**SOFTWARE REQUIREMENTS OF SPECIFICATIONS**

FOR

<PROJECT>

AI CHATBOT WITH PYTHON

**Version 1.0 approved**

**Prepared by Team\_249**

**Parul Institute Of Engineering And Technology**

**24-01-2024**



# Table of Contents

[Table of Contents ii](#_TOC_250026)

[Revision History ii](#_TOC_250025)

1. [Introduction… 1](#_TOC_250024)
   1. [Purpose 1](#_TOC_250023)
   2. Document Conventions 1
   3. [Intended Audience and Reading Suggestions 1](#_TOC_250022)
   4. [Product Scope 1](#_TOC_250021)
   5. [References 1](#_TOC_250020)
2. Overall Description 2
   1. [Product Perspective 2](#_TOC_250019)
   2. [Product Functions 2](#_TOC_250018)
   3. [User Classes and Characteristics 2](#_TOC_250017)
   4. [Operating Environment 2](#_TOC_250016)
   5. [Design and Implementation Constraints 2](#_TOC_250015)
   6. [User Documentation 2](#_TOC_250014)
   7. [Assumptions and Dependencies 3](#_TOC_250013)
3. External Interface Requirements 3
   1. User Interfaces 3
   2. [Hardware Interfaces 3](#_TOC_250012)
   3. [Software Interfaces 3](#_TOC_250011)
   4. [Communications Interfaces 3](#_TOC_250010)
4. [System Features 4](#_TOC_250009)
   1. [System Feature 1 4](#_TOC_250008)
   2. [System Feature 2 4](#_TOC_250007)
5. [Other Nonfunctional Requirements 4](#_TOC_250006)
   1. [Performance Requirements 4](#_TOC_250005)
   2. [Safety Requirements 5](#_TOC_250004)
   3. [Security Requirements 5](#_TOC_250003)
   4. [Software Quality Attributes 5](#_TOC_250002)
   5. [Business Rules 5](#_TOC_250001)
6. [Other Requirements 5](#_TOC_250000)

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

1. **Introduction:**

## Purpose:

The purpose of an AI chatbot developed with Python can vary depending on the specific goals of the project or application. Generally, AI chatbots are designed to interact with users in a natural language conversation. Here are some common purposes and use cases for AI chatbots with Python:

1. **Customer Support**: Many businesses use chatbots to provide instant and automated customer support. These bots can answer frequently asked questions, assist with common issues, and direct users to relevant resources.

2. **Information Retrieval**: Chatbots can be designed to retrieve information from databases or external sources. This can be useful for providing users with real-time information, such as weather updates, stock prices, or news.

3. **Task Automation**: Chatbots can automate repetitive tasks, saving time and effort. For example, a chatbot might help users schedule appointments, order products, or book reservations.

4**. Language Understanding and Processing**: AI chatbots can be equipped with natural language processing (NLP) capabilities to understand and interpret user input. This allows them to respond intelligently to a wide range of queries and statements.

5. **Entertainment:** Some chatbots are designed for entertainment purposes, providing users with a conversational and engaging experience. These bots may tell jokes, play games, or simulate casual conversation.

6. **E-learning and Training**: Chatbots can be used in educational settings to assist learners, answer questions, and provide information. They can also be integrated into training programs to simulate real-world scenarios.

7. **Personal Assistants:** Chatbots can function as personal assistants, helping users with tasks such as setting reminders, sending messages, or providing recommendations based on user preferences.

8. **Data Collection and Analysis:** Chatbots can be used to collect data from users through conversations, which can then be analyzed to gain insights into user preferences, behavior, or feedback.

When using Python to develop AI chatbots, popular frameworks and libraries like NLTK (Natural Language Toolkit), spaCy, and TensorFlow can be employed to implement natural language processing and machine learning functionalities.

* 1. **Document Convention:**

1. Document Structure

This document follows a standardized structure comprising several sections and subsections to organize project requirements systematically.

1. Formatting Guidelines

Headings: Bold font for headings, classic for description Subsections: Numbered (14pt) for clarity

Font Style: Times New Roman Font Size: 14 pt

Spacing: Single spacing between lines, double spacing between sections

Alignment: Based on content

1. References

All external sources and documents referenced in this SRS document follow the IEEE citation style.

4.Termnology:

Chatbot

Natural Language Processing (NLP

Python

Machine Learning (ML )

NLTK (Natural Language Toolkit)

API

User Interface (UI)

1. Revision History

Version 1.0: Initial draft (Date: 01/24/2024, Author: Team\_249)

## Intended Audience and Reading Suggestions

1. Intended Audience

This Software Requirements Specification (SRS) document is intended for the following audience:

* + Project Team: Developers, designers, testers, project managers, and other team members involved in the development of the I.R.I.S. system.
  + Stakeholders: Clients, sponsors, end-users, regulatory bodies, and any other individuals or organizations with a vested interest in the project.

1. Reading Suggestions
   * Project Team: Team members are encouraged to read the document thoroughly to gain a comprehensive understanding of project requirements, goals, and constraints. Developers should focus on functional requirements, testers on testing requirements, and project managers on project scope and objectives.
   * Stakeholders: Stakeholders are recommended to focus on sections relevant to their interests or responsibilities. Clients should pay particular attention to project scope and objectives, while regulatory bodies should focus on compliance requirements.
2. Supplementary Materials

Additional materials, such as design documents, user manuals, and technical specifications, are available upon request for stakeholders seeking further information.

## Product Scope:

## The product scope of an AI chatbot developed with Python can vary based on the specific goals and requirements of the project. However, here is a general outline of the product scope for an AI chatbot using Python:

## Functional Requirements:

## User Interaction: Define how the chatbot will interact with users. This includes understanding user input, generating responses, and various conversation scenarios.

## Natural Language Processing (NLP): Specify the level of NLP required for the chatbot to understand and process user input effectively.

## Intent Recognition: Identify the intents the chatbot should recognize and respond to. For example, handling inquiries, providing information, or assisting with specific tasks.

## Integration: Determine the external services or APIs the chatbot will integrate with, such as databases, external data sources, or other applications.

## User Experience (UX) Design:

## Conversation Flow: Design the flow of conversations to ensure a smooth and natural interaction between users and the chatbot.

## User Interface (UI): Define the chatbot's UI elements, whether it's a text-based interface or includes multimedia elements like images, buttons, or interactive components.

## Technological Stack:

## Programming Language: Specify Python as the primary programming language for development.

## Frameworks and Libraries: Identify the libraries and frameworks to be used for NLP (e.g., NLTK, spaCy), machine learning (e.g., TensorFlow), and web development

## 

## Reference: Creating an AI chatbot with Python involves using various libraries and frameworks. Here are some references that can help you get started:

## 1. NLTK (Natural Language Toolkit):

## - Official NLTK Documentation: [NLTK Documentation] (https://www.nltk.org/)

## - Book: "Natural Language Processing in Python" by Steven Bird, Ewan Klein, and Edward Loper.

## 2. spaCy:

## 

## (https://spacy.io/usage/spacy-101)

## 3. TensorFlow

## - Official TensorFlow Documentation: [TensorFlow Documentation] (https://www.tensorflow.org/)

## - TensorFlow Tutorials: [TensorFlow Tutorials] (https://www.tensorflow.org/tutorials)

## 

## https://codelabs.developers.google.com/codelabs/dialogflow-python-web#0)

## 4. Flask (Web Framework):

## - Official Flask Documentation: [Flask Documentation]( https://flask.palletsprojects.com/)

## - Flask Mega-Tutorial: [The Flask Mega-Tutorial Part I: Hello, World!](https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world)

## 5. Django (Web Framework)

## - Official Django Documentation: [Django Documentation] (https://docs.djangoproject.com/)

## - Django for Beginners: [Django for Beginners] (https://djangoforbeginners.com/)

## 6. GitHub Repositories:

## - [rasa/Chatito] (https://github.com/rasa/Chatito) - A tool to generate training data for chatbots.

## - [microsoft/ botpress] (https://github.com/microsoft/botpress) - An open-source conversational platform for building bots.

1. **Introduction:**

## Product Perspective:

* + 1. System Overview

From a productive perspective, an AI chatbot developed with Python can offer several benefits across various domains. Here's a breakdown of the productive perspectives of an AI chatbot:

Efficient Customer Support:

24/7 Availability: AI chatbots can provide round-the-clock customer support, addressing user queries and issues at any time.

Instant Responses: Chatbots can deliver quick responses to frequently asked questions, improving customer satisfaction and reducing wait times.

Task Automation:

Time Savings: Chatbots automate repetitive tasks, freeing up human resources to focus on more complex and value-added activities.

Consistency: Automation ensures a consistent user experience, reducing the likelihood of errors.

## Product Functions:

* **Functional Requirements**
  + 1. Object Recognition:

The system shall be able to identify objects in images or videos using computer vision techniques.

It shall support real-time object recognition for efficient processing of user queries.

* + 1. Natural Language Processing:

The system shall process and understand human language input, including text and voice commands.

It shall perform tasks such as sentiment analysis, named entity recognition, and language generation to extract meaning from user queries.

* + 1. Intelligent Response Generation:

Based on the recognized objects and the context of the input, the system shall generate intelligent responses to user queries.

Responses may include descriptions of recognized objects, relevant information, or personalized recommendations.

* + 1. User Interaction:

Users can interact with the system through web interfaces, mobile apps, or voice commands.

The system shall provide an intuitive and user-friendly interface for seamless interaction.

## User Classes and Characteristics:

* User Classes
  + **End Users:**
  + The end-users of an AI chatbot developed with Python can vary widely based on the purpose and application of the chatbot. Here are some examples of end-users for AI chatbots with Python in different domains:
  + **Customer Service Chatbots:**
  + Consumers: Individuals seeking support, information, or assistance with products or services.
  + **E-commerce Chatbots:**
  + Shoppers: Users looking for product recommendations, pricing information, or assistance with the online shopping process.
  + **Healthcare Chatbots:**
  + Patients: Individuals seeking information about symptoms, appointment scheduling, or general health inquiries.
  + **Healthcare Professionals**: Doctors and medical staff using chatbots for administrative tasks or accessing medical information.
  + **Finance Chatbots:**
  + **Banking Customers**: Users looking for account information, transaction history, or assistance with common banking tasks.
  + **Investors:** Individuals seeking financial advice, market updates, or portfolio management.

.

* Characteristics and Roles
  + End Users:

Technical Proficiency: Varies among users, ranging from novice to advanced.

Access Privileges: Limited to accessing and using the system's features and functionalities.

Usage Patterns: Regularly interact with the system to retrieve information or perform tasks.

* + Administrators:

Technical Proficiency: High level of technical expertise required for system administration tasks.

Access Privileges: Full access to system configuration settings and administrative functions.

Usage Patterns: Perform administrative tasks such as user management, system configuration, and performance monitoring.

* + Developers:

Technical Proficiency: Highly skilled in software development, programming languages, and system architecture.

Access Privileges: Access to source code, development environments, and testing tools.

Usage Patterns: Develop, debug, and maintain the system's software components.

* User Requirements
  + End Users:

Functional Requirements: Intuitive user interface, accurate responses to queries, personalized recommendations.

Non-functional Requirements: Usability, performance, accessibility.

* + Administrators:

Functional Requirements: Comprehensive system management tools, role-based access control, audit logs.

Non-functional Requirements: Security, scalability, reliability.

* + Developers:

Functional Requirements: Development environment setup, version control integration, documentation.

Non-functional Requirements: Code quality, maintainability, extensibility.

* + Use Cases

End User: A user uploads an image and receives descriptions of recognized objects.

Administrator: An administrator configures user permissions and system settings through the administrative dashboard.

Developer: A developer troubleshoots and debugs issues in the system's codebase.

## Operating Environment:

* Hardware Requirements

Processor: Dual-core CPU, 2.0 GHz or higher Memory: 4 GB RAM minimum

Storage: 100 GB available disk space Software Requirements

Operating System: Windows 10, macOS 10.15, Ubuntu 20.04 LTS Frameworks/Libraries: Python 3.7+, TensorFlow 2.0+, NLTK, Flask

Third-party Services: Google Vision API, Natural Language Processing APIs

* Network Infrastructure

Internet Connectivity: Broadband internet connection (minimum 10 Mbps)

Network Security: Firewall configuration to allow incoming/outgoing traffic on required ports

* Compatibility

Supported Browsers: Chrome, Firefox, Safari Mobile Compatibility: Android 8.0+, iOS 12.0+ Deployment Considerations

Deployment Platform: Compatible with on-premises servers or cloud platforms (e.g., AWS, Google Cloud)

* Configuration: Requires setup of database connections, API keys, and environment variables

Scaling: System architecture designed for horizontal scaling to accommodate increased user load.

## Design and Implementation Constraints:

* Technical Constraints

1. Platform Compatibility: The system must support Windows, macOS, and Linux operating systems.
2. Performance Constraints: Real-time object recognition must be achieved with minimal latency (under 1 second).
3. Integration Challenges: Dependency on third-party APIs for object recognition and natural language processing.

* Resource Constraints

1. Time Constraints: Project delivery deadline set for six months from project initiation.
2. Budget Constraints: Limited budget for purchasing hardware and software licenses.
3. Skillset Constraints: Limited availability of personnel with expertise in computer vision and natural language processing.

* Regulatory Constraints

1. Data Privacy Regulations: Compliance with GDPR for handling and storing user data.
2. Security Standards: Adherence to ISO 27001 standards for information security.
3. Accessibility Requirements: Ensuring the system meets WCAG 2.0 accessibility standards for users with disabilities.

* Organizational Constraints

1. Organizational Culture: Emphasis on iterative development and continuous improvement.
2. Stakeholder Expectations: Alignment with stakeholders' goals for user engagement and system performance.
3. Communication Channels: Use of Slack for internal team communication and JIRA for project management.

## User Documentation:

* User Manual

Comprehensive user manual covering system overview, getting started, user interface, functionality, and troubleshooting.

Tutorials and Guides

Video tutorials demonstrating key features and workflows.

How-to guides providing step-by-step instructions for common tasks.

* FAQs and Help Resources

Frequently asked questions addressing common user inquiries. Online help centre with documentation, tutorials, and support resources.

* Support Channels

Helpdesk contact information for support inquiries.

Community forums for peer-to-peer support and discussions. Email support for contacting the support team directly.

* Updates and Release Notes

Release notes documenting new features, enhancements, and bug fixes in each system update.

Notification mechanisms for informing users of upcoming updates and changes.

## Assumptions and Dependencies:

* Assumptions

Pre-existing Data: It is assumed that a sufficient amount of labeled data for object recognition and natural language processing tasks is available for model training.

* User Behaviour: Users are assumed to interact with the system primarily through web and mobile interfaces, with a preference for intuitive and user-friendly interactions.

Third-party Services: It is assumed that third-party APIs for object recognition and natural language processing are reliable and provide accurate results.

* Dependencies

External APIs: Dependency on the Google Vision API for object recognition and the Natural Language Toolkit (NLTK) for natural language processing.

* Hardware Components: Dependency on webcams or mobile device cameras for capturing images for object recognition.

Software Libraries/Frameworks: Dependency on Python programming language, TensorFlow framework, and Flask web framework for system development.

1. **External Interface Requirement:**
   1. **User Interface:**

Graphical User Interface (GUI)

* Screen Layout:

The GUI shall include a homepage with options for uploading images, entering text queries, and accessing user settings.

Results shall be displayed in a separate section below the input options.

* Functionality:

Users can upload images by clicking on the "Upload Image" button and selecting the desired image file.

Text queries can be entered in a text input field provided on the homepage.

Results shall be displayed dynamically as objects are recognized or responses are generated.

* Navigation:

Users can navigate between different sections of the GUI using a navigation bar located at the top of the screen.

The GUI shall provide breadcrumbs or back buttons for easy navigation within the interface.

* Visual Design:

The GUI shall have a clean and modern design with intuitive navigation and user-friendly features.

Visual elements such as colours, fonts, and graphics shall be chosen to enhance usability and aesthetics.

## Hardware Interfaces:

* Sensors
* Cameras:

The system shall interface with webcams for capturing images for object recognition tasks.

Supported webcam formats include USB-connected webcams and built-in cameras on mobile devices.

* Microphones (if applicable):

The system shall support audio input for processing voice commands or speech-to-text conversion.

Supported microphones include built-in microphones on devices or external USB-connected microphones.

* Other Sensors (if applicable):

The system may interface with other sensors such as temperature sensors, motion sensors, or GPS receivers for additional input data.

* + Actuators
* Displays:

The system shall present visual information to users through displays such as monitors, screens, or mobile device screens.

Supported display formats include standard resolutions and aspect ratios.

* Speakers (if applicable):

The system may output audio information through speakers for providing feedback or alerts to users.

Supported speaker formats include built-in speakers on devices or external speakers connected via audio ports.

* Communication Protocols

The system shall communicate with hardware devices using standard communication protocols such as USB and Bluetooth.

USB-connected devices shall use USB protocols for data transfer and communication.

Bluetooth-connected devices shall use Bluetooth protocols for wireless communication.

## Software Interfaces

* Object Recognition APIs:

The system shall utilize the Google Cloud Vision API for object recognition tasks.

API requests shall be made over HTTPS using RESTful endpoints provided by the Google Cloud Vision API.

* Natural Language Processing (NLP) APIs:

The system may utilize the spaCy library for NLP tasks such as entity recognition and sentiment analysis.

Integration with spaCy shall be via Python APIs provided by the library.

* Relational Database:

The system shall interact with a PostgreSQL database for storing user profiles and system configurations.

Database communication shall be performed using the psycopg2 Python library for PostgreSQL.

* NoSQL Database:

The system may use MongoDB for storing unstructured data such as image metadata.

Database interaction shall be facilitated using the pymongo Python library.

## Communications Interfaces

* Internal Communication Protocols
  + HTTP/HTTPS:

1. The system components shall communicate via HTTP/HTTPS for request-response interactions.
2. RESTful APIs shall be used for communication between different modules or services within the system.
   * Message Queues:
3. The system may utilize Apache Kafka for asynchronous message-based communication between components.
4. Kafka topics shall be used for publishing and subscribing to messages.

* External Communication Protocols
  + HTTP/HTTPS:

1. The system shall communicate with external APIs and web services using the HTTPS protocol.
2. API requests and responses shall be exchanged over secure HTTP connections.
   * WebSocket’s:
3. If real-time bidirectional communication is required, the system may utilize WebSockets for efficient data exchange.
4. WebSocket connections shall be established for persistent, low-latency communication with external clients or services.

* Data Formats
  + JSON:

1. Data exchanged between system components or with external APIs shall be in JSON format.
2. JSON schema definitions shall be used to validate the structure of JSON data.
   * Protobuf:

I. For efficient serialization of data, Protobuf shall be used for message encoding in message-based communication.

1. **System Features**
   1. **System Feature 1**

The "Image Recognition and Text Extraction" feature allows the system to analyse images uploaded by users and extract both visual content and textual information. Using advanced image processing and optical character recognition (OCR) techniques, the system identifies objects within the images and extracts any text present within them.

* Purpose:

This feature serves multiple purposes:

* + Enhance user experience: By providing comprehensive analysis of uploaded images, including both visual and textual elements, the system offers valuable insights to users.
  + Enable content indexing: Extracting text from images allows the system to index and search for images based on their textual content, improving searchability and organization.
  + Support accessibility: Extracted text can be used to provide alternative text descriptions for visually impaired users or facilitate translation into different languages.
* Scope:
* Upon receiving an uploaded image, the system processes it using image recognition algorithms to identify objects, scenes, and other visual elements.
* Simultaneously, the system performs OCR to extract any text present within the image, including printed text, handwriting, or captions.
* The extracted visual content and textual information are stored and made available for further analysis or presentation to users.
  + Use Cases:

Identifying objects and extracting text from a photograph:

* + - Scenario: A user uploads a photograph containing various objects and a signboard with textual information.
    - Action: The system analyzes the image, identifies objects, and extracts text using image recognition and OCR algorithms.
    - Outcome: The system provides a list of recognized objects along with their descriptions and extracts the text from the signboard, making it available for further processing.
  + Creating searchable archives of scanned documents:
    - Scenario: A company uploads scanned documents containing textual content and accompanying images.
    - Action: The system processes the scanned documents, extracting text from images and converting it into searchable text.
    - Outcome: The system creates searchable archives of scanned documents, allowing users to easily locate specific documents based on their textual content.
  + Dependencies:
    - Integration with image recognition models or APIs capable of accurately detecting objects and scenes within images.
    - Integration with OCR engines or services capable of extracting text from images with high accuracy and reliability.
    - Adequate computational resources to support the processing of image data and execution of image recognition and OCR algorithms.
  + Acceptance Criteria:
    - The system should accurately identify objects and scenes within uploaded images, providing detailed descriptions and annotations where applicable.
    - OCR should accurately extract text from images, supporting various languages and fonts with high precision.
    - The extracted visual content and textual information should be presented to users in a user-friendly format, allowing for easy access and understanding.

## System Feature 2

* System Feature: Natural Language Processing (NLP)
  + Description:

The "Natural Language Processing (NLP)" feature enables the system to understand and analyze text input provided by users. Using advanced NLP techniques, the system interprets natural language queries, extracts meaningful information, and performs various text processing tasks, such as sentiment analysis, entity recognition, and language understanding.

* + Purpose:

This feature serves several purposes:

* + - Facilitate user interaction: By understanding natural language queries, the system allows users to communicate with it in a more intuitive and conversational manner.
    - Extract insights from textual data: NLP enables the system to extract valuable information and insights from textual content, such as user feedback, reviews, or social media posts.
    - Enhance decision-making: By analyzing text data, the system can make informed decisions, generate responses, or provide recommendations based on user queries or input.
  + Scope:
    - The system processes text input provided by users, such as search queries, messages, or comments.
    - Using NLP algorithms and techniques, the system analyzes the text input to identify relevant entities, sentiments, intents, and other linguistic features.
    - The extracted information is used to generate responses, perform actions, or provide relevant information to users.
  + Use Cases:

Understanding user queries in a chatbot application:

* + - Scenario: A user interacts with a chatbot to inquire about product availability.
    - Action: The system analyzes the user's text input using NLP techniques to understand the user's intent and extract relevant information.
    - Outcome: The system provides a response indicating the availability of the requested product or offers alternative options based on user preferences.

Sentiment analysis of customer reviews:

* + - Scenario: A company analyzes customer reviews to gauge sentiment and identify areas for improvement.
    - Action: The system processes customer reviews using NLP algorithms to analyze sentiment and extract key insights.
    - Outcome: The system generates reports highlighting sentiment trends, identifies positive and negative feedback, and suggests actions based on the analysis.
  + Dependencies:
    - Integration with NLP libraries, frameworks, or services capable of performing various NLP tasks, such as tokenization, part-of- speech tagging, and named entity recognition.
    - Availability of labeled training data for training and fine-tuning NLP models, if applicable.
    - Adequate computational resources to support the processing of text data and execution of NLP algorithms.
  + Acceptance Criteria:
    - The system should accurately understand and interpret natural language queries provided by users, identifying relevant entities, intents, and sentiments.
    - NLP analysis results should be presented to users in a clear and understandable format, facilitating effective communication and interaction.
    - The system should be able to perform various text processing tasks, such as entity recognition, sentiment analysis, and language understanding, with high accuracy and reliability.

1. **Other Nonfunctional Requirements:**

## Performance Requirements

* Response Time:
  + The system should respond to user queries within 3 seconds under normal load conditions.
  + Object recognition processing time for uploaded images should not exceed 5 seconds per image.
* Throughput:
  + The system should support a minimum of 50 concurrent user interactions without degradation in performance.
  + Object recognition tasks should be able to process a minimum of 20 images per minute during peak usage hours.
* Scalability:
  + The system should be able to scale horizontally to accommodate a 30% increase in user load within 30 minutes.
  + The system should support a minimum of 10,000 image uploads per day without significant degradation in performance.
* Resource Utilization:
  + The system should utilize no more than 70% of available CPU and memory resources under peak load conditions.
  + Database queries for natural language processing tasks should be optimized to minimize CPU and disk I/O usage, ensuring efficient resource utilization.
* Availability:
  + The system should achieve a minimum uptime of 99.5% per month, excluding scheduled maintenance windows.
  + The maximum allowable downtime for system maintenance should not exceed 6 hours per month.
* Reliability:
  + The system should have a mean time between failures (MTBF) of at least 400 hours.
  + The probability of data loss in the event of a system failure should be less than 0.05%.
* Load Testing:
  + The system should undergo load testing to verify performance under expected peak loads, simulating concurrent user interactions and image uploads.
  + Load testing should be conducted using realistic usage scenarios to ensure accurate performance assessment.
* Scalability Testing:
  + The system should undergo scalability testing to validate its ability to handle increasing user loads by adding additional resources or scaling out horizontally.
  + Scalability testing should simulate various growth scenarios to identify potential bottlenecks and scalability limitations.

## Safety Requirements

* Data Security:
  + All user data, including uploaded images and text input, should be encrypted both in transit and at rest using industry-standard encryption algorithms.
  + Access controls should be implemented to restrict access to sensitive user data, ensuring that only authorized personnel can view or modify it.
* User Privacy:
  + The system should adhere to relevant data protection regulations (e.g., GDPR, CCPA) to protect user privacy rights.
  + Personal information collected from users, such as images or text input, should only be used for the intended purposes of the system and not shared with third parties without user consent.
* System Integrity:
  + Measures should be implemented to prevent unauthorized access to system resources, including authentication mechanisms and role-based access controls.
  + Regular security assessments, including vulnerability scans and penetration testing, should be conducted to identify and address potential security vulnerabilities.
* Error Handling:
  + The system should implement robust error handling mechanisms to detect and recover from errors or failures gracefully.
  + Error messages presented to users should be informative and non-threatening, avoiding the disclosure of sensitive system information.
* Backup and Recovery:
  + Regular backups of system data should be performed to ensure data integrity and facilitate recovery in the event of data loss or system failure.
  + Disaster recovery plans should be in place to restore system functionality in the event of catastrophic failures or disasters.
* User Safety:
  + The system should provide appropriate warnings and disclaimers regarding the limitations and potential risks associated with its use, particularly in critical or safety-critical applications.
  + Users should be encouraged to use the system responsibly and follow best practices for safe and secure interactions.
* Compliance:
  + The system should comply with relevant safety standards and regulations applicable to its domain, such as ISO 27001 for information security management.

## Security Requirements

* Authentication and Authorization:
  + Users must authenticate using strong, password-based authentication mechanisms before accessing the system's features.
  + Passwords should be securely hashed and stored in the database to prevent unauthorized access in the event of a data breach.
  + Role-based access control (RBAC) should be implemented to restrict access to system functionalities based on user roles and privileges.
* Data Encryption:
  + All data transmitted between the client and server, including user inputs and responses, should be encrypted using industry- standard encryption protocols (e.g., TLS).
  + Stored user data, including uploaded images and textual information, should be encrypted at rest using encryption algorithms such as AES-256.
* Secure APIs:
  + APIs exposed by the system for client-server communication should implement secure authentication mechanisms (e.g., OAuth 2.0) to prevent unauthorized access.
  + API endpoints should be protected against common security threats such as cross-site request forgery (CSRF), cross-site scripting (XSS), and SQL injection attacks.
* Input Validation:
  + All user inputs, including text queries and uploaded images, should be validated and sanitized on the server-side to prevent injection attacks and malicious file uploads.
  + Image files should be scanned for malware and checked against a whitelist of allowed formats before processing.
* Logging and Auditing:
  + The system should maintain detailed logs of user activities, system events, and security-related incidents for audit and forensic purposes.
  + Access to log files should be restricted to authorized personnel, and log entries should be timestamped and immutable to ensure data integrity.
* Security Updates and Patch Management:
  + Regular security updates and patches should be applied to the system's underlying software components (e.g., operating system, web server, database) to address known vulnerabilities.
  + Vulnerability scanning tools should be employed to identify and remediate security weaknesses proactively.
* Data Privacy and Compliance:
  + The system should comply with relevant data privacy regulations, such as GDPR or HIPAA, ensuring that user data is collected, processed, and stored in accordance with legal requirements.
  + Data anonymization techniques should be applied to sensitive user data where appropriate to minimize privacy risks.
* Incident Response Plan:
  + An incident response plan should be developed outlining procedures for detecting, reporting, and responding to security incidents and data breaches.
  + The incident response team should be trained and prepared to execute the plan effectively in the event of a security incident.

## Software Quality Attributes

* Reliability:
  + The system should operate reliably under normal and peak load conditions, minimizing downtime and service disruptions.
  + The mean time between failures (MTBF) should exceed 500 hours, ensuring consistent and uninterrupted operation.
* Availability:
  + The system should achieve a minimum uptime of 99.9% per month, ensuring that it is available to users whenever needed.
  + Redundancy and failover mechanisms should be implemented to mitigate the impact of hardware or software failures on system availability.
* Performance:
  + The system should respond to user interactions within 3 seconds under normal load conditions, ensuring a responsive user experience.
  + Throughput should be sufficient to handle peak user loads without degradation in performance, with a minimum of 50 concurrent user sessions supported.
* Scalability:
  + The system should be able to scale horizontally to accommodate increasing user loads by adding additional resources or scaling out.
  + Scalability testing should be conducted to validate the system's ability to handle growing user bases and workloads.
* Usability:
  + The system's user interface should be intuitive and easy to navigate, requiring minimal training for users to effectively use the system.
  + User feedback mechanisms should be implemented to gather input on usability issues and areas for improvement.
* Maintainability:
  + The system's codebase should be well-structured, modular, and thoroughly documented to facilitate ongoing maintenance and future enhancements.
  + Automated testing and continuous integration practices should be employed to ensure code quality and detect regressions early.
* Security:
  + The system should adhere to industry-standard security practices, including encryption of sensitive data, secure authentication mechanisms, and protection against common security threats.
  + Regular security audits and vulnerability assessments should be conducted to identify and address potential security vulnerabilities.
* Portability:
  + The system should be designed to run on multiple platforms and environments, supporting deployment across various operating systems and cloud providers.

## Business Rules

* Image Upload Rules:
  + Users must agree to the terms and conditions before uploading images to the system.
  + Uploaded images must comply with acceptable use policies and legal regulations, prohibiting offensive, explicit, or copyrighted content.
* User Authentication Rules:
  + Users must create an account and authenticate themselves before accessing certain features or functionalities.
  + Passwords must meet minimum complexity requirements, including a minimum length and a combination of alphanumeric characters.
* Data Privacy Rules:
  + User data, including uploaded images and text input, should be treated with utmost confidentiality and privacy.
  + Personal information collected from users must be handled in accordance with applicable data protection regulations (e.g., GDPR, CCPA).
* Response Generation Rules:
  + Responses generated by the system should be accurate, relevant, and tailored to the user's query or input.
  + Responses should not contain offensive, discriminatory, or inappropriate content, adhering to community guidelines and ethical standards.
* Feedback Handling Rules:
  + Users should have the option to provide feedback on system- generated responses or analysis results.
  + Feedback should be reviewed and considered for continuous improvement of the system's performance and accuracy.
* Compliance Rules:
  + The system must comply with relevant laws, regulations, and industry standards governing data privacy, security, and intellectual property rights.
  + Compliance with industry-specific regulations (e.g., healthcare, finance) should be ensured to avoid legal liabilities and penalties.
* Error Handling Rules:
  + The system should provide informative error messages and guidance to users in the event of errors or failures.
  + Error handling mechanisms should prevent the disclosure of sensitive system information and maintain user trust.
* Service Level Agreement (SLA) Rules:
  + The system should adhere to predefined service level agreements (SLAs) regarding uptime, response times, and support availability.
  + SLAs should be communicated to users and stakeholders, with penalties or incentives defined for non-compliance or exceptional performance.
* Content Moderation Rules:
  + User-generated content, including text inputs and uploaded images, should be monitored for compliance with community guidelines and acceptable use policies.
  + Automated content moderation algorithms should be employed to detect and flag inappropriate or harmful content for review by moderators.
* Access Control Rules:
  + Access to sensitive system functionalities or administrative features should be restricted to authorized personnel only, based on predefined roles and permissions.
  + Role-based access control (RBAC) should be implemented to enforce access control policies and prevent unauthorized access to critical resources.
* Data Retention Rules:
  + User-generated data, including uploaded images and textual input, should be retained for a specified period based on legal requirements and business needs.
  + Personal data should be anonymized or deleted upon user request or when no longer necessary for the purposes for which

**Appendex:**

It seems like there might be a slight confusion in your request. If by you mean an appendix, it's typically a supplementary material that provides additional information, documentation, or details related to a main document or project.

For an AI chatbot with Python, an appendix could include:

1. Code Samples:

- Include snippets of Python code that demonstrate key functionalities of the chatbot.

2. Data Structures and Flowcharts

- Illustrate the data structures used in the chatbot and provide flowcharts to explain the logic and decision-making process.

3. Integration Documentation:

- Details on how the chatbot integrates with external services, APIs, or databases.

4. Testing Documentation:

- Test cases, scenarios, and results to ensure the chatbot functions correctly in various situations.

5. User Guides

- Instructions for end-users on how to interact with the chatbot, including sample conversations.

6. Development Environment Setup:

- Information on how to set up the development environment for working with the Python chatbot code.

7. Dependencies and Libraries:

- A list of Python libraries and dependencies used in the project, along with their versions.

8. Technical Specifications:

- Detailed technical specifications, such as the architecture, algorithms used, and any third-party tools incorporated.

1. Appendix B: Analysis Models
   1. Use Case Diagram

A use case diagram illustrates the interactions between actors and the system, representing various use cases and their relationships.

[Insert Use Case Diagram Here]

* 1. Activity Diagram

An activity diagram depicts the flow of activities within the system, showing the sequence of actions and decision points.

[Insert Activity Diagram Here]

* 1. Class Diagram

A class diagram illustrates the structure of the system in terms of classes, attributes, and relationships.

[Insert Class Diagram Here]

* 1. Sequence Diagram

A sequence diagram shows the interactions between objects in a particular scenario, depicting the sequence of messages exchanged between them.

[Insert Sequence Diagram Here]

* 1. State Diagram

A state diagram represents the lifecycle of an object or system, showing the transitions between different states and the events triggering those transitions.

[Insert State Diagram Here]

* 1. Data Flow Diagram

A data flow diagram (DFD) illustrates the flow of data within the system, showing how data moves between various components and processes.

[Insert Data Flow Diagram Here]

1. Appendix C: To Be Determined List

* Integration Points: Specific integration points with external systems or APIs are yet to be determined pending further discussions with stakeholders and third-party vendors.
* User Interface Mockups: Detailed user interface mockups and wireframes are pending design iterations and user feedback sessions.
* System Architecture: The final system architecture, including hardware requirements and cloud infrastructure, is subject to further analysis and architectural design discussions.
* Data Storage and Retrieval Mechanism: The mechanism for storing and retrieving user data, including images and textual information, will be determined based on scalability, performance, and cost considerations.
* Third-Party Services and Libraries: The selection and integration of third-party services, libraries, or APIs for object recognition, natural language processing, and other functionalities are yet to be finalized pending evaluation and testing.
* Security Measures: Detailed security measures, including encryption protocols, access controls, and authentication mechanisms, will be defined based on security assessments and compliance requirements.