Week 1 Testing Instructions

Setting Up and Testing the Backend

Follow these step-by-step instructions to set up and test your Week 1 implementation:

1. Environment Setup

First, create a virtual environment and install the required dependencies:

```bash
# Create the project directory structure
mkdir -p document-qa-rag/backend
cd document-qa-rag/backend

# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate

# Copy all the code files into their respective locations according to the folder structure

# Install dependencies
pip install -r requirements.txt
```

2. Manual Testing

You can run the application and test it manually:

```bash
# Run the application
python run.py
```

The API will be available at `http://localhost:8000`. You can access the following endpoints:

- API documentation: `http://localhost:8000/docs`
- Health check: `http://localhost:8000/`

- Document upload: `http://localhost:8000/api/v1/documents/upload`
- List documents: `http://localhost:8000/api/v1/documents/`

Testing with Swagger UI

1. Open `http://localhost:8000/docs` in your browser
2. Navigate to the `/api/v1/documents/upload` endpoint
3. Click "Try it out"
4. Upload a test file (PDF, DOCX, TXT, etc.)
5. Click "Execute"
6. Verify the response contains the document details with a "processed" status

Testing Document List and Retrieval

1. After uploading a document, go to the `/api/v1/documents/` endpoint
2. Click "Try it out" and then "Execute"
3. Verify that your uploaded document appears in the list
4. Copy the `id` of your document
5. Go to the `/api/v1/documents/{document_id}` endpoint
6. Paste the ID and click "Execute"
7. Verify you can retrieve the specific document details

3. Running Automated Tests

To run the automated tests to verify your implementation:

```bash
# Make sure you're in the backend directory
cd document-qa-rag/backend

# Run tests
pytest
```

You should see output indicating that all tests have passed. Pay attention to any failures or errors and fix the issues if necessary.

4. Creating Test Files for Manual Testing

For more thorough manual testing, create sample files of different types:

Sample Text File

Create a file named `test_sample.txt` with content:

```
This is a sample text file for testing document processing.
It contains multiple lines to test text extraction.
The Document Q&A system should be able to process this file correctly.
```

Sample PDF File

If you have PDF creation tools, create a simple PDF with text, or download a sample PDF from the internet.

Sample DOCX File

Create a Word document with various formatting:

- Some bold text
- Bullet points
- Multiple paragraphs
- A simple table

5. Verification Checklist

Use this checklist to verify your implementation:

- [ ] Application starts without errors
- [ ] Can upload text files (.txt) successfully
- [ ] Can upload PDF files (.pdf) successfully (if you have sample PDFs)
- [ ] Can upload Word documents (.docx) successfully (if you have sample DOCXs)
- [ ] Invalid file types are rejected with appropriate error messages
- [ ] Can list all uploaded documents
- [ ] Can retrieve a specific document by ID
- [ ] Can delete a document
- [ ] All automated tests pass

6. Verifying Text Extraction

To verify that text extraction works correctly:

1. Upload a document

2. Note the document ID from the response
3. Check the `processed` directory in your project:

   ```bash
   cat data/processed/{document_id}.txt
   ```

4. Verify that the extracted text matches the content of your original document

7. Common Issues and Troubleshooting

- **Database errors**: Ensure SQLite database is properly initialized
- **File permission issues**: Check that your application has permission to write to the data directories
- **Missing dependencies**: Verify all packages are installed with the correct versions
- **PDF extraction not working**: PDFPlumber requires some additional dependencies; you might need to install them separately
- **CORS errors when testing from a web client**: Update the CORS settings in main.py if needed

8. Docker Testing (Optional)

If you want to test the Docker setup:

```bash
# Build the Docker image
docker build -t document-qa-rag-backend .

# Run the container
docker run -p 8000:8000 document-qa-rag-backend
```

Access the API at `http://localhost:8000` as before, but now it's running in a container.

9. Next Steps After Successful Testing

Once you've verified that Week 1 implementation works correctly:

1. Commit your code to version control
2. Document any issues or customizations you made
3. Prepare for Week 2 by understanding how the document processing code will integrate with the embedding system

Congratulations! You've completed and tested the Week 1 implementation of your Document Q&A RAG system.