DocAI System Sequence Diagram

Participants: User, Gateway Service, Master Service, Ingestion Service, Storage Service, Database Service, Retrieval Service, Vectorstore Service, Generation Service

"Document Ingestion"

1. Upload Raw Document (e.g., PDF) — User → Gateway Service
2. Forward Raw Document — Gateway Service → Master Service
3. Send Raw Document Data (via Message Queue) — Master Service → Ingestion Service
4. Store Document File — Ingestion Service → Storage Service
5. Acknowledgment
   - *Create Document Object with metadata: {id, file_name, pages, state:"created"}.*
6. Ack with Minimal Document Object (state:"created")
   - *Document {doc_id} now in state "created"*
   - *Documents wait in a queue to be processed*
7. Process Document (convert to images, generate metadata) — Ingestion Service (self)
8. Store Page Images — Ingestion Service → Storage Service
9. Acknowledgment
10. Persist Document Object — Ingestion Service → Database Service
    - *Documnet is stored with state "processed"*
11. Acknowledgment
12. Notify Document Processed — Ingestion Service → Master Service

"Document Indexing"

13. Initiate Indexing for Processed Document {doc_id} — Master Service → Retrieval Service
14. Request Document Object for {doc_id} — Retrieval Service → Database Service
15. Return Document Object
16. Request Page Images for Document {doc_id} — Retrieval Service → Storage Service
17. Return Page Images
    - *Pages wait in a queue to be indexed*
18. Generate & Store Embeddings for Each Page — Retrieval Service → Vectorstore Service
19. Confirm Embeddings Stored
20. Update Doucmnet State to "indexed" — Retrieval Service → Database Service
21. Acknowledgment
22. Notify Document Indexed — Retrieval Service → Master Service
23. Notify Document Indexed — Master Service → Gateway Service
24. Notify Document (doc_id) Indexed — Gateway Service → User
    - *User has to upload atleast one document in order to sumbit a query.*
    - *Document indexing is considered a bottlneck to the system.*

"Query Ingestion"

25. Submit Query Text — User → Gateway Service
26. Forward Raw Query — Gateway Service → Master Service
27. Send Raw Query Data (via Message Queue) — Master Service → Ingestion Service
    - *Create Query Object with metadata: {id, text, state:"created"}*
28. Ack with Query Object (state:"created")
29. Retrieve list of Documents created prior to Query {query_id} — Ingestion Service → Database Service
30. Return Document IDs list [doc_id1, doc_id2, ...]
    - *Associate Query {query_id} with these Document IDs*
31. Process Query Data — Ingestion Service (self)
32. Persist Query Object — Ingestion Service → Database Service
33. Acknowledgment
    - *Query is stored with state "processed"*
34. Notify Query Processed — Ingestion Service → Master Service

"Query Indexing"

35. Initiate Indexing for Processed Query {query_id} — Master Service → Retrieval Service
36. Request Query Object for {query_id} — Retrieval Service → Database Service
37. Return Query Object
    - *Queries wait in a queue to be indexed*
38. Generate & Store Embedding for Query — Retrieval Service → Vectorstore Service
39. Confirm Query Embedding Stored
40. Update Query Status to "indexed" — Retrieval Service → Database Service
41. Acknowledgment
42. Notify Query Indexed — Retrieval Service → Master Service

"Context Retrieval"

43. Check states for associated Documents [doc_id1, doc_id2, ...] — Retrieval Service → Database Service
44. Return Document States

alt [One or more Documents not indexed]
- *Context retrieval blocked due to incomplete document indexing*
45. Return "Documents processing. Please wait..." — Retrieval Service → Gateway Service

[All Documents are indexed]
46. Initiate Context Reitrieval for Indexed Query {query_id} — Master Service → Retrieval Service
47. Request Similarity Search for Query {query_id} — Retrieval Service → Vectorstore Service
48. Perform Similarity Search with indexed documents — Vectorstore Service (self)
    - *Returns list of relevant Document/Page IDs*
49. Return relevant IDs
50. Persist Query Context (retrieved IDs) — Retrieval Service → Database Service
51. Notify Query Contect Retrieved — Retrieval Service → Master Service
    - *Update Query {query_id} state to "context-retrieved"*

"Answer Generation"

52. Initiate Answer Generation for {query_id} — Master Service → Generation Service
53. Request Query Object — Generation Service → Database Service
54. Return Query
55. Request Page Images for context IDs — Generation Service → Storage Service
56. Return Page Images
57. Generate Answer using Provided Context — Generation Service (self)
58. Return Generated Answer — Generation Service → Master Service
    - *Update Query {query_id} state to "answered"*
59. Forward Final Answer to User — Master Service → Gateway Service
60. Deliver Generated Answer — Gateway Service → User