Software Requirements Specification

for

Chart Understanding LLM System

with Chart2Table Pipeline & Model Comparison

Version 1.0

December 11, 2025

Supervisors:
Dr. Ahmed Tawfik
Dr. Cherif Salama
Dr. Hossam Sharara

Department of Computer Science and Engineering
The American University in Cairo

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the Chart Understanding LLM System with Chart2Table Pipeline and Model Comparison capabilities. The system is designed to analyze chart images using advanced vision-language models, extract tabular data through different extraction pipelines (PaddlePaddle/Text2Chart and DePlot), and provide accurate answers to user questions about chart content.

The purpose of this document is to:

- Define all functional and non-functional requirements for the system
- Establish a basis for agreement between stakeholders and development team
- Provide a reference for system validation and verification

1.2 Document Conventions

This document follows IEEE Std 830-1998 guidelines for Software Requirements Specifications. The following conventions are used throughout:

- **Requirement Priority Levels:**
  - *Critical*: Essential for system operation
  - *High*: Important for system functionality
  - *Medium*: Desirable but not essential
  - *Low*: Optional enhancement
- **Naming Conventions:**
  - Requirements are identified with unique IDs (e.g., FR-1.1, NFR-2.3)
  - System components use PascalCase (e.g., ChartAnalyzer)
  - API endpoints use lowercase with hyphens (e.g., /extract-base64)
- **Technical Terms**: Defined in Appendix A

1.3 Intended Audience

This document is intended for:

1. **Development Team**: The students conducting the research and developing the application
2. **Academic Advisors**: Faculty members supervising the graduation project
3. **End Users/Stakeholders**: Researchers and data analysts who will use the system when it's published

1.4 Additional Information

Project Context: This system is developed as a thesis graduation project focusing on evaluating prompting techniques for chart understanding using vision-language models.

Research Focus: The project investigates multiple strategies including baseline approaches, chain-of-thought reasoning, few-shot learning, and hybrid chain-of-models approaches (Chart2Table + LLM, DePlot + LLM).

Dataset: The system is evaluated using the ChartQA benchmark dataset (HuggingFaceM4/ChartQA) and PlotQA-v2 (jrc/cleaned-plotqa-v2).

Name	**Contact**
Andrew Aziz	andrewwassem@aucegypt.edu
Kirolous Fouty	kirolous_fouty@aucegypt.edu
MagdElDin AbdalRaaof	magdeldin@aucegypt.edu
Mohamed Abbas	Mohamed_Ragab02@aucegypt.edu
Shaza Ali	shazamamdouh@aucegypt.edu
Tarek Kassab	tarek_kassab@aucegypt.edu

1.5 Contact Information/SDS Team Members

1.6 References

1. IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
2. Qwen2.5-VL Documentation: `https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct`
3. PaddlePaddle PP-Chart2Table: `https://github.com/PaddlePaddle/PaddleOCR`
4. DePlot Model: `https://huggingface.co/google/deplot`
5. ChartQA Dataset: `https://huggingface.co/datasets/HuggingFaceM4/ChartQA`
6. PlotQA Dataset: `https://huggingface.co/datasets/jrc/cleaned-plotqa-v2`

2 Overall Description

2.1 Product Perspective

The Chart Understanding LLM System is a standalone research prototype designed to evaluate different approaches to automated chart analysis. The system consists of three main architectural components:

1. **Client Application**: PyQt6-based desktop GUI for user interaction and model comparison
2. **Model Server Infrastructure**:
   - Base Model Server: Standalone Qwen2.5-VL inference server (Port 8001)
   - Pipelined Model Server: Integrated Chart2Table + Qwen2.5-VL server (Port 8001)
   - PaddlePaddle Server: Chart extraction microservice (Port 8000)
3. **Text2Struct Pipeline**: DePlot-based alternative extraction approach

The system operates in a distributed client-server architecture where:

- Clients communicate with servers via HTTP/REST APIs
- Servers can run on different machines (configurable endpoints)
- Each server handles specific model inference tasks
- The system supports both single-model and multi-model pipeline approaches

2.2 Product Functions

The system provides the following major functions:

1. **Chart Image Analysis**
   - Accept chart images in common formats (PNG, JPG, JPEG, BMP)
   - Process images using state-of-the-art vision-language models
   - Support multiple chart types (bar, pie, line, scatter, etc.)
2. **Question Answering**
   - Accept natural language questions about chart content
   - Generate accurate answers using multiple prompting strategies
   - Support numerical calculations and reasoning tasks
3. **Data Extraction**
   - Extract tabular data from charts using PaddlePaddle PP-Chart2Table
   - Alternative extraction using Google DePlot
   - Clean and format extracted data for LLM consumption
4. **Model Comparison**
   - Simultaneously query two different model endpoints
   - Display responses side-by-side for evaluation
   - Track inference time and performance metrics
5. **Strategy Evaluation**
   - Implement 8 different prompting strategies
   - Benchmark accuracy on standard datasets
   - Generate comprehensive evaluation reports

2.3 User Classes and Characteristics

1. **Research Scientists/Data Analysts**
   - Technical expertise: High
   - Primary use: Evaluating model performance and prompting strategies
   - Frequency of use: Regular during research phase
   - Key needs: Accurate results, detailed metrics, reproducibility
2. **End Users (General)**
   - Technical expertise: Medium
   - Primary use: Analyzing charts and extracting insights
   - Frequency of use: Occasional
   - Key needs: User-friendly interface, quick responses, clear answers
3. **ML Engineers/Developers/Future Researchers**
   - Technical expertise: Very High
   - Primary use: Extending system capabilities and implementing new strategies
   - Frequency of use: During development phase
   - Key needs: Clean code architecture, comprehensive documentation, modularity

2.4 Operating Environment

Server Environment:

- Operating System: Linux (Ubuntu 20.04+ recommended) or Windows 10/11
- Python Version: 3.8 or higher
- GPU Requirements: NVIDIA GPU with CUDA support
  - Minimum VRAM: 16GB (for Qwen2.5-VL-7B)
  - Recommended VRAM: 24GB or higher
  - CUDA Version: 11.8 or higher
- RAM: Minimum 32GB, Recommended 64GB
- Storage: 50GB minimum for models and data

Client Environment:

- Operating System: Windows 10/11, macOS 10.15+, or Linux
- Python Version: 3.8 or higher
- Display: Minimum resolution 1280x720
- Network: Stable internet connection for API communication

2.5 User Environment

The typical user environment consists of:

- **Single User Operation**: Individual researchers working on desktop/laptop computers
- **Network Configuration**:
  - Local network deployment (servers on same subnet as client)
  - Remote server deployment (servers on different machines, accessed via IP)
- **Concurrent Users**: System supports multiple client connections to same server
- **Typical Session Duration**: 15-60 minutes per analysis session
- **Data Volume**: Processing 10-100 charts per session during evaluation

2.6 Design/Implementation Constraints

1. **Hardware Constraints**
   - GPU memory limitation affects batch processing capabilities
   - System requires dedicated GPU resources (cannot share with other intensive tasks)
2. **Software Constraints**
   - PyTorch framework dependency (version compatibility)
   - PaddlePaddle requires separate environment due to dependency conflicts
   - Model weights total 15GB+, requiring significant storage
3. **Performance Constraints**
   - Inference time: 2-10 seconds per chart depending on complexity
   - Memory usage: 70-80% GPU memory allocation per model server
   - Network latency affects multi-server pipeline approaches
4. **Architectural Constraints**
   - Microservice architecture requires multiple server instances
   - REST API communication limits real-time streaming responses
   - PaddlePaddle and PyTorch environments cannot coexist easily
5. **Data Constraints**
   - Image size limitations (max 10MB per image recommended)
   - Supported formats limited to PNG, JPG, JPEG, BMP
   - Chart complexity affects extraction accuracy

2.7 Assumptions and Dependencies

Assumptions:

- Users have basic understanding of chart types and terminology
- Input charts are reasonably clear and well-formatted
- Network connectivity between client and servers is stable
- GPU resources are available and properly configured
- Users have necessary permissions to install required software

Dependencies:

- **Python Libraries**:
  - PyTorch (2.0+)
  - Transformers (4.30+)
  - qwen-vl-utils
  - FastAPI
  - Uvicorn
  - PyQt6
  - PaddlePaddle
  - PaddleOCR
  - PIL/Pillow
  - NumPy, Pandas

- **Pre-trained Models**:
  - Qwen2.5-VL-7B-Instruct (from Hugging Face)
  - PP-Chart2Table (from PaddlePaddle)
  - DePlot (google/deplot from Hugging Face)
- **External Services**:
  - Hugging Face Hub (for model downloads)
  - Internet connection (initial setup only)
- **Operating System Dependencies**:
  - CUDA toolkit and drivers
  - System libraries for PyQt6
  - Network stack for HTTP communication

3 External Interface Requirements

3.1 User Interfaces

3.1.1 UI-1: Main Application Window

- **Description**: PyQt6-based desktop application with split-pane layout
- **Components**:
  - Left panel: Input controls (question text area, image upload, endpoint configuration)
  - Right panel: Dual response display (Model A and Model B side-by-side)
  - Status bar: Progress indicator and connection status
- **Screen Resolution**: Minimum 1280x720, Recommended 1920x1080
- **Layout**: Resizable splitter between panels for user customization

3.1.2 UI-2: Question Input Interface

- **Component**: Multi-line text edit widget
- **Features**:
  - Placeholder text: "Type the user's question here..."
  - Auto-expanding based on content
  - Copy/paste support
  - Character count display (optional)

3.1.3 UI-3: Image Upload Interface

- **Components**:
  - "Attach image" button
  - "Clear image" button
  - Image preview area (360x270 pixels)
- **Behavior**:
  - Click to open file dialog
  - Preview scales to fit while maintaining aspect ratio
- **Supported Formats**: PNG, JPG, JPEG, BMP

3.1.4 UI-4: Model Response Display

- **Components** (per model):
  - Header: Model name (configurable)
  - Info label: Confidence and response time
  - Answer text area: Read-only, scrollable
  - Action buttons: "Show raw JSON", "Save text", "Save raw JSON"
- **Formatting**:
  - Automatic escape sequence handling
  - Removal of markdown formatting symbols
  - Clear error message display

3.1.5 UI-5: Endpoint Configuration

- **Components**:
  - Two text input fields for Model A and Model B URLs
  - Default values: http://10.40.32.8:8001/predict and http://10.40.32.12:8001/predict
- **Validation**: URL format checking before sending requests

3.2 Hardware Interfaces

3.2.1 HW-1: GPU Interface

- **Type**: NVIDIA CUDA-enabled GPU
- **Interface Protocol**: CUDA API via PyTorch
- **Data Transfer**: Model weights and tensors via PCIe bus
- **Memory Management**:
  - Configurable GPU memory fraction (default 75%)
  - TF32 support for Ampere+ architectures
  - Automatic device placement via device_map="auto"

3.2.2 HW-2: Network Interface

- **Type**: Standard Ethernet or Wi-Fi
- **Protocol**: TCP/IP
- **Bandwidth Requirements**:
  - Minimum: 10 Mbps
  - Recommended: 100 Mbps for optimal performance
- **Ports Used**:
  - 8000: PaddlePaddle API Server
  - 8001: Model Inference Servers

3.2.3 HW-3: Storage Interface

- **Type**: Local filesystem (HDD/SSD)
- **Access Method**: Standard file I/O
- **Usage**:
  - Model weights storage (15GB+)
  - Temporary image files during processing
  - Benchmark datasets
  - Results and logs

3.3 Software Interfaces

3.3.1 SW-1: Qwen2.5-VL Model Interface

- **Component**: Qwen2_5_VLForConditionalGeneration
- **Source**: Hugging Face Transformers library
- **Version**: transformers = 4.30.0+
- **Communication**: Direct Python API calls
- **Data Format**:
  - Input: Processed tensors (image + text)
  - Output: Generated token IDs decoded to text
- **Configuration**:
  - dtype: torch.bfloat16
  - device_map: "auto"
  - trust_remote_code: True

3.3.2 SW-2: PaddlePaddle PP-Chart2Table Interface

- **Component**: PPStructureV3 / create_model('PP-Chart2Table')
- **Source**: PaddlePaddle framework
- **Version**: paddlepaddle-gpu = 2.5.0+
- **Communication**: REST API (HTTP POST)
- **Endpoints**:
  - /health: Health check
  - /extract: File upload extraction
  - /extract-base64: Base64 image extraction
- **Data Format**:
  - Input: Image file or base64 encoded string
  - Output: JSON with raw_table, table_clean, rows, headers

3.3.3 SW-3: DePlot Model Interface

- **Component**: Pix2StructForConditionalGeneration
- **Source**: google/deplot (Hugging Face)
- **Version**: transformers = 4.30.0+
- **Communication**: Direct Python API calls
- **Prompt**: "Generate underlying data table of the figure below:"
- **Output Format**: Linearized table with special tokens (0x0A for newlines)

3.3.4 SW-4: FastAPI Server Interface

- **Component**: FastAPI application framework
- **Version**: fastapi = 0.100.0+
- **Server**: Uvicorn ASGI server
- **Features**:
  - Automatic OpenAPI documentation
  - Request validation via Pydantic
  - Async request handling
  - CORS support (if needed)

3.3.5 SW-5: PyQt6 GUI Framework

- **Component**: PyQt6 widget library
- **Version**: PyQt6 = 6.4.0+
- **Key Modules**:
  - QtWidgets: UI components
  - QtCore: Threading and signals
  - QtGui: Graphics and image handling

3.4 Communication Protocols and Interfaces

3.4.1 COMM-1: HTTP REST API Protocol

- **Protocol**: HTTP/1.1
- **Method**: POST
- **Content-Type**: multipart/form-data
- **Request Structure**:

```
POST /predict HTTP/1.1
Content-Type: multipart/form-data; boundary=---...
Content-Length: [size]

---[boundary]
Content-Disposition: form-data; name="question"

[question text]
---[boundary]
Content-Disposition: form-data; name="image";
                     filename="image.jpg"
Content-Type: image/jpeg

[binary image data]
---[boundary]--
```

- **Response Structure**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "answer": "string",
```

```
  "confidence": float (optional),
  "cleaned_table": "string" (for pipelined),
  "strategy": "string" (for pipelined),
  "_raw_text": "string" (optional),
  "_client_elapsed_ms": int (optional)
}
```

- **Timeout**: 60 seconds (configurable)
- **Error Handling**: HTTP status codes (400, 500) with error details in JSON

3.4.2 COMM-2: PaddlePaddle API Protocol

- **Endpoint**: /extract-base64
- **Method**: POST
- **Content-Type**: application/json
- **Request Format**:

```
{
  "image_base64": "base64_encoded_image_string"
}
```

- **Response Format**:

```
{
  "success": boolean,
  "raw_table": "string",
  "table_clean": "string",
  "rows": [array of objects],
  "headers": [array of strings],
  "num_rows": int,
  "num_cols": int
}
```

- **Timeout**: 30 seconds (configurable)

3.4.3 COMM-3: Inter-Process Communication

- **Client-Server**: HTTP REST (as described above)
- **Model Server - Paddle Server**: HTTP REST API calls
- **Thread Communication**: PyQt6 signals/slots for UI updates
- **Data Serialization**: JSON for all API communications

4 System Features

4.1 Dual Model Comparison Mode

4.1.1 Description and Priority

Priority: High
Description: The system shall allow users to simultaneously query two different model endpoints and display their responses side-by-side for comparison and evaluation purposes.

4.1.2 Stimulus/Response Sequences

1. User configures two model endpoint URLs
2. User enters a question and attaches an image
3. User clicks "Send to both models"
4. System initiates two concurrent HTTP requests
5. System displays responses as they arrive
6. System shows performance metrics for each model

4.1.3 Functional Requirements

FR-1.1: The system shall support configurable endpoint URLs for two independent models
Priority: Critical
FR-1.2: The system shall send identical question and image data to both endpoints simultaneously
Priority: Critical
FR-1.3: The system shall use separate worker threads for each model request to prevent UI blocking
Priority: High
FR-1.4: The system shall display both responses in separate, clearly labeled panels
Priority: High
FR-1.5: The system shall track and display response time for each model
Priority: Medium
FR-1.6: The system shall display confidence scores if provided by the model
Priority: Low
FR-1.7: The system shall handle cases where one model succeeds and the other fails gracefully
Priority: High
FR-1.8: The system shall allow users to save responses from either model independently
Priority: Medium

4.2 Image Attachment and Processing

4.2.1 Description and Priority

Priority: Critical
Description: The system shall accept chart images from users, display a preview, and transmit them to model servers for analysis.

4.2.2 Stimulus/Response Sequences

1. User clicks "Attach image" button
2. System opens file selection dialog
3. User selects an image file
4. System validates file format
5. System displays scaled preview
6. System converts image to JPEG format in memory
7. System includes image in API requests

4.2.3 Functional Requirements

FR-2.1: The system shall accept image files in PNG, JPG, JPEG, and BMP formats
Priority: Critical
FR-2.2: The system shall display a scaled preview of the attached image (360x270 pixels) while maintaining aspect ratio
Priority: High
FR-2.3: The system shall convert attached images to JPEG format for transmission
Priority: High
FR-2.4: The system shall provide a "Clear image" button to remove the attached image
Priority: Medium
FR-2.5: The system shall validate image files before accepting them
Priority: High
FR-2.6: The system shall display an error message if an invalid image file is selected
Priority: Medium
FR-2.7: The system shall require an image to be attached before allowing submission
Priority: Critical
FR-2.8: The system shall store image data in memory as byte arrays for API transmission
Priority: High

4.3 Question Input and Natural Language Processing

4.3.1 Description and Priority

Priority: Critical
Description: The system shall accept natural language questions from users about chart content and process them through various prompting strategies.

4.3.2 Stimulus/Response Sequences

1. User types a question in the text area
2. System validates non-empty input
3. System applies appropriate prompting strategy
4. System sends formatted prompt to model(s)
5. System receives and displays answer
6. System extracts final answer using pattern matching

4.3.3 Functional Requirements

FR-3.1: The system shall provide a multi-line text input area for questions
Priority: Critical
FR-3.2: The system shall validate that a question is entered before submission
Priority: High
FR-3.3: The system shall support questions of any reasonable length (up to 1000 characters)
Priority: Medium
FR-3.4: The system shall automatically select the appropriate strategy based on server configuration
Priority: Medium
FR-3.5: The system shall extract final answers from verbose model responses using pattern matching
Priority: High
FR-3.6: The system shall handle JSON-formatted responses and extract answer fields
Priority: Medium

4.4 Chart Data Extraction Pipeline

4.4.1 Description and Priority

Priority: High
Description: The system shall extract tabular data from chart images using dedicated extraction models (PaddlePaddle or DePlot) before passing to the main LLM.

4.4.2 Stimulus/Response Sequences

1. Model server receives chart image
2. System invokes extraction model (PaddlePaddle or DePlot)
3. Extraction model returns raw tabular data
4. System cleans and formats extracted data
5. System constructs enhanced prompt with table data
6. System passes prompt + image to Qwen model
7. Qwen generates answer using both visual and tabular information

4.4.3 Functional Requirements

FR-4.1: The pipelined server shall extract tabular data using PaddlePaddle PP-Chart2Table
Priority: High
FR-4.2: The alternative pipeline shall extract data using Google DePlot
Priority: High
FR-4.3: The system shall clean extracted data by:

- Replacing hex tokens (0x0A) with newlines
- Replacing hex tabs (0x09) with pipes
- Normalizing spacing
- Removing empty lines

Priority: Critical
FR-4.4: The system shall format cleaned table data in a readable structure for LLM consumption
Priority: High
FR-4.5: The system shall combine visual chart data with extracted tabular data in prompts
Priority: High
FR-4.6: The system shall handle extraction failures gracefully and log errors
Priority: High

4.5 Response Processing and Display

4.5.1 Description and Priority

Priority: High
Description: The system shall process model responses, format them appropriately, and display them to users with relevant metadata.

4.5.2 Functional Requirements

FR-5.1: The system shall remove escape sequences from responses for better readability
Priority: Medium
FR-5.2: The system shall remove markdown formatting symbols (backslashes, asterisks) from displayed answers
Priority: Low

FR-5.3: The system shall display raw JSON responses when requested by users
Priority: Medium
FR-5.4: The system shall allow users to save responses as text files
Priority: Medium
FR-5.5: The system shall allow users to save raw JSON responses
Priority: Low
FR-5.6: The system shall display error messages clearly when requests fail
Priority: High
FR-5.7: The system shall show progress indicators during processing
Priority: Medium

4.6 Evaluation and Benchmarking

4.6.1 Description and Priority

Priority: High
Description: The system shall evaluate model performance on standard benchmarks and generate comprehensive reports.

4.6.2 Functional Requirements

FR-6.1: The system shall load benchmark datasets (ChartQA, PlotQA)
Priority: High
FR-6.2: The system shall evaluate answers against ground truth with 5% relative tolerance
Priority: Critical
FR-6.3: The system shall handle multiple ground truth formats (single values, lists)
Priority: High
FR-6.4: The system shall normalize answers for comparison:

- Word-to-digit conversion
- Currency symbol removal
- Percentage format normalization
- Case-insensitive matching

Priority: High
FR-6.5: The system shall calculate accuracy metrics per strategy
Priority: High
FR-6.6: The system shall track average inference time per strategy
Priority: Medium
FR-6.7: The system shall save detailed results in JSON format
Priority: High
FR-6.8: The system shall generate summary reports comparing all strategies
Priority: Medium

5 Other Nonfunctional Requirements

5.1 Performance Requirements

NFR-1.1: The base model server shall process single chart queries in under 10 seconds on recommended hardware
Priority: High

NFR-1.2: The pipelined server (Chart2Table + Qwen) shall complete extraction and inference in under 15 seconds
Priority: High

NFR-1.3: The PaddlePaddle extraction server shall extract tables in under 5 seconds
Priority: Medium

NFR-1.4: The client GUI shall remain responsive during all operations (no UI freezing)
Priority: Critical

NFR-1.5: The system shall support concurrent requests from multiple clients without significant performance degradation
Priority: Medium

NFR-1.6: GPU memory usage shall not exceed 80% to prevent out-of-memory errors
Priority: High

NFR-1.7: The system shall handle images up to 10MB in size without timeout errors
Priority: Medium

NFR-1.8: Network communication timeout shall be configurable (default 60 seconds)
Priority: Low

5.2 Safety Requirements

NFR-2.1: The system shall validate all input files before processing to prevent malicious file execution
Priority: High

NFR-2.2: The system shall limit maximum file upload size to prevent denial-of-service attacks
Priority: High

NFR-2.3: The system shall handle GPU out-of-memory errors gracefully without crashing
Priority: High

NFR-2.4: The system shall implement request timeouts to prevent indefinite hanging
Priority: High

NFR-2.5: The system shall log all errors with stack traces for debugging purposes
Priority: Medium

NFR-2.6: The system shall not expose sensitive server information in error messages
Priority: Medium

5.3 Security Requirements

NFR-3.1: The system shall validate and sanitize all user inputs before processing
Priority: High

NFR-3.2: The system shall not store uploaded images permanently on servers
Priority: Medium

NFR-3.3: The system shall use temporary files with automatic cleanup for image processing
Priority: Medium

NFR-3.4: The system shall not log sensitive user data or API keys
Priority: High

NFR-3.5: Network communication should use HTTPS
Priority: Low

5.4 Software Quality Attributes

5.4.1 Reliability

NFR-4.1: The system shall maintain 95% uptime during evaluation periods
Priority: Medium

NFR-4.2: The system shall recover gracefully from individual request failures
Priority: High
NFR-4.3: The system shall continue operating if one server component fails (degraded mode)
Priority: Medium

5.4.2 Maintainability

NFR-4.4: The system code shall follow PEP 8 Python style guidelines
Priority: Medium
NFR-4.5: All major functions shall include docstrings with parameter descriptions
Priority: Medium
NFR-4.6: The system architecture shall be modular to facilitate component replacement
Priority: High
NFR-4.7: Configuration parameters shall be clearly documented and easily modifiable
Priority: High

5.4.3 Portability

NFR-4.8: The client application shall run on Windows, macOS, and Linux
Priority: High
NFR-4.9: The server components shall be deployable on any Linux distribution with GPU support
Priority: High
NFR-4.10: The system shall use standard Python packaging (pip) for dependencies
Priority: Medium

5.4.4 Usability

NFR-4.11: The GUI shall be intuitive enough for users with basic computer skills
Priority: High
NFR-4.12: Error messages shall be clear and actionable
Priority: High
NFR-4.13: The system shall provide visual feedback for all long-running operations
Priority: High
NFR-4.14: Default configurations shall work out-of-the-box for standard deployments
Priority: Medium

5.4.5 Scalability

NFR-4.15: The system architecture shall support horizontal scaling by adding more server instances
Priority: Low
NFR-4.16: The system shall handle benchmark datasets with thousands of examples
Priority: High

5.5 Project Documentation

NFR-5.1: The project shall include a comprehensive README with:

- System overview and architecture
- Installation instructions
- Configuration guide
- Usage examples
- Troubleshooting section

Priority: High
NFR-5.2: The project shall include API documentation for all server endpoints
Priority: Medium
NFR-5.3: The project shall include architecture diagrams showing component interactions
Priority: Medium

NFR-5.4: The project shall document all prompting strategies with examples
Priority: High
NFR-5.5: The project shall include benchmark results and evaluation methodology
Priority: High

5.6 User Documentation

NFR-6.1: User guide shall include step-by-step instructions for:

- System setup
- Running servers
- Using the client application
- Interpreting results

Priority: High

6 Other Requirements

6.1 Database Requirements

OR-1: The system does not require a persistent database. All data is processed in-memory or stored as JSON files.
Priority: N/A

6.2 Internationalization Requirements

OR-2: The current version supports English language only. Future versions may support multilingual chart analysis.
Priority: Low

6.3 Legal and Compliance Requirements

OR-3: The system shall comply with model licenses:

- Qwen2.5-VL: Apache 2.0 License
- PaddlePaddle: Apache 2.0 License
- DePlot: Apache 2.0 License

Priority: Critical
OR-4: The system shall include appropriate attribution for all pre-trained models and datasets
Priority: High

6.4 Reuse and Third-Party Requirements

OR-5: The system shall use only open-source libraries with permissive licenses
Priority: High
OR-6: The system shall not include proprietary code or models
Priority: Critical

6.5 Future Enhancements

OR-7: Future versions may include:

- Batch processing capabilities
- Web-based interface
- Real-time streaming responses
- Support for additional chart types (3D, heatmaps)
- Integration with business intelligence tools
- Fine-tuning capabilities for domain-specific charts

Priority: Low

Appendix A: Terminology/Glossary/Definition List

API (Application Programming Interface) A set of protocols and tools for building software applications and enabling communication between systems.

Base64 Encoding A binary-to-text encoding scheme used to transmit binary data over text-based protocols.

Benchmark Dataset A standardized dataset used to evaluate and compare model performance.

Chain-of-Models An approach where multiple specialized models are chained together, with the output of one serving as input to another.

Chain-of-Thought (CoT) A prompting technique that encourages models to show step-by-step reasoning.

Chart2Table The process of extracting structured tabular data from chart images.

ChartQA A benchmark dataset containing chart images and question-answer pairs for evaluating chart understanding LLM systems.

CUDA (Compute Unified Device Architecture) NVIDIA's parallel computing platform for GPU programming.

DePlot A model by Google designed to convert chart images into linearized table representations.

FastAPI A modern Python web framework for building APIs with automatic documentation.

Few-Shot Learning A technique where models learn from a small number of examples provided in the prompt.

GPU (Graphics Processing Unit) Specialized hardware for parallel processing, essential for deep learning inference.

Inference The process of using a trained model to make predictions on new data.

LLM (Large Language Model) Deep learning models trained on vast amounts of text data for natural language tasks.

Multimodal Model AI models that can process and understand multiple types of data (text, images, etc.).

PaddlePaddle An open-source deep learning platform developed by Baidu.

PlotQA A dataset designed for question answering over plot images.

PP-Chart2Table PaddlePaddle's model for extracting tables from chart images.

Prompting Strategy A technique for formulating inputs to language models to elicit desired responses.

PyQt6 Python bindings for the Qt GUI framework, used for building desktop applications.

Qwen2.5-VL A vision-language model by Alibaba Cloud capable of understanding both images and text.

REST (Representational State Transfer) An architectural style for designing networked applications using HTTP.

Structured Output A prompting technique that requests responses in specific formats like JSON.

TF32 A numerical format for GPU computing that balances performance and precision.

Uvicorn A fast ASGI server for Python web applications.

Vision-Language Model (VLM) AI models trained to understand and generate content based on both visual and textual inputs.

Zero-Shot Learning The ability of models to perform tasks without specific training examples.

Appendix B: To Be Determined

The following items require further specification and will be addressed in future document revisions:

B.1 Performance Benchmarks

Detailed performance metrics on specific hardware configurations:

- Exact inference times on RTX 3090, 4090, A100
- Memory consumption profiles
- Throughput measurements for concurrent requests
- Network latency impact analysis

B.2 Deployment Scenarios

Specific deployment configurations:

- Docker containerization specifications
- Kubernetes orchestration details
- Cloud deployment guidelines (AWS, Azure, GCP)
- Load balancing strategies

B.3 Extended Evaluation Metrics

Additional evaluation criteria:

- Chart type-specific accuracy breakdowns
- Question complexity categories and performance
- Error analysis taxonomy
- Robustness testing results

B.4 Integration Specifications

Details for system integration:

- Integration with data visualization tools
- Export formats for results
- Batch processing API specifications
- Webhook support for async processing

B.5 Security Hardening

Production security requirements:

- Authentication mechanisms (OAuth2, JWT)
- Rate limiting specifications
- Input validation rules
- Audit logging requirements
- Penetration testing results

B.6 User Study Results

Usability evaluation data:

- User satisfaction surveys
- Task completion rates
- Common user errors and confusion points
- Suggested UI improvements

B.7 Model Update Procedures

Protocols for updating system components:

- Model versioning strategy
- Backward compatibility requirements
- A/B testing framework
- Rollback procedures

B.8 Licensing and Distribution

Final decisions on:

- Software license selection
- Third-party license compliance verification
- Distribution platforms
- Commercial use considerations

Note: Items in this appendix will be specified and moved to appropriate sections as the project progresses and requirements are finalized.