

CHAPTER 1

1 Introduction

1.1 INTRODUCTION OF THE PROJECT

Certainly! Here are some simple lines about creating an AI chatbot with Python: “Crafting an AI chatbot in Python is like planting a digital seed and watching it grow into a conversational tree, branching out with every interaction.” “With Python, a few lines of code can breathe life into a chatbot, transforming scripts into meaningful conversations.” “An AI chatbot in Python is not just a program; it’s a virtual confidant, built on algorithms and nurtured through dialogue.” “Python’s simplicity paves the way for complex AI chatbots, making them accessible to coders and companies alike.” “From a programmer’s keyboard to a user’s screen, Python’s AI chatbots bridge the gap with every typed word.”

1.2 SCOPE OF THE PROJECT

The scope of an AI chatbot project with Python is quite broad and can encompass various aspects depending on the specific goals and requirements of the project. Here are some key points that typically define the scope of the project:

1. Functionality
2. Technologies and Platforms
3. Process Flow
4. Training Data
5. Testing and Deployment
6. Maintenance and Support

1.3 AIM OF THE PROJECT

The aim of an AI chatbot project with Python is to create a virtual assistant that can simulate human-like conversations, understand and process user inputs, and provide helpful, accurate responses. Here are some specific goals for such a project

CHAPTER 2

2 Literature survey

2.1 PAPER:1 AI Chatbot with python Research Report

A literature survey for AI chatbots with Python would encompass a review of recent advances, challenges, and future directions in the field. Here's a summary based on recent literature:

- **Recent Advances:** Chatbots have seen significant improvements, especially in areas of Natural Language Processing (NLP) and Machine Learning (ML), allowing them to provide automated guidance and support across various industries¹².
- **Challenges:** Despite advancements, there are still challenges in creating chatbots that can truly understand and respond to human language in a way that feels natural. Issues such as context handling, emotion recognition, and maintaining a coherent long-term conversation are areas that need further work¹².
- **Future Research:** Recommendations for future research include improving conversational models, enhancing the chatbot's ability to understand complex user inputs, and developing better methods for chatbots to learn from interactions¹.
- **Applications:** Chatbots are being applied in diverse fields such as education, e-commerce, healthcare, and entertainment, providing both support and social interaction².
- **Methodologies:** The literature also discusses various methods and algorithms used in chatbot development, highlighting the importance of a robust design that can handle the intricacies of human conversation³.
- **Impact Factor:** Studies have shown that well-designed chatbots can significantly impact customer satisfaction and operational efficiency, making them valuable assets for businesses³. Based on the recent literature, here's a summary of a research report on AI chatbots developed with Python:
- **AI Chatbot Using TensorFlow:** A project report from the University of Allahabad discusses an AI chatbot created using Google's TensorFlow. The chatbot aims to provide a universal system that can assist customers, reducing time and human resources. The report highlights the use of natural language processing (NLP) and the potential for chatbots to become more effective with ongoing research in machine learning¹.
- **Chatbot in Python:** Another study presents a chatbot designed to provide genuine and accurate answers for queries using Artificial Intelligence Markup Languages (AIML) and Python. The project aims to add a chatbot feature and API for Yioop, a search engine. The chatbot is capable of performing tasks such as responding to users, informing them, helping to purchase products, and providing better customer services².

- Development of Chatterbot using Python: This paper discusses the overall development and working of a chatbot using the Chatterbot library in Python. The chatbot is user-friendly and has been studied under SCSVMV University for its feasibility and effectiveness³.
- Chatbot with AIML and LSA: A design of a chatbot that provides accurate answers using Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA) with the Python platform is detailed in a project. The chatbot aims to be a genuine conversational agent that can handle any query intelligently.

CHAPTER 3

3 Analysis / Software Requirements Specification (SRS)

3.1 Introduction

The purpose of this SRS is to outline the specifications for an AI chatbot system developed with Python. The chatbot will serve as a virtual assistant, providing users with information and support through conversational interfaces.

3.2 Document Conventions

In the context of a Software Requirement Specification (SRS) for an AI chatbot with Python, “Document Conventions” refer to the agreed-upon standards and formats used throughout the document to ensure consistency and clarity. Here are some common conventions that might be included:

- Formatting Standards:
- Terminology:
- Versioning:
- Referencing:
- Notation:
- Diagrams and Figures:
- Language:

3.3 Intended Audience and Reading Suggestions

An AI chatbot built with Python is a computer program designed to simulate conversation with human users. It uses Python’s programming capabilities and AI techniques to understand and respond to user queries in a natural way. The

goal is to provide quick and accurate assistance without the need for human intervention. Product Managers and Business Analysts: Both product managers and business analysts are expected to read the document in its entirety to gain a holistic understanding of the system's requirements, functionalities, and constraints. All Intended Readers: All stakeholders, including product managers, business analysts, and other team members, can refer to the appendix and diagrams provided in the document to clarify technical details and enhance their comprehension as needed.

3.4 Product Scope

The product scope of an AI chatbot in Python is to automate interactions and provide instant responses to user queries using natural language processing.

3.5 References

For references on creating a chatbot with Python, you can explore the following resources:

- Real Python Tutorial: A guide on building a chatbot using the ChatterBot library¹.
- GeeksforGeeks Tutorial: Instructions on making a chatbot using the Cohere API in Python².
- Picasso Blog Guide: A comprehensive guide on mastering AI chatbots using Python³.
- Codecademy Course: A learning path from Python beginner to coding your own AI chatbot⁴.
- The Coded Dose Guide: A step-by-step guide to creating your first AI chatbot using Python⁵. These references cover a range of topics from basic setup to more advanced concepts in chatbot development with Python.

3.6 User Interfaces

User interfaces for AI chatbots with Python can range from simple command-line interfaces to more sophisticated graphical user interfaces (GUIs). Here are some common types:

- Command-Line Interface (CLI):
- Web-Based Interface:
- Desktop Application:
- Voice Interfaces:
- Messaging Platforms:
- Mobile Apps:

3.7 Hardware Interface

The project will be accessible by standard desktop and mobile devices with internet connectivity. It will not require any specialized hardware interface.

3.8 Functional Requirements

- Natural Language Understanding:
- Response Generation:
- Learning and Adaptation:
- Multi-Turn Conversation:
- Integration:
- User Management:
- Scalability:
- Error Handling:
- Reporting and Analytics:

3.9 Non Functional Requirements

The non-functional requirements of an AI chatbot with Python typically include:

- Scalability
- Usability
- Miantability
- Realability
- Performance

3.9.1 Performance

The system should be able to handle a large number of concurrent users without significant performance degradation. Response times for common operations (e.g., booking) should be fast to ensure a smooth user experience.

3.9.2 Safety Requirements

Data Security and Privacy : The system must ensure the confidentiality, integrity, and availability of user data, including personal information, and rental history. Equipment Safety and Compliance : The system must provide clear guidelines and safety instructions for the proper handling and use of rental equipment, adhering to relevant safety standards and regulations.

3.10 Software Quality Attributes

Reliability: The system must be reliable, ensuring that it operates consistently and performs as expected under normal conditions . Usability: The system must be user-friendly and intuitive, with clear navigation, logical understandable interfaces. Performance: The system must be responsive and performant, with fast loading times, efficient search functionality. Maintainability: The system must be maintain with well-organized code, clear documentation, and modular architecture

4 System Design

4.1 Design

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.

4.1.1 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.

4.1.2 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.

4.1.3 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.

s

4.1.4 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.

4.1.5 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.

4.1.6 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.

4.2 System Feature 2

System Feature: Natural Language Processing (NLP)

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.2.4 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.

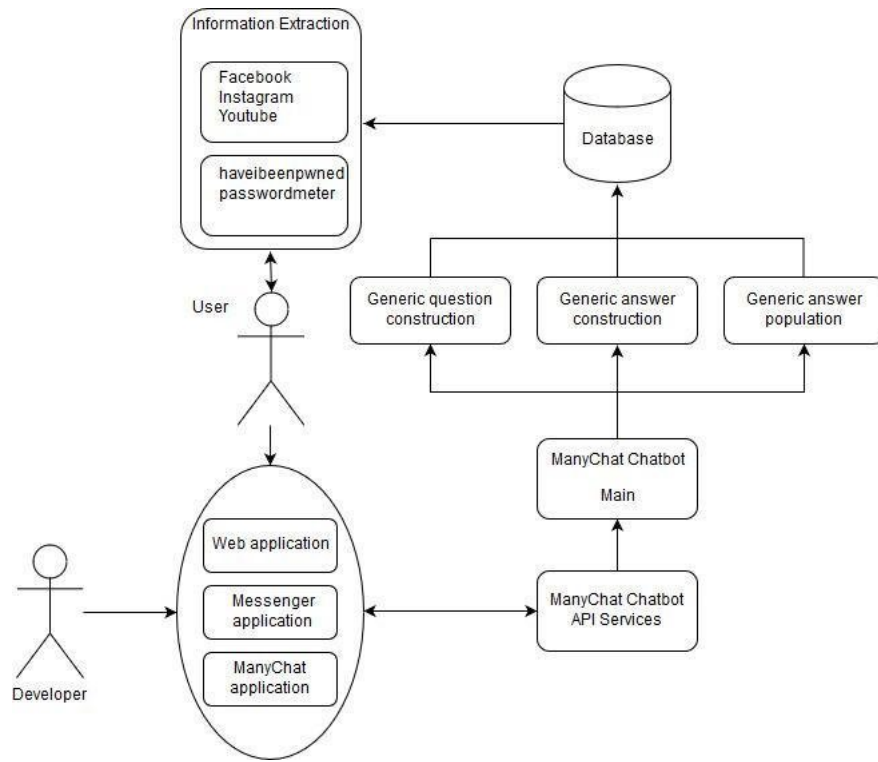


Figure 1: Usecase daigram 1

- 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.



Figure 2: usecase daigram

4.2.5 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.

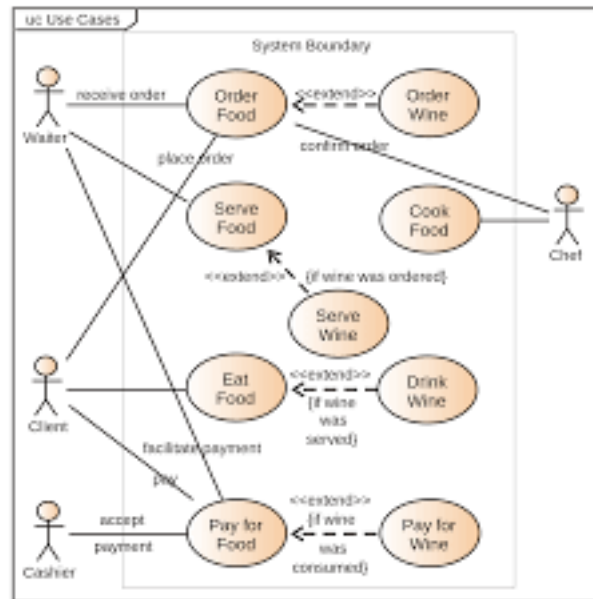


Figure 3: Usecase daigram 2

4.2.6 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.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 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.

5.1.2 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.

5.1.3 Scalability

- The system should be able to scale horizontally to accommodate a 30
- The system should support a minimum of 10,000 image uploads per day without significant degradation in performance.

5.1.4 Resource Utilization

- The system should utilize no more than 70
- Database queries for natural language processing tasks should be optimized to minimize CPU and disk I/O usage, ensuring efficient resource utilization.

5.1.5 Availability

- The system should achieve a minimum uptime of 99.5
- The maximum allowable downtime for system maintenance should not exceed 6 hours per month.

5.1.6 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

5.1.7 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.

5.1.8 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.

5.2 Safety Requirements

5.2.1 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.

5.2.2 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.

5.2.3 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.

5.2.4 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.

5.2.5 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.

5.2.6 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.

5.2.7 Compliance

- The system should comply with relevant safety standards and regulations applicable to its domain, such as ISO 27001 for information security management.

5.3 Security Requirements

5.3.1 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.

5.3.2 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.

5.3.3 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.

5.3.4 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.

5.3.5 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.

5.3.6 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.

5.3.7 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.

5.3.8 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.

5.4 Software Quality Attributes

5.4.1 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.

5.4.2 Availability

- The system should achieve a minimum uptime of 99.9
- Redundancy and failover mechanisms should be implemented to mitigate the impact of hardware or software failures on system availability.

5.4.3 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.

5.4.4 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.

5.4.5 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.

5.4.6 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.

5.4.7 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.

5.4.8 Portability

- The system should be designed to run on multiple platforms and environments, supporting deployment across various operating systems and cloud providers.

5.5 Business Rules

5.5.1 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.

5.5.2 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.

5.5.3 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).

5.5.4 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.

5.5.5 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.

5.5.6 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.

5.5.7 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.

5.5.8 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.

5.5.9 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.

5.5.10 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

5.5.11 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

Appendence: 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.

A. 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.
2. Activity Diagram An activity diagram depicts the flow of activities within the system, showing the sequence of actions and decision points.
3. Class Diagram A class diagram illustrates the structure of the system in terms of classes, attributes, and relationships.
4. Sequence Diagram A sequence diagram shows the interactions between objects in a particular scenario, depicting the sequence of messages exchanged between them.
5. 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.
6. A data flow diagram (DFD) illustrates the flow of data within the system, showing how data moves between various components and processes.

B. 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.

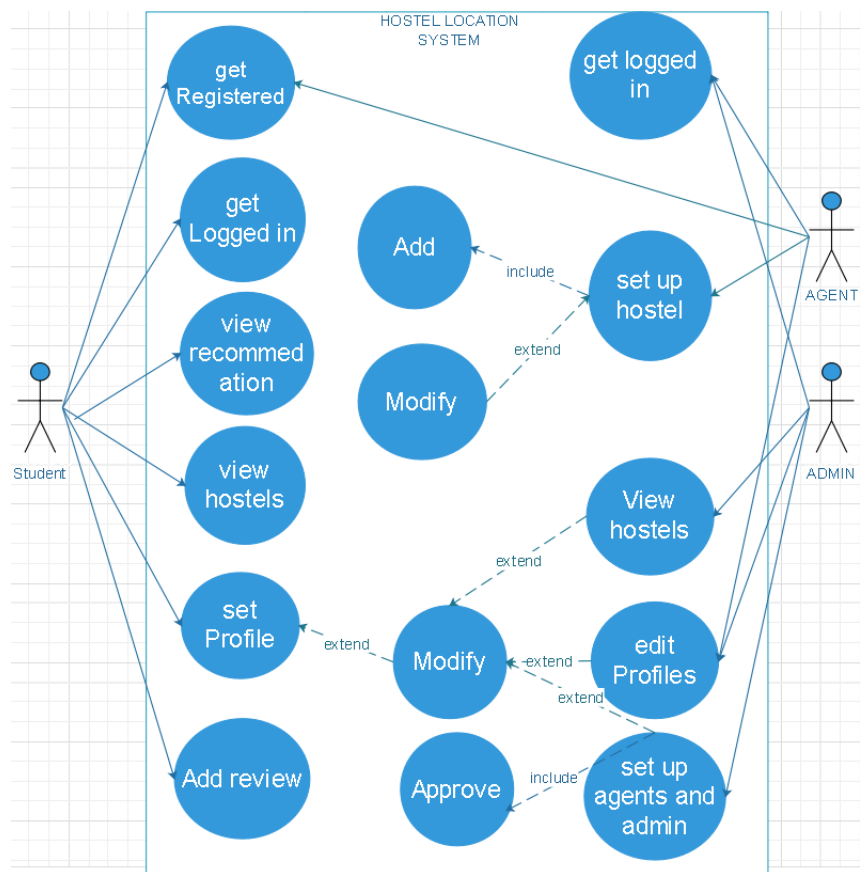


Figure 4: usecase diagram

6 Implementation

6.1 Implementation

To implement an AI chatbot with Python, you typically:

- Choose a Python library for AI and NLP.
- Train the chatbot with conversation data.
- Integrate with APIs if needed.
- Test and refine the chatbot's responses.
- Deploy the chatbot for users to interact with.

6.1.1 System Architecture

The system architecture of an AI chatbot in Python includes the user interface, chatbot application, NLP engine, dialogue management, response generation, machine learning models, data storage, and integration with external APIs, all hosted on a server or cloud platform.

6.1.2 Technology Stack

- PYTHON
- NATURAL LANGUAGE PROCESSING
- APIs
- CLOUD SERVICES
- DATABASE
- VERSION CONTROL
- MACHINE LEARNING FRAMEWORK
- CHATTERBOT


6.1.3 Deployment and Maintenance

Deploying and maintaining an AI chatbot built with Python involves several steps, including choosing a hosting platform, setting up the necessary infrastructure, deploying the chatbot code, monitoring its performance, and updating it as needed. Below is a general guide on how to deploy and maintain an AI chatbot using Python:


- Deploy Chatbot Code:
- Choose a Hosting Platform:
- Set up Infrastructure:
- Updating the Chatbot:

6.2 Register

- Sign Up for OpenAI API Access:
- Get Your API Key:
- Install OpenAI Python Client:
- Use the OpenAI API in Your Code:



Tell us about you




By clicking "Continue", you agree to our [Terms](#) and
acknowledge our [Privacy policy](#)

Figure 5: Chatgpt login page

Reset your password

Enter a new password below to change your
password.






Figure 6: Chatgpt register page