**COMPANION – GenAI Based Application**

|  |
| --- |
| **CONFIDENTIALITY STATEMENT**  Information and data contained herein are proprietary and confidential. This information should not be disclosed to any third party without the prior written consent of Amgen Incorporated. |

**Revision History**

| **Version** | **Change Reference** | **Changes** | **Initiator** |
| --- | --- | --- | --- |
| 0 | <<Change No.>> | Initial version of the document | Kishore |
| 1 | - | Version-2 | Vinay Shankar |

Contents

[1. SYSTEM OVERVIEW 6](#_Toc178613733)

[**Purpose & Scope** 6](#_Toc178613734)

[2. DEFINITION AND ABBREVIATION 6](#_Toc178613735)

[**Definitions** 6](#_Toc178613736)

[**Abbreviations** 6](#_Toc178613737)

[**References** 7](#_Toc178613738)

[3. SYSTEM OVERVIEW 8](#_Toc178613739)

[3.1 General 8](#_Toc178613740)

[3.2 Scope 8](#_Toc178613741)

[3.2.1 High Level Architecture 8](#_Toc178613742)

[3.2.2 Companion Web Application Layer 8](#_Toc178613743)

[3.2.3 API Layer – Django 10](#_Toc178613744)

[ **Authentication and Access Control:** 21](#_Toc178613775)

[**1.1.1.** **LLM Layer** 21](#_Toc178613776)

[**2.1.** **System Design Constraints** 26](#_Toc178613777)

[**2.2.** **System Design Strategies** 26](#_Toc178613778)

[**System Classification** 26](#_Toc178613779)

[**Security** 26](#_Toc178613780)

[**3.1.** **System Configuration Settings** 34](#_Toc178613781)

[**3.3.** **GIT Repository** 36](#_Toc178613782)

[**3.3.1.** **Front End** 36](#_Toc178613783)

[**3.3.2.** **Backend** 36](#_Toc178613784)

[**3.3.3.** **User Interface** 36](#_Toc178613785)

[ **https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=212-7124&t=304qaOGuhPx1zzcW-0** 36](#_Toc178613786)

[ **https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=3-5&t=jxw4BM5uZQ5elSP7-0** 36](#_Toc178613787)

[ **https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=74-5628&t=OHVt7SYTUbPPDVRw-0** 36](#_Toc178613788)

[ **https://url.usb.m.mimecastprotect.com/s/vcIpCp9WQ8snVgqkiP3se4?domain=figma.com** 36](#_Toc178613789)

[**3.4.** **Use Cases** 37](#_Toc178613790)

[***3.4.1.*** **Use Cases -** 37](#_Toc178613791)

[**3.5.** **DEPLOYMENT** 37](#_Toc178613792)

# SYSTEM OVERVIEW

**Purpose & Scope**

This document describes the system design required to support the Companion, which is a Generative AI assistant, that enables medical affairs team to use natural language to search and synthesis content from documents and generates summaries & insights. The document describes the system design, configuration and integration and forms a basis for the testing and acceptance of the system.

# DEFINITION AND ABBREVIATION

**Definitions**

| **Term** | | **Definition** |
| --- | --- | --- |
| Architecture | The organizational structure of a system or component. | |
| AWS S3 | Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. | |
| Django | Python Framework for building scalable server-side application | |
| EC2 | Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. | |
| React JS | React is a free and open-source front-end JavaScript library for building user interfaces or UI components. | |
| MongoDB | NO-SQL based database for storing JSON like documents. | |
| Milvus DB | It is a powerful vector **database** tailored for processing and searching extensive vector data. | |
| Okta | Okta Authentication is a cloud-based identity management service that adds authentication and authorization services to applications. | |
| AWS Textract | AWS Textract is an Amazon Web Services (AWS) service that allows the user to extract text and data from scanned documents. | |
| Embedding Model | Algorithms are trained to encapsulate information into dense representations in a multi-dimensional space. | |
|  |  | |

**Abbreviations**

| **Abbreviation** | **Meaning** |
| --- | --- |
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| S3 | Simple Storage Service |
| LLM | Large Language Model |
| GPT | Generative Pre-Trained Transformers |
|  |  |

**References**

The following Amgen standards and SOPs are used for guidance and/or reference throughout this Design Specification:

| **Document Number** | **Document Title** |
| --- | --- |
|  |  |
|  |  |

# SYSTEM OVERVIEW

## General

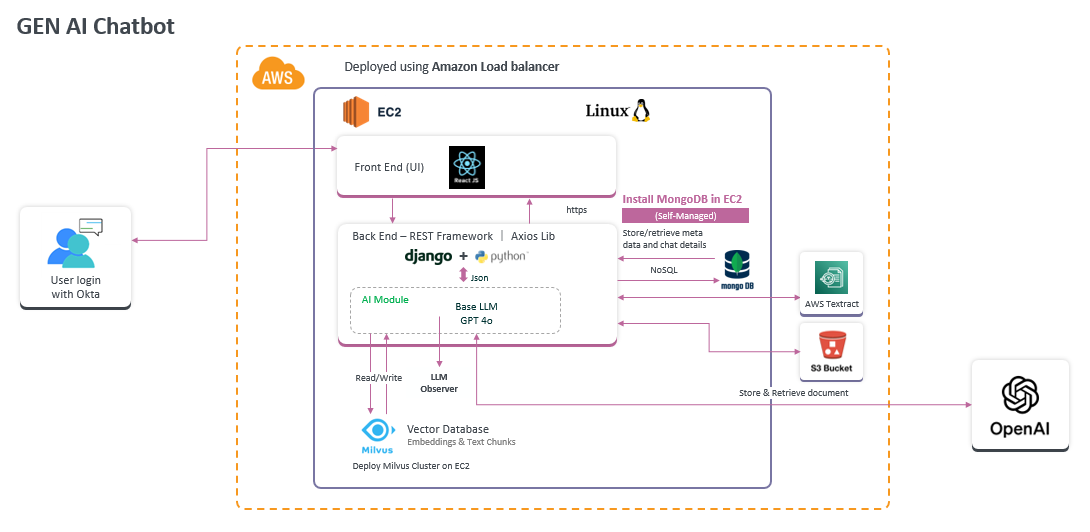
Companion is a Generative AI assistant, which enables medical affairs team to use natural language to search and synthesis content from documents and generates summaries & insights.

The key functionality of the application is :

* + - Generate content, insights, key points & responses by querying in natural language.
    - Ability to quickly digest large amounts of publications, congress materials etc.
    - Effortlessly access citations for each sentence of the response, along with links to corresponding documents for further reference.
    - Tailored response formats such as tables, bullet points, and more, based on the user's query.

## Scope

### High Level Architecture



### Companion Web Application Layer

The application provides following features and functionality

* Okta Login
  + Okta Authentication is a cloud-based identity management service that adds authentication and authorization services to applications.
  + Okta validates the user in particular group and application in the okta portal and returns the validation.
  + If the authentication is successful, the application allows the user else it will block with warning message.
* Temporary File Upload and Querying
  + The home page of the application has the functionality to upload the pdf documents and query on top of it.
  + Once the backend process of vectorizing the documents is complete, the user can either summarize the documents or ask any question related to it.
* Navigation of Chats and History
  + Each Questions/Answers sessions are recorded in MongoDB and showed as history on the sidebar of the application.
  + Users can access till the last 30 days of records in history.
  + Chat histories are shown with proper title as per the Q/A and indicates if it is from upload document session.
  + Each chat session is active so that the user can ask the following questions to it.
* Knowledge base search
  + Each user will be tagged with a BU name as a default configuration, and he/she can query only from that knowledge case.
  + Assigned BU name is shown in header of the application and clicking of it, shows all list of all documents it contains with readable links.
  + Each chat has the option to give feedback, like or dislike and the ability to copy the response.
* Image based response
  + Each response will contain 2 additional fields in the answer block i.e., Sources and Related Images
  + If the answer block has relevant images, user can expand the accordion to see the images displayed under respective document name with below details:
    - * Pdf link from where the image is extracted
      * Image preview
      * Expand icon to view image details
      * Pdf name and image title
      * Image caption
  + Expand Icon: Each image will have expand icon and clicking on it, user can see detailed description, image preview and link to the pdf page from where image has been extracted. Users also have the option to provide feedback for each image in the pop-up.
  + If user asked to load only images, then related images accordion will get expanded default and directly images will get loaded.

### API Layer – Django

* **FAQ**
  + URL - /api/v1/faq
  + GET - Fetches FAQ details for Companion.
  + POST - Creates a new FAQ entry in the Mongo collection.
* **USER**
  + URL - /api/v1/users/validate
  + POST – Validates user access to the application.
    - Params - { “email”: [xyz@amgen.com](mailto:xyz@amgen.com)}
* **Stream-Data**
  + URL - /api/v1/stream\_data/
  + POST – Returns streaming data in response to a prompt.
    - Params - { “prompt”:”xyz”, “session\_id”:”xyz”, “regenerate”: Bool, “user\_bu”:”xyz”, “is\_upload”:Bool, [email”:”xyz@amgen.com](mailto:email)}
* **ChatPrompt**
  + URL - /api/v1/chat\_prompt/<msg\_id>
  + GET – Returns additional details related to streaming data for the given message ID, such as sources, related images and document URLs.
* **UploadPrompt**
  + URL - /api/v1/upload\_prompt/upload
  + POST – Uploads user-provided documents to S3 and stores in vector DB.
    - Params {“email”:”xyz@amgen.com”,”Files”: file-objects}
* **Feedback**
  + URL - /api/v1/feedback
  + PUT – Updates feedback posted on a prompt-response document in Mongo.
    - Params - [{“email”:”xyz@amgen.com](mailto:%7b)”, “ratings”:”xyz”, “feedback”:”xyz”,”tag\_ratings”:”xyz”}
* **Config**
  + URL - /api/v1/config
  + POST – Provides details on upload\_rate, history\_list, and user\_bu.
    - Params - [{“email”:”xyz@amgen.com](mailto:%7b)”}
  + URL - /api/v1/config/doc\_list/<bu\_name>
  + GET – Provides a list of document URLs based on the provided BU.
  + URL - /api/v1/config/history/<session\_id>
  + GET – Provides all chats based on the given session ID.

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Endpoint** | **Request** | **Response** |
| **Okta** | okta\_url/default/v1/token | client\_id: 0oa22q39o5uGOWESV0h8  redirect\_uri: app\_url  grant\_type: authorization\_code  code\_verifier: 7439a34a3b94fc3deb54c9a5cdd473e8b95c1c75774  code: UeigNgAXINSyXigRHmYh8FRsn-F1JvH-MoTRZDqYcoI | {  "token\_type": "Bearer",  "expires\_in": 3600,  "access\_token": "eyJraWQiOiJvclk3ZVhoT3lVLVVaY0VEUzI4ejlKeDhLOHVHNFZXVE5mSWJabDNVT2NVIiwiYWxnIjoiUlMyNTYifQ",  "scope": "email profile openid",  "id\_token": "mZl9pZCI6IjM3MzgyNiIsIlVzZXJfSUQiOiJ2c2hhbmswMiJ9.zvzSCmay0iLC6z5fmeFAvtm-y9\_9RHeyYgLSAFxxfdp74vkrcRvqeJBWrhh\_sYCpkS "  } |
| **Application** | /api/v1/config | {  "email": "[vshank02@amgen.com](mailto:vshank02@amgen.com)"  } | {  "history\_list":{  "seven\_days":[  {  "chat\_title":"\"Uplizna: NMOSD Treatment Overview\"",  "isUpload":false,  "session\_id":"667e9c7fe77efe5b192673d7"  }  ],  "thirty\_days":[  {  "chat\_title":"\"Uplizna: NMOSD Treatment Overview\"",  "isUpload":false,  "session\_id":"667abfc63e12f3e14d3eea8f"  }  ],  "yesterday":[  {  "chat\_title":"\"Uplizna: NMOSD Treatment Overview\"",  "isUpload":false,  "session\_id":"6683f6d9e77efe5b192674de"  },  {  "chat\_title":"\"Overview of Uplizna (Inebilizumab)\"",  "isUpload":false,  "session\_id":"6683ab54e77efe5b19267437"  }  ],  "today":[  {  "chat\_title":"\"Uplizna: NMOSD Treatment Overview\"",  "isUpload":false,  "session\_id":"6683f6d9e77efe5b192674de"  }  ]  },  "upload\_rate":189207,  "user\_bu":"UPLIZNA"  } |
| Faq | /api/v1/faq | - | {  "FAQs":[  {  "faq\_string":"Uploaded Docs",  "questions":[  {  "answer":"Documents uploaded will not automatically be added to the internal corpus of data. As such, your colleagues are not able to access files that you have uploaded to the tool. Documents will be added to the internal corpus after yearly assessment by the owner of the internal database.",  "question":"Will documents that I upload be added to the internal corpus of data? Can my colleagues view documents that I upload?"  }  ],  "type":"faq"  },  {  "faq\_string":"Internal Corpus of Data",  "questions":[  {  "answer":"Medical Affairs and Medical Information team members for each BU determine the key documents that Companion should have access to. These documents are combined into a collection.",  "question":"How are Companion's data collections started?"  }  ],  "type":"faq"  },  {  "faq\_string":"External Data Sources",  "questions":[  {  "answer":"Currently Companion has no access to external data sources.",  "question":"What external databases does the tool have access to?"  } ],  "type":"faq"  },  {  "faq\_string":"Analyzing Documents & Generations Results",  "questions":[  {  "answer":"This tool is based on a Generative AI Large Language Model that is able to process user queries in natural language and intake many data sources to provide an answer.",  "question":"How does the tool analyze documents?"  } ],  "type":"faq"  },  {  "faq\_string":"Exporting & Sharing Results",  "questions":[  {  "answer":"This tool is currently able to produce its outputs in text format along with source details and link to document. In the future, the tool may be able to generate images with its responses.",  "question":"In what formats is the Companion able to produce its outputs? Can Companion generate images with its results?"  },  {  "answer":"The tool is not currently able to produce results in any pre-defined templates. In the future, it will be included in its functionality.",  "question":"In what templates is the tool able to produce its results (i.e. standard response letter, literature review, etc.)?"  } ],  "type":"faq"  },  {  "faq\_string":"Querying",  "questions":[  {  "answer":"Simply ask Companion a question in natural language in order to find specific pieces of information in documents.",  "question":"How do I find more specific pieces of information within documents?"  }  ],  "type":"faq"  }  ],  "\_id":"6666d10def94eec5b0f15245"  } |
| BU DOC List | /api/v1/config/doc\_list/TEPEZZA |  | [  {  "02-TED 2022 JCEM supplement.pdf": "<https://companion-app-document.s3.amazonaws.com/KNOWLEDGEBASE/TEPEZZA/02-TED%202022%20JCEM%20supplement.pdf?response-content-disposition=inline&response-content-type=application%2Fpdf>"  },  {  "A New Era in the Treatment of Thyroid Eye Disease-01.pdf": "<https://companion-app-document.s3.amazonaws.com/KNOWLEDGEBASE/TEPEZZA/A%20New%20Era%20in%20the%20Treatment%20of%20Thyroid%20Eye%20Disease-01.pdf?response-content-disposition=inline&response-content-type=application%2Fpdf>"  },  {  "Cockerham\_Quality of Life in Chronic Thyroid Eye Disease Patients in the United States.pdf": "<https://companion-app-document.s3.amazonaws.com/KNOWLEDGEBASE/TEPEZZA/Cockerham_Quality%20of%20Life%20in%20Chronic%20Thyroid%20Eye%20Disease%20Patients%20in%20the%20United%20States.pdf?response-content-disposition=inline&response-content-type=application%2Fpdf>"  }  ] |
| History Session | /api/v1/config/history/668543b3cd24d19a703477cf/ |  | [  {  "\_id": "668543cbcd24d19a703477d5",  "created\_at": "2024-07-03T12:27:55.168000",  "is\_stream": true,  "llm\_call\_time": 7.261878,  "message": "## Understanding TED in the Medical Context\n\nTED, or Thyroid Eye Disease, is a complex autoimmune disorder that primarily affects the eyes and the surrounding tissues. It is often associated with immune-mediated thyroid dysfunction, such as Graves' disease. Below is a detailed explanation of TED, including its pathogenesis, symptoms, and treatment options.\n\n### Pathogenesis of TED\n\nTED is an autoimmune-mediated, inflammatory, and fibrotic orbitopathy.",  "prompt": "What is Ted?",  "session\_id": "668543b3cd24d19a703477cf",  “image\_details: [  {  "caption": "Fig. 1 Pathogenesis of AQP4-seropositive NMOSD and targets of biologics. IL-6 promotes T-cell differentiation into Th17, B-cell survival, and anti-AQP4-IgG production by plasma cells. In turn, B cells and Th17 release IL-6 and B-cell activating factor (BAFF), respectively, strengthening the circuit. Plasma cells produce anti-AQP4-IgG, which cross the blood–brain barrier, bind the AQP4 water channel on astrocytes, and trigger the complement cascade, with subsequent oligodendrocyte injury and demyelination. Anaphylatoxins and other cytokines recruit eosinophils and neutrophils, which exert a non-selective damage on nearby tissue. Biologics act on IL-6 receptors (tocilizumab and satralizumab, red cross), CD20+ cells (rituximab, orange cross), and CD19+ cells (inebilizumab, green cross) or on the complement cascade (eculizumab, blue cross). Figure is created with BioRender.com. AQP4 aquaporin-4, BAFF B-cell activating factor.",  "document\_name": "Cacciaguerra-Targeting NMOSD Pathogenesis.pdf",  "document\_url": "https://companion-knowledge-base-prd.s3.amazonaws.com/KNOWLEDGEBASE/UPLIZNA/Cacciaguerra-Targeting%20NMOSD%20Pathogenesis.pdf?response-content-disposition=inline&response-content-type=application%2Fpdf&AWSAccessKeyId=AKIA6GBMCA6HZKKEWD7T&Signature=%2BEZL4LWgcctOpx4OvPDDm8cBUB8%3D&Expires=1727607435",  "img\_id": "448871447457714447",  "page\_number": 7,  "path": "KnowledgeBase\_Images/UPLIZNA/Cacciaguerra-Targeting NMOSD Pathogenesis/page\_7\_image\_6.png",  "pdf\_path": "KNOWLEDGEBASE/UPLIZNA/Cacciaguerra-Targeting NMOSD Pathogenesis.pdf",  "relevance\_score": 0.0010237869573757052,  "source": "internal",  "title": "Pathogenesis of AQP4-seropositive NMOSD and Targets of Biologics",  "url": "https://companion-knowledge-base-prd.s3.amazonaws.com/KnowledgeBase\_Images/UPLIZNA/Cacciaguerra-Targeting%20NMOSD%20Pathogenesis/page\_7\_image\_6.png?response-content-disposition=inline&response-content-type=image%2Fpng&AWSAccessKeyId=AKIA6GBMCA6HZKKEWD7T&Signature=1Pme6Qjwkki1hlbGhM3%2FrxYC870%3D&Expires=1727607435"  }  ]  "source": [  {  "ChunkNumber": 1,  "DocumentName": "Do\_Thyroid Eye Disease - Pathogenic Risk Factors.pdf",  "author": "Thai H Do, Alon Kahana",  "content": [  {  "pageNumber": 7,  "score": 0.998161256313324,  "source": "## Conclusion\n\nTED is an autoimmune mediated, inflammatory/congestive/fibrotic orbitopathy that is often associated with immune-mediated thyroid dysfunction.",  "tag": "1a"  }  ],  "flag": "Internal",  "url": "<https://companion-app-document.s3.amazonaws.com/KNOWLEDGEBASE/TEPEZZA/Do_Thyroid%20Eye%20Disease%20-%20Pathogenic%20Risk%20Factors.pdf?response-content-disposition=inline&response-content-type=application%2Fpdf&AWSAccessKeyId=AKIATPU2MLEGA4IIR7HT&Signature=p0eK%2BxCvPo3qZxJSXZClBANz0SQ%3D&Expires=1720182475>"  }  ],  "status": "Success",  "streaming\_response\_time": 15.935582,  "updated\_at": "2024-07-03T12:27:55.168000"  }  ] |
| Doc Upload | /api/v1/upload\_prompt/upload | Yen\_Thyroid Eye Disease\_Smallest\_Document.pdf: (binary)  email: [apatel27@amgen.com](mailto:apatel27@amgen.com) | {  "data": {  "session\_id": "668d13d6be510d4244fda939",  "status": "success"  },  "is\_exists": false,  "status": "success",  "success": true  } |
| Streaming Response | /api/v1/stream\_data/ | {  "prompt": "What is Present in the uploaded document?",  "session\_id": "668d13d6be510d4244fda939",  "email": "[apatel27@amgen.com](mailto:apatel27@amgen.com)",  "regenerate": false,  "user\_bu": "TEPEZZA",  "is\_upload": true  } | Streaming Objects as below -  {'type': 'answer', 'data': ''}@@    {'type': 'answer', 'data': '##'}@@    {'type': 'answer', 'data': ' Thy'}@@    {'type': 'answer', 'data': 'roid'}@@ |
| Additional details for the streaming data | /api/v1/chat\_prompt/668d1575be510d4244fda963 |  | {  "data": {  "\_id": "668d1575be510d4244fda963",  "chat\_title": "What is TED?",  "created\_at": "2024-07-09T10:48:21.605000",  "is\_stream": true,  "llm\_call\_time": 3.066014,  "message": "## Thyroid Eye Disease: Current Concepts and State of the Art Treatment\n\n### Introduction\nThe year 2020 has been significant for the medical field, not only due to the global pandemic but also because of groundbreaking advancements in the treatment of Thyroid Eye Disease (TED).",  "prompt": "What is Present in the uploaded document?",  "session\_id": "668d13d6be510d4244fda939",  "source": [  {  "DocumentName": "Yen\_Thyroid Eye [Disease\_Smallest\_Document\_@\_777312e3-a07f-4082-bd62-b601747e0b98.pdf](mailto:Disease_Smallest_Document_@_777312e3-a07f-4082-bd62-b601747e0b98.pdf)",  "PageNumber": null,  "author": "Not Available",  "content": "",  "flag": "Uploaded By You",  "id": 1,  "url": "<https://companion-app-document.s3.amazonaws.com/UPLOADED_DOCS/Yen_Thyroid%20Eye%20Disease_Smallest_Document_%40_777312e3-a07f-4082-bd62-b601747e0b98.pdf?response-content-disposition=inline&response-content-type=application%2Fpdf&AWSAccessKeyId=AKIATPU2MLEGA4IIR7HT&Signature=unH0%2BVlEmZMMf1pnElZL%2BxVR9LA%3D&Expires=1720694901>"  }  ],  "status": "Success",  "streaming\_response\_time": 16.564552,  "updated\_at": "2024-07-09T10:48:21.605000"  },  "is\_exists": false,  "status": "success",  "success": true  } |

* **Authentication and Access Control:**
  + The application uses Okta for user authentication and authorization. Users are validated based on their credentials and group permissions managed through Okta.
  + Successful authentication grants access to the application's features, while unauthorized attempts are blocked with appropriate messages.
* **Document Management and Querying:**
  + Users can upload PDF documents to the application.
  + Uploaded documents undergo backend processing to extract and analyze content.
  + Functions allow users to summarize uploaded documents or query specific information from them.
* **Chat History and Interaction:**
  + Each interaction session (Q&A) is recorded and stored in MongoDB.
  + Users can view their chat history, which includes sessions categorized by titles indicating whether they originated from document uploads.
  + Access to chat history is available for up to the last 30 days.
* **Knowledge Base and Search Functionality:**
  + Users are associated with specific Business Units (BU), restricting their search and access to knowledge bases relevant to their assigned BU.
  + The application provides a searchable index of documents linked to each BU, accessible through user-friendly links.
* **Feedback and Interaction Features:**
  + Users can provide feedback on chat sessions, expressing satisfaction (like) or dissatisfaction (dislike) with responses.
  + Responses can be copied directly from the chat interface for further use or sharing.
* **API Integration:**
  + The application includes a backend API layer built on Django, supporting various endpoints for managing FAQs, user validations, data streaming, document uploads, feedback updates, and configuration settings.
  + These APIs facilitate seamless integration of different functionalities within the application, supporting its interactive and data-driven features.
* This overview encapsulates the main functionalities of the application, highlighting its emphasis on secure user authentication, document handling, interactive chat history, knowledge base search capabilities, user feedback mechanisms, and robust API integration.
  + 1. **LLM Layer**

Explain LLM flow for JSON creation and Vectorization here

Backend Module calls AI Core Module (ai\_core.py) for different functionalities -

1. **ai\_core.py module**

* **Chat Response** - own\_chat\_ai\_call
  + Function is called for each query entered by user in the chat box.
  + Returns a dictionary with final response as a streaming object along with the list of sources (pdf name & text chunks) from which answer is formulated.
  + To generate response this function calls the LLM Agent workflow which might includes creating multiple LLM calls depending on question asked.
* **Upload & Vectorize Document** – vectorizePdfs
  + Upload document flow –
    - Upload to S3 -> Extract PDF Data (Textract) -> Chunking -> Vectorizing -> Milvus Vector DB
  + When a document is uploaded, Backend uploads document to S3 and calls vectorizePdfs function.
  + The function calls **extractPDFData** which calls Textract. Textract generates a JSON with all the extracted data out of PDF.
  + PDF Text is extracted and chunked using semantic chunking from this JSON.
  + This data is then vectorized using OpenAI Embedding Model “text-embedding-3-small” and saved as a collection Milvus vector DB.
* **Generate Chat Title** – generate\_chat\_title
  + For each chat session, a chat title is generated which is used to display chat in chat history.

1. **openai\_client.py module ->**

* Module defines open ai client and functions for calling Open AI APIs.
* defines functions for calling open ai api to create embeddings of the text.
* Defines open ai client, async open ai client and functions for chat completion api calls (Async, streaming and normal chat completion call).

1. **milvus\_client.py module ->**

* Module defines milvus client to connect to milvus vector DB
* Variable to store list of collections in the DB.

1. **openai\_prompts.py module ->**

* Defines all prompts used in the generating response, summary, comparison or chat title
* Following are the list of prompts defined –
  + systemMessage
  + summarySystemPrompt
  + summaryContentPrompt
  + combineSummaryTemplate
  + qaPrompt
  + GenerateChatTitlePrompt
  + ComparisonPrompt

1. **conversation\_memory.py module ->**

* Defines Conversation class providing following methods –
  + Initialize empty conversation.
  + Get conversation id.
  + Add message to conversation.
* It is used to store all chat history of user and AI messages.

1. **openai\_tools.py module ->**

* Tool definitions along with description of tools and parameters.
* Plant & Execute Agent would have option to choose from these tools according to the tool description and generate the tool inputs (parameter values) as per the parameter descriptions.
* **Knowledgebase tools –**
  + KnowledgeBaseAssistant
* **Upload tools –**
  + UploadedKnowledgeBase
  + InternalKnowledgeBase
  + SummarizeUploadedDocuments
  + ComparingDocumentsAssistant

1. **pdfParserTextract.py module ->**

* Used to process the uploaded pdf files and generate chunks along with metadata. (Refer figure 5.4.)
* Takes input as s3 location of the pdf files and their names.
* Parses pdf files using AWS Textract API call -> generates raw json -> calls chunking functions.
* Parallel processing of pdf files.

1. **getChunks.py module ->**

* Takes Raw JSON from Textract as input and extracts text and tables chunks in required format along with metadata.
* Extract text in Markdown format and tables as strings from all pages of the document.
* Concatenates all pages text in a single string.
* Semantic Chunking of the text using MarkdownSplitter of semantic\_text\_splitter library which gives us chunks of 500 to 1000 tokens each.
* These chunks are stored along with metadata like author, page number etc.
* Author of document is extracted by utilizing NER of Spacy on first 4 chunks of the document.
* Each table is returned as separate chunk.

1. **doc\_retriever.py module ->**

* Retrieves and returns relevant chunks from the vector database by utilizing the user question and the tool inputs like search\_queries, preliminary\_answer & answer\_depth.
* It firsts creates a semantic search query = user question + search queries + preliminary answer
* Extracts chunks from milvus vector db. Number of chunks depends on answer depth, 30 if answer\_depth == "High", 20 if answer\_depth == "Medium" else 15 if answer\_depth == "Low".
* These chunks are then reranked according to user question using a Reranker model.

After reranking, we choose top 10 chunks for “High” answer depth, 7 for “Medium” else 5 for “Low”.

1. **image\_retriever.py module ->**

* Retrieves and returns relevant images from the vector database by utilizing the user question and the tool inputs like search\_queries, preliminary\_answer & answer\_depth.
* It firsts creates a semantic search query = user question + search queries + preliminary answer
* Extracts chunks from milvus vector db. Number of images depends on answer depth, 20 if answer\_depth == "High", 15 if answer\_depth == "Medium" else 10 if answer\_depth == "Low".
* These images are then reranked according to user question using a Reranker model.
* After reranking, we choose top 6 images for “High” answer depth, 6 for “Medium” else 4 for “Low”. Images are filtered based on reranker score to be greater than **85**.
* It uses get\_only\_images parameter indicating if user asked only to show imgaes in UI. If this flag is True, then logic is applied to show at least 2 images even if reranker score is below 85.

1. **QA\_tools.py module ->**

* Defines Question Answering Tool function which is called when any of the below tool calls are made by Plan & Execute Agent –
  + KnowledgeBaseAssistant
  + UploadedKnowledgeBase
  + InternalKnowledgeBase
* Milvus Collection Name is changed if it is for uploaded documents.
* It takes inputs generated by Agent and retrieves relevant chunks.
* Then prompt is formulated using context, user question, answer depth & additional instructions.
* Response along with sources from open ai call are returned back as streaming object.

1. **doc\_summary.py module ->**

* Defines tool function for generating summary of the given documents.
* Summary Tool function is called when ‘SummarizeUploadedDocuments‘ tool call is made by Plan & Execute Agent.
* Milvus vector db collection is queried for the given documents and all chunks for them are extracted.
* Then summary prompt is formulated using chunks, summary\_objective, key\_focus\_areas, desired\_length & additional instructions.
* If chunk length < 75000 tokens then document is summarized in a single open ai call and returns the streaming object.
* If chunks > 75000 tokens then for each chunk of 75K tokens a summary is generated in parallel and then a combined summary of these summaries are returned as a streaming object.

1. **doc\_comparison.py module ->**

* Defines tool function for generating comparison of the 2 uploaded documents.
* Comparison Tool function (getComparsion) is called when ‘ComparingDocumentsAssistant’ tool call is made by Plan & Execute Agent.
* Milvus vector db collection is queried for the given documents names and all chunks for them are extracted.
* For both the uploaded documents, it is checked whether the total size of document content is < 40000 tokens. If greater than 40k tokens then standard message is returned to be shown in UI.
* Comparison prompt is formulated using chunks, document names & key\_focus\_areas and OpenAI is called.
* Finally, Comaprison of the 2 documents is returned as a streaming object.

1. **agent\_workflow.py module ->**

* Defines the flow of LLM calls to generate a response with sources for the given user question.
* It calls conversation memory class and adds formatted system message, chat history and user question.
* Calls Plan & Execute Agent (LLM). If it returns direct response then response is returned as streaming object otherwise a Tool call is made.
* Processes the output of the Plan & Execute agent, identifies which tool to call and its arguments.
* It defines functions complete Tool calls depending on output of Agent.
* Response from the tool call is returned as streaming object which will have answer to user question along with source details.

1. **DESIGN CONSTRAINTS AND STRATEGIES** 
   1. **System Design Constraints**

For ongoing internal support and continued development of the system are employed using Amgen standard tools and platforms.

* All data in the source system is refreshed periodically.
* All the data type of EDF layers are supported to IEG layer.
* Data Quality and Integrity issues shall have to be primarily addressed in the Source System as the proposed System, per se, shall not change any data values and will only restrict itself in aggregating and sorting data at its end.
* The Amgen Logo should be present using a tint of Amgen Blue (RGB: 0-99-195).
* The EDF source table & column names are currently hardcoded in the notebooks.
  1. **System Design Strategies**

**System Classification**

The Smart Study Search (S3) solution is classified as Custom Developed Software according to the definitions provided in Non-Validation of GxP Business Systems.

**Security**

Access to the Companion Application will be restricted to authorized users through Amgen SSO Login.

*Authentication*

All users are authenticated using Okta

1. **DETAIL DESIGN**

A screenshot of a computer

Description automatically generated

Figure 5.1. Functional Flow Diagram

A screenshot of a computer

Description automatically generated

Figure 5.2. Gen AI high Level Process Flow Diagram

A screenshot of a computer

Description automatically generated

Figure 5.3. Document Processing: AWS Textract Parser and Milvus Flow Diagram

A screenshot of a computer

Description automatically generated

Figure 5.4. AWS Textract Parser Implementation

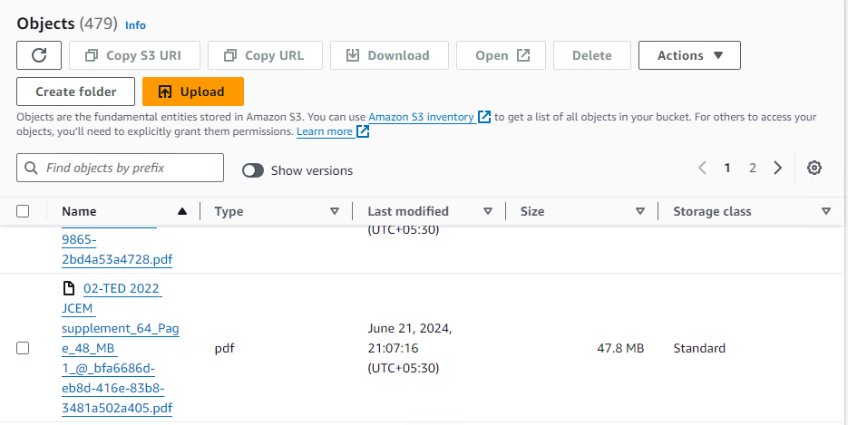
**Overview:**

Below is the list of components/technologies used and their high-level use case -

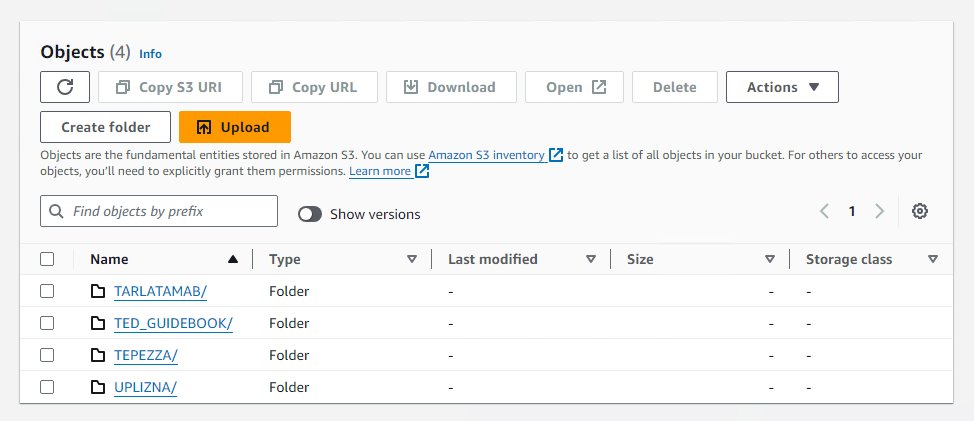
* **AWS Textract**
* **Milvus DB**
* **Vectorization**
* **Embedding Model (text-embedding-3-small)**

**S3 Folder structure:**

Amazon S3 > Buckets > companion-app-documents > UPLOADED\_DOCS



Amazon S3 > Buckets > companion-app-documents > KNOWLEDGEBASE



Amazon S3 > Buckets > companion-app-documents > KnowledgeBase\_Extracts

A screenshot of a computer

Description automatically generated

Amazon S3 > Buckets > companion-app-documents > KnowledgeBase\_Metadata

A screenshot of a computer

Description automatically generated

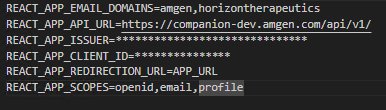
Amazon S3 > Buckets > companion-app-documents > KnowledgeBase\_Images

A screenshot of a computer

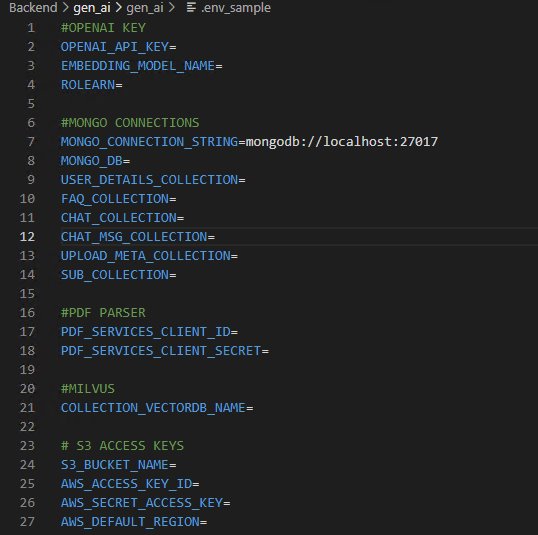
Description automatically generated

**Environment Variables:**

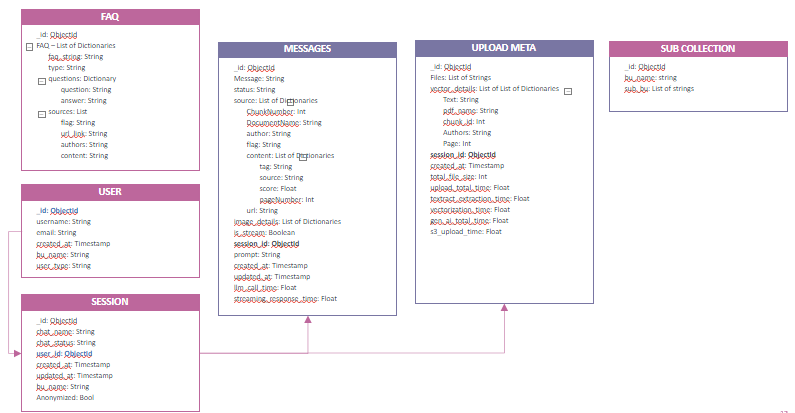
**Frontend:**



**Backend:**



**Database Architecture – No SQL**



**Milvus Database and Structure**

A screenshot of a computer

Description automatically generated

* 1. **System Configuration Settings**

**Development Environment**

The development environment will be used for development & testing.

**Production Environment**

The production environment is a controlled environment used by Amgen Companion users only. Installation Testing will be performed in the production environment as part of the deployment.

* **Security configuration:**

This application shall be implemented on IEG platform and all the security configurations of the platform per SPEC-430126 shall be applied to this application.

* **User accounts:**
  + All user access accounts shall be maintained under the Okta group.
  + Each user should be added to okta group and the application database as a default configuration to access the application.
  1. **GIT Repository**
     1. **Front End**

The Companion application is maintained in GitLab.

Following GIT repositories are being used for Front End

**URL:https://gitlab.nimbus.amgen.com/gitlab-gmaap/companion-ai-group/companion-ai.git**

**Implementation branch: frontend**

**Dev branch: dev**

**Production branch: main**

* + 1. **Backend**

The Companion application is maintained in GitLab.

Following GIT repositories are being used for Back End

**URL:https://gitlab.nimbus.amgen.com/gitlab-gmaap/companion-ai-group/companion-ai.git**

**Branch: backend**

**Dev branch: dev**

**Production branch: main**

* + 1. **User Interface**
* [**https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=212-7124&t=304qaOGuhPx1zzcW-0**](https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=212-7124&t=304qaOGuhPx1zzcW-0)
* [**https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=3-5&t=jxw4BM5uZQ5elSP7-0**](https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=3-5&t=jxw4BM5uZQ5elSP7-0)
* [**https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=74-5628&t=OHVt7SYTUbPPDVRw-0**](https://www.figma.com/design/IGIdgoaAjBTmQld4FdVaXu/%F0%9F%96%A5%EF%B8%8F-MUI-for-Amgen%3A-Flows?node-id=74-5628&t=OHVt7SYTUbPPDVRw-0)
* [**https://url.usb.m.mimecastprotect.com/s/vcIpCp9WQ8snVgqkiP3se4?domain=figma.com**](https://url.usb.m.mimecastprotect.com/s/vcIpCp9WQ8snVgqkiP3se4?domain=figma.com)

Prod URL: https://companion.amgen.com

Dev URL: https://companion-dev.amgen.com

* 1. **Use Cases**
     1. **Use Cases -**
  2. **DEPLOYMENT**

**DEPLOYMENT IN AWS SERVER**

**Frontend port: 8001**

**Backend port: 8000**

**IP:**

**Dev - 10.23.3.143**

**Prod – 10.23.131.167**

**Steps:**

* 1. Open the EC2 instance in root.
  2. Navigate to - /var/snap/amazon-ssm-agent/7983/Genai
  3. For Dev server - Clone the repo from **dev** branch – *git pull origin dev*
  4. For Prod serv - Clone the repo from **main** branch – *git pull origin main*

**Deployment steps for Frontend:**

* Install the dependency packages –> *npm install –force*
* Build the react project -> *npm run build*
* Attach screen –> *screen –x frontend*
* Restart the application
  + *Ctrl+c*
  + *node server.js*

**Deployment steps for backend**

* Attach screen –> *screen –x backend*
* Activate the environment -> *source env/bin/activate*
* Install dependencies -> *pip install –r requirement.txt*
* Restart the application
  + *Ctrl+c*
  + *python manage.py runserver 0.0.0.0:8000*