# VANTAGE

# Software Requirements Specification

## Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 04/07/2026 | 1.0 | Initial SRS draft aligned to the approved team template and tailored to the implemented VANTAGE platform. | Project Documentation Team |

## Table of Contents

# 1. Introduction

## 1.1 The Purpose of VANTAGE

VANTAGE (Visual Autonomous Navigation and Task-driven Agentic Ground-to-air Engine) is an intelligent drone-control platform that combines autonomous flight support, multimodal perception, and agentic reasoning. The System enables an operator to issue natural-language commands through text or voice and then translates those commands into coordinated drone actions, perception tasks, and simulation-supported mission steps. VANTAGE is intended to reduce the technical barrier to drone-assisted inspection, search, survey, and analysis workflows while improving safety through simulation-first validation and fail-safe flight controls.

## 1.2 The Purpose of this Document

This Software Requirements Specification (SRS) defines the functional and non-functional requirements for VANTAGE. It is intended for stakeholders, developers, testers, instructors, and reviewers who need a single authoritative description of the System's expected behavior, architectural boundaries, interfaces, operational constraints, safety requirements, and deployment expectations. This document also serves as a baseline for validation, demonstration, and future maintenance.

## 1.3 Document Conventions

- "The System" refers to the VANTAGE platform as a whole.
- Requirements are written in verifiable "shall" statements wherever practical.
- Requirement identifiers use the format <Category>-<Number> such as FR-1, SEC-2, and SAF-3.
- "Real drone mode" refers to operation against a DJI Tello drone.
- "Simulation mode" or "digital twin mode" refers to Gazebo/ROS2-based dry-run execution.

### 1.4 References

- Vision and Scope.pdf
- Project Glossary.pdf
- Use-cases.pdf
- Development Plan.pdf

## 2. Project Glossary

It defines core domain and platform terminology used throughout this SRS, including AI Agent, MCP, ROS2, ROSBridge, Gazebo, VoiceVision, VQA, and related VANTAGE-specific concepts.

## 3. Vision and Scope

The high-level business vision, product scope, subsystem breakdown, and stated limitations are maintained in Vision and Scope.pdf. At a high level, VANTAGE delivers a unified multimodal platform for drone control, speech and vision processing, and simulation-backed mission execution. The platform combines agentic mission planning, real-time perception tools, dashboard-driven media analysis, and a ROS2/Gazebo digital twin so that missions can be validated safely before or instead of hardware flight.

## 4. Software Architecture

### 4.1 System Context Diagram

At the highest level, the System sits between the operator and a set of execution, perception, and reasoning services. Operators interact with VANTAGE through a web mission console, a tool dashboard, or command-line workflows using text and optional voice input. VANTAGE then coordinates an LLM-backed mission agent, perception models, the DJI Tello control layer, and the Gazebo/ROS2 digital twin. External dependencies include OpenAI- or Ollama-compatible model providers, the local filesystem for generated artifacts, ROSBridge for simulation command transport, and the DJI Tello hardware SDK in real-drone mode.

### 4.2 Container Diagram

The current implementation is composed of the following major logical containers:

1. Frontend Web Client
   A Vue 3 and Vite application that provides:
   - A mission-control page with live video, telemetry, mission chat, and emergency land control.
   - A dashboard page for direct use of speech, vision, segmentation, and object-detection tools.
2. Unified API Server
   A FastAPI application that:

    - Serves the frontend build.
    - Exposes REST routes for status, emergency landing, tool invocation, and dashboard workflows.
    - Exposes WebSocket routes for live video and mission chat.
    - Hosts runtime coordination for the mission agent and model execution.
3. Mission Agent Runtime
   A smolagents tool-calling agent that:
    - Uses LiteLLM-compatible backends.
    - Combines local drone-control tools with MCP-hosted perception tools.
    - Executes multi-step missions and reports progress to the UI.
4. MCP Tool Server
   A local MCP server that exposes perception tools including speech-to-text, text-to-speech, captioning, visual question answering, segmentation, and object detection.
5. Model Runtime Layer
   A Python runtime that loads and caches models for:
    - Whisper speech-to-text
    - WhisperSpeech text-to-speech
    - BLIP image captioning
    - BLIP or InternVL visual question answering
    - YOLO-based object detection
    - DeepLabV3 semantic segmentation
6. Drone Control Layer
   A thread-safe DJI Tello controller that manages connection state, telemetry, takeoff, landing, rotation, linear movement, frame capture, and emergency landing.
7. Simulation Stack
   A ROS2 and Gazebo digital twin, with ROSBridge and optional VNC access, used for dry-run flight validation and simulation-based operation.
8. File Storage Layer
   Local filesystem directories used for uploaded media, captured frames, generated artifacts, and navigation logs.

Together, these containers allow the System to accept user intent, translate that intent into tool-assisted reasoning, execute flight or simulation commands, run multimodal analysis, and return structured feedback to the operator.

## 4.3 Operating Environment

- OE-1: The System shall run the web interface in modern Chromium-, Safari-, Firefox-, or Edge-class browsers.
- OE-2: The backend services shall run in a Python environment capable of supporting FastAPI, PyTorch, OpenCV, and the required model runtimes.

- OE-3: The simulation environment shall run in Docker containers with ROS2, ROSBridge, and Gazebo support.
- OE-4: Real-drone operation shall require a workstation connected to the DJI Tello over Wi-Fi.
- OE-5: GPU acceleration should be supported where CUDA-capable hardware is available, but CPU execution shall remain possible for supported tool flows.
- OE-6: Audio transcription for compressed audio formats shall require an `ffmpeg` binary installed on the host.

## 4.4 Design and Implementation Constraints

- CO-1: The web application shall use the existing Vue 3 and Vite frontend architecture.
- CO-2: The unified backend shall use the existing FastAPI application structure and Uvicorn runtime.
- CO-3: Mission orchestration shall use a LiteLLM-compatible model backend and `smolagents` tool-calling flow.
- CO-4: Perception tools shall remain callable through the local MCP server boundary used by the current runtime.
- CO-5: Real-drone control shall conform to the capabilities and limits of the DJI Tello SDK as accessed through `djitellopy`.
- CO-6: Simulation mode shall depend on ROS2, ROSBridge, and Gazebo components provided by the Docker-based environment.
- CO-7: Linear drone movement commands shall respect the Tello-safe range of 20 to 500 centimeters per command.
- CO-8: Rotation commands shall respect the current 1 to 360 degree command range.
- CO-9: Current persistent artifact storage shall use the local filesystem; the current implementation does not include a production database layer.

## 4.5 Assumptions and Dependencies

- AS-1: A compatible LLM provider is available, either through OpenAI-compatible APIs or an Ollama-compatible backend.
- AS-2: Required model weights and Python dependencies can be installed on the target machine.
- AS-3: The DJI Tello remains reachable over the local wireless network during hardware operation.
- AS-4: Docker, ROS2, ROSBridge, and Gazebo remain available for simulation workflows.
- AS-5: Operators have sufficient permissions to access microphone, browser media devices, and local networking when needed.
- AS-6: The local filesystem has enough capacity to store uploaded media and generated outputs.
- AS-7: Development and demo environments may rely on local storage and environment-variable-based secrets rather than enterprise deployment infrastructure.

# 5. Functional Requirements

## 5.1 Use Cases

The detailed EARS-style use cases are maintained in use-cases.pdf. Those use cases organize VANTAGE behavior into the following major capability areas:

- Agentic intelligence and mission planning
- Drone operation and Tello control
- Perception intelligence and computer vision
- Audio processing and voice interaction
- Digital twin simulation and ROS2 integration
- Web frontend and user interface
- Safety, bounds checking, and recovery behavior

## 5.2 Non-Use Case Functional Requirements

- FR-1: The System shall provide a mission-control web interface at the root route /.
- FR-2: The System shall provide a tool dashboard web interface at the route /dashboard.
- FR-3: The System shall expose a REST endpoint at /api/status that returns current telemetry and mission-busy state.
- FR-4: The System shall expose a REST endpoint at /api/emergency-land that triggers emergency landing behavior.
- FR-5: The System shall expose a WebSocket endpoint at /ws/video that streams live JPEG-encoded camera frames to connected clients.
- FR-6: The System shall expose a WebSocket endpoint at /ws/chat for mission submission, mission progress events, and status responses.
- FR-7: The System shall reject a new mission request while another agent mission is already running.
- FR-8: The System shall allow mission input through typed text and through voice-transcribed text in the mission console.
- FR-9: The System shall support dashboard execution of speech-to-text, text-to-speech, image captioning, visual question answering, semantic segmentation, and object detection.
- FR-10: The System shall expose registered tools through /v1/tools/list and /v1/tools/call.
- FR-11: The System shall save dashboard uploads and generated files into isolated, timestamped run directories.
- FR-12: The System shall save captured drone images with timestamp-based filenames for later analysis.
- FR-13: The System shall allow the mission agent to use both drone-control tools and MCP-provided perception tools during mission execution.
- FR-14: The System shall cache loaded model instances and reuse them for subsequent requests when the model and device combination match.
- FR-15: The System shall provide downloadable or renderable access to generated dashboard artifacts through validated file routes.

- FR-16: The System shall support both simulation-backed and hardware-backed mission execution workflows.
- FR-17: The System shall publish mission lifecycle feedback including mission start, completion, and mission errors to the web client.
- FR-18: The System shall provide an emergency landing control in the user interface.

# 6. Business Rules

- BR-1: Emergency landing shall take precedence over all active missions and manual flight commands.
- BR-2: The System shall not allow movement commands when the drone is disconnected or not airborne.
- BR-3: Linear movement commands shall be issued only in discrete increments between 20 cm and 500 cm.
- BR-4: Rotation commands shall be issued only in discrete increments between 1 degree and 360 degrees.
- BR-5: Autonomous target-navigation missions shall not begin when the starting battery level is below 15%.
- BR-6: The System shall enforce a single active flight operation at a time through shared flight locking.
- BR-7: Mission-agent reasoning shall execute one tool action at a time and evaluate the result before the next step.
- BR-8: New or high-risk flight workflows should be validated in the digital twin before real-drone execution whenever practical.

# 7. Data Requirements

## 7.1 Data Acquisition, Integrity, Retention, and Disposal

- DI-1: The System shall acquire operational data from operator text input, optional voice input, uploaded media files, drone telemetry, and drone camera frames.
- DI-2: The System shall generate derived artifacts including transcriptions, synthesized audio files, captions, VQA answers, annotated detections, segmentation masks, overlays, and navigation logs.
- DI-3: Dashboard uploads and outputs shall be stored under the local outputs/dashboard storage tree.
- DI-4: Captured live-drone images shall be stored under the local captures directory or another configured capture directory.
- DI-5: Mission progress events, mission busy state, and overlay state may be held in in-memory queues and runtime objects during active execution.
- DI-6: File-serving routes shall validate that requested files remain inside approved dashboard storage paths.
- DI-7: Generated artifact names shall preserve run-level separation through timestamped or unique directory naming.

- DI-8: Retention and cleanup procedures for locally stored artifacts shall be defined operationally before production use because the current implementation stores outputs on disk by default.

# 8. External Interface Requirements

## 8.1 User Interfaces

The current user interface has two primary views:

1. Mission Console
    - Live video canvas
    - Telemetry badges for battery, height, flight time, temperature, and flight state
    - Emergency land button
    - Mission chat feed
    - Text input and microphone-triggered voice input
2. Tool Dashboard
    - Upload-and-run panels for STT, TTS, image captioning, VQA, segmentation, and object detection
    - Preview panels for generated images, video, audio, and JSON-like results

UI requirements:

- UI-1: The mission console shall display live telemetry and flight-state indicators.
- UI-2: The mission console shall show mission status messages and error responses in a chat-style feed.
- UI-3: The mission console shall allow operators to submit typed missions and optional voice-captured missions.
- UI-4: The tool dashboard shall allow direct media upload for supported perception workflows.
- UI-5: The UI shall present generated media outputs back to the user when those outputs are available.

## 8.2 Software Interfaces

- SI-1: The frontend shall communicate with the backend over HTTP/JSON for REST requests.
- SI-2: The frontend shall communicate with the backend over WebSockets for live video and chat events.
- SI-3: Dashboard upload interfaces shall use multipart/form-data requests for audio, image, and video file submission.
- SI-4: The mission agent shall communicate with model providers through a LiteLLM-compatible API layer.
- SI-5: The mission runtime shall communicate with the MCP tool server through the local MCP stdio client/server boundary.
- SI-6: Real-drone control shall communicate with DJI Tello hardware through the djitellopy SDK.

- SI-7: Simulation publishing shall communicate through ROSBridge and ROS2 topic infrastructure.
- SI-8: Gazebo simulation visualization shall be externally accessible through a VNC endpoint in the containerized setup.
- SI-9: The System shall support rendering generated audio through normal browser media playback controls.

# 9. Quality Attributes

## 9.1 Usability

- USE-1: The System shall support natural-language mission input to reduce the need for manual command scripting.
- USE-2: The mission console shall present flight telemetry in a persistent, visible HUD-style format.
- USE-3: The tool dashboard shall separate media-analysis workflows into clearly labeled panels.
- USE-4: The System shall provide operator-readable success, failure, and mission-completion messages.

## 9.2 Performance

- PER-1: The mission console shall support telemetry refresh at approximately one-second intervals during operation.
- PER-2: The live video stream shall target near-real-time refresh, subject to available camera, CPU, GPU, and network resources.
- PER-3: The System shall cache loaded ML models in memory to reduce repeated cold-start latency.
- PER-4: The mission agent shall use a configurable maximum reasoning step count to bound mission execution.

## 9.3 Security

- SEC-1: Provider API keys and related secrets shall be supplied through environment variables or equivalent deployment configuration, not hard-coded in source.
- SEC-2: File retrieval interfaces shall reject path traversal and requests for files outside approved storage roots.
- SEC-3: Production deployments shall use HTTPS and secure WebSocket transport.
- SEC-4: Access to real-drone control functions should be restricted to authorized operators in production deployments.

## 9.4 Safety

- SAF-1: The System shall provide an emergency landing path that can be triggered by UI control, signal handling, or mission failure.
- SAF-2: The System shall block movement when emergency-stop state is active.
- SAF-3: The System shall validate movement and rotation ranges before issuing commands to the drone.

- SAF-4: The System shall land or attempt safe shutdown when a critical mission exception occurs during flight.
- SAF-5: The platform shall support simulation-based dry runs to reduce hardware-flight risk.

### 9.5 Availability

- AVL-1: When the drone, camera stream, or model provider is unavailable, the System shall return a structured error rather than failing silently.
- AVL-2: The mission console shall attempt to reconnect WebSocket sessions after disconnect events.
- AVL-3: The System shall support local development and containerized simulation workflows so that work can continue when hardware is unavailable.

### 9.6 Robustness

- ROB-1: Drone-control operations shall use thread-safe access patterns around the shared controller.
- ROB-2: The System shall prevent overlapping flight operations through shared locking.
- ROB-3: Runtime state shall isolate uploaded and generated artifacts by run directory to reduce file collisions.
- ROB-4: The unified server shall include automated route and WebSocket tests for core integration behavior.

## 10. Deployment

VANTAGE supports two primary deployment modes:

1. Local or hardware-connected mode
    - FastAPI backend running on a workstation
    - Vue frontend served in development through Vite or as built static assets through the backend
    - DJI Tello connected over Wi-Fi
    - Optional cloud or Ollama-compatible LLM backend
2. Containerized simulation mode
    - `ros2` service for ROS2 workspace and topic infrastructure
    - `rosbridge` service exposing WebSocket access to ROS topics
    - `gazebo` service for the digital twin and physics simulation
    - `vantage` service for the Python agent runtime

Deployment requirements:

- DEP-1: The backend shall be configurable by environment variables including model selection, API base, API key, secret key, and service port.
- DEP-2: The frontend shall be buildable independently through `npm` and deployable as static assets.

- DEP-3: Simulation deployment shall be orchestrated through the provided Docker Compose configuration.
- DEP-4: Simulation deployment shall expose ROSBridge and VNC ports as configured by environment variables.
- DEP-5: The deployment process shall support both local model execution and remote model-provider execution.
- DEP-6: Production-grade deployment beyond prototype or demonstration usage should add authentication, centralized logging, managed artifact retention, and formal operational monitoring.

## 11. Internationalization and Localization Requirements

The current implementation is primarily English-first in its user interface, prompts, and operating assumptions.

- INT-1: The initial release shall support English-language UI text and English mission workflows.
- INT-2: The System may accept additional spoken languages where supported by the selected speech model, but multilingual behavior is not guaranteed across all workflows.
- INT-3: Future releases should externalize user-facing strings and prompt templates if multilingual deployment becomes a project objective.