🎨 Pixel Art Agent

System Architecture & Integration Documentation

Generated: November 22, 2025

Table of Contents

1. System Overview

The Pixel Art Agent is an AI-powered system that generates detailed pixel art specifications and actual sprite images for video game development. It combines natural language processing through Ollama with advanced image generation via Stable Diffusion to create both textual descriptions and visual assets.

🍃 Spring Boot 3.3.5

Core application framework providing REST APIs and dependency injection.

🤖 Ollama (Qwen 2.5:3b)

Local LLM for generating detailed pixel art specifications and descriptions.

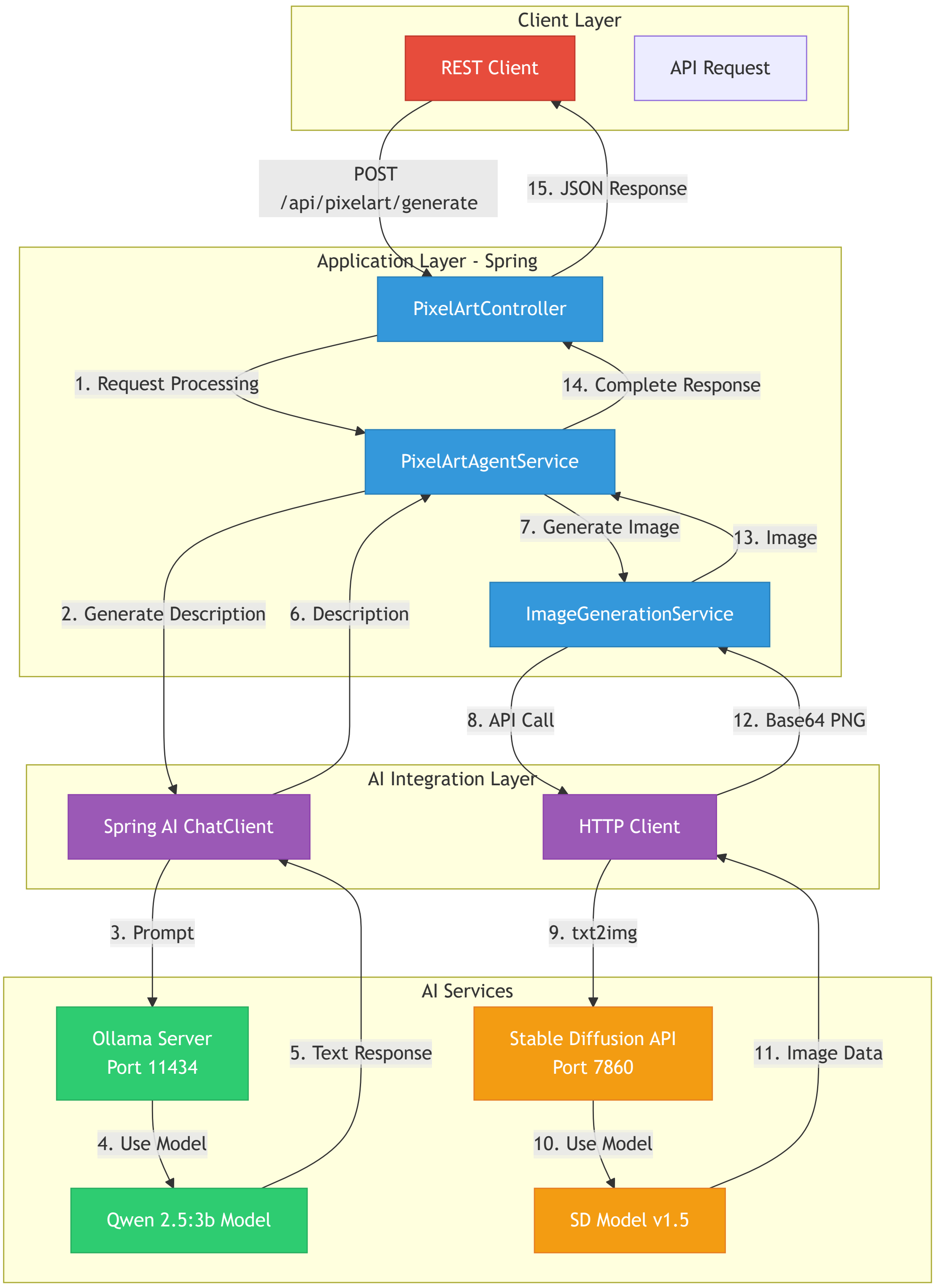
🎨 Stable Diffusion

Image generation model for creating actual pixel art sprites from descriptions.

🔧 Spring AI 1.0.0-M3

Abstraction layer for AI model integration with Spring Boot.

2. High-Level Architecture



3. Component Details

3.1 PixelArtController

Purpose: REST API endpoint layer that handles HTTP requests and responses.

Endpoints:

- `POST /api/pixelart/generate` - Generate complete pixel art specification with image
- `POST /api/pixelart/generate/image` - Generate only the image asset
- `POST /api/pixelart/variations` - Generate multiple variations
- `POST /api/pixelart/refine` - Refine based on feedback

Responsibilities:

- Request validation and sanitization
- Response formatting (JSON)
- Error handling and HTTP status codes
- CORS configuration for cross-origin requests

3.2 PixelArtAgentService

Purpose: Core business logic orchestrating AI interactions and response generation.

Key Capabilities:

- **Prompt Engineering:** Constructs detailed prompts for the Qwen model with specific technical requirements
- **Response Parsing:** Extracts structured data from LLM responses (colors, animations, specifications)
- **Spritesheet Detection:** Automatically determines if multiple frames are needed based on context
- **Image Orchestration:** Coordinates between description generation and image creation

```
 // Example prompt template
You are an expert pixel art and sprite design consultant.

Generate detailed technical description for:
- Asset Type: CHARACTER
- Description: brave knight
- Style: 16-bit pixel art
- Size: 32x32

Provide:
1. Visual description (shapes, proportions)
2. Color palette (5-8 hex codes)
3. Technical specs (dimensions, layers)
4. Animation suggestions
5. Special effects
```

3.3 ImageGenerationService

Purpose: Manages communication with Stable Diffusion for actual image generation.

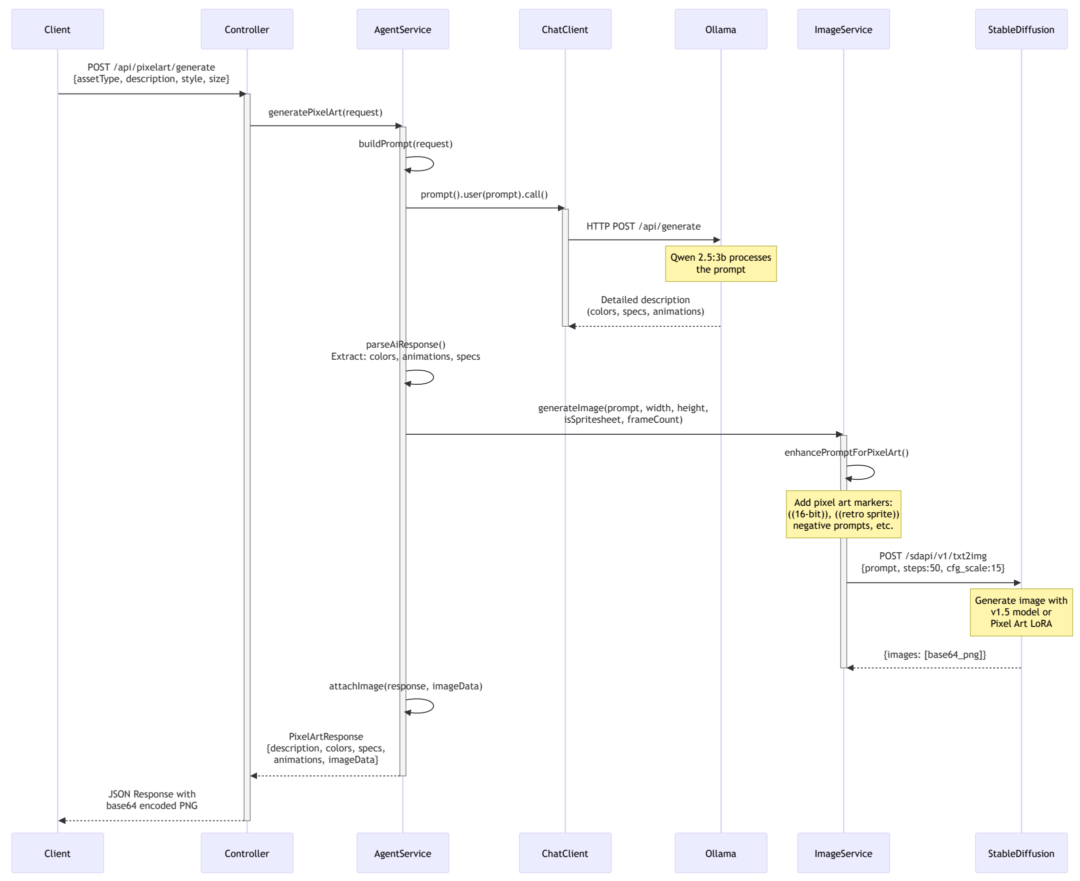
Features:

- **Prompt Enhancement:** Adds pixel art style markers and constraints
- **LoRA Support:** Can integrate specialized pixel art LoRA models
- **Spritesheet Generation:** Creates horizontal sprite strips with multiple frames
- **Fallback Handling:** Provides placeholder images if SD is unavailable

Image Generation Parameters:

- Steps: 50 (higher for better adherence to pixel art style)
- CFG Scale: 15 (maximum guidance for style consistency)
- Sampler: Euler a
- Resolution: 8x upscaled then downscaled for pixel effect

4. Data Flow & Integration

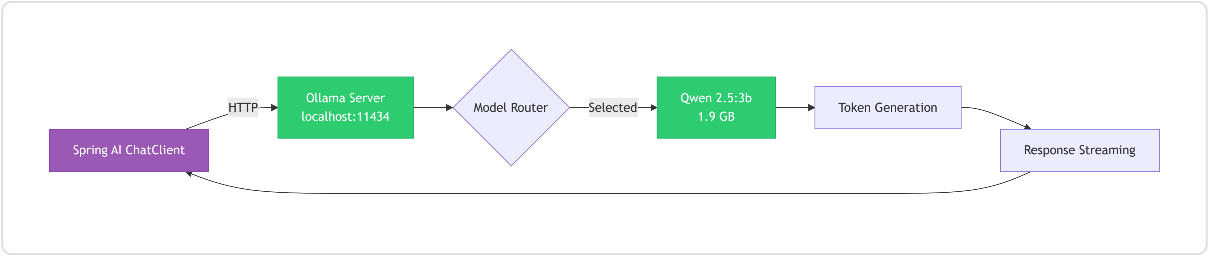

Client
Controller
AgentService
ChatClient
Ollama
ImageService
StableDiffusion
POST /api/pixelart/generate {assetType, description, style, size}
generatePixelArt(request)
buildPrompt(request)
prompt().user(prompt).call()
HTTP POST /api/generate
Qwen 2.5:3b processes the prompt
Detailed description (colors, specs, animations)
parseAiResponse() Extract: colors, animations, specs
generateImage(prompt, width, height, isSpritesheet, frameCount)
enhancePromptForPixelArt()
Add pixel art markers: ((16-bit)), ((retro sprite)) negative prompts, etc.
POST /sdapi/v1/txt2img {prompt, steps:50, cfg_scale:15}
Generate image with v1.5 model or Pixel Art LoRA
{images: [base64_png]}
attachImage(response, imageData)
PixelArtResponse {description, colors, specs, animations, imageData}
JSON Response with base64 encoded PNG
Client
Controller
AgentService
ChatClient
Ollama
ImageService
StableDiffusion

Flow Description:

1. **Request Reception:** Client sends JSON request with asset requirements
2. **Prompt Construction:** Service builds detailed prompt with technical specifications
3. **LLM Processing:** Ollama/Qwen generates comprehensive pixel art description
4. **Response Parsing:** Extract structured data (hex colors, animation types, dimensions)
5. **Image Prompt Enhancement:** Add extreme pixel art style constraints and negative prompts
6. **Image Generation:** Stable Diffusion creates actual sprite with specified parameters
7. **Response Assembly:** Combine text description with base64-encoded image
8. **Delivery:** Return complete JSON response to client

5. AI Model Integration

5.1 Ollama Integration (Text Generation)

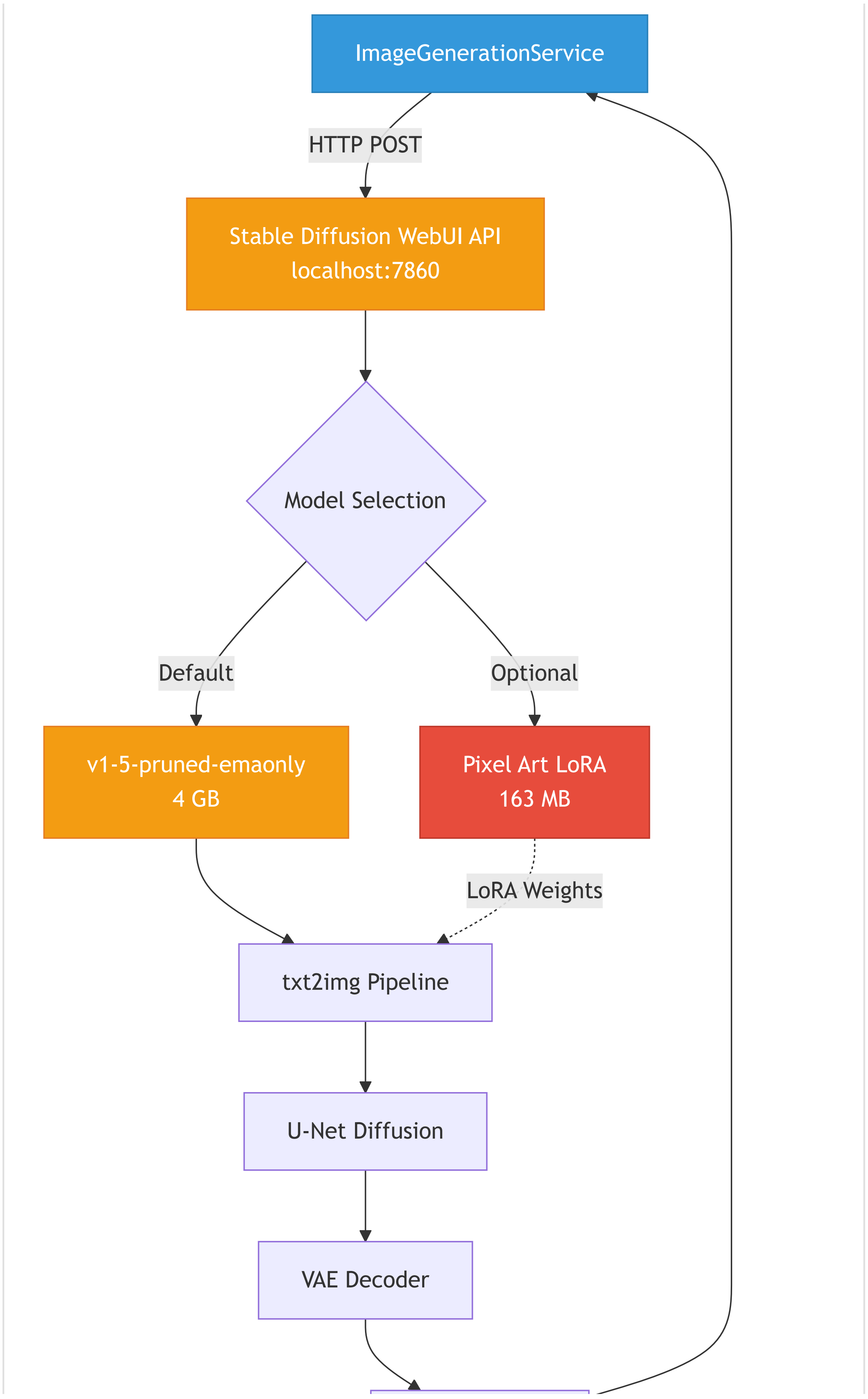


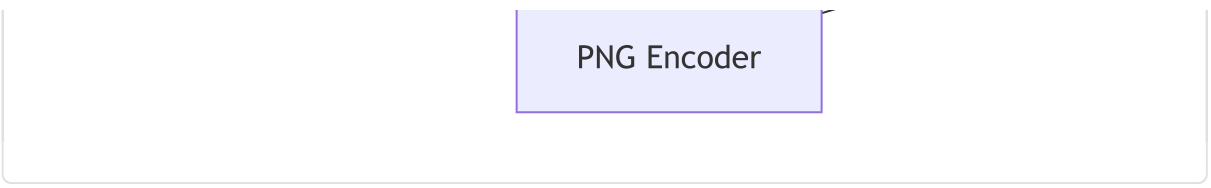
Qwen 2.5:3b Model Characteristics

Property	Value	Description
Model Size	1.9 GB	Quantized version for efficient local execution
Parameters	3 billion	Balanced for quality and performance
Context Window	32K tokens	Can handle detailed prompts and responses
Temperature	0.8	Balanced creativity vs consistency
Top-P	0.9	Nucleus sampling for varied but coherent output
Use Case	Technical Writing	Generates precise pixel art specifications

⚡ **Performance Note:** Qwen 2.5:3b processes requests in 2-5 seconds on modern hardware, making it suitable for real-time API responses. The model excels at structured technical descriptions and understands pixel art terminology.

5.2 Stable Diffusion Integration (Image Generation)


ImageGenerationService
HTTP POST
Stable Diffusion WebUI API
localhost:7860
Model Selection
Default
Optional
v1-5-pruned-emaonly
4 GB
Pixel Art LoRA
163 MB
LoRA Weights
txt2img Pipeline
U-Net Diffusion
VAE Decoder



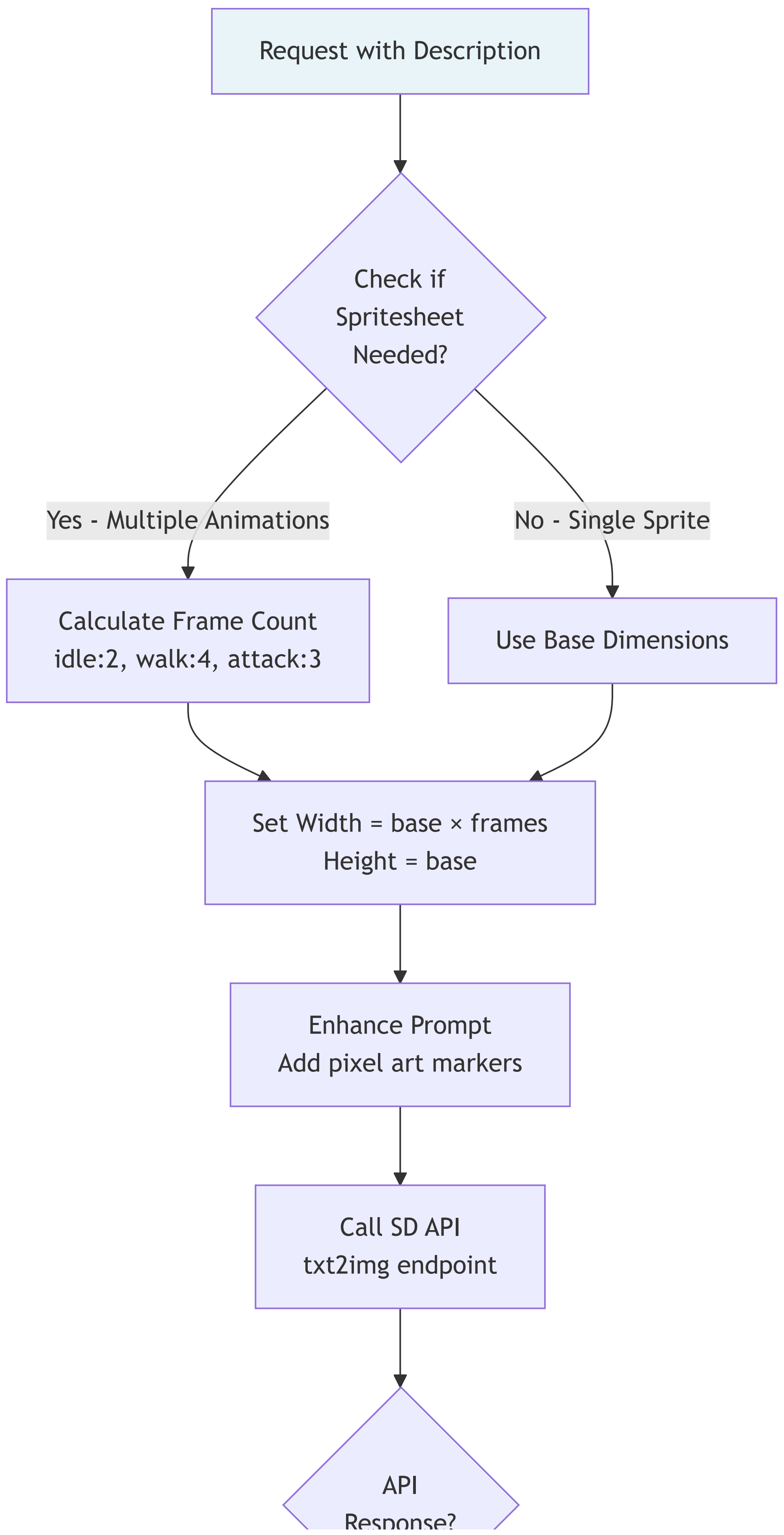
Stable Diffusion Configuration

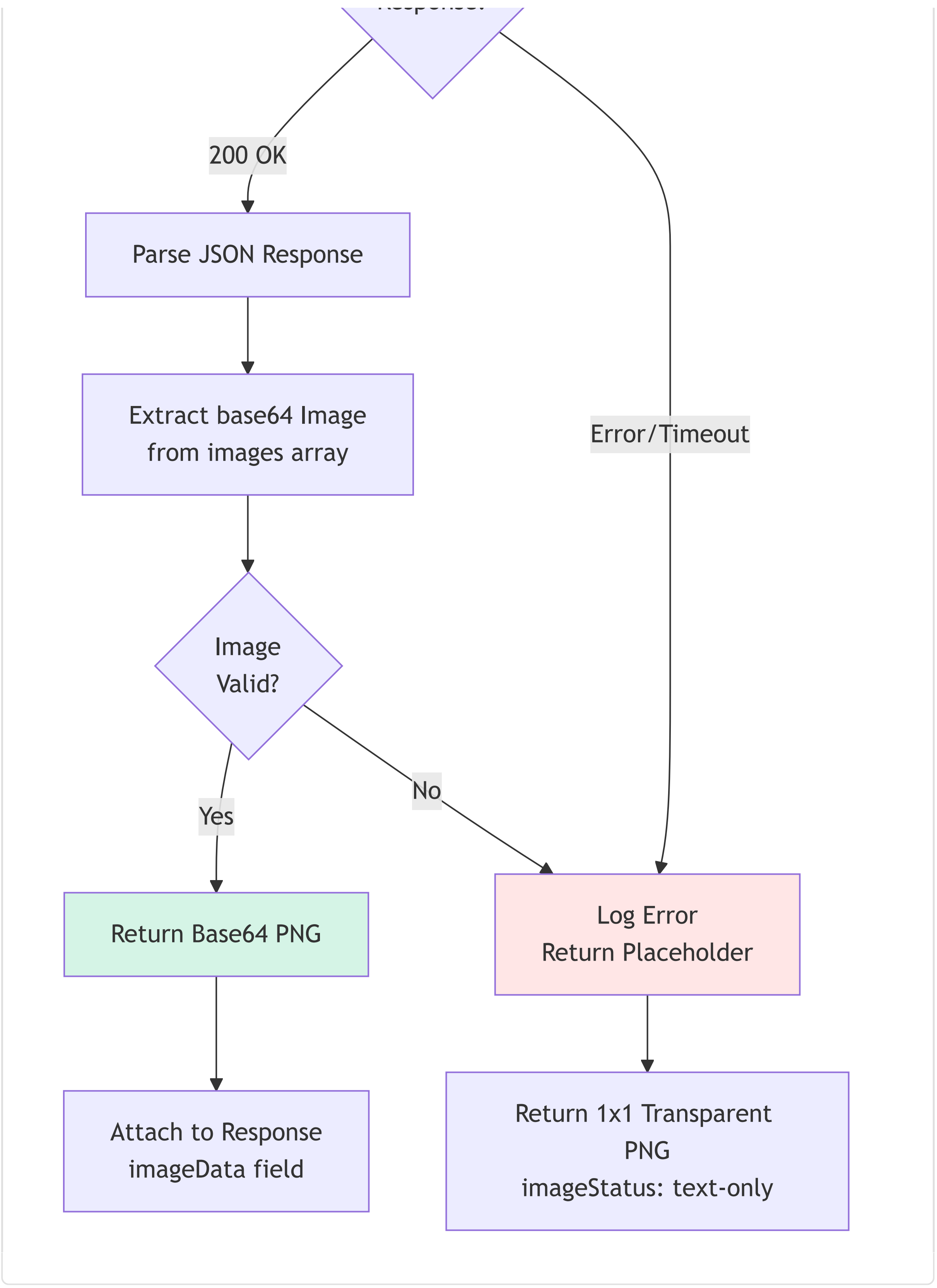
Parameter	Value	Purpose
Base Model	v1-5-pruned-emaonly	Stable Diffusion 1.5 optimized weights
Steps	50	More iterations for style adherence
CFG Scale	15	Maximum prompt guidance
Sampler	Euler a	Good for pixel art generation
Resolution	8x requested	Generate high then downscale for pixel effect
Denoising	0.4	Control randomness in generation
CPU Flags	--no-half --precision full --disable-nan-check	CPU mode optimizations

5.3 Prompt Enhancement for Pixel Art

```
 // Enhanced prompt structure
StringBuilder enhancedPrompt = new StringBuilder();

// LoRA activation (if configured)
if (loraModel != null) {
    enhancedPrompt.append(" ");
}

// STRONG pixel art style markers
enhancedPrompt.append("((pixel art)), ((16-bit)), ((retro game sprit
enhancedPrompt.append("((NES style)), ((SNES style)), ");

// User description
enhancedPrompt.append(userDescription);

// EXTREME quality constraints
enhancedPrompt.append(", ((8-bit graphics)), ((low resolution)), ");
enhancedPrompt.append("((chunky pixels)), ((limited colors)), ");
enhancedPrompt.append("((no anti-aliasing)), ((no gradients)), ");

// Reference games for style
enhancedPrompt.append("Super Nintendo, Pokemon Red Blue style, ");
enhancedPrompt.append("Final Fantasy 6 sprite, Chrono Trigger sprite

// Negative prompt
negative = "blurry, smooth, realistic, 3d render, gradients,
           anti-aliasing, detailed textures, soft edges";
```

6. Image Generation Pipeline


Request with Description
Check if Spritesheet Needed?
Yes - Multiple Animations
No - Single Sprite
Calculate Frame Count idle:2, walk:4, attack:3
Use Base Dimensions
Set Width = base × frames Height = base
Enhance Prompt Add pixel art markers
Call SD API txt2img endpoint
API Response?


200 OK
Parse JSON Response
Extract base64 Image from images array
Error/Timeout
Image Valid?
Yes
No
Return Base64 PNG
Log Error Return Placeholder
Attach to Response imageData field
Return 1x1 Transparent PNG imageStatus: text-only

Pipeline Stages:

1. **Spritesheet Detection:** Analyze request context and animation suggestions to determine if multiple frames needed
2. **Frame Calculation:** Assign typical frame counts per animation type (idle: 2, walk: 4, jump: 3, attack: 3)
3. **Dimension Adjustment:** For spritesheets, multiply width by frame count for horizontal layout
4. **Prompt Enhancement:** Add ((pixel art)), ((16-bit)) markers and strong negative prompts
5. **API Communication:** POST to /sdapi/v1/txt2img with full configuration
6. **Response Handling:** Parse JSON, extract base64 image from images array
7. **Error Recovery:** Return placeholder on failures, never crash the request
8. **Integration:** Attach base64 PNG to main response object

7. Configuration Guide

7.1 Application Properties

```
 # application.properties

# Server Configuration
server.port=8080
spring.application.name=pixel-art-agent

# Ollama Configuration
spring.ai.ollama.base-url=http://localhost:11434
spring.ai.ollama.chat.options.model=qwen2.5:3b
spring.ai.ollama.chat.options.temperature=0.8
spring.ai.ollama.chat.options.top-p=0.9

# Logging
logging.level.com.pixelart.agent=DEBUG
logging.level.org.springframework.ai=DEBUG

# Agent Settings
pixelart.agent.max-iterations=3
pixelart.agent.default-style=pixel-art

# Image Generation Settings
pixelart.image.generation.enabled=true
pixelart.image.generation.api-url=http://localhost:7860
pixelart.image.generation.model=stable-diffusion

# Optional: LoRA Model
pixelart.image.generation.lora=pixel-art-xl
pixelart.image.generation.lora-strength=0.8
```

7.2 Docker Container Configuration

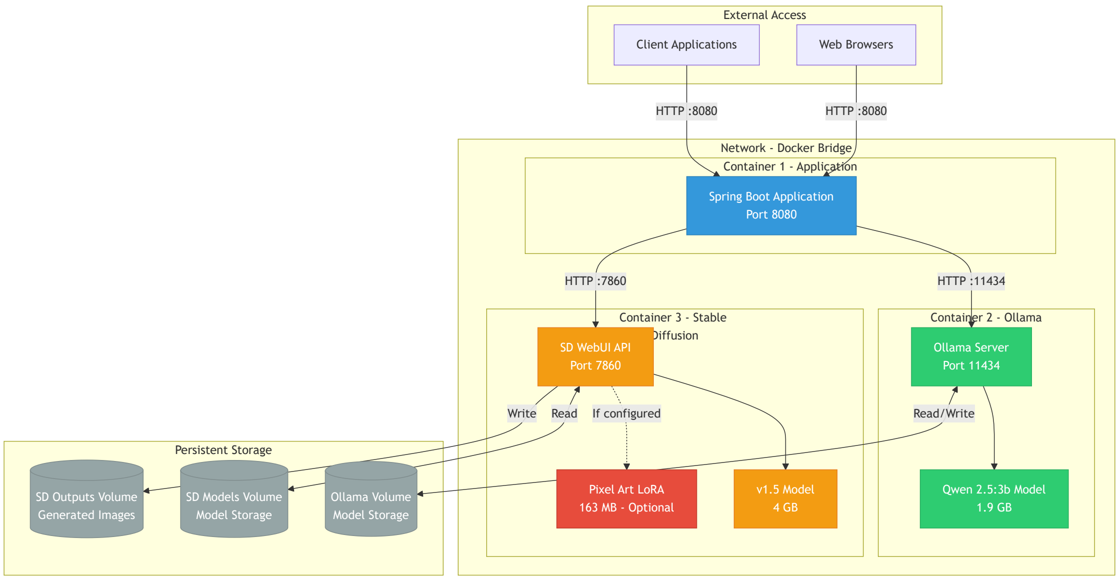
Ollama Container

```
 docker run -d --name ollama \
  -p 11434:11434 \
  -v ollama-data:/root/.ollama \
  ollama/ollama

# Pull the Qwen model
docker exec ollama ollama pull qwen2.5:3b
```

Stable Diffusion Container

```
 docker run -d --name stable-diffusion \
  -p 7860:7860 \
  -v sd-models:/data/StableDiffusion \
  -v sd-outputs:/data/outputs \
  -e CLI_ARGS="--api --listen --port 7860 \
    --no-half --precision full \
    --skip-torch-cuda-test \
    --disable-nan-check" \
  ghcr.io/neggles/sd-webui-docker:main
```

8. Deployment Architecture



System Requirements

Component	RAM	Storage	CPU
Spring Boot Application	1-2 GB	100 MB	2 cores recommended
Ollama + Qwen 2.5:3b	4-6 GB	2 GB	4 cores recommended
Stable Diffusion (CPU)	8-16 GB	6-8 GB	8+ cores (generation is slow)
Total Minimum	**16 GB**	**10 GB**	**8 cores**

🚀 **Performance Tip:** For production deployments, consider using GPU acceleration for Stable Diffusion. This can reduce image generation time from 30-60 seconds (CPU) to 2-5 seconds (GPU). Ollama also benefits from GPU acceleration, reducing response time from 3-5 seconds to under 1 second.

9. API Examples

9.1 Generate Pixel Art with Image

```
 POST http://localhost:8080/api/pixelart/generate
Content-Type: application/json

{
  "assetType": "CHARACTER",
  "description": "brave knight with sword and shield",
  "style": "16-bit pixel art",
  "colorPalette": "medieval fantasy",
  "size": "32x32",
  "additionalContext": "animation frames for walk cycle"
}

Response:
{
  "detailedDescription": "### Character Asset Specification...",
  "suggestedColors": ["#2C3E50", "#E74C3C", "#ECF0F1", "#3498DB"],
  "specifications": {
    "size": "32x32",
    "assetType": "CHARACTER",
    "frameCount": 4,
    "orientation": "front-facing",
    "layers": ["background", "base", "details", "highlights"]
  },
  "animationSuggestions": ["idle", "walk"],
  "style": "16-bit pixel art",
  "imageData": "iVBORw0KGgoAAAANSUhEUgAA...[base64]",
  "imageStatus": "spritesheet-generated",
  "generatedAt": "2025-11-22T14:00:00"
}
```

9.2 Generate Image Only

```
 POST http://localhost:8080/api/pixelart/generate/image
Content-Type: application/json

{
  "assetType": "ITEM",
  "description": "health potion bottle",
  "style": "pixel art",
  "size": "16x16"
}

Response: Binary PNG image file
```

10. Troubleshooting

Common Issues and Solutions

Issue: Ollama Connection Refused

Symptom: Error connecting to localhost:11434

Solution:

- Verify Ollama container is running: `docker ps | grep ollama`
- Check port mapping: `docker port ollama`
- Ensure model is pulled: `docker exec ollama ollama list`

Issue: Stable Diffusion Returns Placeholder

Symptom: imageStatus is "text-only", 70-byte images

Solution:

- Check SD container logs: `docker logs stable-diffusion --tail 50`
- Verify API is accessible: `curl http://localhost:7860/sdapi/v1/sd-models`
- Check application logs for "ERROR" messages related to Stable Diffusion
- Ensure --disable-nan-check flag is set for CPU mode

Issue: Out of Memory

Symptom: Container crashes or system freezes

Solution:

- Reduce Qwen model size to smaller variant (1.8b instead of 3b)
- Limit concurrent requests with rate limiting
- Increase Docker memory limits in Docker Desktop settings
- Consider using GPU for better memory management

11. Future Enhancements

- ✅ LoRA model support for specialized pixel art styles
- ✅ Spritesheet generation with multiple frames
- 🔄 GPU acceleration for faster image generation
- 🔄 Persistent volume configuration for model storage
- 📋 Animation sequence generation (idle → walk → attack)
- 📋 Color palette extraction from reference images
- 📋 Batch generation API for multiple assets
- 📋 WebSocket support for real-time generation progress
- 📋 Integration with game engines (Unity, Godot)