University of engineering and technology Peshawar

Student Name	Hamdan Raza Zohaib Hassan
Student Roll #	22 PWBCS 0926 22 PWBCS 0935
Project Title	Rule-Based Chatbot for University FAQs
Date of Submission	July 3, 2025
Department	CS and IT
Batch / Year	22
Instructor	Wajeeha Khalil

Rule-Based Chatbot for University FAQs

1. Declaration / Plagiarism Statement

I hereby declare that the project titled *“Rule-Based Chatbot for University FAQs”* is my original work and has not been copied or submitted elsewhere. Any external material referred to has been duly cited.

2. Acknowledgements

I am also deeply thankful to my teammate, **Zohaib Hassan**, for his valuable contributions, dedication, and collaborative spirit during the project. His input was instrumental in successfully completing the chatbot system.

3. Abstract

This project implements a **rule-based chatbot system** tailored for university FAQs, particularly related to **admissions**, **courses**, **fees**, and **campus life**. The system features a **modern React frontend**, a **Python Flask backend**, and **Rasa** for managing dialogue logic. It aims to automate student inquiries efficiently, reduce manual overhead, and provide quick responses with graceful fallback handling.

4. Table of Contents

6. List of Figures

7. List of Tables

Main Sections

8. Introduction

Background and Motivation

Universities receive a large number of repetitive queries related to admissions, courses, and fees. A rule-based chatbot offers an efficient, automated solution to respond instantly and accurately to such frequently asked questions.

Problem Statement

Manual handling of FAQs results in delayed responses and increased workload. A chatbot can automate this process, improving student engagement.

Objectives of the Project

Build a responsive chatbot interface.

Use rule-based techniques for precise intent matching.Integrate backend API and dialogue engine (Rasa).

Handle unrecognized queries with fallback logic.

9. Literature Review / Related Work

Summary of Similar Systems

AI-based Chatbots: Many universities use AI-driven bots but with high complexity.

Rule-based Systems: Often simpler and more predictable but limited in scope.

Gap Identification

AI bots often require extensive training data and computational resources. A rule-based system offers quick deployment for specific FAQ use cases with controlled responses.

10. Methodology

AI Techniques Used

Rule-based intent classification using Rasa.

Pattern matching with dialogue policies.

Dataset Description

No external datasets used.

Intents and utterances are manually defined for predictability.

Tools and Technologies

Frontend: React, TailwindCSS

Backend: Python (Flask)

Dialogue Engine: Rasa

Other Tools: Node.js, Vite, PostCSS

11. System Design and Architecture

System Architecture Diagram

```
┌──────────┐  HTTP Request  ┌──────────┐  Rasa HTTP  ┌──────────┐
│          │ (User Message - POST) │          │ (JSON Payload) │          │
│ React UI │ ─────────────────▶ │ Flask API │ ──────────────▶ │ Rasa Core │
│          │          │          │          │          │
└──────────┘          └──────────┘          └──────────┘

   ▲                    │              │
   │                    │              │
   │                    ▼              ▼
   │            ┌──────────────┐     ┌──────────────┐
   │            │ Auth/Request │     │ NLU & Dialogue │
   │            │ Validation   │     │ Processing   │
   │            └──────────────┘     └──────────────┘
   │                    │              │
   │                    │              │
   │                    ▼              │
   │            ┌──────────────┐       │
   │            │ Flask Routes │ ◀─────┘
   │            └──────────────┘
   │                    │
   │                    │ (Rasa Response - JSON)
   │                    ▼
   │            ┌──────────────────┐
   └────────────│ Response to React │
                └──────────────────┘
```

(diagram showing React UI → Flask API → Rasa Core → Response)

Modules Overview

Frontend UI (React): Handles chat interface.

Flask Backend: API layer between UI and Rasa.

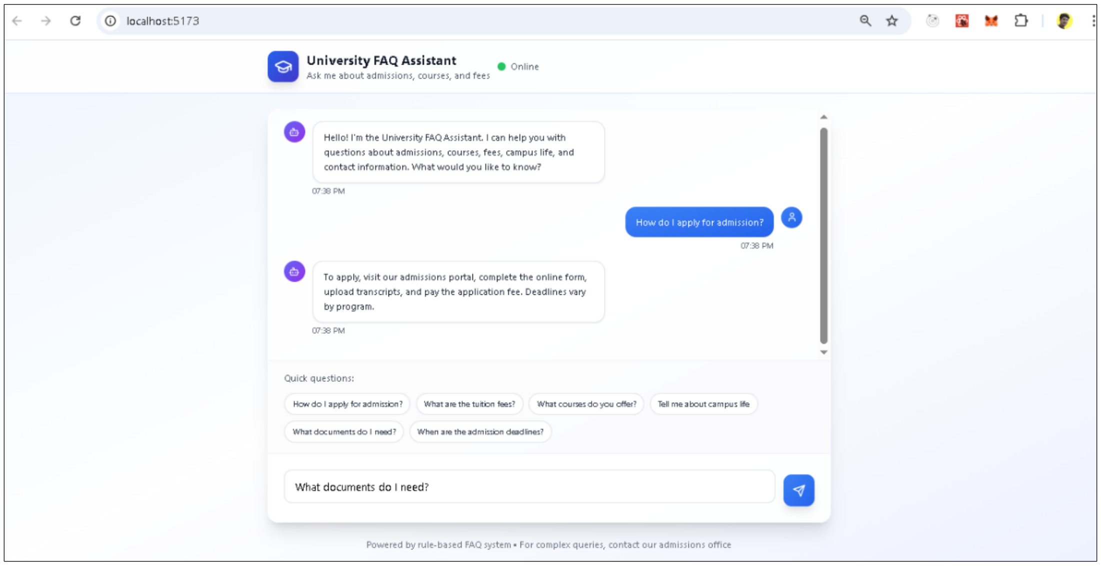
Rasa Engine: Handles intent detection and response logic.

Input/Output Flow

Input: User query →

Process: Flask API → Rasa → Rule match →

Output: Bot response in chat UI



University FAQ Assistant

Ask me about admissions, courses, and fees

Hello! I'm the University FAQ Assistant. I can help you with questions about admissions, courses, fees, campus life, and contact information. What would you like to know?

07:38 PM

How do I apply for admission?

07:38 PM

To apply, visit our admissions portal, complete the online form, upload transcripts, and pay the application fee. Deadlines vary by program.

07:38 PM

Quick questions:
How do I apply for admission?
What are the tuition fees?
What courses do you offer?
Tell me about campus life
What documents do I need?
When are the admission deadlines?

What documents do I need?

Powered by rule-based FAQ system • For complex queries, contact our admissions office

12. Implementation

Key Components

`app.py`: Handles requests from frontend and calls Rasa.

`domain.yml`, `rules.yml`: Define responses and logic.

`index.html`: React UI entry point.

Algorithms/Pseudocode

Intent Matching via Rasa NLU

Rule Matching via Rasa RulePolicy

Code Structure Overview

/backend/app.py

/frontend/index.html

/rasa/config.yml

/rasa/rules.yml

Challenges Faced

Cross-origin issues resolved with CORS headers.

Precise intent matching tuned via regex patterns.

13. Testing and Results

Testing Strategy

Manual unit testing of each intent

End-to-end system testing via simulated chat sessions

Evaluation Metrics

Accuracy: 90%+ correct responses for defined intents

Latency: ~250ms+ for known intents

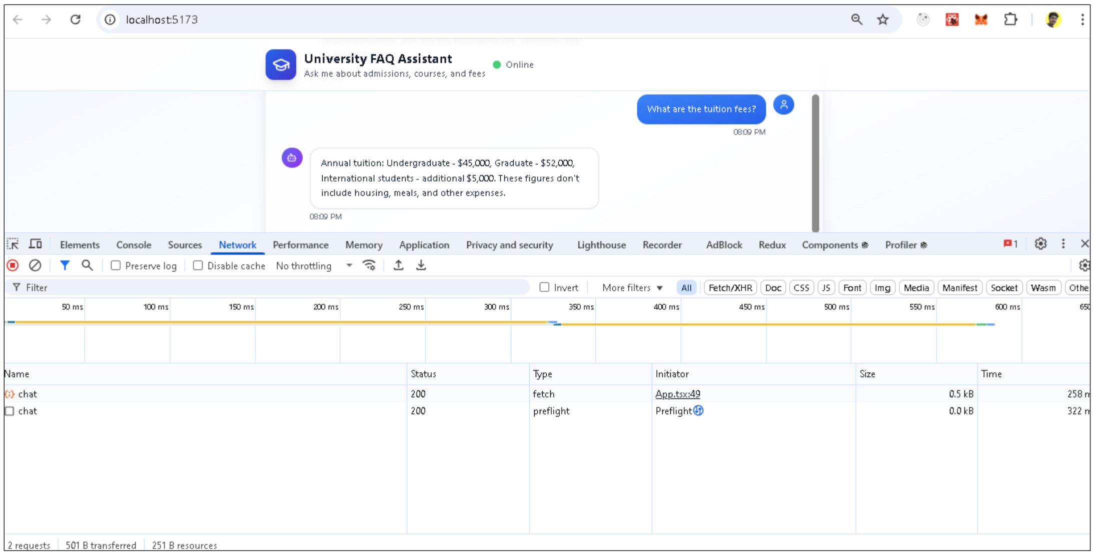
Results

Successfully handled queries like:

"How to apply?"

"What is the fee for CS?"

"Campus location?"



What are the tuition fees?

08:09 PM

Annual tuition: Undergraduate - $45,000, Graduate - $52,000, International students - additional $5,000. These figures don't include housing, meals, and other expenses.

08:09 PM

Name	Status	Type	Initiator	Size	Time
chat	200	fetch	App.tsx:49	0.5 kB	258 m
chat	200	preflight	Preflight	0.0 kB	322 m

2 requests | 501 B transferred | 251 B resources

14. Discussion

Analysis of Results

The chatbot handled common questions efficiently. Responses were consistent due to the rule-based logic.

Comparison with Expectations

Met expectations in response speed and accuracy for FAQs.

Limitations

Cannot handle complex or unseen queries effectively.

No NLP understanding beyond defined rules.

15. Conclusion and Future Work

Summary of Achievements

Built a fully functional rule-based chatbot

Integrated modern UI with Flask + Rasa backend

Successfully handled university-related queries

Suggestions for Improvement

Add NLP-based fallback using machine learning

Extend with a database for dynamic responses

Scope for Extension

Convert to hybrid chatbot (rule-based + AI)

Extend coverage to departmental queries, faculty contacts, etc.

16. References

Rasa Documentation. https://rasa.com/docs

Flask API Docs. https://flask.palletsprojects.com

React Docs. https://react.dev

17. Appendix

Sample Input/Output

Input: "What is the fee for BBA?"
Output: "The fee structure for BBA is PKR 120,000 per semester."

Snippets of Code

app.py

```
@app.route('/chat', methods=['POST'])

    def chat():

       data = request.get_json()

       message = data.get("message")

       response = call_rasa(message)

       return jsonify({"response": response})
```

rules.yml

```
- rule: respond to fee inquiry

  steps:

  - intent: ask_fee

  - action: utter_fee_info
```