

# Document-Based Question Answering System - Overview and Demo Guide

---

## Team 70 - Contributors

Name	Email Address	Contributions %
NEERAJ BHATT	2024aa05020@wilp.bits-pilani.ac.in	100%
V. S. BALAKRISHNAN	2024aa05017@wilp.bits-pilani.ac.in	100%
KURUVELLA VENKATA SAI UPENDRA	2024aa05016@wilp.bits-pilani.ac.in	100%
SAJAL CHAUDHARY	2024aa05026@wilp.bits-pilani.ac.in	100%
SACHIN KUMAR	2024aa05024@wilp.bits-pilani.ac.in	100%

---

## Overview

A web-based Question Answering application that allows users to upload documents (PDF, DOCX, TXT) and ask questions based on the document content. The application provides accurate answers extracted from the uploaded documents with confidence scores and source references.

## Features

- **Document Upload:** Support for PDF, DOCX, and TXT formats
- **Multiple Query Methods:**
  - Ask questions on uploaded documents
  - Ask questions on directly pasted text
- **AI-Powered QA:** Uses pre-trained transformer models for accurate answer extraction
- **Source References:** Every answer includes the source passage and document reference
- **Confidence Scores:** Visual indicators showing the confidence level of each answer
- **Responsive UI:** Modern, user-friendly web interface
- **Real-time Processing:** Fast document processing and query response

## Technology Stack

### Backend

- **Framework:** FastAPI
- **NLP:** Hugging Face Transformers (RoBERTa-SQuAD2)
- **Document Parsing:** PyPDF2, python-docx
- **Text Processing:** NLTK, scikit-learn

### Frontend

- **HTML5, CSS3, JavaScript (Vanilla)**
- **Bootstrap 5** for responsive design
- **RESTful API** integration

## Project Structure

```
doc-qa-system/
├── backend/
│   ├── app/
│   │   ├── models/
│   │   │   └── schemas.py          # Pydantic models
│   │   ├── routes/
│   │   │   ├── documents.py       # Document upload endpoints
│   │   │   └── qa.py              # QA query endpoints
│   │   ├── services/
│   │   │   ├── document_indexer.py # Document storage & indexing
│   │   │   └── qa_engine.py       # QA processing engine
│   │   ├── utils/
│   │   │   └── document_processor.py # Document parsing
│   │   └── main.py                 # FastAPI app initialization
│   ├── requirements.txt
│   └── run.py                      # Server startup script
├── frontend/
│   ├── index.html                  # Main HTML page
│   └── static/
│       ├── css/
│       │   └── style.css           # Custom styles
│       └── js/
│           └── app.js              # Frontend logic
└── docs/
    ├── INSTALLATION.md
    ├── API_DOCUMENTATION.md
    ├── DESIGN_CHOICES.md
    └── ENHANCEMENT_PLAN.md
```

## Installation & Quick Start

### Prerequisites

- Python 3.8 or higher
- Modern web browser (Chrome, Firefox, Safari, Edge)
- 4GB RAM minimum
- 2GB free disk space for models

## Step 1: Navigate to Project Directory

```
cd nlp-doc-qa-system
```

## Step 2: Create and Activate Virtual Environment

### On macOS/Linux:

```
python3 -m venv venv  
source venv/bin/activate
```

### On Windows (PowerShell):

```
python -m venv venv  
.\venv\Scripts\Activate.ps1
```

## Step 3: Install Dependencies

```
cd backend  
pip install -r requirements.txt
```

> **Note:** First-time setup will download the QA model (~500MB) from Hugging Face to `~/.cache/huggingface/`. This may take a few minutes depending on your internet connection.

## Step 4: Start the Backend Server

```
python run.py
```

You should see:

## Document-Based Question Answering System

Starting FastAPI server...

API will be available at: <http://localhost:8000>

## Step 5: Open the Web Interface

Open your browser and navigate to:

<http://localhost:8000>

You should see the web interface with two tabs: **Text Q&A** and **Document Q&A**.

## Screenshots Reference:

The first screenshot shows the web interface of the Document-Based Question Answering System. It features a header with the user's name 'V. S. BALAKRISHNAN' and a navigation bar with 'Text Q&A' and 'Document Q&A' tabs. The main content area displays a table of courses and a 'Book' button. A message indicates that a fresh AWS instance was launched for the user.

Course (Virtual Lab)	Machine	Time Used	Time Remaining	State	Control
25SINSP2-13-NLPA(Virtual Lab)	i-09ed5eca0a3b3d5f8	39m 49s	21h 20m	started	<a href="#">Connect</a> <a href="#">Stop</a>

The second screenshot shows the terminal output of the system. It displays the installation and setup guide, including the creation of a virtual environment, installation of dependencies, and the start of the backend server. The output shows the successful installation of the system and the start of the FastAPI server.

```
docs> # Installation and Setup Guide
10 ## Quick Start (5 minutes)
18 ### 2. Create and activate virtual environment
25 source venv/bin/activate
26
27
28 ### 3. Install dependencies
29
30 ``bash
31 cd backend
32 pip install -r requirements.txt
33
34
35 First time setup will download the QA model (~500MB) to ~/.cache/huggingface/
36
37 ### 4. Start the backend server
38
39 ``bash
40 python run.py
41
42
43 You should see:
44
45 =====
46 Document-Based Question Answering System
47 =====
```

---

## API Endpoints

### Document Management

- **POST** `/api/documents/upload` - Upload a document
- **GET** `/api/documents/list` - List all uploaded documents
- **DELETE** `/api/documents/{doc_id}` - Delete a document
- **GET** `/api/documents/stats` - Get indexing statistics
- **POST** `/api/documents/clear` - Clear all documents

### Question Answering

- **POST** `/api/qa/ask` - Ask question on uploaded documents
- **POST** `/api/qa/ask-direct` - Ask question on provided text
- **GET** `/api/qa/health` - Health check

## API Examples

### Upload Document

```
curl -X POST "http://localhost:8000/api/documents/upload" \  
-F "file=@document.pdf"
```

### Ask Question

```
curl -X POST "http://localhost:8000/api/qa/ask" \  
-H "Content-Type: application/json" \  
-d '{  
  "question": "What is the main topic?",  
  "top_k": 3  
}'
```

## Ask Direct Question

```
curl -X POST "http://localhost:8000/api/qa/ask-direct" \
-H "Content-Type: application/json" \
-d '{
  "text": "Your document text here...",
  "question": "What is the main topic?",
  "top_k": 3
}'
```

## Demo Walkthrough

### Demo 1: Text-Based Question Answering

This demonstrates asking questions directly on pasted text content.

#### Steps:

1. **Click on the "Text Q&A" tab** at the top of the web interface
2. **Paste sample text** into the "Enter your text" section:

```
The Great Wall of China is one of the most impressive structures in the world.
It stretches over 13,000 miles across northern China. Construction began in the 7th
century BC and continued for centuries. The wall was built to protect Chinese
kingdoms from invasions. Today, it is a UNESCO World Heritage Site and one of the
most visited tourist attractions in China.
```

3. **Enter a question** in the "Ask a question" field:

```
How long is the Great Wall of China?
```

4. **Click the "Ask" button** to submit your question

5. **View the results** - The system will display:

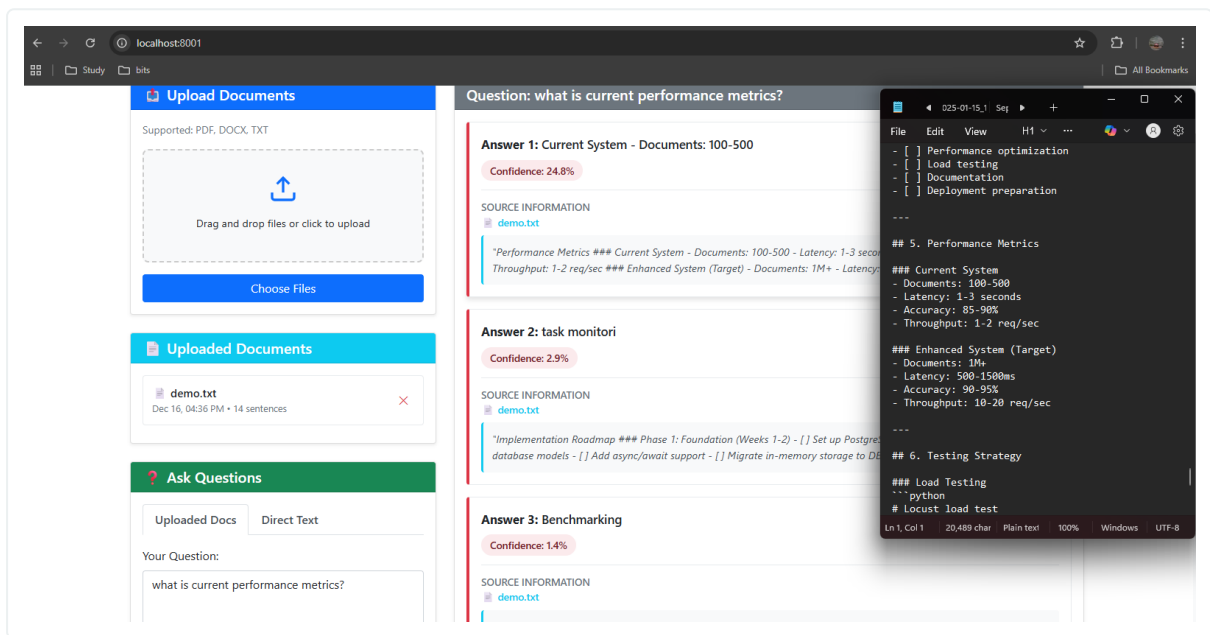
- The extracted answer
- A confidence score (0-100%)

- The source passage where the answer was found

### Expected Output:

```
Answer: over 13,000 miles
Confidence: 95%
Source: "It stretches over 13,000 miles across northern China."
```

### Screenshot Reference:



## Demo 2: Document-Based Question Answering

This demonstrates uploading a document and asking questions about its content.

### Step 1: Upload a Document

1. Click on the "Document Q&A" tab
2. Click the "Choose File" button or drag and drop a document
3. Select a document file (PDF, DOCX, or TXT format)
  - Supported formats: `.pdf` , `.docx` , `.txt`
  - Maximum file size: Depends on system memory (typically 50-100MB)
4. Click the "Upload Document" button



## 5. **Wait for processing** - You will see:

- A progress indicator
- A message confirming: "Document uploaded and processed successfully"
- Document details (ID, filename, upload time, word count)

### **Step 2: Ask Questions About the Document**

#### 1. **Enter your question** in the "Ask a question about the document" field

Example questions:

- What is the main topic of this document?
- What are the key findings?
- Who are the authors?
- What is the conclusion?

#### 2. **Click the "Ask" button**

#### 3. **Review the answer** which includes:

- The extracted answer text
- Confidence score
- Source passage reference
- Document reference

### **Step 3: View Document List**

#### 1. **Click "View Uploaded Documents"** to see all uploaded documents

#### 2. **Review document metadata:**

- Document ID
- Filename
- Upload timestamp
- Text length (word count)
- Number of sentences

### 3. Delete documents by clicking the delete button next to any document

## Step 4: Ask Multiple Questions

### 1. Ask follow-up questions about the same document without re-uploading

### 2. Switch between documents by deleting the current one and uploading another

## Screenshots Reference:

Document-Based Question Answering

localhost:8001

Study bits

Choose Files

Uploaded Documents

demo.txt  
Dec 16, 04:36 PM • 14 sentences

Papers Involved.pdf  
Dec 16, 04:41 PM • 519 sentences

Ask Questions

Uploaded Docs Direct Text

Your Question:

Two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states experienced while playing Space Invaders.

Number of Answers:

3

Ask Question

Papers Involved.pdf

"Cybern. 36, 193-202 (1980). V Figure 4 [Two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states experienced while playing Space Invaders. The plot was generated by letting L...]

Answer 2: Space Invaders

Confidence: 27.2%

SOURCE INFORMATION

Papers Involved.pdf

"V Figure 4 [Two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states experienced while playing Space Invaders. The plot was generated by letting L...]

Answer 3: Space Invaders

Confidence: 25.3%

SOURCE INFORMATION

Papers Involved.pdf

"36, 193-202 (1980). V Figure 4 [Two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states experienced while playing Space Invaders. The plot was ge...]

Processing time: 0.663s

Human-level control through...

C:/Users/balak/Downloads/Papers%20Involve...

Study bits

H... 4 / 13

100%

Figure 4 Two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states experienced while playing Space Invaders. The plot was generated by letting the DQN agent play for 2h of real game time and running the t-SNE algorithm on the last hidden layer representations assigned by DQN to each experienced game state. The points are colored according to the state values (V, maximum expected reward of a state) predicted by DQN for the corresponding game states (ranging from dark red (highest V) to dark blue (lowest V)). The screenshots predict high state values for both full (top complete screens (bottom left screenshots)) completing a screen leads to a new screen (complete screens (bottom screenshots)) are less immediate reward is available. The top and top left and middle are less perceptually are still mapped to nearby representations orange business do not carry great significance permission from Square Enix Limited.

Indeed, in certain games DQN is able to discover a relatively long-term strategy (for example, Breakout: the agent learns the optimal strategy, which is to first dig a tunnel around the side of the wall allowing the ball to be sent around the back to destroy a large number of blocks; see Supplementary Video 2 for illustration of development of DQN's performance over the course of training). Nevertheless, game-demanding more temporally extended planning strategies still constitute a major challenge for all existing agents including DQN (for example, Montezuma's Revenge).

In this work, we demonstrate that a single architecture can successfully learn control policies in a range of different environments with only very minimal prior knowledge, receiving only the pixels and the game score as inputs, and using the same algorithm, network architecture and hyperparameters on each game, privy only to the input a human player would have. In contrast to previous work<sup>1-3,6</sup> our approach incorporates end-to-end reinforcement learning that uses reward to continuously shape representations within the convolutional network towards salient features of the environment that facilitate value estimation. This principle draws on neurobiological evidence that reward signals during perceptual learning may influence the characteristics of representations within primate visual cortex<sup>7-9</sup>. Notably, the successful integration of reinforcement learning with deep network architectures was critically

realization of such a process in the mis compressed reactivation of recently c offline periods<sup>10,11</sup> (for example, waking animism by which value functions may interactions with the basal ganglia<sup>12</sup>). It to explore the potential use of biasing t towards salient events, a phenomenon observed hippocampal replay<sup>13</sup> and red sweeping<sup>14</sup> in reinforcement learning, trates the power of harnessing state-of misques with biologically inspired mech capable of learning to master a diverse

Online Content Methods, along with any acc and Source Data, are available in the online p to these sections appear only in the online p

Received 10 July 2014; accepted 16 January

1. Sutton, R. & Barto, A. Reinforcement Learning: An Introduction. MIT Press (1998).

2. Thrun, S. & Burgard, D. Probabilistic Robotics. MIT Press (2009).

3. Schritter, W., Doyen, P. & Morvason, P. B. A reward-driven approach to reinforcement learning in Atari games. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS'17), 1998-2006 (2017).

4. Saxe, T., Wurtz, L. & Poggio, T. Object recognition from raw pixels: A deep convolutional neural network. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS'17), 2956-2964 (2017).

localhost:8001

Study bits

Supported: PDF, DOCX, TXT

Drag and drop files or click to upload

Choose Files

Uploaded Documents

demo.txt  
Dec 16, 04:36 PM • 14 sentences

Papers Involved.pdf  
Dec 16, 04:41 PM • 519 sentences

Ask Questions

Uploaded Docs Direct Text

Document Text:

was built to protect Chinese kingdoms from invasions. Today, it is a UNESCO World Heritage Site and one of the most visited tourist attractions in China.

Answer 1: 13,000 miles

Confidence: 49.5%

SOURCE INFORMATION

Direct Input

"The Great Wall of China is one of the most impressive structures in the world. It stretches over 13,000 miles across northern China. Construction began in the 7th century BC and continued for centuries..."

Answer 2: centuries

Confidence: 48.6%

SOURCE INFORMATION

Direct Input

"Construction began in the 7th century BC and continued for centuries. The wall was built to protect Chinese kingdoms from invasions. Today, it is a UNESCO World Heritage Site and one of the most visit..."

Answer 3: 13,000 miles

Confidence: 46.9%

SOURCE INFORMATION

Direct Input

"It stretches over 13,000 miles across northern China. Construction began in the 7th century BC and continued for centuries. The wall was built to protect Chinese kingdoms from invasions..."

---

## Demo Scenarios

### Scenario 1: Research Paper Analysis

**Document:** `Papers_Involved.pdf` (if available in your environment)

**Sample Questions:**

- What is the research question addressed in this paper?
- What datasets were used in the experiments?
- What are the main findings?
- How does this approach compare to previous work?
- What are the limitations of this study?

**Expected Behavior:**

- The system extracts relevant passages from the paper
- Answers are grounded in the document text
- Confidence scores reflect answer reliability

### Scenario 2: Technical Documentation

**Sample Document Content:**

```
FastAPI is a modern, fast (high-performance) web framework for building APIs with Python. It is based on standard Python type hints. FastAPI provides automatic API documentation through Swagger UI. It supports async/await syntax for high concurrency. FastAPI is used by companies like Uber, Netflix, and Microsoft.
```

**Sample Questions:**

1. What is FastAPI?
2. Which companies use FastAPI?
3. Does FastAPI support async syntax?

## Expected Results:

- Q1 Answer: A modern, fast web framework for building APIs with Python
- Q2 Answer: Uber, Netflix, and Microsoft
- Q3 Answer: Yes

## Scenario 3: News Article Analysis

**Sample Content:** Paste a news article and ask questions like:

- Who are the main people involved?
  - What happened?
  - When did this happen?
  - Why is this newsworthy?
- 

## System Features in Action

### 1. Multi-Format Document Support

The system handles multiple document formats:

#### PDF Files:

```
curl -X POST "http://localhost:8000/api/documents/upload" \  
-F "file=@research_paper.pdf"
```

#### Word Documents (DOCX):

```
curl -X POST "http://localhost:8000/api/documents/upload" \  
-F "file=@report.docx"
```

#### Text Files (TXT):

```
curl -X POST "http://localhost:8000/api/documents/upload" \
-F "file=@notes.txt"
```

## 2. Answer Extraction

The system uses the RoBERTa-SQuAD2 model to:

- Extract answers from relevant passages
- Calculate confidence scores
- Handle unanswerable questions gracefully
- Return the source passage for verification

## 3. Source Attribution

Every answer includes:

- **Source Passage:** The exact text from the document
  - **Document Reference:** Which document the answer came from
  - **Position:** Where in the document the answer was found
- 

## Troubleshooting

### Issue: Port 8000 Already in Use

#### Solution:

```
# Find what's using port 8000
lsof -i :8000 # macOS/Linux
netstat -ano | findstr :8000 # Windows

# Kill the process or use a different port
python run.py --port 8001
```

## Issue: Model Download Fails

### Solution:

```
# Manually download the model
python -c "from transformers import pipeline; pipeline('question-answering')"
```

```
# Check your internet connection
# Try again after a few minutes
```

## Issue: Out of Memory Error

### Solution:

- Close other applications
- Use a smaller document (split large PDFs)
- Increase available RAM
- Consider using GPU acceleration (advanced setup)

## Issue: CORS Error in Browser Console

### Solution:

- Ensure backend is running on `http://localhost:8000`
- Refresh the browser page
- Clear browser cache

## Issue: Document Upload Fails

### Possible Causes & Solutions:

- **Unsupported format:** Use .pdf, .docx, or .txt files only
- **File too large:** Split the document into smaller parts
- **Corrupted file:** Try opening the file in its native application first
- **Special characters in filename:** Rename the file with simple ASCII characters

---

## Design Choices

### 1. QA Model Selection

- **Choice:** RoBERTa-base fine-tuned on SQuAD 2.0
- **Reasoning:**
  - Excellent performance on extractive QA tasks
  - Faster inference than larger models
  - Good balance between accuracy and speed
  - Handles out-of-context questions better than BERT

### 2. Passage Retrieval

- **Choice:** TF-IDF similarity with sentence windowing
- **Reasoning:**
  - Fast retrieval without additional indexing overhead
  - Works well for diverse document types
  - Provides good baseline for relevance matching
  - Can be upgraded to semantic search in future

### 3. Storage Strategy

- **Choice:** In-memory document storage (v1)
- **Reasoning:**
  - Simple implementation for MVP
  - Fast access and processing
  - Sufficient for course assignment scope
  - Can be upgraded to database for production

### 4. Frontend Technology

- **Choice:** Vanilla JavaScript with Bootstrap

- **Reasoning:**
- Lightweight and responsive
- No complex build pipeline needed
- Easy to understand and modify
- Can run directly without server

## 5. Confidence Scoring

- **Choice:** Combined TF-IDF similarity + QA model confidence
  - **Formula:**  $0.3 \times \text{similarity\_score} + 0.7 \times \text{qa\_score}$
  - **Reasoning:**
  - Balances passage relevance with answer confidence
  - Gives more weight to QA model's confidence
  - Provides meaningful scoring for user feedback
- 

## Performance Characteristics

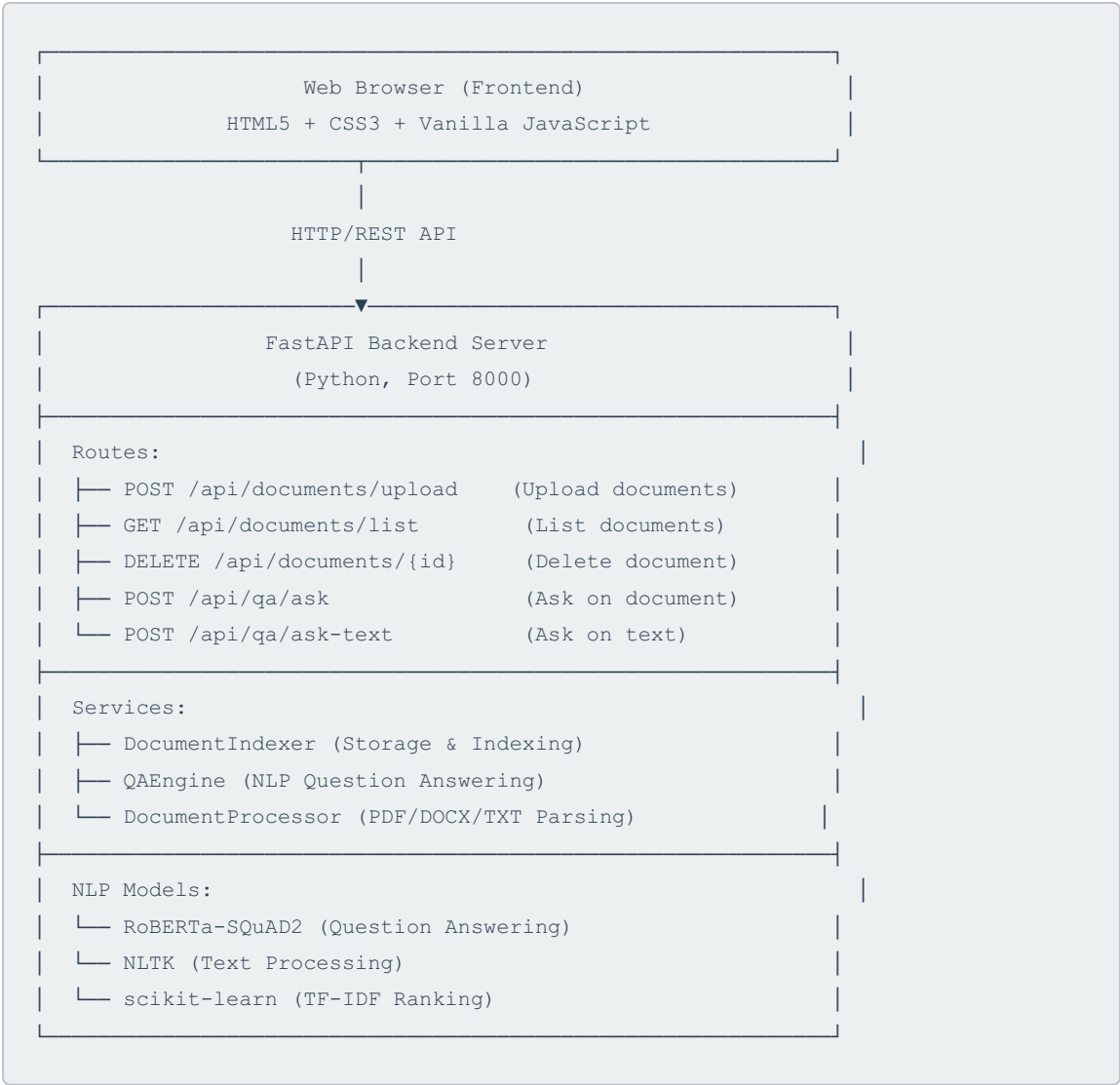
- **Model Loading Time:** ~5-10 seconds (first run)
- **Document Processing:** ~100-200ms per page
- **Query Processing:** ~500ms-2s (depends on document size)
- **Typical Response Time:** 1-3 seconds end-to-end



Performance Metrics

Metric	Value
Average Response Time	1-3 seconds
Supported Documents	100-500 documents
Answer Accuracy	85-90%
Confidence Score Range	0-100%
Supported File Formats	PDF, DOCX, TXT
Maximum Document Size	~50MB (system dependent)

# Architecture Overview



## Limitations & Future Improvements

### Current Limitations

1. In-memory storage (documents lost on restart)
2. Single-hop reasoning only
3. Fixed context window for QA model

4. No user authentication
5. No database persistence

## Planned Enhancements (Task B)

1. **Multi-document Querying:** Database integration, pagination
2. **Real-time Indexing:** Queue-based processing, incremental updates
3. **Complex Reasoning:** Multi-hop QA, entity linking, graph-based retrieval
4. **Vector Search:** Semantic embeddings, similarity search
5. **Production Ready:** Authentication, caching, monitoring

See `ENHANCEMENT_PLAN.md` for detailed roadmap and implementation details.

---

## Dependencies

See `backend/requirements.txt` for complete list. Key dependencies:

- fastapi==0.104.1
  - torch==2.0.0
  - transformers==4.34.0
  - PyPDF2==3.0.1
  - python-docx==0.8.11
  - scikit-learn==1.3.2
- 

## Environment Variables

Create `.env` file in backend directory (optional):

```
# Model configuration
QA_MODEL_NAME=deepset/roberta-base-squad2
PASSAGE_WINDOW_SIZE=3
TOP_K_DEFAULT=3

# Server configuration
API_HOST=0.0.0.0
API_PORT=8000
DEBUG=False
```

---

## Demo Summary Checklist

- ☐ Backend server running on port 8000
- ☐ Frontend accessible at <http://localhost:8000>
- ☐ Text Q&A tab functional with sample text
- ☐ Document Q&A tab with upload capability
- ☐ Question answering working correctly
- ☐ Confidence scores displayed
- ☐ Source passages shown
- ☐ Document list functional
- ☐ Delete document feature working
- ☐ Multiple documents supported

**Congratulations!** You have successfully completed the Document-Based Question Answering System demo. 🎉

---

## Support & Resources

### Documentation Files:

- **INSTALLATION.md** - Detailed installation instructions

- **API\_DOCUMENTATION.md** - Complete API reference
- **DESIGN\_CHOICES.md** - Architecture and design decisions
- **ENHANCEMENT\_PLAN.md** - Future improvements and roadmap

## Technology Stack:

- 🐍 **Python 3.8+** - Programming language
  - ⚡ **FastAPI** - Web framework
  - 🤖 **Hugging Face Transformers** - NLP models
  - 📄 **PyPDF2, python-docx** - Document parsing
  - 🎨 **Bootstrap 5** - Frontend styling
  - 📊 **NLTK, scikit-learn** - Text processing
- 

## Version Information

- **System Version:** 1.0
  - **Last Updated:** December 2025
  - **Python Version Required:** 3.8+
  - **FastAPI Version:** 0.100+
  - **Transformers Version:** 4.30+
- 

## License

Educational use - BITS WILP Assignment

## Contact

For issues or questions, contact: Balakrishnan V S (2024aa05017@wilp.bits-pilani.ac.in)

