Project Design Phase-II
Technology Stack (Architecture & Stack)

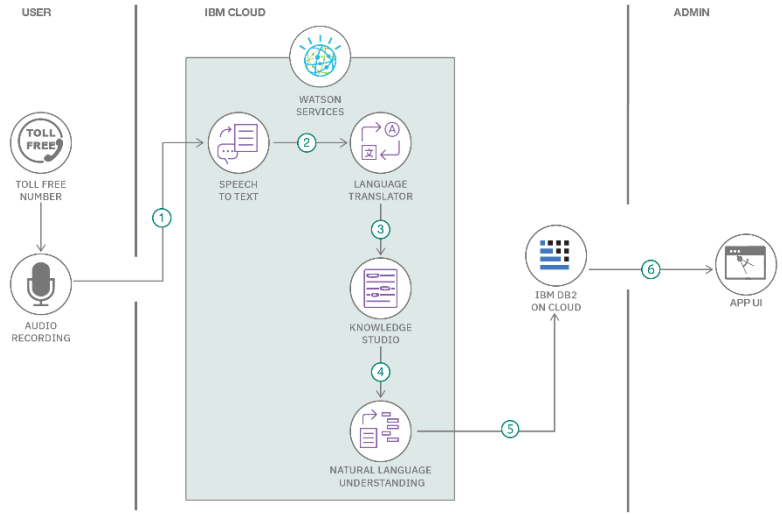
Date	31 January 3035
Team ID	
Project Name	DocuQuery: AI-Powered PDF Knowledge Assistant Using Google PALM
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Example: Order processing during pandemics for offline mode

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/



Guidelines:

1. **Frontend**:
   - **React.js**: A powerful library for building dynamic user interfaces.
   - **Bootstrap/Tailwind CSS**: Pre-designed components for a polished UI.
   - **react-dropzone**: Simplifies drag-and-drop file uploads.
2. **Backend**:
   - **Flask/FastAPI**: Lightweight frameworks for building RESTful APIs.
   - **Hugging Face Transformers**: Pre-trained models for question-answering.
   - **SpaCy/NLTK**: Tools for document preprocessing.

S.No	Component	Description	Technology
1	Frontend	User interface for document upload, query input, and answer display.	React.js, Bootstrap/Tailwind CSS
2	File Upload	Drag-and-drop functionality for uploading documents.	react-dropzone
3	State Management	Manage global state (e.g., document text, query, answer).	Redux or Context API
4	Routing	Navigation between pages (e.g., upload page, query page).	React Router
5	Backend	RESTful API for handling document uploads and queries.	Flask or FastAPI
6	AI Model	Pre-trained models for question-answering.	Hugging Face Transformers
7	Document Preprocessing	Clean and preprocess uploaded documents (e.g., tokenization, lemmatization).	SpaCy or NLTK
8	Database	Store metadata about documents and queries.	SQLite (local) or PostgreSQL (production)
9	Vector Database	Store and query document embeddings for semantic search.	Pinecone or Weaviate
10	File Storage	Store uploaded documents securely.	AWS S3 or Google Cloud Storage
11	Authentication	User authentication and authorization.	Flask-Login or JWT
12	Deployment (Backend)	Host the backend API.	Heroku, AWS Elastic Beanstalk, or Render
13	Deployment (Frontend)	Host the frontend application.	Netlify, Vercel, or GitHub Pages
14	Containerization	Package the application for consistent deployment.	Docker
15	Orchestration	Manage containers in production.	Kubernetes
16	CI/CD	Automate testing and deployment.	GitHub Actions or GitLab CI/CD

S.No	Component	Description	Technology
17	Monitoring	Track system performance and set up alerts.	Prometheus and Grafana
18	Logging	Centralized logging for debugging and analysis.	ELK Stack or AWS CloudWatch
19	Version Control	Manage code versions and collaboration.	Git and GitHub/GitLab
20	API Testing	Test APIs during development.	Postman or Insomnia
21	Code Formatting	Automatically format code for consistency.	Black (Python) and Prettier (JavaScript)
22	Linting	Catch errors and enforce style guidelines.	Flake8 (Python) and ESLint (JavaScript)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology/Approach
1	User-Friendly Interface	Intuitive and responsive UI for document upload, query input, and answer display.	React.js, Bootstrap/Tailwind CSS
2	Drag-and-Drop File Upload	Allow users to upload documents by dragging and dropping files.	react-dropzone
3	Real-Time Query Processing	Process user queries in real-time and provide instant answers.	Flask/FastAPI, Hugging Face Transformers
4	Document Preprocessing	Clean and preprocess uploaded documents (e.g., tokenization, lemmatization).	SpaCy or NLTK
5	AI-Powered Question Answering	Use pre-trained models to answer questions based on the uploaded document.	Hugging Face Transformers

S.No	Characteristics	Description	Technology/Approach
6	Scalability	Handle large documents and multiple users efficiently.	Docker, Kubernetes, Pinecone/Weaviate
7	Semantic Search	Enable semantic search for better query results.	Sentence Transformers, Pinecone/Weaviate
8	Secure File Storage	Store uploaded documents securely.	AWS S3 or Google Cloud Storage
9	Metadata Management	Store and manage metadata about documents and queries.	PostgreSQL or MongoDB
10	Cross-Platform Compatibility	Ensure the application works seamlessly on different devices and browsers.	Responsive Design (Bootstrap/Tailwind CSS)
11	User Authentication	Secure user access with authentication and authorization.	Flask-Login or JWT
12	API Integration	Connect the frontend with the backend via RESTful APIs.	Flask/FastAPI, Axios (Frontend)
13	Error Handling	Provide meaningful error messages for invalid inputs or system failures.	Flask/FastAPI Error Handling, React Alerts
14	Performance Optimization	Optimize the application for fast query processing and response times.	FastAPI, React.js, Caching Mechanisms
15	Monitoring and Logging	Track system performance and log errors for debugging.	Prometheus, Grafana, ELK Stack
16	Automated Testing	Ensure code quality and functionality through automated tests.	Pytest (Backend), Jest (Frontend)
17	CI/CD Pipeline	Automate testing, building, and deployment processes.	GitHub Actions, GitLab CI/CD
18	Document Embeddings	Generate and store embeddings for efficient document retrieval.	Sentence Transformers, Pinecone/Weaviate
19	Multi-Language Support	Support documents and queries in multiple languages.	Hugging Face Multilingual Models
20	Accessibility	Ensure the application is accessible to users with disabilities.	ARIA Tags, Semantic HTML

References:

https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d