B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

OOMD Mini Project Report

LANGUAGE AGNOSTIC CHATBOT

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

Navneet Pitani
Neha Susan Baiju
Nidhi Pramod Nambiar
Nishta Khariwal

1BM23CS208
1BM23CS210
1BM23CS212
1BM23CS217

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2025-26

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Navneet Pitani (1BM23CS208), Neha Susan Baiju (1BM23CS210), Nidhi Pramod Nambiar (1BM23CS212), Nishta Khariwal (1BM23CS217) students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this OOMD Mini Project entitled "LANGUAGE AGNOSTIC CHATBOT" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester August 2025- December 2025. I also declare that to the best of our knowledge and belief, the OOMD mini Project report is not from part of any other report by any other students.

Signature of the Candidate

Navneet Pitani (1BM23CS208)

Neha Susan Baiju (1BM23CS210)

Nidhi Pramod Nambiar (1BM23CS212)

Nishta Khariwal (1BM23CS217)

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOMD Mini Project titled "**LANGUAGE AGNOSTIC CHATBOT**" has been carried out by Navneet Pitani (1BM23CS208), Neha Susan Baiju (1BM23CS210), Nidhi Pramod Nambiar (1BM23CS212), Nishta Khariwal (1BM23CS217) during the academic year 2025-2026.

Signature of the Faculty in Charge (Your Guide Name)

Table of Contents

Chapter 1: Problem Statement

Problem Statement: Language Agnostic Chatbot for Educational Institutions

Educational institutions such as schools and colleges often accommodate students, parents, and faculty members from diverse linguistic backgrounds. Communication gaps arise when institutional information, announcements, academic queries, or administrative support is available primarily in one or two dominant languages. This leads to misunderstandings, delays in information access, and reduced inclusivity for non-native speakers.

To address this challenge, there is a need for an intelligent system that can **interact seamlessly in multiple languages**, ensuring equitable access to information for all stakeholders in an educational ecosystem.

This project focuses on designing and modelling a **Language Agnostic Chatbot specifically for schools and colleges**. The chatbot should be capable of accepting queries in any supported language, automatically detecting the language, converting the input into a unified intermediate representation, and generating accurate responses in the user's preferred language. It will help users obtain information related to timetables, admission queries, fee details, academic schedules, exam dates, notices, faculty information, campus facilities, and general FAQs without language barriers.

The system will be designed using **object-oriented modelling principles**, incorporating abstraction, encapsulation, inheritance, and polymorphism. Major components include language detection modules, translation handlers, a context-aware conversation manager, domain-specific knowledge bases for academic workflows, and extensible interfaces to easily add new languages or institutional data. The chatbot must maintain conversational context, provide accurate information, and offer a consistent multilingual experience.

The final outcome of the project will include detailed **UML diagrams** such as use case diagrams, class diagrams, sequence diagrams, and activity diagrams that collectively describe a scalable, maintainable, and robust multilingual chatbot tailored for the needs of educational institutions.

Chapter 2: Software Requirement Specification

SRS : Language Agnostic Chatbot for Campus Offices

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to outline the requirements and specifications for the development of a Multilingual Campus Chatbot. It will provide a clear understanding of the project objectives, scope, and deliverables for a solution that makes campus information accessible to students in multiple languages.

1.2 Scope of this Document

This document defines the overall functioning and objectives of the Language Agnostic Campus Chatbot. It includes a description of estimated development cost and timeline for the project.

1.3 Overview

The Campus Chatbot is a software solution designed to streamline communication between students and campus offices, automating the handling of routine queries such as fee deadlines,timetable changes, and scholarship information. It enables students to receive conversational guidance in major regional languages, minimizing queues and easing communication issues.

2. General Description

The Campus Chatbot will serve the needs of students and administrative staff by answering frequently asked questions in Hindi, English, and at least three other regional languages. Features include context-aware conversation, intent recognition, human fallback for unresolved queries, and daily interaction logs for iterative improvement. The platform is accessible via the college website and mainstream messaging apps, and is designed for ongoing maintenance by student volunteers after the hackathon.

3. Functional Requirements

3.1 Multilingual Query Handling

Understands and responds to queries in at least five Indian languages (including Hindi and English).
Allows language switching mid-conversation without loss of context.

3.2 Intent and Context Management

Accurately identifies user intent from conversational input.
Maintains conversational context across multiple question-and-answer turns.

3.3 FAQ Integration and Content Curation

Ingests and indexes circulars, PDFs, and FAQ documents for answer retrieval.
Provides up-to-date answers based on curated institutional content.

3.4 Human Escalation and Fallback

Detects when unable to answer and seamlessly passes users to designated staff.
Notifies users when their query is outside of chatbot scope.

3.5 Conversation Logging and Monitoring

Logs all interactions with users for analytics and improvement.
Supports daily export/viewing of conversation logs by administrators.

3.6 Platform Integration

Embeds chatbot widget on college website.
Integrates with WhatsApp, Telegram, or other mainstream messaging platforms.

4. Interface Requirements

4.1 User Interface

Simple, chat-based UI accessible on web and mobile.
Supports regional language input via keyboard and speech (where possible).
Clear prompts for language and category selection.

4.2 Integration Interfaces

Connects to cloud-based NLP/conversational AI platforms with drag-and-drop workflow
(e.g., Dialogflow, Rasa).
Integrates with document repositories (for circulars, PDFs).
APIs for human handover and IT support.

5. Performance Requirements

5.1 Response Time

Message response should not exceed 2 seconds during normal operation.

5.2 Scalability

Handles at least 1000 simultaneous user sessions during peak times.

5.3 Data Integrity

Ensures accurate mapping from questions to official answers across all languages.

6. Design Constraints

6.1 Hardware Limitations

Compatible with existing college website infrastructure and standard mobile devices.

6.2 Software Dependencies

Utilizes cloud/NLP services for language support and conversational intelligence.
Built for ease of content update and maintenance by student volunteers.

7. Non-Functional Attributes

7.1 Security

Implements secure authentication/authorization for administrators.
Protects user data and maintains privacy in all logs.

7.2 Reliability

Designed for high uptime, redundant hosting, and graceful error handling.

7.3 Scalability

Easily extendable to more languages or campuses as needs evolve.

7.4 Portability

Operable on all common mobile and desktop browsers.

7.5 Usability

Intuitive interface, supports beginners and students not comfortable with English.

7.6 Reusability

Modular code and architecture to enable easy feature extension and handover.

7.7 Compatibility
Compatible with Chrome, Firefox, Edge, and Safari browsers.

7.8 Data Integrity
Ensures correct and up-to-date answers by syncing with official sources.

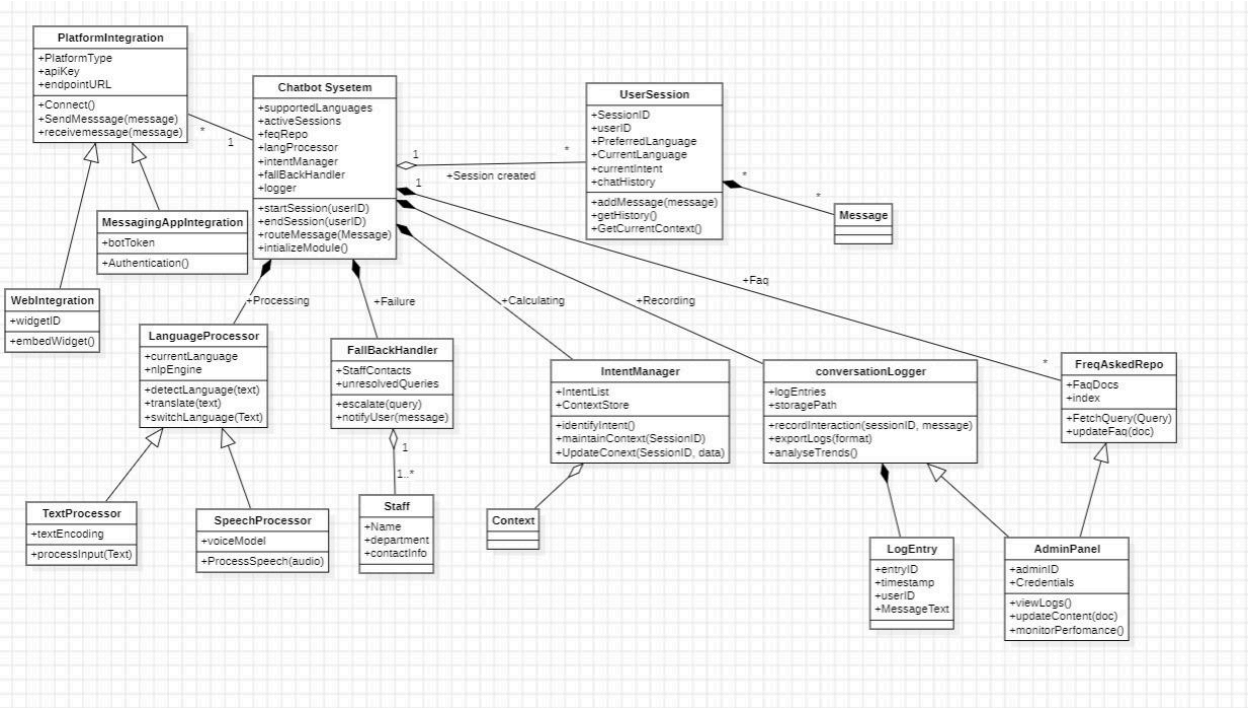
8. Preliminary Schedule and Budget

The development of the Campus Chatbot is estimated to take 3-4 months with a budget of Rs 25,000. This covers planning, design, multilingual content curation, platform integration, testing, and documentation for future student volunteers.

Chapter 3: Class Modeling

Class diagram with explanation (Advanced features to be included)

Note: Explain the relevance of each class.



3.1 Class Diagram

EXPLANATION OF EACH CLASS (Relevance + Advanced Features) :

1. ChatbotSystem

Role: Central orchestrator of the chatbot.
Why it's important:

- Manages user sessions
- Coordinates interactions between all modules
- Handles routing of messages
  Advanced Features:

- Uses composition to include LanguageProcessor, IntentManager, Logger
- Implements facade pattern to simplify interactions
- Provides scalable modular initialization

2. UserSession

Role: Represents one user's active chat session.
Relevance:

- Stores language preference, context, history — vital for personalized responses
- Maintains continuity across multiple messages
  Advanced Features:
- Uses encapsulation for chat history
- Session context is shared with IntentManager

3. Message

Role: Represents an individual message from user or bot.
Relevance:

- Tracks who said what and when
- Acts as a data carrier between components
  Feature:
- Lightweight, immutable design

4. LanguageProcessor

Role: Handles language detection, translation, and switching.
Relevance:

- Core component of a *language-agnostic* system
- Ensures all user inputs are processed uniformly
  Advanced Features:

- Uses injecting of different NLP engines (plug-and-play architecture)

5. TextProcessor

Role: Handles text normalization (tokenization, encoding).
Relevance:

- Prepares raw text for NLP & translation
  Advanced Features:

- Acts as a helper class to LanguageProcessor

6. SpeechProcessor

Role: Converts speech to text.
Relevance:

- Adds optional advanced feature: voice support
  Advanced Features:

- Allows model upgrades without changing other components

7. IntentManager

Role: Determines what the user wants (intent extraction).
Relevance:

- Core to chatbot intelligence

- Helps generate the correct response
  Advanced Features:

- Implements Context Management

- Stores conversation context in `contextStore`

8. Context

Role: Represents a piece of contextual information.
Relevance:

- Helps understand follow-up queries:
  → “When is my exam?” → “What about math?”
  Advanced Features:

- Simple but extensible structure

9. FreqAskedRepo

Role: Stores FAQ documents and manages semantic search.
Relevance:

- Used for answering common queries

- Reduces load on school staff
  Advanced Features:

- Maintains an index for faster query retrieval

- Supports content updates via AdminPanel

10. FallBackHandler

Role: Manages errors, escalation and unresolved queries.
Relevance:

- Ensures chatbot doesn’t get stuck

- Escalates to human staff when needed
  Advanced Features:

- Maintains log of unresolved queries

- Integrates with Staff class for escalation

11. Staff

Role: Represents human staff members (admin office, faculty).
Relevance:

- Supports escalation when chatbot can't answer
  Advanced Features:

- Provides department-structured contact routing

12. ConversationLogger

Role: Logs interactions for monitoring and analysis.
Relevance:

- Crucial for improving chatbot accuracy

- Helps admins track usage patterns
  Advanced Features:

- Performs analytics (trend analysis)

- Exports data in multiple formats

13. LogEntry

Role: Represents one log record.
Relevance:

- Base unit of the analytics system

14. AdminPanel

Role: Backend management panel for institution staff.
Relevance:

- Allows updating FAQs

- Monitoring system performance

- Accessing logs
  Advanced Features:

- Integrates with ConversationLogger and FreqAskedRepo

15. PlatformIntegration (Abstract Class)

Role: Base class for different communication platforms.
Relevance:

- Allows integration with WhatsApp, Telegram, Websites, etc.
  Advanced Features:

- Abstract class with overridable methods

- Enforces structure of platform connectors

16. MessagingAppIntegration

Role: Works for apps like WhatsApp/Telegram.
Relevance:

- Handles authentication & message routing
  Advanced Features:

- Extends PlatformIntegration

- Adds botToken for secure messaging

17. WebIntegration

Role: Connects chatbot to websites/portals.
Relevance:

- Most schools embed chatbot in website
  Advanced Features:

- Minimalist class with widget embedding

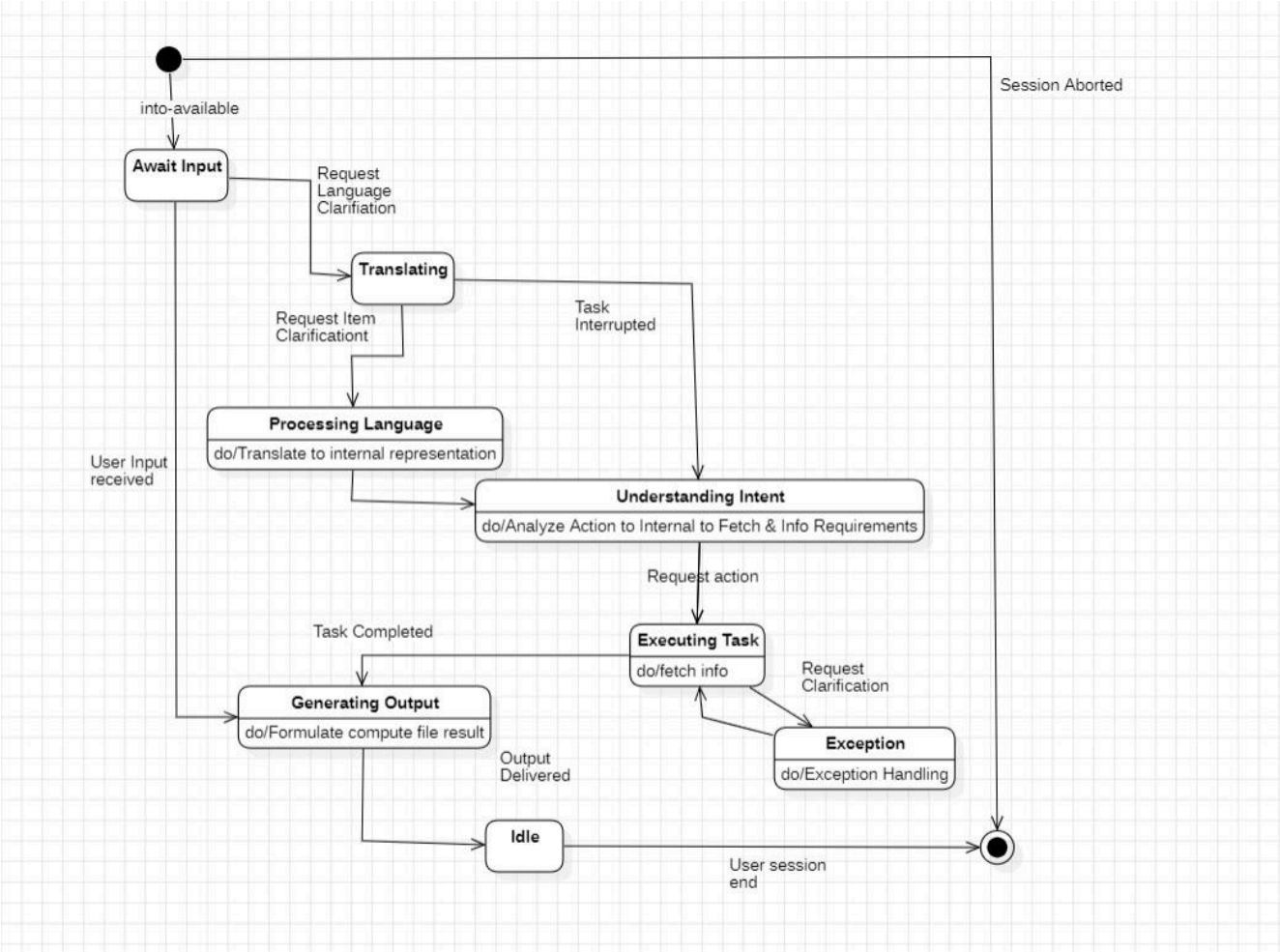
Chapter 4: State Modeling

State diagram with explanation

(Advanced features to be included)

Note: Explain the relevance of each state

Explain the relevance of each Event



4.1 State Model

State Machine Diagram – Explanation

This state diagram models the **dynamic behavior of the Language-Agnostic Chatbot** used in schools and colleges. It shows how the chatbot behaves from the moment a session starts until it ends, including translation, intent processing, task execution, output generation, and exception handling.

1. Initial State (Black Dot)

Purpose:
Marks the start of the chatbot's operational cycle for a new session.

Relevance:
Indicates that the system is ready and waiting for the first user input.

2. Await Input

Purpose:
The primary idle state where the chatbot waits for a user's text or voice message.

Relevance:

- All conversations begin here.
- Central listening point for user messages.
- Triggers the language-processing pipeline once input arrives.

Advanced Feature:
Supports timeout or inactivity detection → may lead to session abort.

3. Translating

Purpose:
Handles language detection, translation to an internal representation, and clarification if needed.

Relevance:

- Critical for a language-agnostic system.
- Ensures consistent processing regardless of the language used.
- Requests clarification if the detected language is uncertain or mixed.

Advanced Feature:
Interruptions handled here (e.g., when a message is malformed or incomplete).

4. Processing Language

Purpose:
Converts the cleaned/translated text into a format understandable by the chatbot's internal modules (tokenization, semantic representation).

Relevance:

- Ensures uniform internal semantics for multilingual input.
- Reduces ambiguity before intent extraction.

Advanced Feature:
Uses NLP pipelines (syntax, entity extraction, semantic parsing).

5. Understanding Intent

Purpose:
Analyzes the processed text to determine what the user wants (e.g., "When is the exam?", "Show timetable", "What is the fee?").

Relevance:

- Core state for chatbot intelligence.
- Produces actionable intent + required information fields.

Advanced Feature:
Context handling → recognizes follow-up queries like "What about the lab?"
Can return to *Await Input* if context is insufficient.

6. Executing Task

Purpose:
Performs the actual work needed to respond:

- Fetching FAQ data
- Retrieving timetable

- Querying student information
- Processing admin rules
- Running computations

Relevance:
The "action" state responsible for producing meaningful output.

Advanced Feature:
Supports clarification requests when information is missing (e.g., “Which semester?”).

7. Exception

Purpose:
Handles problems such as:

- Unsupported queries
- System errors
- Missing information
- Internal issues

Relevance:
Preserves user experience by avoiding abrupt failures.

Advanced Feature:
Implements fallback logic → may escalate to staff or request clarification.

8. Generating Output

Purpose:
Forms the final, user-readable reply in the appropriate output language.

Relevance:

- Applies translation back to user’s preferred language.
- Ensures response correctness and formatting.
- Could include file generation (PDF, timetable, receipt).

Advanced Feature:
Output can be in text, voice, or structured file formats.

9. Idle

Purpose:
A neutral waiting state after sending the reply.

Relevance:
Prepares system for next input or session termination.

10. Final State (Bullseye Symbol)

Purpose:
Session ends due to:

- User logout
- Inactivity
- Admin termination
- Error recovery

Relevance:
Marks termination of conversation and clears session context.

Relevance of Each Event / Transition :

Below is a detailed description of every arrow/transition shown in the diagram.

1. Event: "User input received" → Processing begins

Relevance:
Triggers the main flow of language detection and translation.

2. Language Clarification Request

Occurs from *Translating* when input language is:

- Ambiguous

- Mixed-language
- Not recognized

Relevance:
Ensures translation accuracy before proceeding.

3. Item Clarification Request

Occurs when Processing Language encounters unknown terms (e.g., “Which department?”, “Which date?”).

Relevance:
Enhances precision.

4. Task Interrupted

Occurs when input is incomplete or user sends a correction.

Relevance:
Enables dynamic conversation correction without restarting.

5. Request Action

From *Understanding Intent* to *Executing Task.*

Relevance:
Defines what operation chatbot must perform (fetch, calculate, verify, retrieve).

6. Request Clarification (From Executing Task or Exception)

Triggered when task cannot proceed due to missing details.

Relevance:
Avoids invalid or incomplete output.

7. Task Completed

Sent by *Executing Task* to *Generating Output.*

Relevance:
Marks the successful completion of backend logic.

8. Output Delivered

From *Generating Output* to *Idle.*

Relevance:
Indicates reply sent; system waits for next input.

9. Session Aborted

Transition from any state to final termination.

Relevance:
Handles:

- User exits
- Timeout
- Fatal error
- Manual admin closure

10. User Session End

Moves from *Idle* to Final State.

Relevance:
Conversational loop completed normally.

Chapter 5: Interaction Modeling

Use case diagram, Sequence Diagram, Activity Diagram with explanation

(Advanced features to be included)

Note: Use case diagram

- Explain the relevance of each actor
- Explain the relevance of each use case

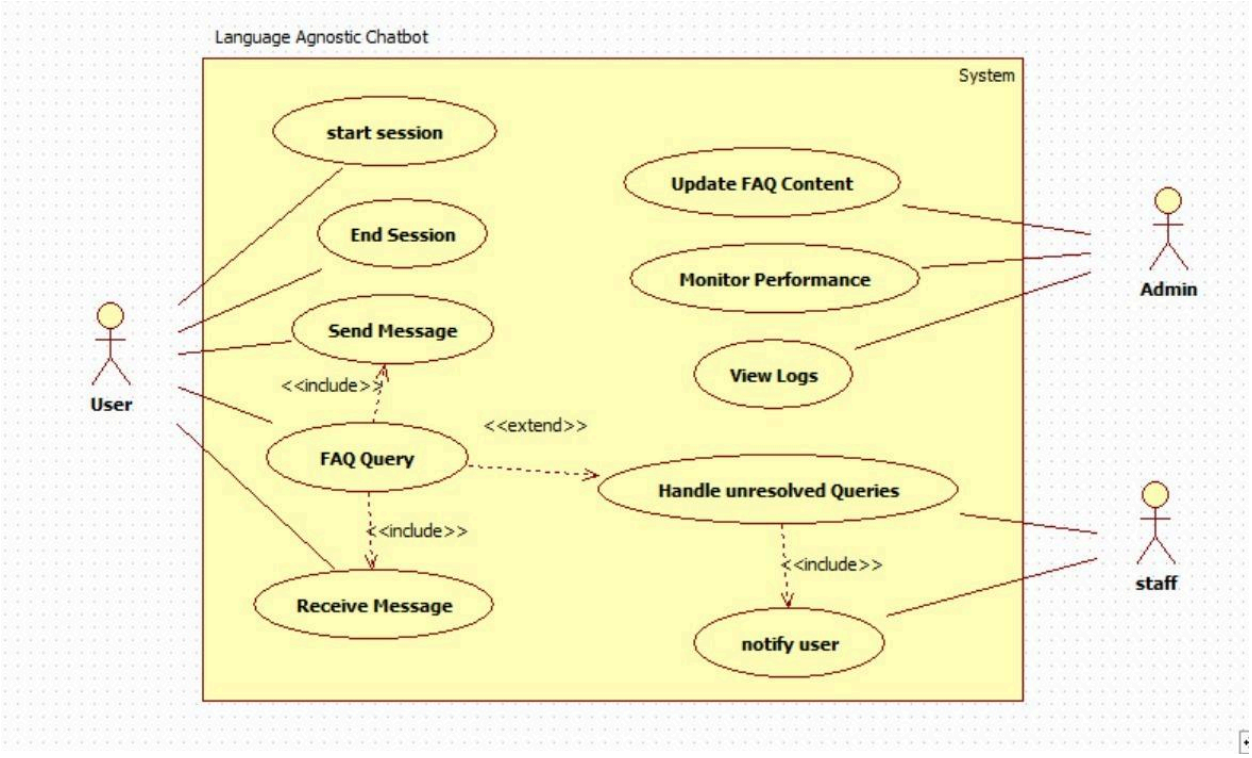
Sequence Diagram

- Explanation of your diagram

Activity Diagram

- Swimlanes
- Splitting and merging of control
- Explain your diagram

5.1 USE CASE DIAGRAM



5.1 : Use Case Diagram

1. USE CASE DIAGRAM – EXPLANATION

Actors and Their Relevance :

1. User
    - Primary actor interacting with the chatbot.
    - Sends queries in any language.
    - Receives translated response.
    - Starts and ends chat sessions.

2. Admin
    - Maintains and updates FAQ content.
    - Monitors system performance metrics.
    - Views system logs for debugging or optimization.

3. Staff
    - Handles unresolved or complex queries that the chatbot cannot answer.
    - Sends manual responses to users when needed.

Use Cases and Their Relevance :

1. Start Session
    - User begins interaction.
    - Initializes session context and language preferences.

2. End Session
    - Closes the interaction.
    - Saves history and logs for analytics.

3. Send Message

- User sends message in any language.
- Chatbot begins processing pipeline.
- Includes language detection step.

4. Receive Message

- User receives final output after translation and processing.

5. FAQ Query

- Chatbot checks internal FAQ database.
- Enables instant responses to common questions.
- Uses: <> Send Message and <> Receive Message.

6. Handle Unresolved Queries

- Triggered when FAQ/NLP engine cannot answer.
- Logs issue and alerts staff.
- Extends FAQ Query.

7. Notify User

- System alerts user about unresolved queries.
- Includes final staff-generated answer.
- Uses: <> Handle Unresolved Queries.

8. Update FAQ Content (Admin)

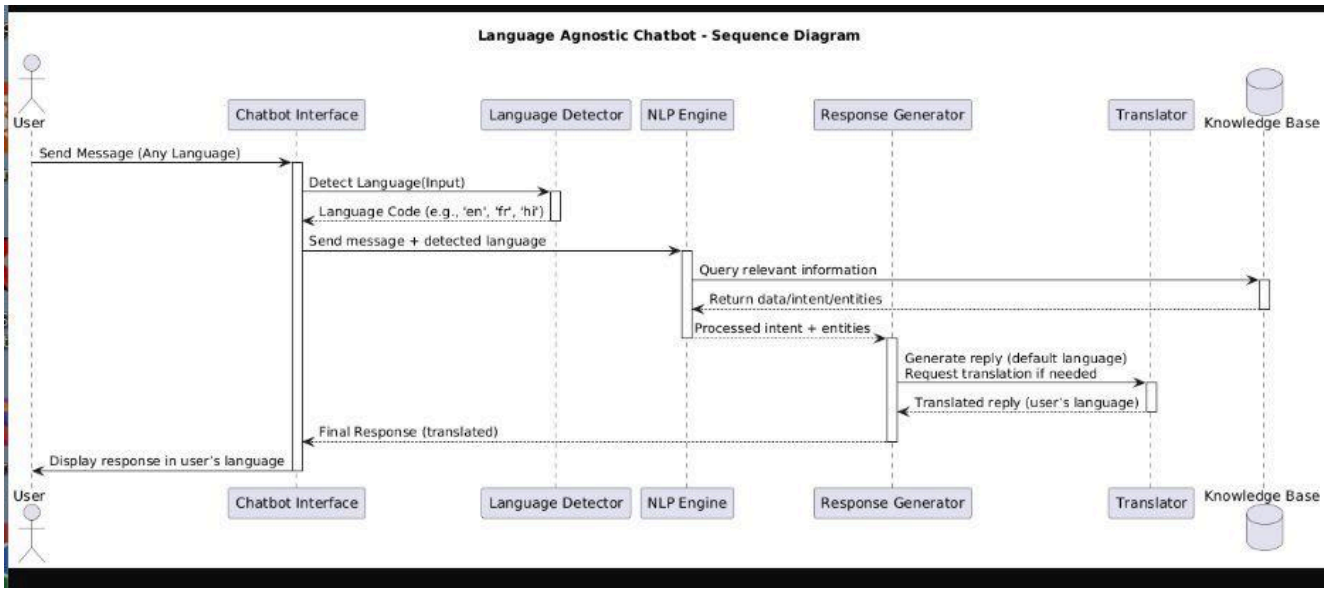
- Admin adds/edits/deletes FAQ items.
- Ensures chatbot is always updated.

9. Monitor Performance (Admin)

- Admin checks system statistics:
- response time
- accuracy rate
- query volume

10. View Logs (Admin)

- Admin reviews conversation logs.
- Used for system improvement and debugging.



5.2 : Sequence Diagram

2. SEQUENCE DIAGRAM – EXPLANATION

The sequence diagram explains the internal flow of message processing in a Language-Agnostic Chatbot.

Step-by-Step Explanation

1. User → Chatbot Interface

- User sends message in any language.
- Example: French, Hindi, Kannada, etc.

2. Chatbot Interface → Language Detector

- Detects the input language.
- Extracts language code (e.g., “en”, “fr”, “hi”).

3. Language Detector → NLP Engine

- Passes detected language + original message.
- NLP engine performs:
- tokenization
- entity extraction
- sentiment analysis
- intent classification

4. NLP Engine → Knowledge Base

- Queries FAQ or knowledge repository.
- If answer exists → returns data.
- If not → message marked as unresolved.

5. Knowledge Base → Response Generator

- Generates reply in default system language.
- Adds structure and grammatical corrections.

6. Response Generator → Translator

- If user’s language ≠ system language:
- translates the final reply into user’s language.

7. Translator → Chatbot Interface

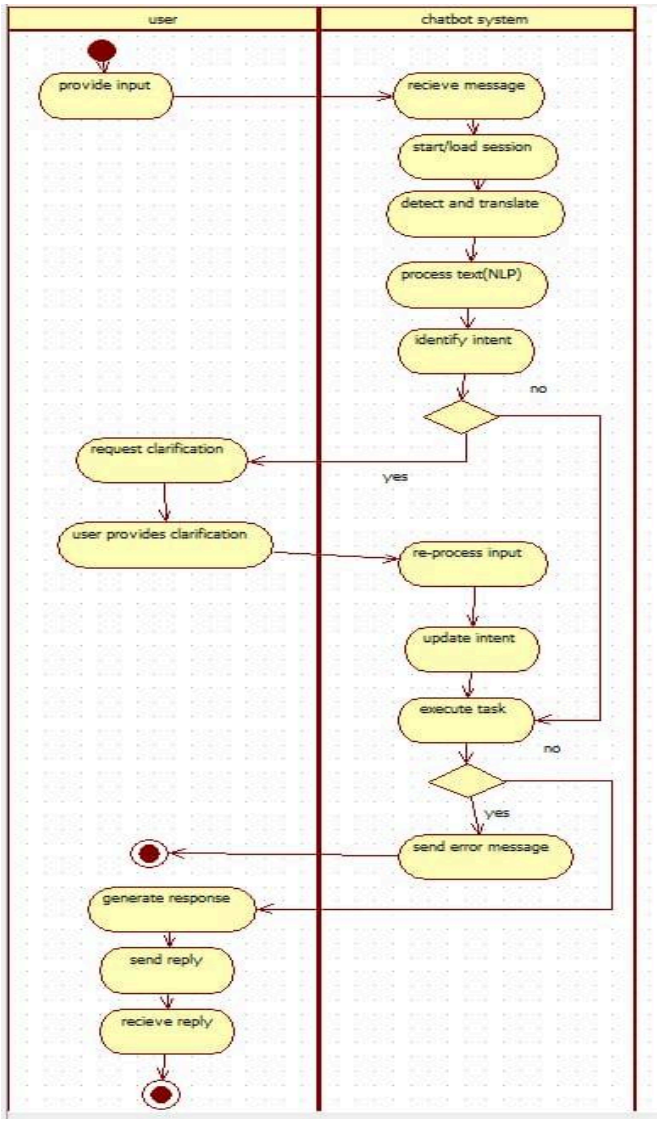
- Final translated response delivered to user.

8. Chatbot Interface → User

- User sees reply in the same language they originally typed.

Advanced Features Highlighted

- Automatic multilingual detection.
- NLP-driven intent recognition.
- Dynamic translation support.
- Knowledge Base query handling.
- Unresolved query routing.
- Context tracking across sessions.



5.3 : Activity Diagram

3. ACTIVITY DIAGRAM – EXPLANATION

The activity diagram shows the workflow using two swimlanes:

(1) User

(2) Chatbot System

Swimlanes

Swimlane 1: User

- Provide input
- Provide clarification (if required)
- Receive final reply

Swimlane 2: Chatbot System

- Receive message
- Start/load session
- Detect & translate
- NLP processing
- Intent identification
- Execute task
- Send error or output
- Generate final response

Step-by-Step Explanation

1. User provides input
    - User enters a message in any language.

2. Chatbot receives message
    - Input text transferred to system.

3. Start or Load Session
    - Loads previous context.
    - Tracks conversation thread.

4. Detect and Translate (if needed)
    - Identifies language of input.
    - Converts to system language for processing.

5. NLP Processing

- Performs:
- parsing
- tokenization
- intent recognition
- entity extraction

6. Decision: Intent Identified?

If NO:

- System asks for clarification.
- User gives additional details.
- Chatbot re-processes and updates intent.

If YES:

- Move to task execution.

7. Execute Task

Examples:

- Search FAQ database
- Retrieve KB article
- Perform computation
- Route unresolved query to staff

8. Decision: Task Successful?

If NO:

- Send error message to user.
- Inform staff if needed.

If YES:

- Proceed to response generation.

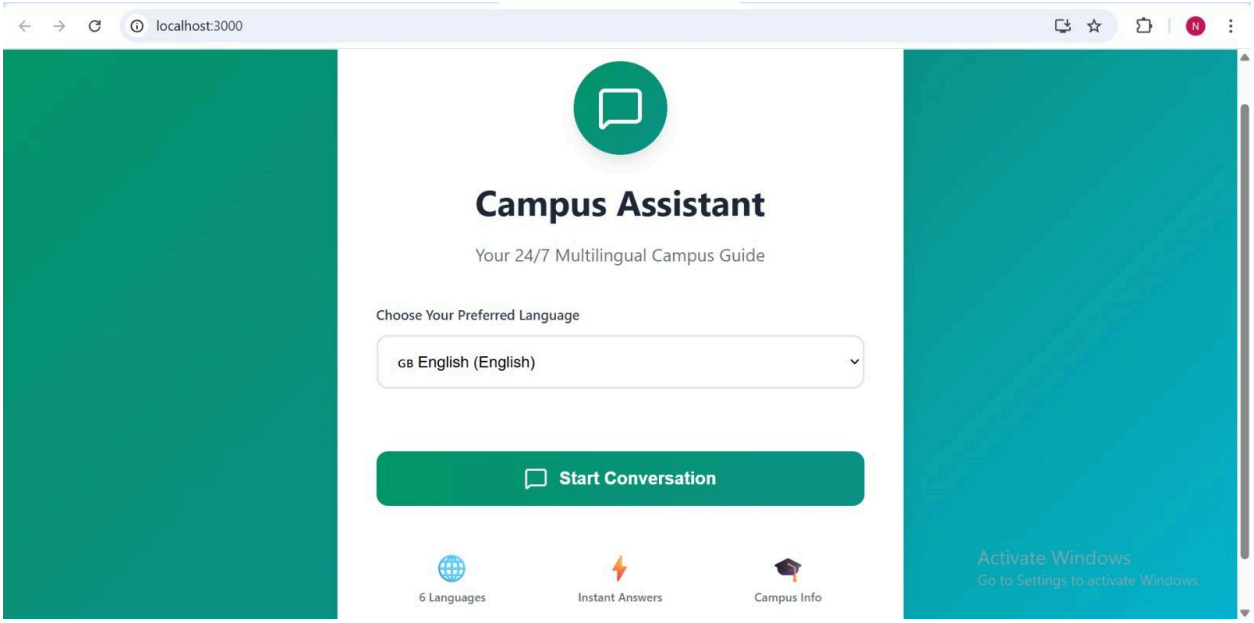
9. Generate Response
    - System prepares the final answer.

10. Send Reply
    - Message translated back into user's language.

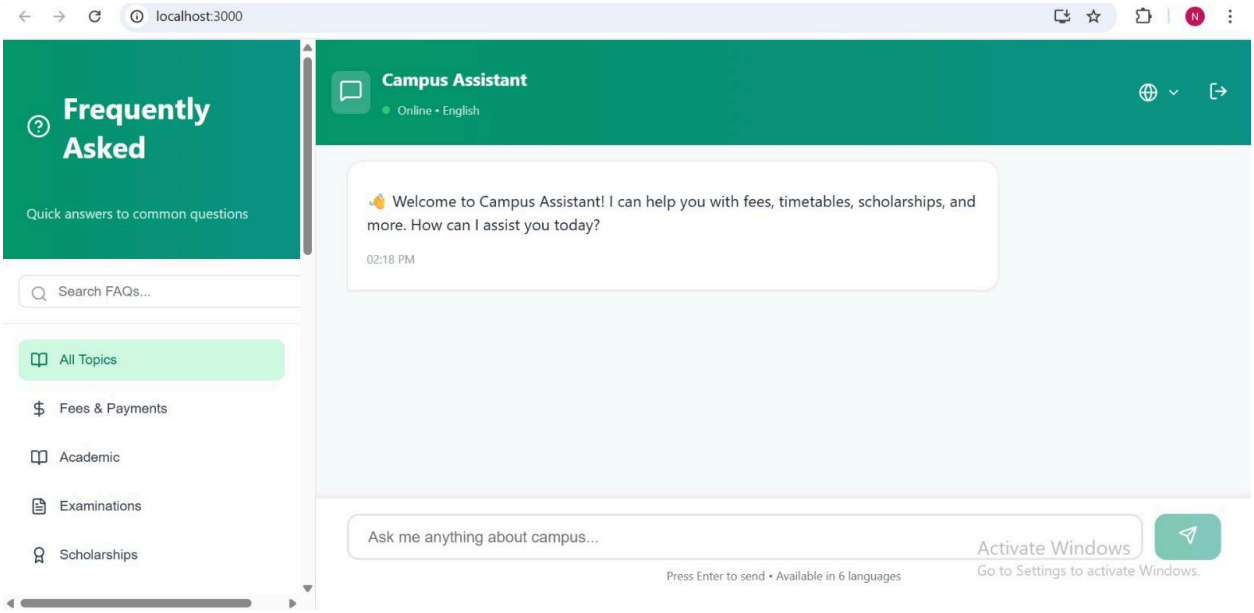
11. User Receives Reply
    - End of workflow.

Chapter 6: UI Design with Screenshots

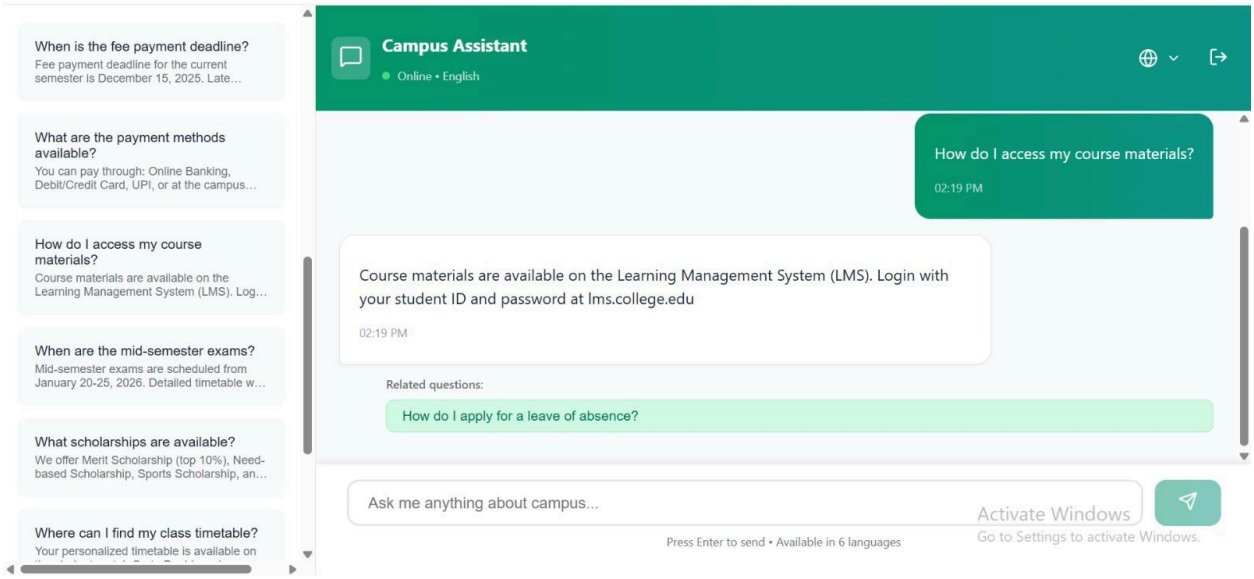
Insert screenshots of UI Design and explain



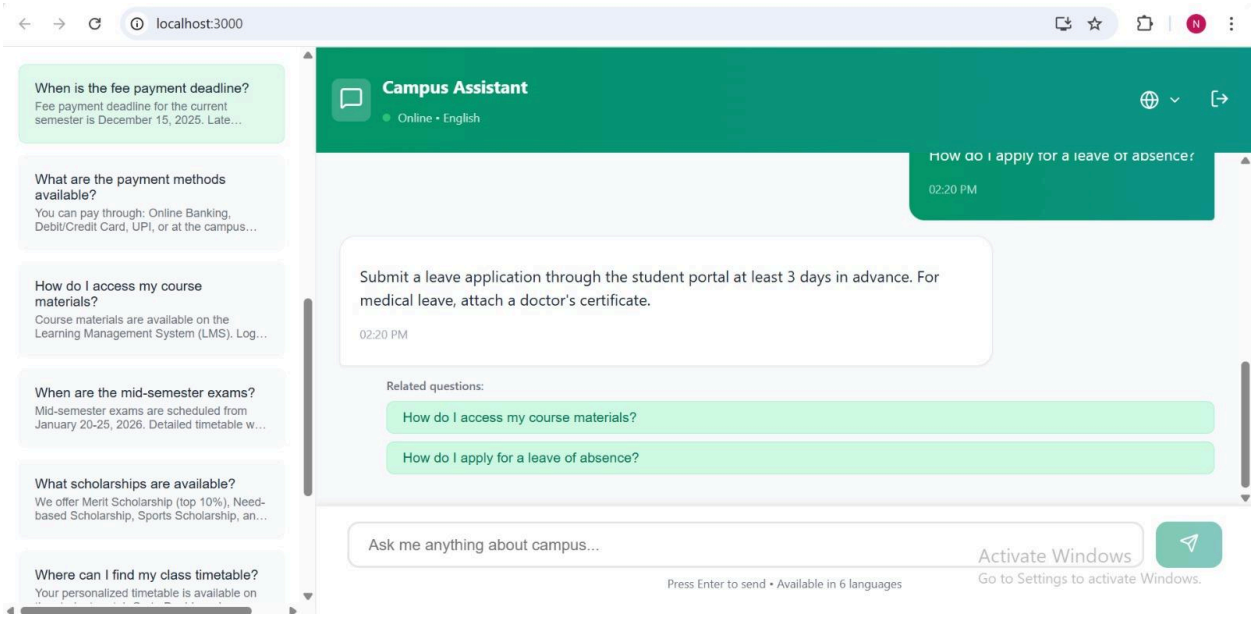
6.1 : Home Page 1



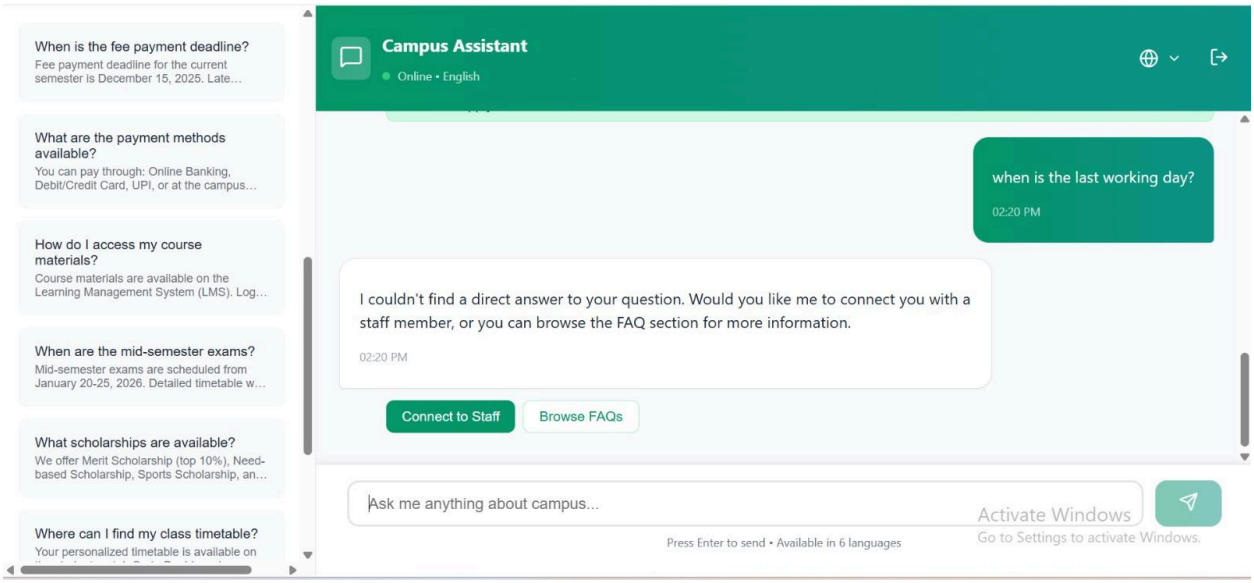
6.2 : Home Page 2



6.3 Input From User 1



6.4 Input From User 2



6.5 Output From Campus Assistant

****