**Design**

**And**

**Approach**

**Design & Approach Document: Document Search Bot**

**Objective:**

The Document Search Bot is designed to facilitate users in searching through uploaded documents using natural language queries. This document outlines the architecture, workflow, user roles, frontend and backend components, and third-party libraries used in the development of the application.

**Workflow:**

**Frontend:** React.js

**Backend:** ASP.NET Core Web API

**Backend Components:**

1. **Login Controller:**
   * Handles user authentication and authorization.
   * Utilizes hardcoded credentials for user and admin authentication.
   * Distinguishes between admin and user roles.
2. **Document Processing Controller:**
   * Manages document upload, deletion, and retrieval operations.
   * Uses Azure Blob Storage for storing uploaded documents.
   * Accepts only .docx, .ppt, and .xlsx file formats for upload.
   * Provides APIs for admin to manage documents.
3. **Chat Controller:**
   * Interacts with the AI model (GPT-3.5) to answer user queries.
   * Extracts text content from uploaded documents in Azure Blob Storage.
   * Sends content along with user query to the AI model for response generation.

**Frontend Components:**

1. **Login Component:**
   * Allows users to log in as admin or user.
   * Displays different UI components based on user role.
   * Redirects to respective components after successful login.
2. **Chat Component:**
   * Provides a chat-like interface for users to interact with the AI model.
   * Sends user queries to the backend for processing and receives AI-generated responses.
3. **Document Upload Component:**
   * Visible only to admin users.
   * Enables admin to upload documents to Azure Blob Storage.
   * Shows a list of uploaded documents.
4. **Logout:**
   * Logs the user out and redirects to the login page.

**User Roles:**

1. **Admin:**
   * Accesses all components (Chat, Document Upload).
   * Manages documents and interacts with the AI model.
2. **User:**
   * Accesses only the Chat component.
   * Asks queries and receives responses.

**Third-party Libraries:**

1. **Aspose.Words:** Extracts text from .docx files.
2. **DocX:** Processes .docx files.
3. **Newtonsoft.Json:** Handles JSON serialization and deserialization.
4. **Microsoft.Extensions.Logging:** Logs application events.
5. **Xceed.Words.NET:** Works with Word documents.
6. **Microsoft.Azure.Storage:** Interacts with Azure Storage services.
7. **DocumentFormat.OpenXml:** Processes XML-based document formats like PowerPoint and Excel.

**Future Enhancements:**

1. Implement user authentication using tokens or JWT for enhanced security.
2. Enhance document processing capabilities to support additional file formats.
3. Implement caching mechanisms for improved performance.
4. Implement error handling and logging for better debugging.
5. Integrate semantic search techniques for more accurate search results.

**Conclusion:**

The Document Search Bot application provides users with a user-friendly platform to search through documents using natural language queries. With a robust architecture and integration of third-party libraries, the application efficiently manages document processing, user authentication, and AI-based response generation. Future enhancements aim to further enrich the application's capabilities and performance.