Software Design Description

for

Automatic Document Summarization and Q&A Using LLMs

Version 1.0

Prepared by

Group Name: SYDAI

Agnishwar Raychaudhuri	BT22GDS056	agnishwar.raychaudhuri22@st.niituniversity.in
Gautam Sharma	BT22GCS187	gautam.sharma22@st.niituniversity.in
Mannat Ashish Nayak	BT22GCS368	mannat.nayak22@st.niituniversity.in
Samarth Tanay	BT22GCS192	samarth.tanay22@st.niituniversity.in
Sarabjoth Bhatia	BT22GCS233	sarabjoth.bhatia22@st.niituniversity.in
Yuvraj Chawla	BT22GCS375	yuvraj.chawla22@st.niituniversity.in

Instructor:	***Manish Hurkat***
Course:	**CS 4191 – Capstone Project II**
Lab Section:	***D1 D3 D4 D6***
Date:	**21st September 2025**

Table of Contents

1. Introduction

Automatic Document Summarization and Q&A Using Large Language Models (LLMs) is a system designed to address the challenges of information overload and inefficient document processing. In today's digital environment, organizations and individuals are required to process vast volumes of text ranging from research articles, policy documents, corporate reports, technical manuals, to legal documents. Manual summarization and information retrieval are time-consuming, prone to human error, and lack consistency.

The proposed system leverages the latest advances in natural language processing (NLP) and LLMs to:

- Automatically generate coherent, concise, and context-aware summaries.
- Allow users to query documents in natural language and obtain accurate, source-backed answers.
- Enhance decision-making by enabling faster access to critical information while ensuring traceability and interpretability of outputs.

Unlike traditional extractive summarization approaches, this system combines retrieval-augmented generation (RAG), semantic embeddings, and fine-tuned LLMs to deliver high-quality abstractive summaries and interactive Q&A capabilities.

This document (SDD) describes the system architecture, design components, data flow, and implementation strategy that will guide developers, researchers, and stakeholders in building and deploying the system.

1.1 Purpose

The purpose of this SDD is to serve as a technical blueprint for the development and deployment of the Automatic Document Summarization and Q&A system.

Key objectives of this document are:

- To define the architecture of the system, including backend, frontend, storage, model layer, and deployment infrastructure.
  To describe data processing workflows, covering ingestion, preprocessing, summarization, Q&A pipeline, and explainability mechanisms.
- To detail the technology stack (Flask/Django, Docker, Kubernetes, CI/CD, LLM APIs or fine-tuned models).
- To outline security and authentication measures, including Google OAuth, secure data handling, role-based access, HTTPS, and JWT session management.

- To ensure all stakeholders (students, mentors, developers, and evaluators) share a common understanding of the project scope and implementation details.
- To act as a reference document for ongoing system development, testing, and maintenance.

1.2 Scope

The Automatic Document Summarization and Q&A System will be a web-based application with backend APIs and a user-friendly interface. The system's scope includes:

1.2.1 Functional Scope

1. Document Ingestion
    - Upload and parse multiple document formats (PDF, DOCX, TXT, HTML).
    - Handle unstructured content through OCR for scanned images.

2. Preprocessing & Storage
    - Text cleaning, tokenization, and chunking for efficient LLM processing.
    - Embedding generation using transformer-based models for semantic search.
    - Secure storage of documents, embeddings, and metadata.

3. Summarization
    - Extractive Summarization: Select key sentences/paragraphs.
    - Abstractive Summarization: Generate coherent human-like summaries using LLMs.
    - Multiple summary lengths: short (bullet points), medium (paragraph), and long (executive summary).

4. Question Answering (Q&A)
    - Retrieve relevant document chunks using embeddings and vector similarity.
    - Generate context-aware, factually consistent answers with citations.
    - Support follow-up questions for conversational Q&A.

5. Explainability & Compliance
    - Show references for summaries and answers (highlighted document segments).
    - Provide confidence scores for Q&A outputs.

    - Ensure transparency to align with academic, corporate, and regulatory standards.

6. User Management & Security
    - Authentication: Google OAuth for sign-in.
    - Authorization: Role-based access (e.g., admin, reviewer, user).
    - Secure session handling (JWTs, CSRF protection, HTTPS).
    - Data privacy controls for sensitive documents.

7. Deployment & Scalability
    - Dockerize application for portability.
    - Deploy to Kubernetes for scalability and fault tolerance.
    - Use CI/CD pipelines for automated testing, container builds, and cloud deployment.

8. Evaluation & Feedback Loop
    - Evaluate summaries using ROUGE, BLEU, and human ratings.
    - Evaluate Q&A with exact match (EM), F1, and human validation.
    - Provide feedback mechanisms for continuous improvement of LLM outputs.

1.2.2 Non-Functional Scope

- Scalability: System should handle large documents (100+ pages).
- Performance: Summaries generated in < 10 seconds for medium documents.
- Security: End-to-end encryption for data in transit and at rest.
- Usability: Simple UI for non-technical users.
- Maintainability: Modular microservice-based architecture.
- Portability: Cloud-native deployment on Kubernetes.

1.2.3 Out of Scope

- Financial scoring, legal compliance checks, or plagiarism detection.
- Offline desktop/mobile apps (initial phase is web-only).
  Training large LLMs from scratch (system will leverage APIs or fine-tuning).

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition

LLM	Large Language Model - An advanced AI model trained on vast amounts of text to understand, generate, and summarize human-like language.
RAG	Retrieval-Augmented Generation - A technique that combines information retrieval (fetching relevant text from documents) with LLM-based generation to produce accurate and context-aware responses.
API	Application Programming Interface - A set of functions and protocols that allow the frontend and backend to communicate.
OAuth	Open Authorization – A secure protocol that allows users to log in using third-party providers (e.g., Google OAuth) without sharing passwords.
UI	User Interface
SDD	Software Design Description
CI/CD	Continuous Integration and Continuous Deployment

1.4 References

1. https://futureagi.com/blogs/revolutionizing-document-management-llm-2025#:~:text=The%20process%20of%20document%20summarization,and%20structure%20of%20the%20document.
2. https://medium.com/@tahirbalarabe2/summarizing-private-documents-with-llms-langchain-and-rag-3582a67212f2
3. https://python.langchain.com/docs/tutorials/summarization/
4. https://docs.paperless-ngx.com

2. System Overview

2.1 System Description

The Automatic Document Summarization and Q&A System is a web-based application designed to process large volumes of documents and provide concise summaries and interactive question-answering capabilities using Large Language Models (LLMs). The system aims to streamline information retrieval for students, researchers, corporate teams, and individuals, reducing manual reading effort and improving decision-making.

System Architecture & Technology Stack

1. Backend Framework (Flask/Django)

- Flask or Django serves as the main API layer, handling user requests, document ingestion, preprocessing, embedding retrieval, and LLM-based inference.
- Provides RESTful endpoints for frontend communication and integration with the LLM pipeline.

2. Frontend (React)

- React.js with Vite for fast development and responsive UI.
- TailwindCSS + shadcn/ui for clean, interactive components.
- Framer Motion for smooth animations.
- Provides a user-friendly interface to upload documents, view summaries, and ask questions.

3. Document Processing & ML Pipeline

- Text Extraction: PyMuPDF or pdfminer.six for PDFs, OCR (Tesseract/EasyOCR) for scanned documents.
- Image & Text Preprocessing: OpenCV and Pillow for scanned images; NLTK/regex for text cleaning and tokenization.
- Embeddings: SentenceTransformers or Ollama embedding models convert text into vector representations for semantic retrieval.
- Vector Database: Qdrant (Dockerized) stores embeddings for fast retrieval during Q&A.
- LLM Integration: Ollama, OpenAI, Groq, or Anthropic models for summarization and Q&A, orchestrated via LangChain.

4. Data Storage

- MongoDB: Stores user data, document metadata, chat history, and session information.

5. Security & Authentication

- Google OAuth for secure login and user authentication.
- HTTPS via Nginx with SSL (Let's Encrypt).
- JWT tokens with refresh mechanisms for session security.
- File Upload Security: MIME-type validation, size limits, and scanning for malicious content.

6. Containerization & Deployment

- Docker for containerizing backend, frontend, embedding DB, Redis, and LLM inference.
- Kubernetes for orchestration, scaling, load balancing, and high availability.
- CI/CD Pipelines: GitHub Actions or Jenkins for automated testing, Docker builds, and deployment.

2.2 System Architecture

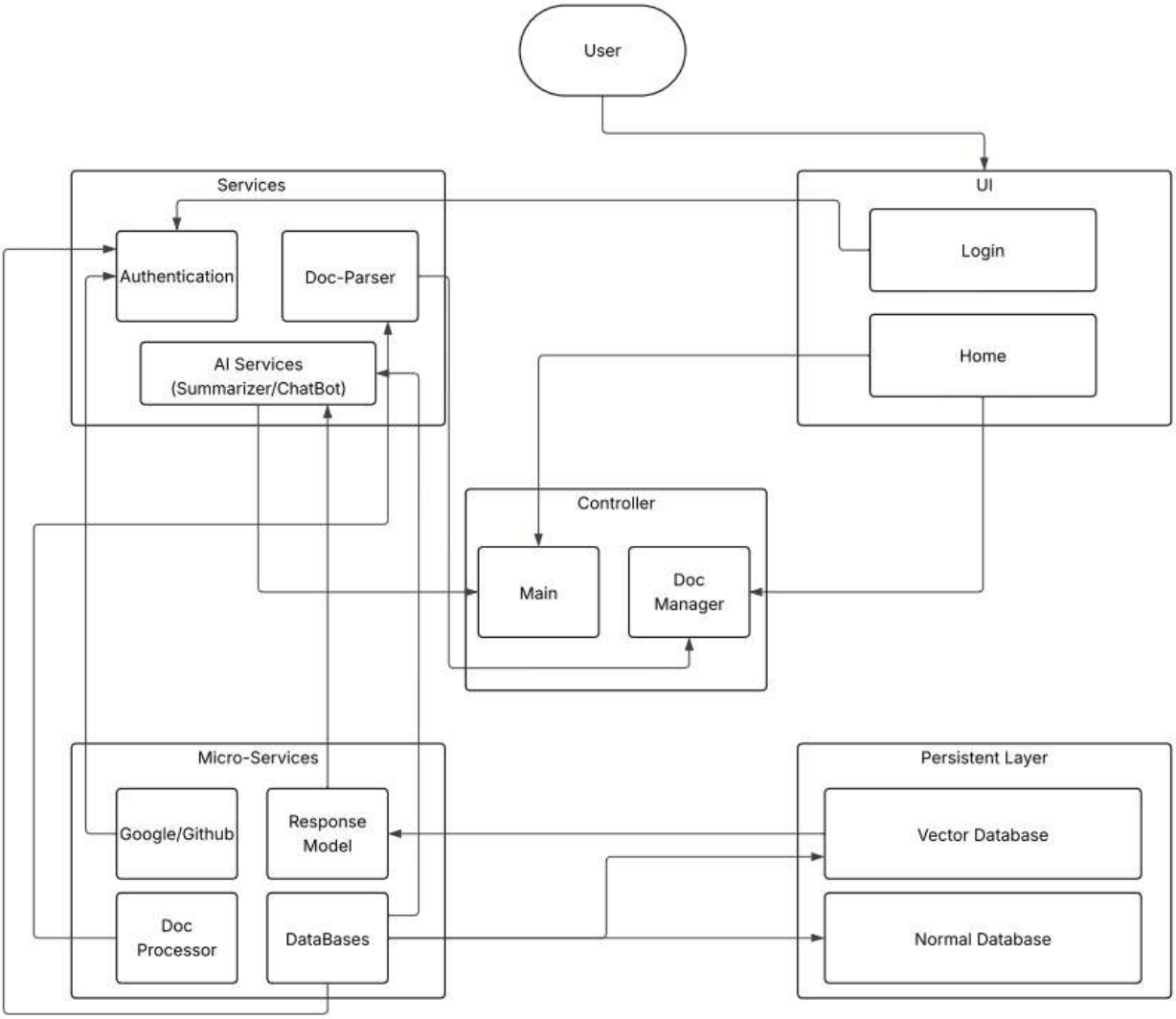


Figure 2 : System Architecture

This is the high-level system architecture of your Document Summarizer and Q&A Application. The system has been designed in a modular way, with each layer focusing on a specific functionality to ensure scalability, security, and smooth user interaction.

At the entry point of the system is the User, who interacts through the UI Layer. The UI consists of two primary components: the Login page, which manages authentication and access control, and the Home page, where the user can upload documents, view summaries, and interact with the Q&A chatbot. This layer ensures that the user experience remains simple and intuitive.

Once a user action is initiated, it is passed into the Services Layer. Here, the Authentication service validates user credentials, while the Doc-Parser handles

document ingestion, converting different file formats such as PDF, DOCX, or TXT into structured formats suitable for analysis. The AI Services (Summarizer/ChatBot) module forms the intelligence of the system. It leverages NLP models to generate concise summaries of the input documents and provides conversational responses to user queries in real time.

To coordinate these services, the Controller Layer is in place, which consists of the Main controller and the Doc Manager. The Main controller routes requests between the UI and the services or microservices, while the Doc Manager specifically deals with document operations such as parsing, indexing, summarization, and Q&A handling. This ensures that each component focuses only on its responsibility, making the architecture clean and modular.

Supporting the services are the Micro-Services Layer, which handles additional but critical functionalities. The Google/GitHub module allows integrations such as OAuth-based authentication or document imports from cloud repositories. The Doc Processor prepares the documents by cleaning, tokenizing, and generating embeddings for AI models. The Response Model generates intelligent answers to queries, and the Databases microservice provides structured interactions with storage.

Finally, all processed and raw data is stored in the Persistent Layer. This includes both a Normal Database for storing user details, authentication records, and document metadata, and a Vector Database for storing embeddings, which power efficient similarity search and retrieval in the Q&A module. This layered persistence ensures that both structured and AI-driven data can be managed effectively.

Altogether, this architecture creates a seamless pipeline where the user uploads a document, the system processes and summarizes it, and intelligent Q&A responses are generated—ensuring both usability and technical robustness.

2.3 Users

The Automatic Document Summarization and Q&A System is designed to serve a variety of stakeholders who interact with documents regularly and require efficient access to information. The system ensures secure access, personalized interactions, and role-based functionalities to meet diverse user needs.

The key users of the application will be:

- Students & Researchers
    - Upload study materials, research papers, or lengthy reports.

    - Use automatic summarization to quickly grasp key points.
      Ask targeted questions for deeper understanding of academic content.
- Corporate Professionals & Teams
    - Summarize business reports, contracts, and meeting transcripts.
    - Use Q&A for fast decision-making without reading full documents.
    - Collaborate securely with version control and document history.
- Legal & Compliance Officers
    - Extract insights from lengthy contracts, compliance documents, and legal case files.
    - Use summarization for efficient review.
    - Ask precise legal/compliance-related queries from uploaded files.
- General Users (Knowledge Seekers)
    - Summarize news articles, e-books, or any uploaded text.
    - Ask natural language questions for easy information retrieval..

3. Data Flow and Processing

The Automatic Document Summarization and Q&A System follows a structured data pipeline, ensuring efficient document ingestion, preprocessing, transformation, and analysis. The data flow is designed to handle both structured and unstructured documents, providing accurate, context-aware summaries and answers in real time.

3.1 Data Sources

The system integrates multiple sources of input documents to build a robust knowledge base:

- User-Uploaded Documents – PDF, DOCX, TXT files uploaded directly through the user interface.
- Scanned Documents – Image-based PDFs or scanned files requiring OCR for text extraction.
- External Knowledge Sources (Optional) – APIs or external corpora that can enrich context for summarization and Q&A.

3.2 Document Preprocessing

Once documents are uploaded, they undergo a preprocessing pipeline to ensure clean, consistent, and reliable text for downstream tasks.

The steps involved are:

- Text Extraction:
    - PyMuPDF/pdfminer.six for PDFs.
    - Tesseract/EasyOCR for scanned images.
- Text Cleaning:
    - Removal of stopwords, special symbols, and formatting tags.
    - Sentence segmentation and tokenization using NLTK/spaCy.
- Chunking:
    - Long documents are split into smaller, semantically meaningful text chunks (e.g., 500–1000 tokens) for embedding and retrieval efficiency.

Preprocessed text is then stored in MongoDB (metadata, document references) and Qdrant (vectorized embeddings).

3.3 Embeddings & Feature Representation

Instead of traditional numerical features, the system uses semantic vector embeddings for document representation.

- Embedding Generator: SentenceTransformers (HuggingFace) or Ollama embedding models (nomic-embed-text).
- Vector Database: Qdrant (Dockerized, scalable) for similarity search.
- Feature Representation: Each text chunk is represented as a high-dimensional embedding vector, enabling semantic similarity search and context retrieval for Q&A.

3.4 LLM Pipeline

The LLM pipeline powers summarization and Q&A by combining retrieval-augmented generation (RAG) with modern large language models.

- Summarization Workflow:
    1. Extract and clean text from document.
    2. Break into smaller chunks.
    3. Pass through embeddings + LLM to generate an abstractive summary.
- Q&A Workflow:
    1. The user asks a natural language query.
    2. Relevant chunks retrieved from Qdrant using embeddings.
    3. Retrieved text provided to LLM through LangChain as context.
    4. LLM generates an accurate, context-aware response with references.

LLMs Used:

- Primary: Ollama (local inference with models like LLaMA 3, Mistral, Phi-3, Gemma).
- Cloud Models: OpenAI GPT, Anthropic Claude, or Groq API as fallback.

3.5 Model Deployment

- The LLM pipeline (embeddings + retrieval + summarization/Q&A) is exposed as RESTful APIs using Flask or Django.
- The React.js frontend communicates with these APIs to display summaries and answers in real time.
- All services are containerized using Docker and deployed via Kubernetes for scalability.

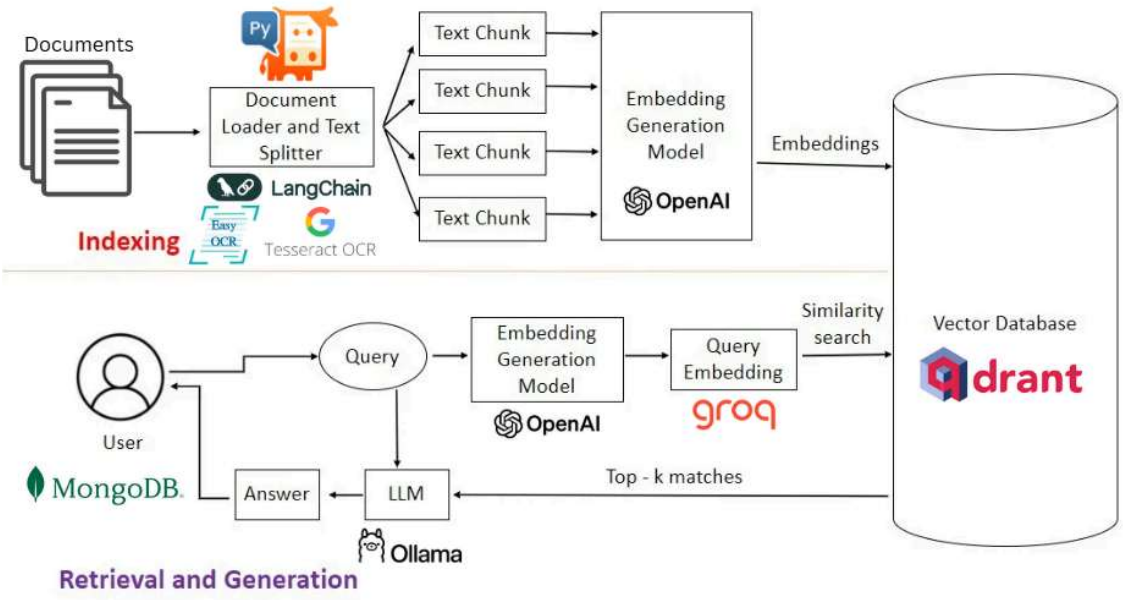


Figure 2 : RAG Architecture

The above diagram represents the RAG Architecture of the Automatic Document Summarization and Q&A System, which is divided into two major stages: Indexing and Retrieval & Generation. Together, these stages ensure efficient document processing, semantic search, and accurate, context-driven answers.

In the Indexing phase, user-uploaded documents such as PDFs, Word files, or text documents are first passed through a Document Loader and Text Splitter. Since large documents can be too extensive for direct processing, they are divided into smaller, semantically meaningful text chunks. Each text chunk is then transformed into a high-dimensional vector using an Embedding Generation Model (such as

SentenceTransformers or Ollama embeddings). These embeddings represent the semantic meaning of the text, making it easier to perform similarity-based retrieval. The generated embeddings, along with metadata, are stored in a Vector Database (Qdrant), which acts as the knowledge store for the system.

In the Retrieval and Generation phase, the user submits a natural language query through the system's interface. The query itself is converted into a semantic query embedding using the same embedding model for consistency. The system then performs a similarity search in the vector database to find the top-k most relevant document chunks. These retrieved text chunks are passed, along with the user query, to the Large Language Model (LLM). The LLM uses this contextual information to generate a precise and coherent answer or summary. Finally, the response is returned to the user, ensuring that answers are both contextually accurate and grounded in the source documents.

Beyond query answering, the system supports Automatic Document Summarization, which allows users to quickly understand the essence of lengthy or complex documents. During summarization, the LLM processes the most important retrieved chunks and condenses them into concise overviews at either a section level or a full-document level. This helps reduce information overload and enables decision-making based on high-level insights without reading the entire text.

Another critical functionality is Q&A with Link Referencing, which improves reliability and transparency. When the user asks a question, the system not only generates a contextual answer but also attaches references or links to the specific sections of the original documents from which the information was derived. This ensures that every answer is verifiable, allowing users to trace responses back to their source for confirmation and deeper exploration.

This modular architecture is scalable and efficient: the Indexing phase is performed once for each uploaded document, while the Retrieval and Generation phase is executed dynamically at runtime. By leveraging embeddings and a vector database, the system moves beyond simple keyword search to true semantic retrieval, delivering context-aware, human-like responses with summarization and verifiable references.

4. Implementation Details

4.1 Technology Stack

The Automatic Document Summarization and Q&A System will be built using a modular, service-oriented architecture. The system integrates frontend, backend, document processing, embeddings pipeline, LLM layer, databases, and DevOps infrastructure to ensure scalability, security, and maintainability.

1. Frontend (User Interface)

- Framework: React (bundled with Vite for faster development builds).
- Styling: TailwindCSS with shadcn/ui component library for clean, responsive design.
- Animations: Framer Motion for smooth UI transitions and polish.
- Charts & Visuals: Recharts for dashboards (document statistics, usage analytics).
- State Management: Redux Toolkit or React Query to manage global state and API communication.
- Authentication: Google OAuth integration (via API) + JWT tokens stored in HttpOnly cookies for security.

2. Backend (API Layer)

- Framework: Flask with Flask-RESTful for building REST APIs.
- Authentication: Flask-JWT-Extended for token-based login and refresh tokens.
- ORM / Database: SQLAlchemy for relational DB operations.
- File Handling: Flask-Uploads for document ingestion, Flask-CORS for cross-origin support.
- Validation: Marshmallow for schema validation of request/response payloads.

3. Document Processing (OCR & Text Extraction)

- PDF Parsing: PyMuPDF (fitz) or pdfminer.six for extracting structured text from documents.
- OCR (Scanned Docs): Tesseract + pytesseract OR EasyOCR for image-to-text extraction.
- Image Preprocessing: OpenCV and Pillow for cleaning scanned documents (deskewing, noise removal).
- Text Cleaning: NLTK + regex for tokenization, stopword removal, and normalization.

4. Embeddings + Vector Database (RAG Pipeline)

- Embeddings Generator: SentenceTransformers (HuggingFace models) OR Ollama embedding models
- Vector Database: Qdrant (Dockerized, scalable, with Python SDK integration).
- Integration: qdrant-client for storing and retrieving embeddings during Q&A.

5. LLM Layer (Response Generation)

- Primary Models: Ollama for local inference (models like Mistral, Llama 3, Phi-3, Gemma).
- Optional Cloud Models: Groq API, OpenAI GPT, or Anthropic Claude (as fallback).
- Framework Integration: LangChain to orchestrate document retrieval, prompt construction, and response generation.

- Capabilities:
    - Abstractive summarization (short, medium, long versions).
    - Context-aware question answering with document citations.
    - Conversational follow-up questions (chat history).

6. Databases

- Relational DB: PostgreSQL (via SQLAlchemy) to store user accounts, permissions, and app metadata.
- NoSQL DB: MongoDB for storing chat history, document metadata, and flexible JSON-like objects.

7. Infrastructure & Deployment

- Containerization: Docker + Docker Compose (packaging services: Flask backend, Qdrant, Ollama, Redis, Postgres).
- Orchestration: Kubernetes for horizontal scaling, high availability, and automated rollouts.
- Reverse Proxy & Load Balancer: Nginx for routing, HTTPS enforcement, and load balancing.
- Environment Management: Python venv / Poetry for backend dependency isolation.
- Secrets Management: dotenv for local; HashiCorp Vault or AWS Secrets Manager for production.
- Testing: pytest for unit and integration testing across modules.

8. Security & Authentication

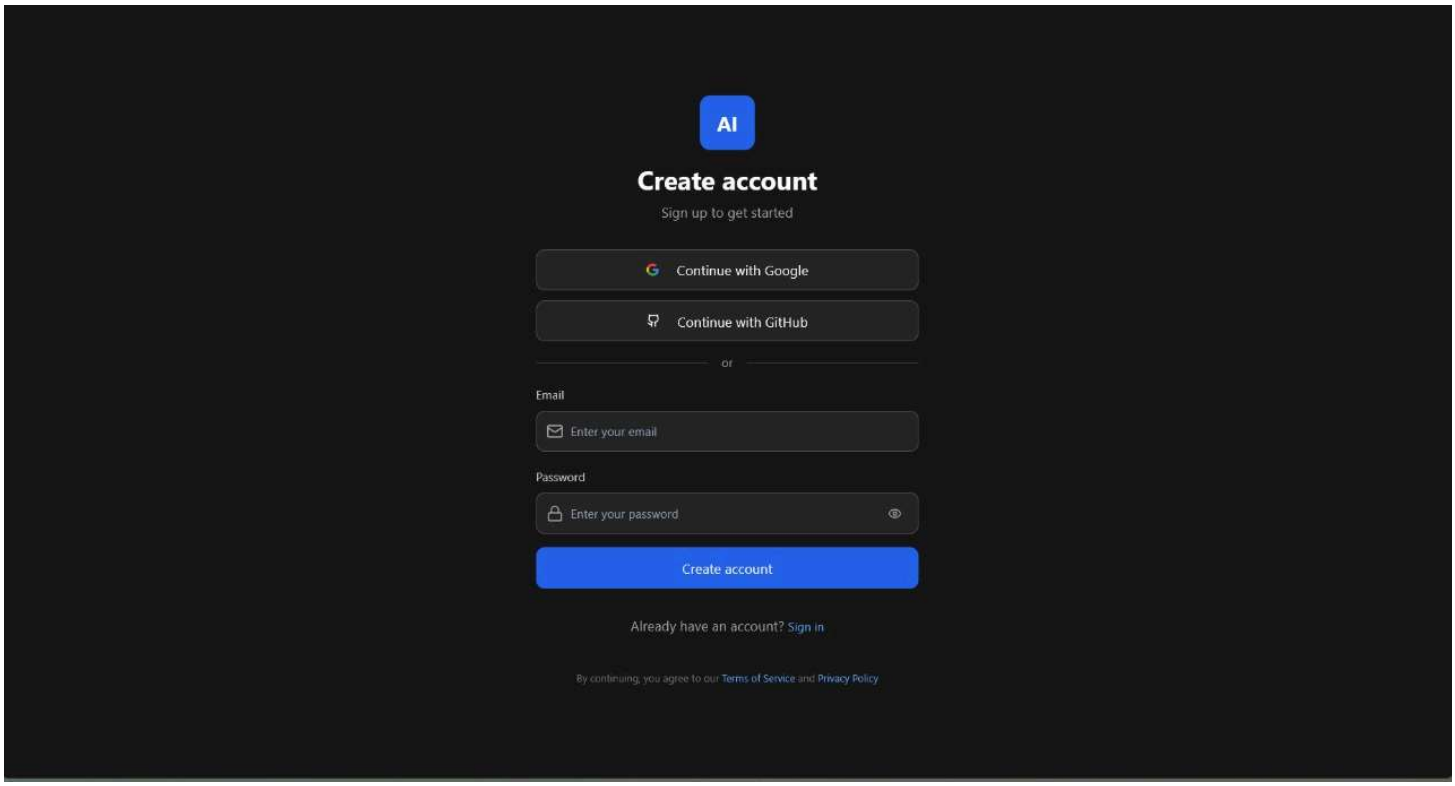
- Transport Security: HTTPS via Nginx with Let's Encrypt SSL.
  Authentication: Google OAuth 2.0 for login; JWT tokens (with refresh tokens) for sessions.
- Password Security: bcrypt for hashing (if local accounts are allowed).
- File Upload Security: MIME-type validation, file size limits, sandboxing OCR processes.
- CORS Handling: Flask-CORS for safe API access across domains.

9. DevOps & Monitoring

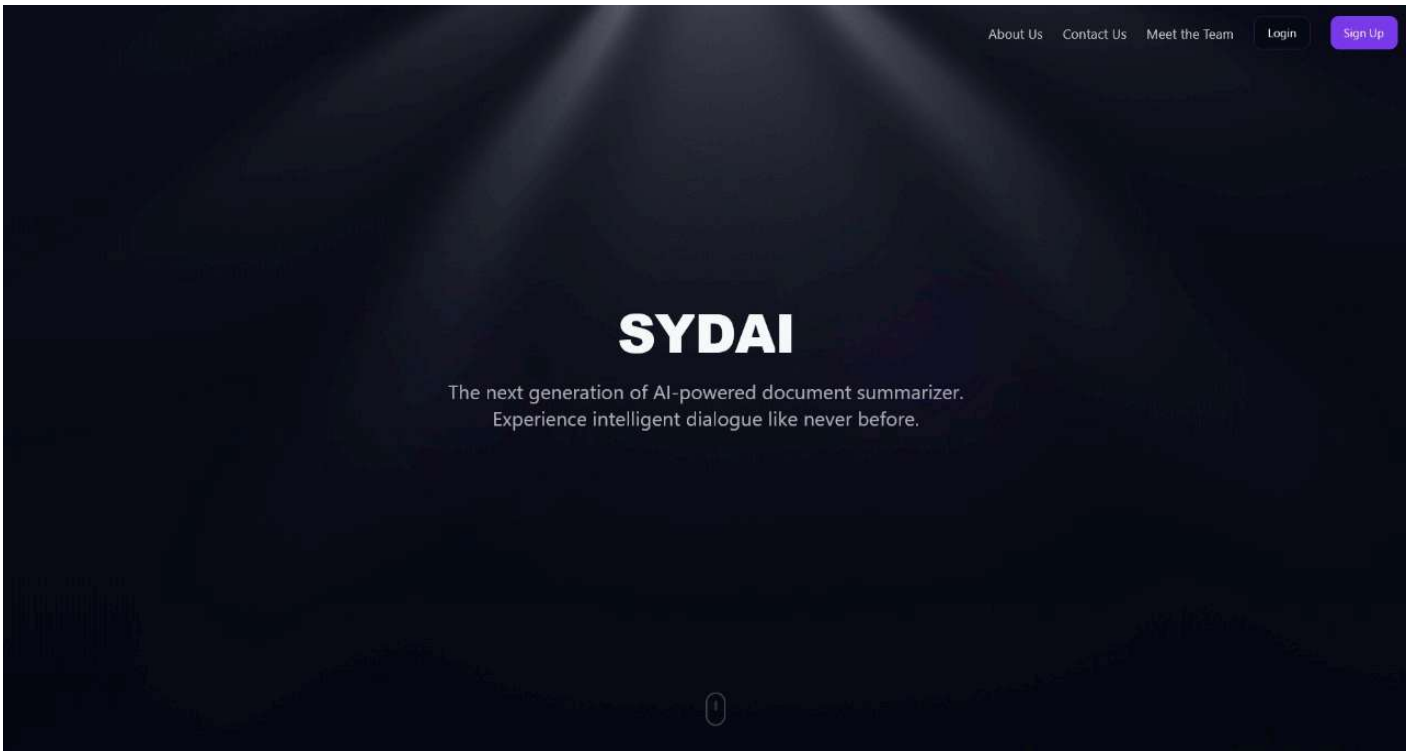
- Version Control: Git + GitHub for collaborative development.
- CI/CD Pipelines: GitHub Actions for automated testing, Docker builds, and deployment.
- Deployment Environments:
  - Dev: Local Docker Compose + Minikube for testing.
  - Prod: Cloud-based Kubernetes cluster.

4.2 GUI (Graphical User Interface)

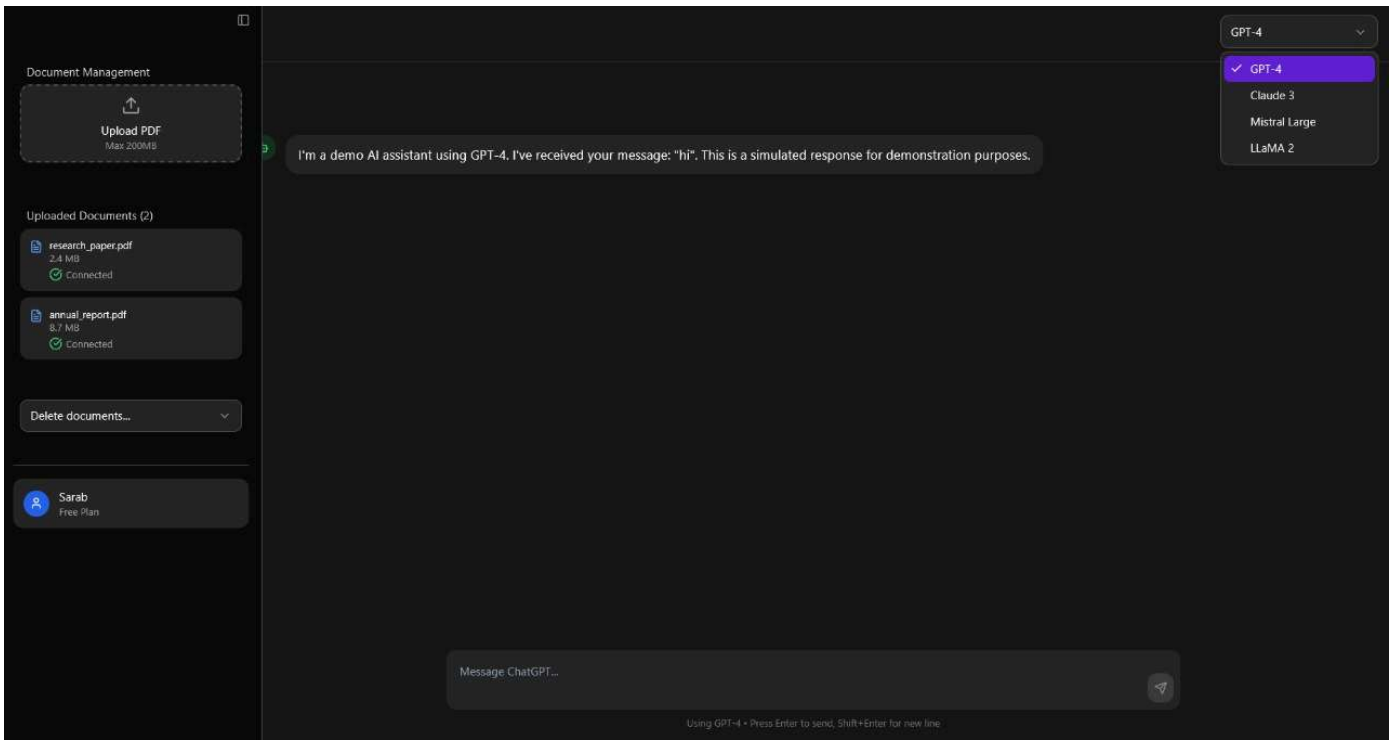
4.2.1 Sign In and Sign Up Page

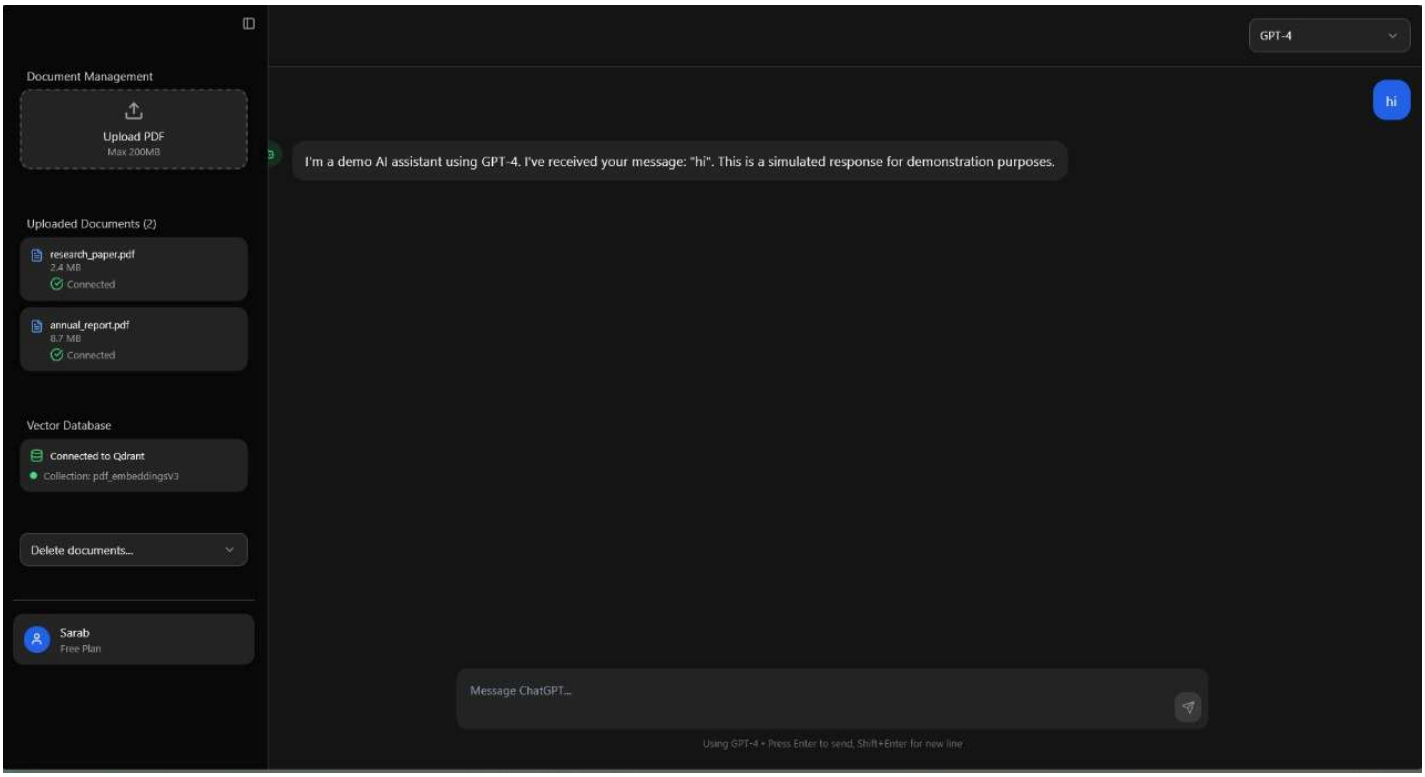


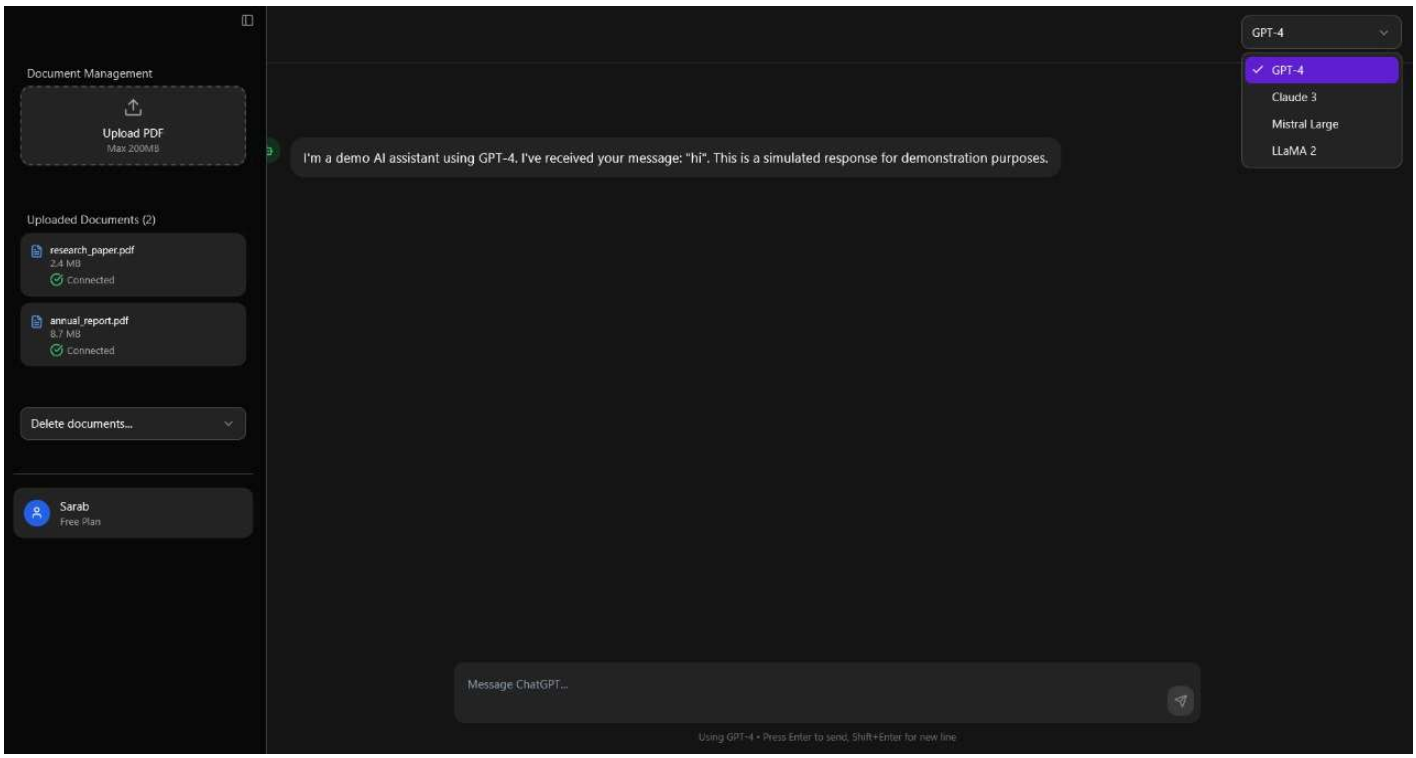
4.2.2 Home Page



4.2.3. Summarizer and Q & A Chat Bot Page




Document Management
Upload PDF
Max 200MB
Uploaded Documents (2)
research_paper.pdf
2.4 MB
Connected
annual_report.pdf
8.7 MB
Connected
Vector Database
Connected to Qdrant
Collection: pdf_embeddingsv3
Delete documents...
Sarab
Free Plan
GPT-4
hi
I'm a demo AI assistant using GPT-4. I've received your message: "hi". This is a simulated response for demonstration purposes.
Message ChatGPT...
Using GPT-4 • Press Enter to send, Shift+Enter for new line


Document Management
Upload PDF
Max 200MB
Uploaded Documents (2)
research_paper.pdf
2.4 MB
Connected
annual_report.pdf
8.7 MB
Connected
Delete documents...
Sarab
Free Plan
GPT-4
GPT-4
Claude 3
Mistral Large
LLaMA 2
I'm a demo AI assistant using GPT-4. I've received your message: "hi". This is a simulated response for demonstration purposes.
Message ChatGPT...
Using GPT-4 • Press Enter to send, Shift+Enter for new line

5. Security Aspects

5.1 Security Measures

Ensuring data security and compliance is essential in the Automatic Document Summarization and Q&A System, since it processes sensitive documents such as research papers, corporate reports, and potentially confidential user data. The system is designed with a security-first approach to protect user-uploaded content, embeddings, and generated outputs.

Authentication will be managed through OAuth-based login (Google Authentication), providing a secure and widely trusted identity verification mechanism. By using a third-party provider, the system avoids storing raw passwords and reduces the risk of unauthorized access. In addition, role-based access controls (RBAC) can be implemented to separate permissions between end users, administrators, and system maintainers.

For data protection, all uploaded documents will be encrypted at rest using AES-256 encryption within the storage system and secured in transit via HTTPS/TLS. Embeddings and metadata stored in the vector database (Qdrant) will also be encrypted to prevent unauthorized access to semantic representations of sensitive documents. Token-based authentication (JWT in HttpOnly cookies) will further protect API interactions, minimizing the risk of session hijacking.

The system also employs file validation checks during upload to prevent malicious files from being ingested. This includes MIME-type verification, file size limits, and secure sandboxing for OCR operations. Logs of all access requests and user interactions will be maintained to enable auditing and detect anomalies.

Finally, security testing is integrated into the CI/CD pipeline with automated vulnerability scans, dependency checks, and container-level security policies. These measures ensure that the summarization and Q&A platform remains secure, reliable, and compliant with modern data protection standards.

6. CI/CD & Kubernetes Strategy

We plan to implement Continuous Integration (CI) and Continuous Deployment (CD) along with a containerized Kubernetes deployment to ensure the Automatic Document

Summarization and Q&A System is reliable, scalable, and maintainable. This setup will allow automated testing, building, and deployment of code changes, reducing downtime and minimizing errors in production.

6.1 Continuous Integration & Continuous Deployment (CI/CD)

- Version Control: Git repositories hosted on GitHub with structured branching (feature branches, development branch, main branch).
- Automated Testing: CI pipelines will run unit tests, integration tests, and security scans whenever new code is pushed.
- Continuous Deployment: CD pipelines using GitHub Actions or Jenkins will automate Docker image builds, testing, and deployment to the Kubernetes cluster.

6.2 Kubernetes & Container Strategy

- Backend Framework: Flask or Django for serving APIs, handling document processing, embeddings, and LLM inference.
- Containerization: All services (backend API, LLM inference, embeddings DB, Redis, MongoDB, frontend) will be containerized using Docker.
- Orchestration: Kubernetes will manage containers for scalability, high availability, and load balancing.
- Deployment Strategy:
    - Separate namespaces for development, testing, and production.
    - Horizontal Pod Autoscaling to handle variable workloads.
    - Persistent volumes for MongoDB and document storage.

7. Conclusion

The Automatic Document Summarization and Q&A System leverages Flask/Django, LLMs, embeddings, and RAG pipelines to deliver fast, accurate, and context-aware document summaries and answers. With secure authentication via Google OAuth, a user-friendly interface, and scalable deployment using Docker and Kubernetes, the system ensures performance, reliability, and maintainability.

Automated CI/CD pipelines streamline testing, building, and deployment, reducing downtime and human errors. Designed with explainability and traceability, the system provides provenance for all outputs. Future improvements may include advanced LLM fine-tuning, optimized document preprocessing, and enhanced cloud integration to further improve performance, security, and user experience.

8. Timeline

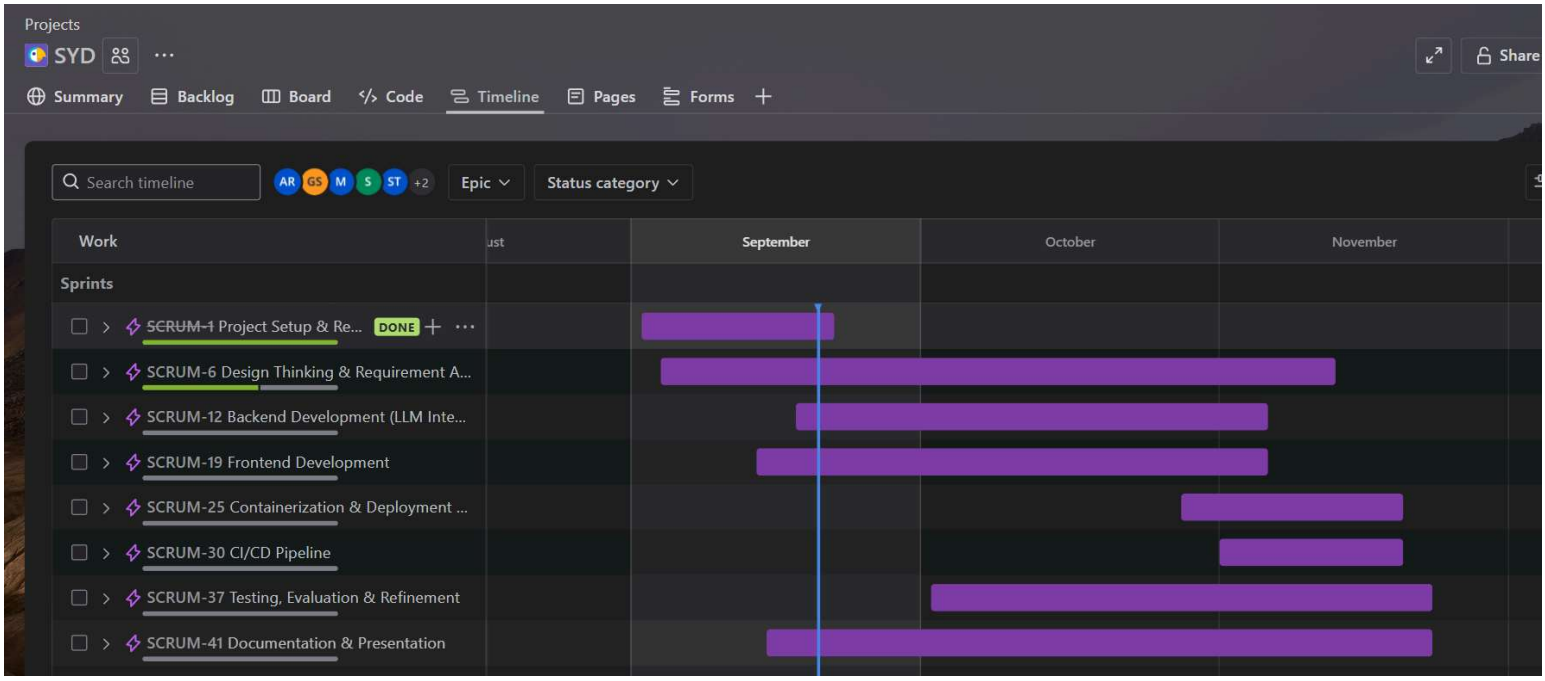


Figure 2: Project Timeline (JIRA Dashboard)

https://documentsummarizer.atlassian.net/jira/software/projects/SCRUM/boards/1/timeline?atlOrigin=eyJpIjoiYjVkZWQwNjIwMTQ3NGJiNmFhZWQzMzhlZGQ5NWE3YjIiLCJwIjoiaiJ9