**Feature:**Implement comprehensive chatbot capabilities using Azure services

**Description:**From a developer's perspective, this user story covers the creation of missing frontend components for natural language question submission and user authentication, the development of multiple backend modules to integrate with Azure OpenAI services and ensure secure, scalable operations, and the establishment of configuration guidelines to comply with Azure policies and manage external constraints.

**frontend**

**Components:**chat\_interface.html, chat\_component.js, login\_component.js

**Details:**

* chat\_interface.html: Create a chat interface page that includes an input field for users to submit natural language questions and a dynamic area to display answers.
* chat\_component.js: Implement JavaScript logic to capture user input from the chat interface, send requests to the backend, and handle asynchronous responses.
* login\_component.js: Develop a user authentication component that enables users to log in and manage sessions, including UI feedback and error handling.

| **Features** | **Components** | **Input/Output Conditions** | **Validations** |
| --- | --- | --- | --- |
| Natural language question submission | chat\_interface.html | Input: User text query from the chat interface; Output: Display answer in chat format | Ensure non-empty and valid user input |
| Dynamic display of chat responses | chat\_component.js | Input: User credentials from login form; Output: Validated session token and UI feedback | Handle AJAX errors and invalid credentials |
| User login and session management | login\_component.js |  | Implement client-side security validations |

**backend**

**Components:**azure\_integration.js, security\_manager.js, scalability\_manager.js, deployment\_config.yaml, logging\_monitor.js, conversation\_manager.js, nlp\_module.js, error\_handler.js, knowledge\_base\_manager.js, auth\_controller.js

**Details:**

* azure\_integration.js: Create a module to integrate with Azure OpenAI API, handling the transmission of user queries and retrieval of responses.
* security\_manager.js: Implement secure communication channels between the chatbot and backend services, enforcing Azure security protocols.
* scalability\_manager.js: Develop a scalable architecture module that allows the system to handle high load and multiple concurrent users.
* deployment\_config.yaml: Create a deployment configuration file for Azure Cloud Services that includes settings for monitoring, maintenance, and auto-scaling.
* logging\_monitor.js: Set up logging and monitoring functionality to track performance metrics and capture errors for debugging.
* conversation\_manager.js: Implement support for multi-turn conversations to allow the chatbot to maintain context across follow-up questions.
* nlp\_module.js: Develop a module for natural language understanding that deciphers user intents and extracts key entities from queries.
* error\_handler.js: Provide mechanisms for fallback responses and user-friendly error messages when queries are unsupported.
* knowledge\_base\_manager.js: Design a component to allow dynamic updating of the chatbot's knowledge base as new information becomes available.
* auth\_controller.js: Implement backend authentication to manage user sessions securely, including token validation and session lifecycle management.

| **Features** | **Components** | **Input/Output Conditions** | **Validations** |
| --- | --- | --- | --- |
| Azure OpenAI integration for natural language processing | azure\_integration.js | Input: API requests from the frontend; Output: Processed natural language responses | Enforce secure API calls and data transmission |
| Secure backend communication using Azure protocols | security\_manager.js | Input: Authentication requests; Output: Valid session tokens and secure data exchange | Test system load under concurrent user sessions |
| Scalable architecture for high concurrency | scalability\_manager.js |  | Validate error handling and fallback scenarios |
| Azure-based deployment and monitoring | deployment\_config.yaml |  | Ensure compliance with Azure authentication standards |
| Robust logging and error tracking | logging\_monitor.js |  |  |
| Support for multi-turn conversations | conversation\_manager.js |  |  |
| Natural language understanding capabilities | nlp\_module.js |  |  |
| Fallback error responses | error\_handler.js |  |  |
| Dynamic knowledge base management | knowledge\_base\_manager.js |  |  |
| Backend user authentication | auth\_controller.js |  |  |

**Others**

**Components:**azure\_policy\_config.json

**Details:**

* azure\_policy\_config.json: Create a configuration file that enforces Azure OpenAI usage policies, token limitations, and security standards. Additionally, update documentation to address potential latency issues, scalability constraints based on budget, dependencies on Azure service availability, and limitations in AI model customization.

| **Features** | **Components** | **Input/Output Conditions** | **Validations** |
| --- | --- | --- | --- |
| Compliance with Azure OpenAI usage policies | azure\_policy\_config.json | Input: Azure policy parameters; Output: Validated and enforceable configuration settings | Check adherence to Azure security standards |
| Clear documentation of latency and scalability constraints |  |  | Validate token limits and policy compliance |