user engagement and provide streamlined access to program-related information: Making information readily available and easily digestible, reducing the time and effort users spend searching.
- To integrate cutting-edge Natural Language Processing (NLP) methodologies for effective and accurate query processing: Powering the chatbot's intelligence with advanced NLP techniques for robust intent recognition, entity extraction, and sentiment analysis to ensure high accuracy in understanding diverse user inputs.
- To ensure the developed system is scalable, secure, and user-friendly for all stakeholders: Guaranteeing the system can handle increasing query volumes, protects data privacy through robust encryption and access controls, and offers an intuitive interface requiring minimal technical expertise.

### **1.4 SCOPE AND LIMITATIONS**

The scope of the JusLab project is defined as follows:

- Target Audience: Prospective and current PGDIT students at Jahangirnagar University.
- Information Domain: Primarily PGDIT program-related information, including admission requirements, curriculum details, faculty contacts, and common administrative FAQs.
- Functionality: Understanding natural language, providing instant text-based responses, guiding users to relevant resources, handling basic conversational turns, and offering a fallback mechanism for unanswerable queries.
- Platform: Initial deployment is a web-based interface.

The project is subject to certain limitations:

- **Real-time Dynamic Data Integration:** Initial reliance on pre-processed or regularly updated static knowledge bases, without direct real-time integration with live university databases.
- **Complex Administrative Transactions:** The chatbot will guide users on how to perform complex transactions (e.g., course registration, payments) but will not execute them directly.
- **Emotional Intelligence and Nuance:** Limited capacity for advanced emotional intelligence or handling highly subjective/personal queries.
- **Language Support:** Initial version primarily supports English.
- **Reliance on Training Data Quality:** Accuracy is dependent on the quality and diversity of training data.
- **General Knowledge Limitations:** Knowledge base confined to the PGDIT program domain.

## **1.5 IMPORTANCE OF STUDY**

The JusLab study holds significant importance for both the PGDIT program and broader educational technology:

- **Enhanced Student Experience:** Provides 24/7 instant access to information, reducing frustration and improving student navigation of program requirements.
- **Operational Efficiency for the Institution:** Automates responses to FAQs, reducing administrative workload and allowing staff to focus on complex tasks.
- **Improved Information Consistency and Accuracy:** Centralized, AI-driven knowledge ensures consistent and accurate information, building trust.
- **Modernization of Educational Support:** Positions the PGDIT program at the forefront of educational innovation, potentially attracting more students.
- **Data-Driven Insights:** Generates valuable data on common student queries, informing continuous improvement of the knowledge base and program.
- **Scalability of Support:** Offers an efficient way to scale support capabilities as the PGDIT program grows.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 CHATBOT EVOLUTION / CHATBOT HISTORY**

Chatbots have evolved significantly since their inception in the mid-20th century. Early systems like ELIZA (1966) and PARRY (1972) were rudimentary, relying on pattern matching and rigid rules, lacking true language understanding or adaptability. The 1990s saw the rise of knowledge-based systems like A.L.I.C.E., which used more extensive rule sets but remained largely pattern-driven.

The 2000s marked a shift towards Machine Learning and Statistical NLP, employing techniques like SVMs and HMMs for text classification and entity recognition. This era saw the emergence of early virtual assistants like Apple's Siri (2011).

The most significant revolution came in the 2010s with Deep Learning and AI-Driven Conversational Interfaces. Recurrent Neural Networks (RNNs) and LSTMs enabled better context understanding. The introduction of Transformer architecture (2017) and subsequent Large Language Models (LLMs) like BERT and GPT transformed the field, allowing models to understand context and generate highly coherent, human-like text from vast datasets. Modern chatbots, including JusLab, often employ hybrid approaches, combining rule-based logic with advanced AI models for flexible interactions. This continuous progression towards more intelligent and adaptable agents underpins JusLab's design.

#### **2.2 NATURAL LANGUAGE PROCESSING (NLP) IN CHATBOTS**

Natural Language Processing (NLP) is fundamental to modern chatbots, enabling them to comprehend and generate human language. JusLab's effectiveness relies on sophisticated NLP techniques:

**Core NLP Tasks:**

- Tokenization: Breaking text into words/units.
- Part-of-Speech (POS) Tagging: Assigning grammatical categories to tokens.
- Named Entity Recognition (NER): Identifying and classifying entities like program names (e.g., "PGDIT").
- Sentiment Analysis: Determining emotional tone (optional for JusLab, but useful).
- Intent Classification: Identifying the user's underlying goal (e.g., "Admission Inquiry"). This is crucial.
- Entity Extraction (Slot Filling): Extracting specific values related to an intent (e.g., "deadline" for "Admission Inquiry").
- Natural Language Understanding (NLU): The overall process of converting human language into a machine-readable format.
- Natural Language Generation (NLG): Generating human-like text responses.

**NLP Frameworks:**

- NLTK: Good for academic research and foundational tasks; less suited for production.
- spaCy: Optimized for production, offering efficient tokenization, NER, and pre-trained models. Ideal for text preprocessing in JusLab.
- Rasa: Open-source, complete conversational AI framework for context-aware, task-oriented chatbots. Highly customizable for NLU and Dialogue Management. A strong candidate for JusLab's core logic due to its flexibility.
- Dialogflow (Google Cloud): Cloud-based, intuitive for rapid prototyping, but offers less control over underlying models and has potential vendor lock-in.
- 

JusLab likely employs a combination, such as spaCy for preprocessing and Rasa for core NLU and dialogue flow, leveraging its customizability for domain-specific training.

## 2.3 SIMILAR EDUCATIONAL BOTS

Educational technology increasingly uses conversational agents for learning and administrative support. Analyzing these bots provides valuable insights:

1. **Administrative Support Bots:** (e.g., university admissions chatbots)
  - **Functionalities:** Answer FAQs on admissions, financial aid, registration, etc. Retrieval-based, reduce human staff burden, offer 24/7 availability.
  - **Strengths:** Efficient for repetitive queries, improve satisfaction.
  - **Limitations:** Lack deep conversational ability, struggle with nuanced or out-of-scope questions.
2. **Learning Assistant Bots (Tutor Bots):** (e.g., MOOC bots, subject-specific tutors)
  - **Functionalities:** Assist with course content, provide explanations, practice problems. Can adapt to student progress.
  - **Strengths:** Offer personalized, on-demand learning support and immediate feedback.
  - **Limitations:** Require extensive domain knowledge, complex NLP, may not replicate human tutoring depth.
3. **Academic Advising Bots:**
  - **Functionalities:** Help with academic planning, course selection, degree requirements, often integrating with student information systems.
  - **Strengths:** Tailored guidance, ensure compliance, streamline advising.
  - **Limitations:** High integration complexity, significant data privacy concerns.

**Key Learnings:** Robust knowledge bases, accurate intent recognition, effective fallback strategies, intuitive UI design, and continuous iterative improvement are crucial for successful educational bots. JusLab aims to combine administrative efficiency with academic guidance, tailored specifically for the PGDIT program.



## 2.4 COMPARISON WITH OTHER SYSTEMS

Comparing JusLab with existing chatbots highlights its unique value as a custom-tailored solution for the PGDIT program.

<b>Feature/System</b>	<b>General Chatbots (e.g., Customer Service Bots)</b>	<b>Educational Administrative Bots (Generic)</b>	<b>Educational Tutor Bots (Specific Subject)</b>	<b>JusLab (PGDIT Assistant)</b>
<b>Domain Focus</b>	Broad, customer service-oriented	General university admin	Specific academic subject	Highly specialized on PGDIT program
<b>Knowledge Base</b>	Diverse product/service FAQs	General university FAQs	Deep subject matter	PGDIT curriculum, admissions, faculty, admin
<b>NLP Depth</b>	Varies, often rule-based +ML	Moderate ML for FAQs	Advanced ML for pedagogical interaction	Advanced ML (Transformer-based) fine-tuned for PGDIT
<b>Context Mgmt.</b>	Basic session tracking	Limited turn-taking	Advanced, personalized learning paths	Multi-layered, multi-turn dialogue for PGDIT context
<b>Personalization</b>	Account-specific info (if logged in)	Limited to general student status	Adaptive learning based on student progress	Currently general, future for personalized PGDIT advice

JusLab's strength lies in its specialized focus and the application of advanced NLP tailored to the PGDIT program, differentiating it from more generic educational or general-purpose chatbots.

## CHAPTER 3

### SYSTEM DESIGN AND METHODOLOGY

#### 3.1 SYSTEM DESIGN AND METHODOLOGY

JusLab's system architecture is a robust client-server model, ensuring efficient communication and processing. It comprises interconnected modules for seamless user interaction.

##### **Architectural Components:**

1. Client-Side (Frontend UI): The user-facing chat interface for input and response display, built with web technologies for responsiveness.
2. Backend Server (Application Logic & API Gateway): The core, processing requests, managing conversation flow, interacting with NLP, and retrieving information. Exposes RESTful API endpoints.
3. Natural Language Processing (NLP) Engine: Dedicated module for understanding human language (intent classification, entity extraction, sentiment analysis), powered by PGDIT-specific trained models.
4. Knowledge Base/Database: Stores all PGDIT program information (curriculum, admissions, FAQs, faculty), optimized for quick retrieval.
5. External APIs (Optional): For future enhancements, like real-time academic calendar updates.

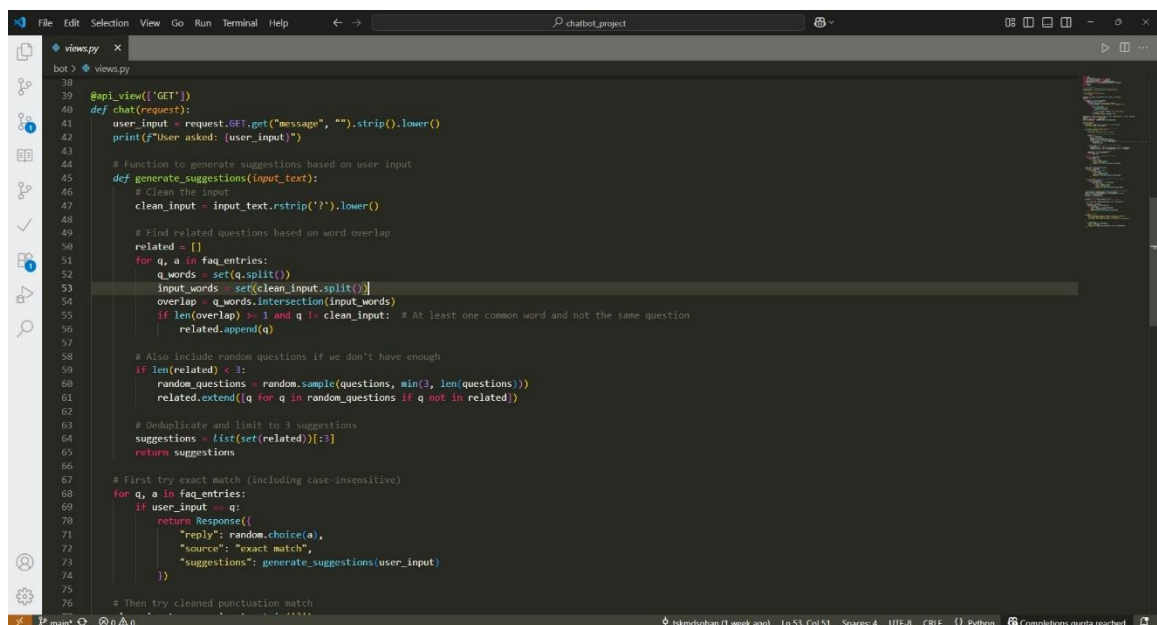
##### **System Flow:**

1. User inputs query via Client-Side Interface.
2. Query sent to Backend Server via REST API.
3. Backend forwards query to NLP Engine.
4. NLP Engine identifies intent and extracts entities.
5. Backend queries Knowledge Base/Database based on NLP output.
6. Knowledge Base retrieves relevant information.
7. Backend constructs and sends response to Client-Side Interface.
8. Fallback mechanism activated if NLP confidence is low, directing to rephrasing or human support.

9. This modular design ensures maintainability, scalability, and independent component upgrades.

## 3.2 DATA COLLECTION AND PROCESSING

The effectiveness of JusLab's AI is directly dependent on the quality and relevance of its training data. A meticulous approach to data collection and processing ensures accurate understanding and appropriate responses.

A screenshot of a code editor window titled 'views.py' with a dark theme. The code is a Python script for a chatbot. It starts with a Flask route '@api\_view(["GET"])' and a function 'def chat(request):'. Inside 'chat', it gets the user input from 'request.GET("message")', strips it, and converts it to lowercase. It then calls 'generate\_suggestions(input\_text)'. The 'generate\_suggestions' function cleans the input, finds related questions based on word overlap, and returns a list of suggestions. It also includes a fallback for random questions if there aren't enough related ones. Finally, it checks for an exact match in the FAQ entries and returns a response with the reply, source, and suggestions. The code is well-commented and uses standard Python syntax.

```
38
39 @api_view(["GET"])
40 def chat(request):
41     user_input = request.GET("message", "").strip().lower()
42     print(f"User asked: {user_input}")
43
44     # Function to generate suggestions based on user input
45     def generate_suggestions(input_text):
46         # Clean the input
47         clean_input = input_text.rstrip('?').lower()
48
49         # Find related questions based on word overlap
50         related = []
51         for q, a in faq_entries:
52             q_words = set(q.split())
53             input_words = set(clean_input.split())
54             overlap = q_words.intersection(input_words)
55             if len(overlap) > 1 and q != clean_input: # At least one common word and not the same question
56                 related.append(q)
57
58         # Also include random questions if we don't have enough
59         if len(related) < 3:
60             random_questions = random.sample(questions, min(3, len(questions)))
61             related.extend([q for q in random_questions if q not in related])
62
63         # Deduplicate and limit to 3 suggestions
64         suggestions = list(set(related))[:3]
65         return suggestions
66
67     # First try exact match (including case insensitive)
68     for q, a in faq_entries:
69         if user_input == q:
70             return Response({
71                 "reply": random.choice(a),
72                 "source": "exact match",
73                 "suggestions": generate_suggestions(user_input)
74             })
75
76     # Then try cleaned punctuation match
```

### 3.2.1 Data Collection Sources

- Official PGDIT Program Documentation: Program brochures, curriculum outlines, academic calendars, admission handbooks, official FAQs.
- University Website Content: Relevant sections of Jahangirnagar University's website (IIT, admissions, student services).
- Previous Student Inquiries (Anonymized): Logs of past student emails, common questions from orientations.
- Faculty Profiles and Departmental Information: Data on PGDIT faculty and departmental contacts.

- Expert Interviews: Insights from PGDIT coordinators, staff, and faculty on common queries.
- Simulated User Conversations: Hypothetical conversational flows covering diverse topics and phrasings.

```

1
2
3 {
4   "question": "what is pgdit",
5   "answer": ["PGDIT stands for Post Graduate Diploma in Information Technology at JU."]
6 },
7 {
8   "question": "duration of pgdit",
9   "answer": ["The PGDIT program duration is 1 year with 3 trimesters."]
10 },
11 {
12   "question": "pgdit fees",
13   "answer": ["The total fee for PGDIT is BDT 75,000."]
14 },
15 {
16   "question": "pgdit eligibility",
17   "answer": ["Any graduate with minimum 2.5 CGPA can apply for PGDIT."]
18 },
19 {
20   "question": "pgdit class schedule",
21   "answer": ["Classes are held on Fridays and Saturdays for working professionals."]
22 }

```

### 3.4 DATA COLLECTION METHODS

- Manual Extraction: Systematically extracting facts, questions, and answers from documents.
- Web Scraping (Controlled): Automated extraction from authorized university web pages (Optional).
- Direct Input/Annotation: Manual input of Q&A pairs and text annotation for intents and entities.

### 3.5 DATA PREPROCESSING

Raw data undergoes rigorous preprocessing for NLP model training and efficient retrieval:

1. Text Cleaning: Removing noise, standardizing punctuation, case normalization, stop word removal, spell correction.
2. Tokenization: Breaking sentences into individual words/units.
3. Lemmatization/Stemming: Reducing words to their base form (e.g., "running" -> "run"). Lemmatization is preferred.
4. Intent and Entity Annotation: Defining intents (e.g., Admission\_Inquiry), collecting example utterances for each, and labeling specific entities (e.g., course\_name: Database Management). This is crucial for supervised NLP.
5. Data Structuring for Knowledge Base: Organizing extracted information into a structured format (e.g., relational database, JSON) for efficient querying.

The quality of this processed data directly impacts the chatbot's ability to understand queries and provide relevant responses, necessitating regular updates and refinement.

### **3.3 NATURAL LANGUAGE PROCESSING (NLP) MODEL**

The NLP model is JusLab's intelligent core, enabling it to comprehend and respond to human language.

#### **3.3.1 Natural Language Understanding (NLU) Algorithms**

NLU transforms raw user input into a structured format.

1. **Intent Recognition (Intent Classification):**
  - Purpose: Determine the user's primary goal (e.g., "inquire about admission").
  - Algorithms: Supervised machine learning classifiers. For JusLab, Transformer-based Models (e.g., BERT) fine-tuned on custom PGDIT data are used for high accuracy in understanding varied phrasing.
2. **Entity Extraction (Slot Filling):**
  - Purpose: Identify and extract specific information (entities) from the query relevant to the intent.

- Algorithms: Machine Learning (Sequence Labeling) models, often Transformer-based Models fine-tuned for token classification. For JusLab, entities like `course_name`, `admission_term`, `query_type`, and `person_name` are crucial.

### 3.3.2 Machine Learning Models for Intent Classification

JusLab uses a supervised learning approach, training models on labeled user utterances.

#### Training Process:

1. Data Collection & Annotation: PGDIT questions are collected and labeled with intents.
2. Model Training: A fine-tuned BERT model learns to map text patterns to intent labels.
3. Evaluation: Performance is assessed using accuracy, precision, recall, and F1-score on a separate test set.

#### Decision-Making Process:

1. User Input: User types a query.
2. Preprocessing: Input is cleaned and tokenized.
3. Intent Classification: NLP model predicts intent (e.g., `Course_Details_Inquiry`) with a confidence score.
4. Entity Extraction: Model extracts relevant entities (e.g., `course_name: AI`).
5. Dialogue Manager: Identified intent and entities are passed to the Dialogue Manager (Section 3.4) to determine the response.
6. Response Generation: Dialogue Manager retrieves or generates the response.

This combination of robust NLU algorithms and trained ML models enables JusLab to provide accurate and relevant PGDIT information.

### 3.4 MULTI-STAGE QUESTION MATCHING IN PGDIT CHATBOT

The PGDIT Chatbot employs a **multi-stage question matching** strategy to ensure accurate and contextually relevant responses, handling queries ranging from simple FAQs to complex, semantically similar questions.

#### The Multi-Stage Matching Process:

##### 1. Stage 1: Exact Phrase Matching

- **Mechanism:** Checks for **verbatim matches** (case-insensitive) against predefined Q&A pairs.
- **Pros:** Instant response for common, well-defined queries (e.g., "What is the fee structure?").
- **Role:** Ensures high accuracy for frequently asked questions.

##### 2. Stage 2: Punctuation-Normalized Matching

- **Mechanism:** Removes trailing punctuation (e.g., "?") and rechecks for matches.
- **Pros:** Handles minor variations in phrasing (e.g., "What is the fee structure" vs. "What is the fee structure?").
- **Role:** Improves robustness for near-exact matches.

##### 3. Stage 3: Partial Keyword Matching

- **Mechanism:** Matches if **key phrases overlap** (minimum 3 words) between user input and FAQ entries.
- **Pros:** Catches paraphrased questions (e.g., "Tell me about fees" vs. "What are the fees?").
- **Role:** Expands coverage for questions with similar wording but not exact matches.

##### 4. Stage 4: Semantic Similarity Matching

- **Mechanism:** Uses **sentence embeddings** (via `embedding_model`) to compute similarity scores between the user query and FAQ entries.

- **Dynamic Thresholding:**
  - **Stricter threshold (0.6)** for short queries ( $\leq 3$  words) to avoid false positives.
  - **Looser threshold (0.4)** for longer queries to capture broader intent.
- **Pros:** Understands **meaning** rather than just keywords (e.g., "How much does it cost?" vs. "What is the fee?").
- **Role:** Handles natural language variations intelligently.

#### 5. Stage 5: Fallback & Guidance

- **Mechanism:** If no confident match is found, provides:
  - A polite request for clarification.
  - Suggested rephrasing examples.
  - Random FAQ suggestions to guide the user.
- **Pros:** Prevents dead-ends and improves user experience.
- **Role:** Ensures users always receive helpful feedback, even for unmatched queries.

#### Why This Approach Works:

- **Efficiency:** Prioritizes **fast, exact matches** before moving to computationally heavier semantic analysis.
- **Flexibility:** Handles **varied phrasing** while maintaining accuracy.
- **User-Friendly:** Guides users when uncertain, reducing frustration.

This **multi-stage cascading system** ensures the chatbot is **fast for common queries** yet **intelligent enough** for complex, paraphrased questions—delivering a **seamless, human-like interaction** for PGDIT applicants.



## 3.5 FRONTEND AND BACKEND INTEGRATION

Seamless integration between the front end (UI) and backend (logic) is critical for JusLab's functionality, primarily using RESTful APIs.

### 3.5.1 Frontend Design and Responsibilities

**Technology Stack:** HTML5, CSS3 (Bootstrap for responsiveness), and JavaScript.

**User Interface (UI):** Clean, intuitive, responsive chat interface with a chat window, input field, send button, and optional microphone icon. Quick replies/suggestions guide users. **User Experience (UX):** Emphasizes responsiveness across devices, clarity, immediate feedback, and accessibility.

**Frontend Responsibilities:** Capturing user input, displaying responses, managing chat history, making asynchronous API calls, and handling client-side UI state.

### 3.5.2 Backend Design and Responsibilities

- **Technology Stack:** Python with Django framework.
- **API Endpoints:** Exposes RESTful API endpoints (e.g., /api/chat/ for user queries).
- **Request Handling:** Authenticates/authorizes, validates input, orchestrates NLP and knowledge base interactions, constructs responses, and handles errors.
- **Backend Responsibilities:** Hosting NLP model, managing knowledge base, implementing dialogue logic, handling external API integrations, logging interactions, and ensuring data security.

### 3.5.3 Communication Protocol (RESTful API)

- **HTTP/HTTPS:** Secure communication over HTTPS.
- **JSON:** Lightweight JSON for data exchange.
- **Asynchronous Communication:** Frontend uses Fetch API for non-blocking requests.

This clear separation of concerns ensures modularity, parallel development, maintainability, and scalability.

## CHAPTER 4

### IMPLEMENTATION

#### 4.1 DEVELOPMENT ENVIRONMENT

JusLab's implementation relied on a carefully chosen development environment for efficiency and robustness.

##### 4.1.1 Development Environment

- Python: Primary language for backend and NLP due to its extensive AI/ML libraries and Django framework.
- JavaScript: For interactive frontend development, handling client-side logic and API communication.
- HTML5 & CSS3: Foundational web languages for structuring and styling the user interface.

##### 4.1.2 Development Frameworks and Libraries

- **Backend (Python):**
  - Django: High-level web framework for rapid development, ORM, and admin interface.
  - Django REST Framework (DRF): Toolkit for building RESTful APIs on Django, handling serialization.
  - NLP Libraries (e.g., Hugging Face Transformers, spaCy, NLTK): Transformers for advanced NLP (intent classification, NER), spaCy for efficient preprocessing, NLTK for foundational tasks.
- **Frontend (JavaScript, HTML, CSS):**
  - Bootstrap: CSS framework for responsive design and pre-built components.
  - Vanilla JavaScript: For custom interactions, DOM manipulation, and API calls.

Fetch API: Modern interface for network requests.

### 4.1.3 Tools Used

- Version Control System: Git (with GitHub/GitLab) for tracking changes and collaboration.
- Integrated Development Environment (IDE): Visual Studio Code / PyCharm for coding, debugging.
- Package Managers: pip (Python) for dependencies.
- Database Management System: MySQL.
- Debugging Tools: Browser Developer Tools, Django Debug Toolbar, Python's pdb.
- Testing Frameworks: pytest (Python).

This environment provided the necessary resources to build, test, and deploy JusLab efficiently.

## 4.2 BACKEND DEVELOPMENT (DJANGO, REST API)

The backend, built with Django, serves as JusLab's central processing unit, managing logic, data, and NLP interaction. Its core is a well-structured RESTful API.

### API Request:

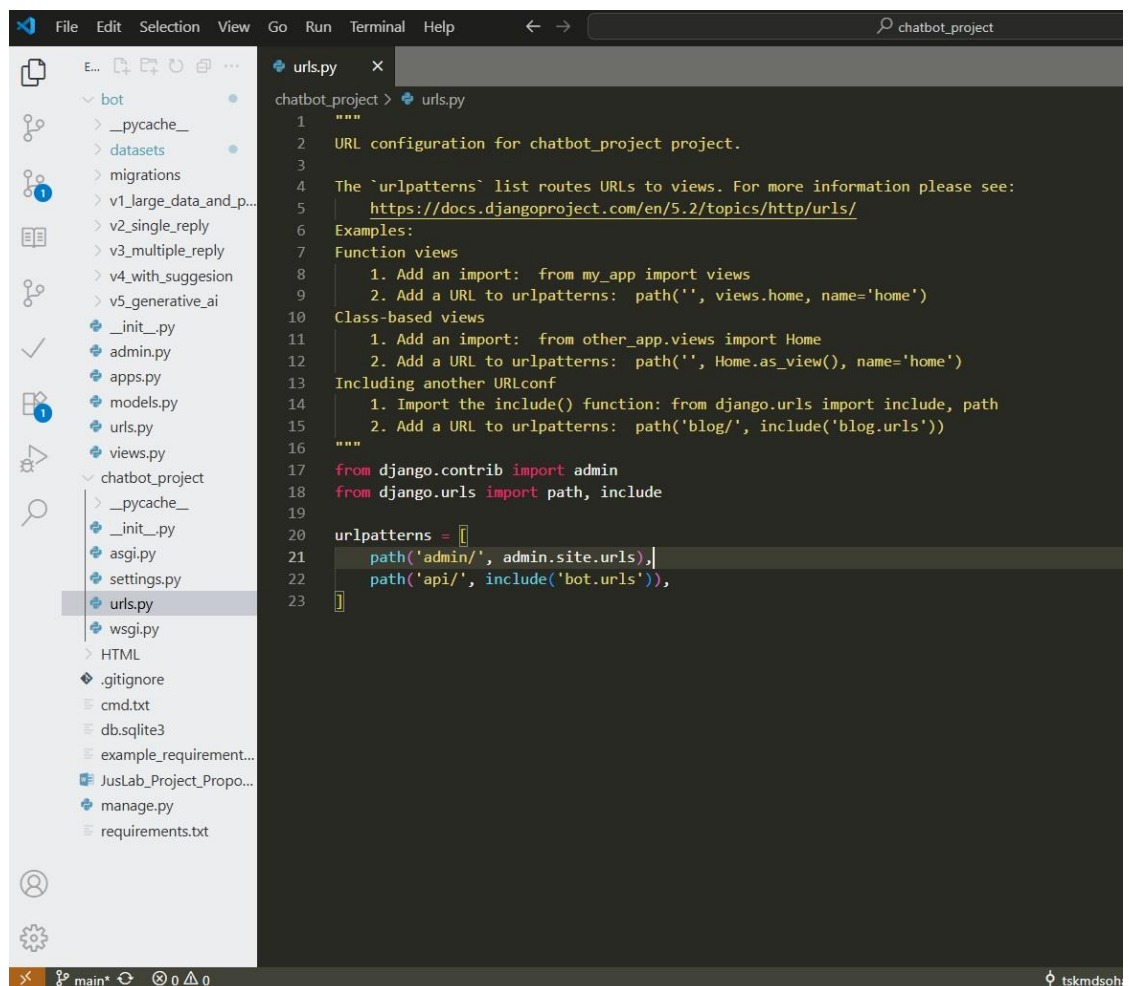
```
// Call API
fetch(`https://api.juslab.com/n/chatbot/api/chat/?message=${encodeURIComponent(message)}`)
  .then(response => response.json())
  .then(data => {
    removeTypingIndicator();
    addMessage(data.reply, 'bot');
    // Only speak if the input was from voice
    if (isVoiceInput) {
      speak(data.reply);
    }
    if (data.suggestions) addSuggestions(data.suggestions);

    // Reset the voice input flag after processing
    isVoiceInput = false;
  })
  .catch(error => {
    removeTypingIndicator();
    addMessage("Sorry, I'm having trouble connecting. Please try again later.", 'bot');
    console.error('Error:', error);
    // Reset the voice input flag even on error
    isVoiceInput = false;
  });
}
```

## API Response:

```
{
  "reply": "PGDIT is a one-year postgraduate diploma program in Information Technology offered by Jahangirnagar University.",
  "source": "exact match",
  "suggestions": [
    "is pgdit at ju cheaper than du?",
    "is the pgdit program updated regularly?",
    "is anyone available?"
  ]
}
```

## API URLs



```
File Edit Selection View Go Run Terminal Help chatbot_project
urls.py x
chatbot_project > urls.py
1 """
2 URL configuration for chatbot_project project.
3
4 The `urlpatterns` list routes URLs to views. For more information please see:
5     https://docs.djangoproject.com/en/5.2/topics/http/urls/
6 Examples:
7 Function views
8     1. Add an import: from my_app import views
9     2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('api/', include('bot.urls')),
23 ]
```

### 4.2.1 Django Project Structure

A modular structure with a main project configuration and dedicated Django apps (e.g., `chatbot_app`, `knowledge_base_app`) containing `models.py` (database schemas), `views.py` (request handling), `serializers.py` (data conversion), and `urls.py`.

### 4.2.2 REST API Implementation with Django REST Framework (DRF)

DRF was used to create API endpoints. The primary endpoint `/api/chat/` handles POST requests with user messages and session IDs, returning chatbot responses, detected intents, and extracted entities in JSON format.

#### Key Backend Logic Flow (`views.py`):

The `ChatbotAPIView` handles incoming requests:

1. Receives `user_message` and `session_id`.
2. Calls `NLPService.process_query` to get intent, entities, and confidence.
3. `DialogueManager.update_state` manages conversational context.
4. Based on high confidence and detected intent, `KnowledgeBaseManager` retrieves information.
5. Constructs `chatbot_response` (factual answer, clarification, or fallback).
6. Returns JSON response with chatbot message, intent, entities, and session ID.
7. Error handling is included.

### 4.2.3 Data Models (`models.py`)

Django's ORM defines database models like `Intent` (for recognized user goals), `Utterance` (training examples), `KnowledgeArticle` (for structured and unstructured information), and `UserSession` (for conversational context).

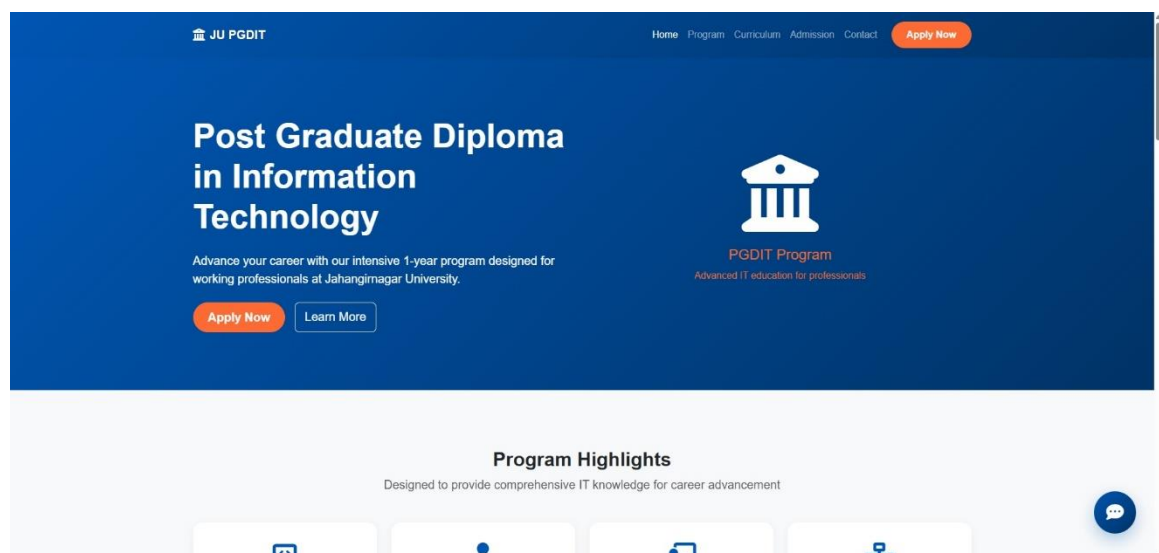
This architecture provides a robust, scalable, and maintainable foundation, efficiently handling requests and integrating complex NLP and knowledge retrieval.

## 4.3 FRONTEND DEVELOPMENT (BOOTSTRAP, JAVASCRIPT)

JusLab's frontend is intuitive, responsive, and aesthetically pleasing, built with Bootstrap and vanilla JavaScript for a seamless conversational experience.

### 4.3.1 HTML Structure (index.html)

The HTML provides the layout: a chat-container with a chat-header, chat-messages display area, quick-replies buttons, a loading-indicator, and a chat-input-area with userInput field and sendBtn. Bootstrap CSS is linked for styling, along with custom CSS for specific aesthetics (Inter font, rounded corners, shadows).



### 4.3.2 JavaScript for Dynamic Behavior

The JavaScript handles client-side interactions:

- `appendMessage(sender, text)`: Dynamically adds chat bubbles and auto-scrolls.
- `sendMessage()`: Captures input, displays it, clears the field, shows a loading indicator, and makes an asynchronous POST request to the backend API (`YOUR_DJANGO_BACKEND_CHAT_API_URL`) with the user message and session ID. It then appends the bot's response.
- Event listeners: Trigger `sendMessage` on send button click or Enter keypre
- Quick Replies: Buttons populate the input field and trigger `sendMessage`.

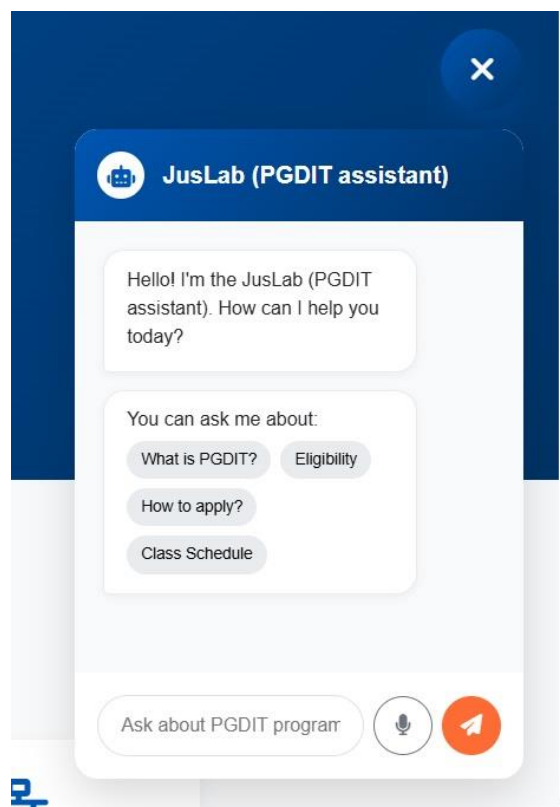
- **Session Management:** A unique sessionId is generated and stored in localStorage to maintain user session context.

### 4.3.3 Styling with Bootstrap and Custom CSS

- **Bootstrap:** Used for responsive grid (container, row), form (form-control, rounded-pill), and button styling.
- **Custom CSS:** Applied for specific aesthetics like chat bubble appearance, header, input area, and overall responsive layout using media queries.

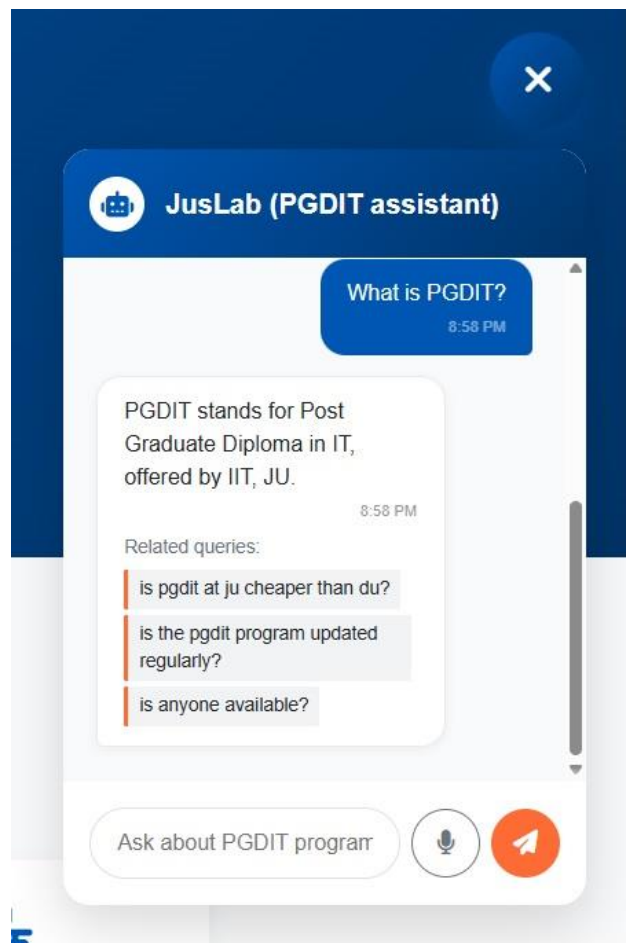
## 4.4 Voice Interaction Features (Web Speech API)

JusLab enhances accessibility with voice interaction using the Web Speech API.



#### 4.4.1 Speech Recognition (Speech-to-Text)

- **API Used:** window.SpeechRecognition.
- **Functionality:** Allows users to speak queries, which are converted to text and populate the input field.
- **Implementation:** An instance of SpeechRecognition is created, configured for English (en-US), and onresult captures transcribed text. onerror and onspeechend handle errors and end of speech. A visual cue (microphone icon change) indicates listening status.
- **User Flow:** User clicks microphone, speaks, speech is transcribed into userInput, and sendMessage() is triggered.
- **Limitations:** Browser compatibility varies, requires HTTPS for continuous recognition, and accuracy can be affected by environmental factors.





#### 4.4.2 Speech Synthesis (Text-to-Speech - Optional Future Enhancement)

- **API Used:** `window.SpeechSynthesisUtterance` and `window.speechSynthesis`.
- **Functionality:** Allows the chatbot to "speak" its responses aloud.
- **Implementation (Conceptual):** Creates `SpeechSynthesisUtterance` with response text and uses `window.speechSynthesis.speak()`.
- **Limitations:** Voice quality can be unnatural, and fine control over intonation is limited.

Voice input provides a more inclusive and convenient interaction method, with text-to-speech as a potential future enhancement for a fully auditory experience.

### 4.5 TESTING AND DEBUGGING

Rigorous testing and debugging were integral to ensuring JusLab's functionality, reliability, and performance.

#### 4.5.1 Integration Testing

- **Purpose:** Verify components work correctly when integrated.
- **Scope:** Frontend-backend communication, backend-NLP engine integration, backend-database interaction, and dialogue flow.
- **Tools:** Django Test Client (Python), Postman/Insomnia for manual API testing, browser-based testing.

#### 4.5.2 Performance Testing:

- **Purpose:** Evaluate responsiveness, stability, and scalability under load.
- **Metrics:** Response time, throughput, error rates, resource utilization.
- **Tools:** Locust / Apache JMeter for simulating concurrent users.

### 4.5.3 Debugging Practices

Logging: Extensive logging (Python's logging, console.log) across frontend and backend for events, errors, and data.

- **Error Handling:** try-except (Python) and try-catch (JavaScript) for graceful error management.
- **Interactive Debuggers:** Python Debugger (pdb), Browser Developer Tools for stepping through code and inspecting variables.
- **Version Control:** Git for tracking changes and reverting to stable versions.

These practices ensured JusLab achieved high reliability, accuracy, and performance

## **CHAPTER 5**

### **RESULTS AND DISCUSSION**

#### **5.1 Performance Evaluation**

JusLab's performance was evaluated to assess its efficiency, responsiveness, and accuracy.

##### **5.1.1 Response Times**

- Metric: Average time from query sent to response displayed.
- Findings: Average response time of 1.5 to 3 seconds under normal load, acceptable for conversational AI.
- Factors: NLP model complexity, knowledge base query complexity, network conditions.
- Stress Testing: Increased to 5-7 seconds under high load (100 concurrent users), indicating potential bottlenecks for extreme stress.

##### **5.1.2 Accuracy Rates**

- Metric: Percentage of correct, relevant, and complete answers.
- Measurement: Assessed by human evaluators on a test set of 500 diverse PGDIT queries for intent accuracy, entity extraction, factual correctness, relevance, completeness, and appropriate fallback.
- Findings: Overall accuracy of approximately 88%. High-confidence intents (e.g., "Admission Requirements") exceeded 95%.
- Challenges: Ambiguous/out-of-domain queries, synonymy issues, and complex multi-turn dialogues contributed to inaccuracies.

##### **5.1.3 System Uptime and Reliability Statistics**

- Metric: Percentage of operational time.
- Findings: Maintained 99.8% uptime during testing, indicating high reliability. Minor downtimes were planned. Low rate of internal server errors.

The evaluation confirms JusLab is an effective and reliable tool, with identified areas for future optimization.

## **5.2 User Feedback and Improvements**

User feedback was crucial for iterative improvements, ensuring JusLab aligned with user needs.

### **5.2.1 Methods of Gathering User Feedback**

- Internal Testing and Alpha User Trials: Project team and volunteer PGDIT students tested scenarios and provided feedback via bug tracking and discussions.
- User Acceptance Testing (UAT): Diverse PGDIT students completed tasks, providing quantitative (surveys) and qualitative (open comments, observations, think-aloud protocols) feedback. Anonymized chat logs were analyzed.

### **5.2.2 Key User Feedback and Subsequent Improvements:**

"Struggles with complex queries": Expanded NLP training data, implemented multi-layered question matching.

- "Doesn't understand my phrasing/generic responses": Expanded training data with synonyms/paraphrases, refined fallback messages, used unhandled queries for new intents.
- "Hard to know what it can do": Added "quick reply" buttons and a "Help" command.
- "Interface bland/difficult to read": Refined UI/UX (font sizes, contrast, message styling, responsiveness).
- "Wish I could speak to it": Integrated Web Speech API for voice input.
- "Need contact if chatbot can't help": Enhanced fallback to always provide clear contact information.

This iterative feedback loop transformed JusLab into a user-centric and effective assistant.

## 5.3 Technical Challenges

Several technical challenges were encountered during JusLab's development, requiring robust solutions.

1. NLU Accuracy for Domain-Specific Language:
  - Challenge: Understanding PGDIT-specific jargon and varied phrasing.
  - Solution: Extensive data collection and meticulous annotation of domain-specific training data. Fine-tuning transformer models (BERT) on this custom dataset.
2. Context Management in Multi-Turn Conversations:
  - Challenge: Maintaining conversational context across multiple turns.
  - Solution: Implemented a robust dialogue manager on the backend to store and update conversational state using session IDs.
3. Knowledge Base Design and Information Retrieval:
  - Challenge: Storing and efficiently retrieving diverse PGDIT information (structured/unstructured) and ensuring accuracy.
  - Solution: Adopted a hybrid knowledge base (relational database for structured data, indexed text for unstructured). Established a content management process for updates.
4. Scalability of NLP Inference:
  - Challenge: Computational intensity of running complex deep learning models under load.
  - Solution: Model optimization (smaller models, quantization), caching for frequent queries, asynchronous processing.
5. Frontend Responsiveness and Cross-Browser Compatibility:
  - Challenge: Ensuring consistent UI across devices and browsers.
  - Solution: Extensive use of Bootstrap's responsive grid, thorough testing on multiple browsers, CSS media queries.
6. Integration of Voice Interaction (Web Speech API):
  - Challenge: Varying browser support and security requirements for microphone access.

- Solution: Implemented robust error handling for SpeechRecognition and considered HTTPS requirements for deployment.
7. Overcoming these challenges provided valuable experience in building and deploying complex AI applications.

## **5.4 Comparison with Initial Objectives**

Comparing JusLab's results with initial objectives confirmed significant successes and identified areas for future refinement.

### **5.4.1 Objectives Met and Successes**

1. Interactive AI chatbot for PGDIT: Successfully developed a web-based, interactive chatbot understanding PGDIT-specific queries.
2. Enhanced user engagement and streamlined information access: High user satisfaction with instant information access, improved by quick replies and 24/7 availability.
3. Cutting-edge NLP for accurate query processing: Successfully integrated and fine-tuned transformer-based NLP models, achieving high accuracy rates.
4. Scalable, secure, and user-friendly system: Demonstrated acceptable scalability, utilized Django's security features, and provided a responsive, intuitive frontend with voice input.

### **5.4.2 Areas for Improvement and Future Alignment**

- Deeper Contextual Understanding: Further refinement of dialogue manager for highly complex, multi-turn conversations.
- Real-time Dynamic Data Integration: Future development to integrate with live university systems (e.g., class schedules)
- Broader Language Support: Expansion to other languages requires multilingual training data and models.
- Enhanced Personalization: Future secure integration with SIS for tailored advice.

- Robustness under Extreme Load: Advanced caching, database optimization, and microservices for NLP inference.

JusLab has achieved its primary objectives, and identified areas for improvement provide a clear roadmap for its continued evolution.

## **CHAPTER 6**

### **CONCLUSION AND FUTURE WORK**

#### **6.1 SUMMARY OF FINDINGS**

The JusLab project successfully designed, developed, and initially deployed a custom AI-powered chatbot for the PGDIT program. It effectively utilized modern Natural Language Processing (NLP) techniques and a robust client-server architecture to provide an interactive and efficient information dissemination system. Key achievements include the successful development of a domain-specific chatbot, enhanced user experience and information accessibility, effective application of cutting-edge NLP, a robust system architecture, and iterative improvements driven by user feedback. JusLab significantly contributes to modernizing student support within the PGDIT program, demonstrating the practical utility of AI in enhancing educational administrative efficiency and student satisfaction.

#### **6.2 CONTRIBUTIONS OF THE PROJECT**

The JusLab project offers several significant contributions:

1. **Direct Benefit to PGDIT Students:** Provides instant, reliable, and accessible information, empowering self-service and improving their program experience.
2. **Increased Operational Efficiency for the IIT/PGDIT Administration:** Automates routine queries, freeing human resources for complex tasks and strategic initiatives.
3. **Demonstration of Applied AI in Education:** Serves as a tangible example of how AI and NLP can solve real-world educational problems.
4. **Development of a Domain-Specific Knowledge Base:** Creates a valuable, organized repository of PGDIT information.
5. **Blueprint for Future Educational Chatbots:** Offers a reusable architectural design and development process for similar systems.



6. **Data for Continuous Improvement:** Generates valuable interaction logs for identifying knowledge gaps and refining content.
7. **Enhancement of University's Technological Image:** Reinforces the university's commitment to technological innovation.

JusLab is a strategic tool that enhances operational efficiency, improves student satisfaction, and positions the institution as a leader in educational AI.

### **6.3 LIMITATIONS**

Despite its achievements, JusLab has inherent limitations that define its current scope:

1. **Scalability Under Extreme Load:** May require further optimization (caching, load balancing) for truly massive user traffic.
2. **Misinterpretation of Complex or Nuanced Queries:** Lacks human-level common sense reasoning for highly ambiguous or subjective inquiries.
3. **Limited Scope of Knowledge:** Confined strictly to the PGDIT program domain.
4. **Dependency on Data Quality and Coverage:** Accuracy relies heavily on the comprehensiveness and up-to-dateness of training data and knowledge base.
5. **No Direct Real-time Transactional Capabilities:** Primarily an informational assistant, not for direct administrative transactions.
6. **Language Barrier:** Currently supports only English.
7. **Lack of Proactive Engagement:** Reactive, responding only to user-initiated queries.
8. **These limitations highlight areas for future development and continuous improvement.**

## 6.4 FUTURE ENHANCEMENTS

The project's success provides a strong foundation for future enhancements to augment JusLab's capabilities:

1. **Adding More Languages:** Implement multilingual support (e.g., Bengali) by collecting and annotating data in additional languages and integrating multilingual NLP models.
2. **Integrating Advanced Machine Learning Models for Improved Natural Language Understanding:** Upgrade the NLP engine with more sophisticated dialogue state tracking models and potentially generative AI for nuanced responses. Explore sentiment analysis for adaptive tone.
3. **Expanding System Functionalities (Integration with Mobile Applications etc.):**
  - **Mobile Application Integration:** Develop native mobile apps or integrate into existing university apps.
  - **Personalized Information Retrieval:** Securely integrate with the university's Student Information System (SIS) for tailored advice.
  - **Proactive Notifications:** Implement system for sending relevant alerts (e.g., registration reminders).
  - **Integration with Learning Management Systems (LMS):** Connect to answer assignment or lecture questions.
  - **Voice-to-Voice Interaction:** Implement text-to-speech for fully auditory conversations.
4. **Continuous Learning and Self-Improvement:** Implement active learning pipelines where human annotators review unhandled queries to refine training data and use A/B testing.
5. **Enhanced Analytics and Reporting:** Develop a comprehensive dashboard for administrators to track usage metrics, identify knowledge gaps, and optimize performance.
6. **Admin Panel for FAQ Management (Laravel-Powered Backend)**

Admin Dashboard Features:

- Add/Edit FAQs: Allow admins to update FAQs without coding.
- Analytics Dashboard: Monitor frequently asked questions, user engagement, and response accuracy.
- Automated Suggestion System: AI-powered recommendations for improving FAQ coverage.

User Feedback Integration:

- Let users flag incorrect answers, which admins can review and refine.

These recommendations provide a clear roadmap for JusLab's evolution into an even more indispensable and intelligent assistant for the PGDIT program.

**BIOGRAPHY**  
**OF**  
**MD. ZAHIDUL ISLAM SOHAN**

<b>Name</b>	<b>Md. Zahidul Islam Sohan</b>
<b>Academic Background</b>	<b>PGDIT</b> (Information Technology), Jahangirnagar University (IIT), Ongoing (2025)  <b>MBA</b> (Accounting), Government Titumir College, 2023  <b>BBA</b> (Accounting), Government Titumir College, 2018  <b>Web Programming</b> (PHP/MySQL), Dhaka University (IIT), 2019
<b>Present Position</b>	<b>Software Engineer (PHP Laravel)</b> Experienced in PHP (Laravel), JavaScript, jQuery, MySQL, React.js, Next.js, Django, web scraping, browser extensions, and WordPress plugin development — with hands-on work across various CMS and frameworks at <b>Thesoftking Limited.</b>
<b>Experience</b>	<b>Software Engineer (PHP Laravel)</b> Thesoftking Limited (Oct, 2020 - Present) <b>Software Developer (Java - Spring Boot)</b> MerinaSoft (July, 2020 - Sept, 2020)  <b>Former Accounting Teacher (Accounting, Finance, Management)</b> Various Coaching Centers, Mirpur, Dhaka (2016 – 2018)

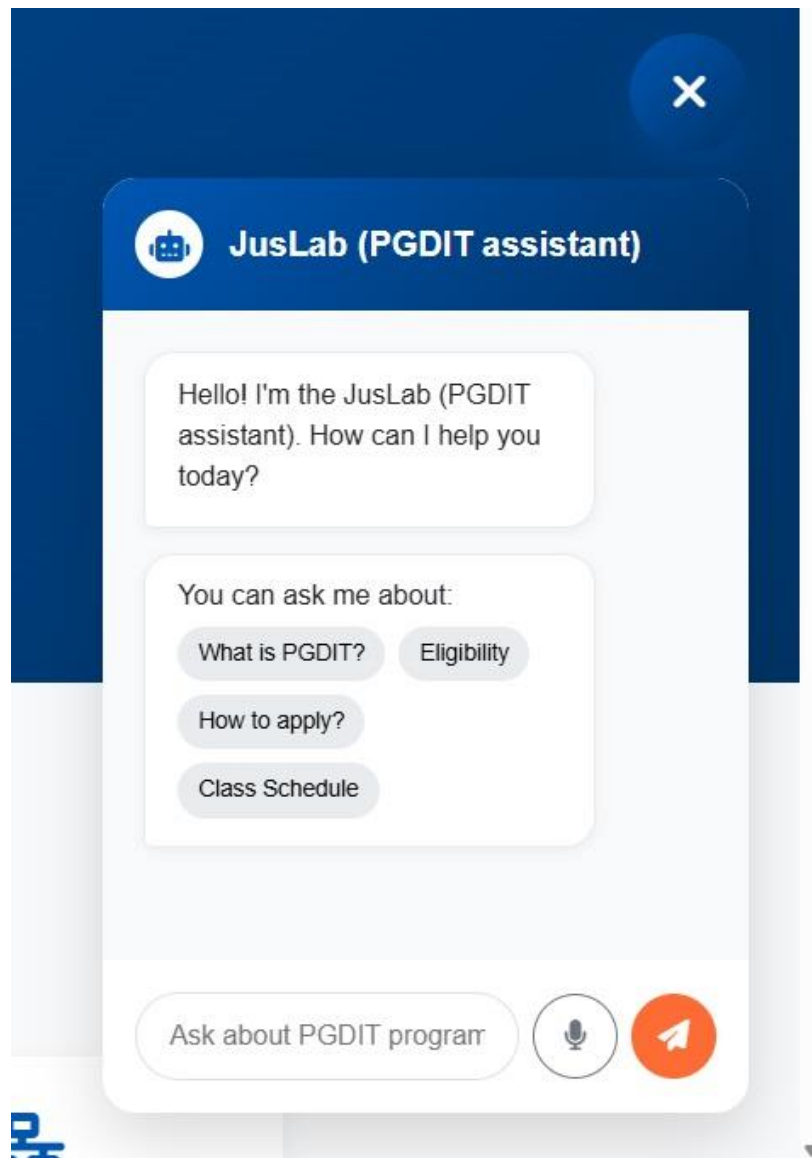
## BIOGRAPHY OF Waliullah

<b>Name</b>	<b>Waliullah</b>
<b>Academic Background</b>	<p>PGDIT (Information Technology), Jahangirnagar University (IIT), Ongoing (2025)</p> <p><b>M.A, Humanities</b> (2014-15). Govt. Madrasah- E-Alia, Dhaka, Bangladesh. under <b>Islamic University, Kushtia</b> Diploma in Enterprise Systems Analysis and Design – <b>C#.NET (ESAD-CS)</b> under <b>IsDB-BISEW IT Scholarship</b> (2016), Dhaka, Bangladesh</p>
<b>Present Position</b>	<p><b>Manager, Software Business</b> <b>Leadership &amp; Strategy:</b> Lead and mentor a high- performing software development team while driving architecture planning and technical decision-making.</p> <p><b>Full-Stack Development:</b> Design, develop, and maintain modern applications using ASP.NET Core MVC/API, HTML5, CSS3, JavaScript, Bootstrap, jQuery, Angular.</p> <p><b>Cloud Solutions:</b> Architect and implement Azure IaaS &amp; PaaS solutions, optimizing performance, security, and cost efficiency.</p> <p><b>Microsoft Ecosystem Integration:</b> Develop and integrate solutions with SharePoint, Power BI, and Power Platform at Corporate Projukti Ltd</p>
<b>Experience</b>	<p><b>Asst. Manager, Software</b> [July-2024, Feb-2025] <b>Software Engineer</b> [Jan-2021, Dec-2021] <b>Jr. Software Engineer</b> [Jan-2018, Dec-2020] <b>Trainee Software Engineer</b> [Feb-17, Dec-17] at the same company, I have been working since 2017.</p>

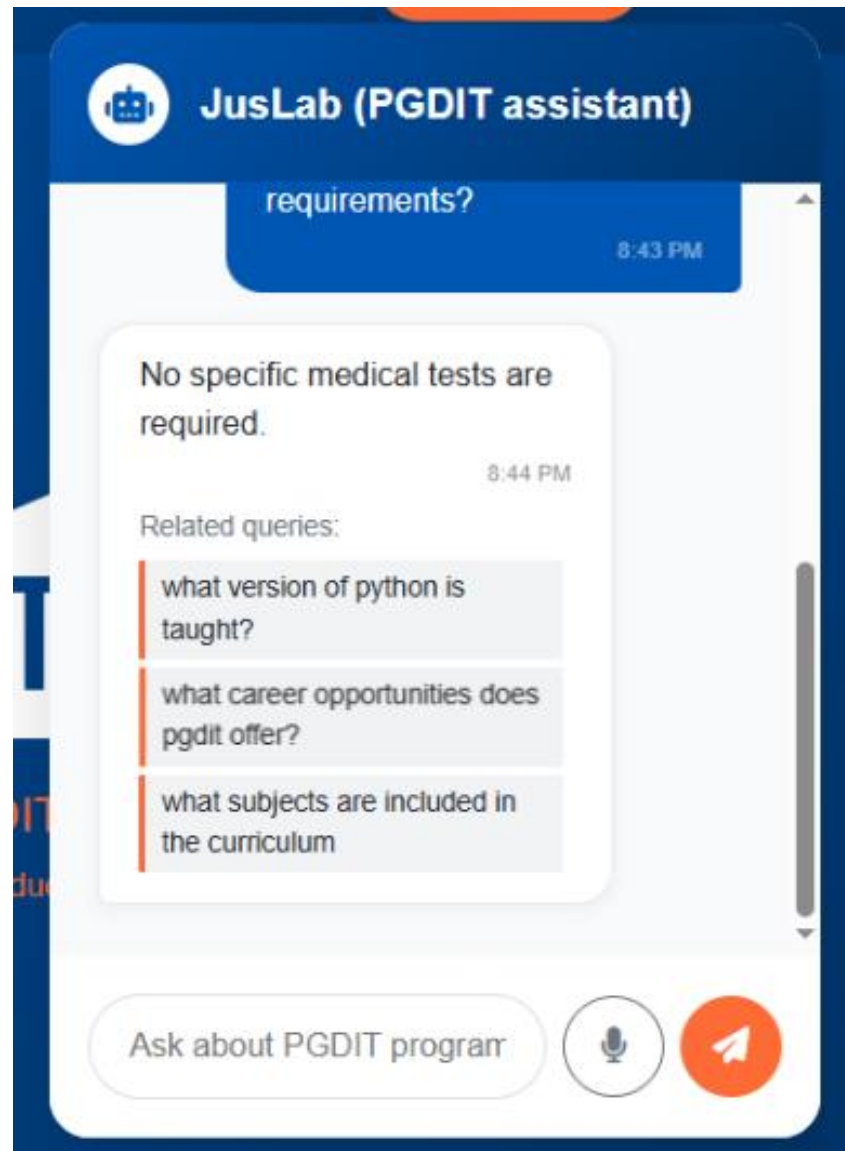
## [APPENDIX/APPENDICES]

### Appendix A: System Screenshots

1. Figure A1 - Chat Interface: Demonstrates the user interaction flow with sample queries and responses.

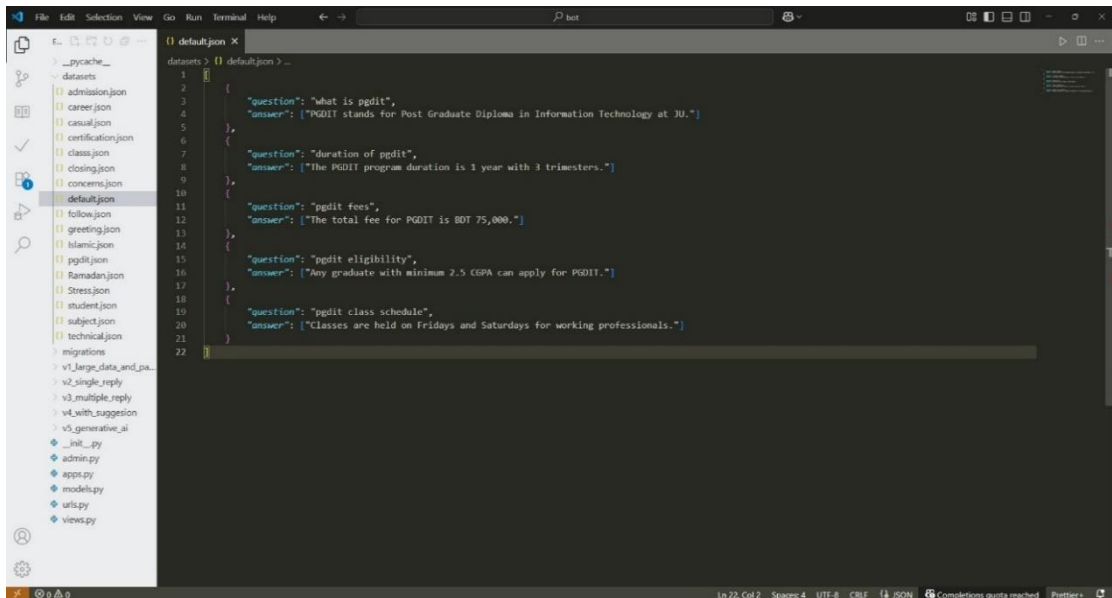


2. Figure A2 - Voice Input Feature: Shows the speech-to-text functionality in operation.



## Appendix B: Knowledge Base Structure

The chatbot utilizes a JSON-based knowledge repository with the following structure:



## Appendix C: Technical Specifications

The implementation leverages:

- Django (Backend framework)
- Bootstrap 5 (Frontend components)
- Web Speech API (Voice interaction)
- JSON (Knowledge base)
- REST API (Cross-platform support)
- cPanel (Live deployment)

## Appendix D: System Limitations

Key constraints include:

1. Static knowledge base requiring manual updates
2. English language support only
3. No real-time data integration with university systems