Software Requirements Specification (SRS)

Project Title: AI Chatbot for Student Support Services
Date: August 2025
Prepared By: Mridula

---

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the **AI Chatbot for Student Support Services**. The chatbot is designed to assist students by answering frequently asked questions (FAQs) related to admissions, fees, hostel facilities, examinations, and other academic queries. It aims to reduce the workload of support staff and provide students with instant, accurate information.

1.2 Scope

The system is a **web-based AI chatbot application** that allows students to type in their queries and receive instant responses. It will:

- Provide responses to predefined FAQs.
- Suggest related questions when users ask something.
- Allow feedback collection to improve responses.
- Be scalable to support future integration with databases, student portals, or multilingual support.

The system will not replace human staff completely but will complement existing support services by handling common and repetitive queries.

1.3 Definitions, Acronyms, Abbreviations

- **FAQ:** Frequently Asked Questions.
- **SRS:** Software Requirements Specification.
- **UI:** User Interface.
- **AI Chatbot for Student Support Services:** The system under development.

1.4 References

- IEEE Standard 830-1998 for SRS.
- Python, Streamlit Documentation.

- Pandas Library Documentation.

---

2. Overall Description

2.1 Product Perspective

The chatbot is a standalone system but may be integrated with a university website or student portal in the future. It relies on a predefined FAQ dataset stored in CSV format.

2.2 Product Functions

- Accept user queries via text input.
- Match queries against FAQ dataset.
- Display the most relevant answer.
- Suggest related questions.
- Allow user feedback (“helpful” or “not helpful”).
- Provide a simple, clean, and responsive user interface.

2.3 User Classes and Characteristics

- **Students:** Primary users, seeking quick answers to academic and administrative queries.
- **Staff/Admins:** Secondary users, responsible for updating FAQs and reviewing feedback.

2.4 Operating Environment

- **Frontend:** Streamlit-based web UI.
- **Backend:** Python.
- **Database:** CSV file for FAQs (extendable to SQL in future).
- **Supported Platforms:** Windows, macOS, Linux (via browser).

2.5 Constraints

- Works only with predefined FAQs (not a full AI chatbot).
- Requires periodic FAQ updates.
- Internet connection needed for web app.

2.6 Assumptions and Dependencies

- Users will have access to internet and web browser.

- FAQs will be comprehensive enough to cover most queries.
- Future versions may integrate Natural Language Processing (NLP).

---

3. System Features

3.1 FAQ Management

- Store questions and answers in structured CSV format.
- Support multiple keywords for each question.

3.2 Query Processing

- Accept user input.
- Match input with closest FAQ.
- Display relevant answer.
- Show related suggestions.

3.3 Feedback System

- Users can mark answers as “Helpful” or “Not Helpful.”
- System logs feedback for improvement.

3.4 Clear Chat Feature

- Allows users to reset conversation history.

---

4. External Interface Requirements

4.1 User Interfaces

- Text input box for questions.
- Display area for chatbot answers.
- Buttons for feedback.
- Option to clear chat.

4.2 Hardware Interfaces

- No special hardware required. Runs on any standard PC or laptop.

4.3 Software Interfaces

- Python 3.10+

- Streamlit
- Pandas

4.4 Communication Interfaces

- Runs on localhost or deployed to a web server (e.g., Streamlit Cloud, Heroku).

---

5. Non-Functional Requirements

5.1 Performance Requirements

- Responses should be displayed instantly (<1 second).
- Support at least 100 users simultaneously when deployed.

5.2 Security Requirements

- Data protection for student queries.
- Secure storage of feedback.

5.3 Usability Requirements

- Simple, minimal UI.
- Works across devices (desktop, mobile).

5.4 Reliability & Availability

- System should be available 24/7.
- Minimal downtime during updates.

---

6. System Design Approach

- **Architecture:** Client-server model.
- **Frontend:** Streamlit UI.
- **Backend:** Python scripts handling FAQ retrieval.
- **Storage:** FAQs in CSV (expandable to DB).
- **Feedback:** Stored temporarily for admin review.

---

7. Appendix

- **Future Enhancements:**
    - Integration with AI/NLP for smarter responses.
    - Multilingual support.
    - Voice-based queries.
    - Integration with University ERP/Student Portal.