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
```

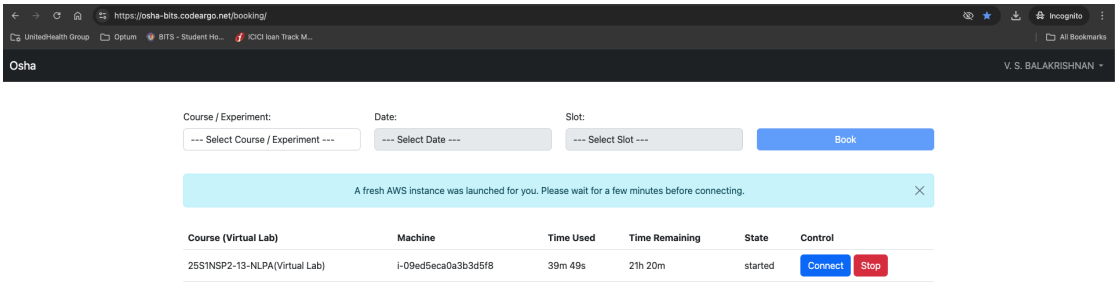
Step 5: Open the Web Interface

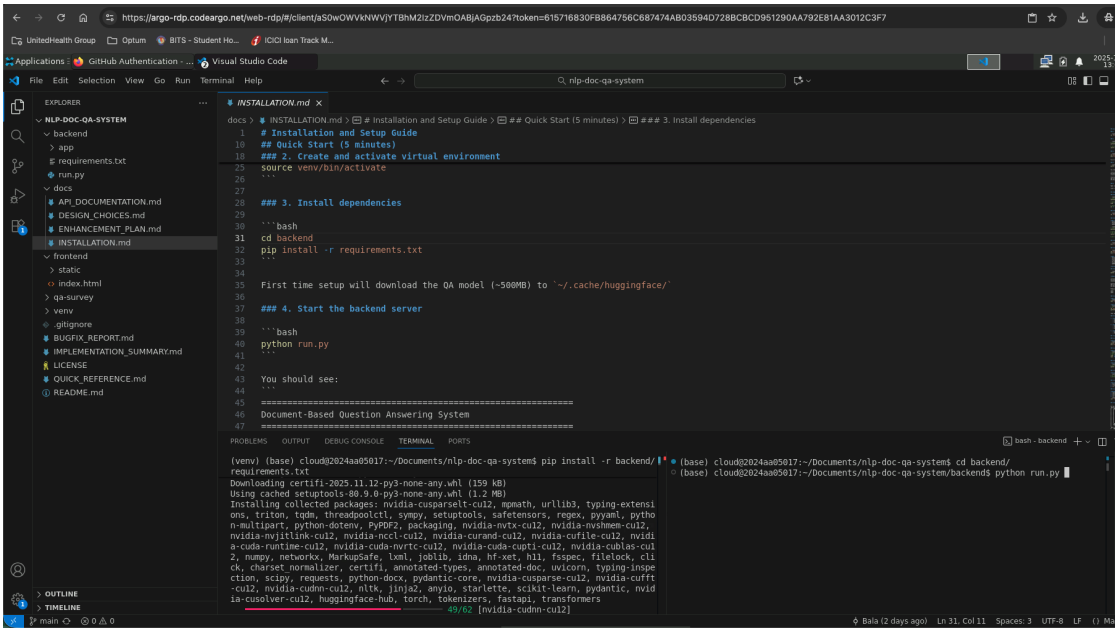
Open your browser and navigate to:

```
http://localhost:8000
```

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

Screenshots Reference:

- 

- 

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:

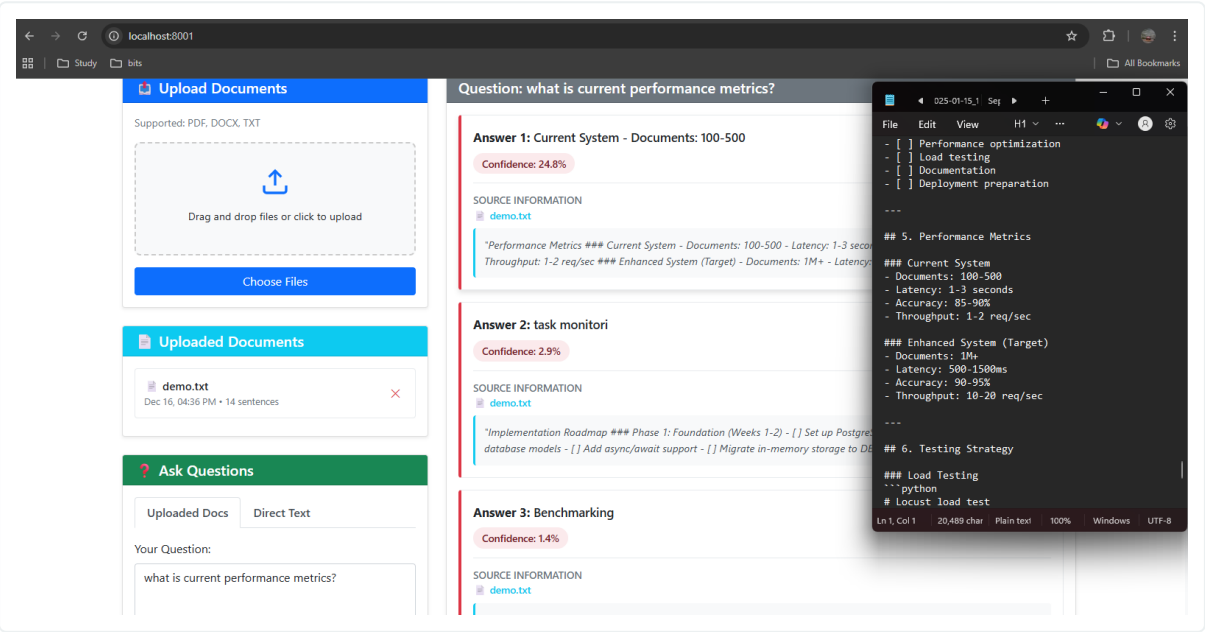
   - The extracted answer
   - A confidence score (0-100%)

- 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

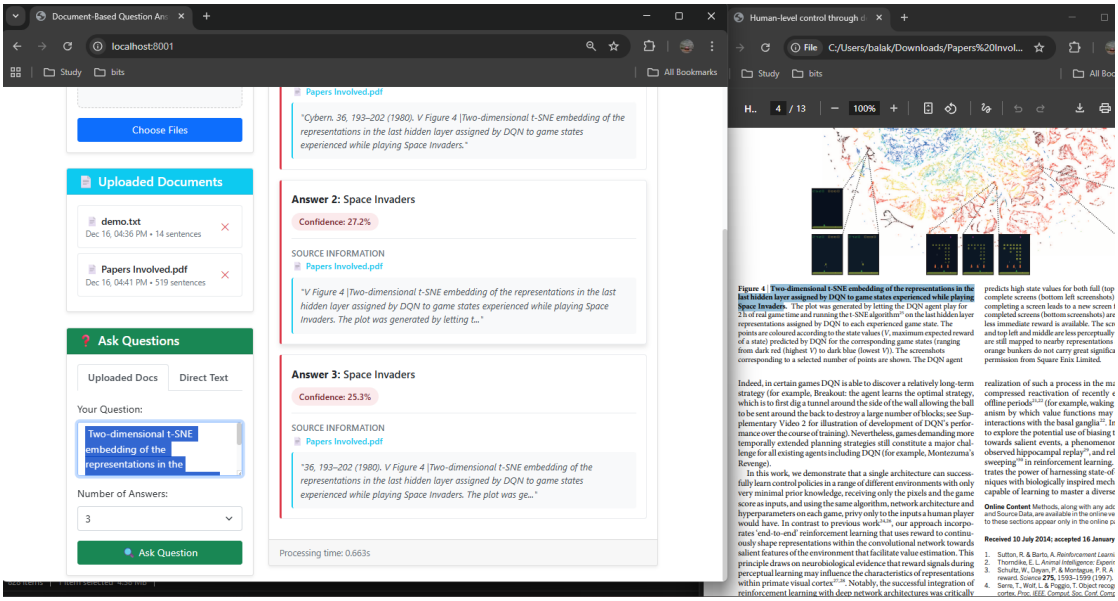
3. **Delete documents** by clicking the delete button next to any document

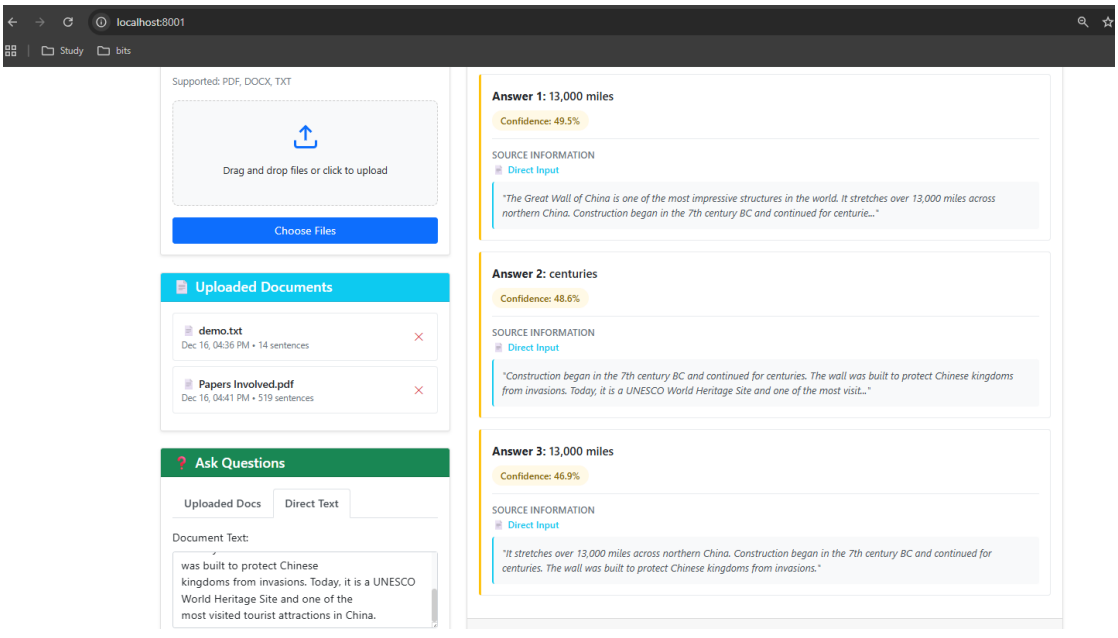
Step 4: Ask Multiple Questions

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:

- 

- 

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**: $0.3 \times similarity_{score} + 0.7 \times qa_{score}$
- **Reasoning**:
- Balances passage relevance with answer confidence
- Gives more weight to QA model's confidence
- Provides meaningful scoring for user feedback

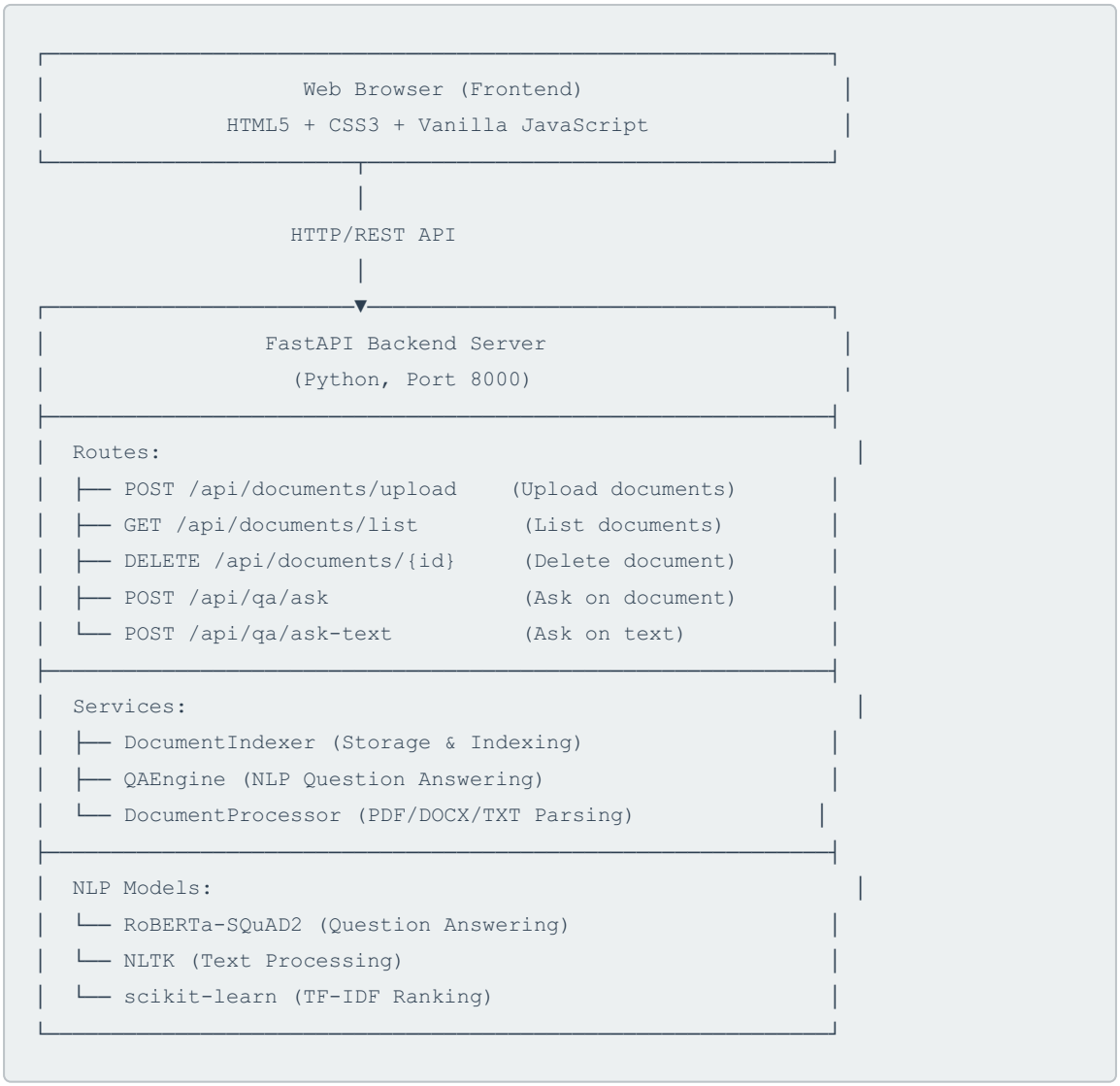
Performance Characteristics

- **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)