```

////////////////////////////////////////////////
// FILE: analyze_size.py
////////////////////////////////////////////////
import os
import sys
def get_files_with_size(start_path):
    """递归获取项目文件大小，并排除标准开发目录。"""
    # 需要排除的目录列表
    EXCLUDE_DIRS = ['venv', 'node_modules', '.git', '__pycache__', '.pytest_cache', 'frontend/
    node_modules', 'docker/']
    file_list = []
    for root, dirs, files in os.walk(start_path, topdown=True):
        # 在 os.walk 运行时排除目录，以提高效率
        dirs[:] = [d for d in dirs if d not in EXCLUDE_DIRS]
        for file_name in files:
            file_path = os.path.join(root, file_name)
            # 再次检查排除列表中的文件（如 .DS_Store 等）
            if any(exclude in file_path for exclude in ['.git', '.vscode', '.idea', 'thumbs.db'])
    :
                continue
            try:
                if os.path.exists(file_path):
                    size_bytes = os.path.getsize(file_path)
                    file_list.append((file_path, size_bytes))
            except OSError:
                # 忽略权限不足的文件
                continue
    return file_list
def main():
    # 从当前目录开始扫描
    start_path = '.'
    print("--- 正在扫描项目文件大小 (已排除 venv, node_modules, .git 等) ---")
    # 获取文件并按大小降序排序
    files_data = get_files_with_size(start_path)
    files_data.sort(key=lambda item: item[1], reverse=True)
    print(f"\n项目总文件数 (不含排除项): {len(files_data)}")
    print("\n--- 最大的 10 个文件 (Largest 10 Files) ---")
    print
    ("-------------------------------------------------------------------------------------------------------")

    print(f"| {'Size (MB)':<10} | {'Size (Bytes)':<15} | {'Path':<50}")
    print
    ("-------------------------------------------------------------------------------------------------------")

    # 打印前 10 个结果
    for path, size in files_data[:10]:
        size_mb = size / (1024 * 1024)
        print(f"| {size_mb:10.2f} | {size:15,} | {path}")
    print
    ("-------------------------------------------------------------------------------------------------------")

    print("\n请根据列表检查 yolov8n.pt 和 drone.db 是否出现在其中。")
if __name__ == "__main__":
    main()

////////////////////////////////////////////////
```

```
// FILE: app\main.py
////////////////////////////////////////////////
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from contextlib import asynccontextmanager
import asyncio
from app.infrastructure.mavsdk.connection import mavsdk_manager
from app.api.routers import missions, telemetry, vision
from app.infrastructure.database.db import engine, Base
from app.infrastructure.database import models
@asynccontextmanager
async def lifespan(app: FastAPI):
    print(">>> System Starting...")
    # Initialize DB Tables
    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)
    # Initialize MAVSDK connection (non blocking)
    asyncio.create_task(mavsdk_manager.connect())
    yield
    print(">>> System Shutting Down...")
app = FastAPI(title="Drone Control System", lifespan=lifespan)
# CORS Configuration
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
# Mount Routers
app.include_router(missions.router, prefix="/api/v1/missions", tags=["Missions"])
app.include_router(telemetry.router, tags=["Telemetry"])
app.include_router(vision.router, prefix="/vision", tags=["Vision"])
@app.get("/")
async def root():
    return {"message": "System Online"}

////////////////////////////////////////////////
// FILE: app\api\routers\missions.py
////////////////////////////////////////////////
from fastapi import APIRouter, HTTPException, Depends
from pydantic import BaseModel
from typing import List
import uuid
from datetime import datetime
import json
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.future import select
from app.infrastructure.mavsdk.mission_service import mavsdk_mission_service
from app.infrastructure.mavsdk.connection import mavsdk_manager
from app.infrastructure.database.db import get_db
from app.infrastructure.database.models import MissionModel
from app.domain.mission import Mission, Waypoint
from app.core.drone_state import drone_state
router = APIRouter()
class WaypointDTO(BaseModel):
    latitude: float
    longitude: float
```

```
    relative_altitude: float
    speed_m_s: float
class MissionDTO(BaseModel):
    name: str
    waypoints: List[WaypointDTO]
@router.post("/upload")
async def upload_mission(mission_data: MissionDTO, db: AsyncSession = Depends(get_db)):
    try:
        # Convert DTO to Domain Entities
        domain_waypoints = [
            Waypoint(
                latitude=wp.latitude,
                longitude=wp.longitude,
                relative_altitude=wp.relative_altitude,
                speed_m_s=wp.speed_m_s
            ) for wp in mission_data.waypoints
        ]
        domain_mission = Mission(
            id=uuid.uuid4(),
            name=mission_data.name,
            waypoints=domain_waypoints,
            created_at=datetime.utcnow(),
            status="UPLOADED"
        )
        # 1. Upload to Drone (Hardware/Sim)
        if mavsdk_manager.system:
            await mavsdk_mission_service.upload_mission(mavsdk_manager.system, domain_mission)
        else:
            print("  Drone system not connected. Skipping hardware upload.")
        # 2. Update Physics Engine Target
        if mission_data.waypoints:
            last_wp = mission_data.waypoints[-1]
            drone_state.target_lat = last_wp.latitude
            drone_state.target_lon = last_wp.longitude
            print(f">>> New Target Set: {drone_state.target_lat}, {drone_state.target_lon}")
        # 3. Persist to Database
        mission_entry = MissionModel(
            name=mission_data.name,
            status="UPLOADED",
            waypoints_json=json.dumps([wp.dict() for wp in mission_data.waypoints])
        )
        db.add(mission_entry)
        await db.commit()
        return {"message": f"Mission '{mission_data.name}' uploaded and saved."}
    except Exception as e:
        print(f"Error uploading mission: {e}")
        raise HTTPException(status_code=500, detail=str(e))
@router.get("/history")
async def get_mission_history(db: AsyncSession = Depends(get_db)):
    result = await db.execute(select(MissionModel).order_by(MissionModel.timestamp.desc()))
    missions = result.scalars().all()
    return missions

////////////////////////////////////////////////
// FILE: app\api\routers\telemetry.py
////////////////////////////////////////////////
from fastapi import APIRouter, WebSocket, WebSocketDisconnect
import asyncio
```

```
from app.core.drone_state import drone_state
from pydantic import BaseModel
class SpeedRequest(BaseModel):
    speed: float
router = APIRouter()
@router.websocket("/ws/telemetry")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    try:
        while True:
            # Update Physics
            drone_state.update_position()
            data = {
                "lat": drone_state.lat,
                "lon": drone_state.lon,
                "heading": drone_state.heading,
                "alt": drone_state.alt
            }
            await websocket.send_json(data)
            await asyncio.sleep(0.05) # 20Hz update rate
    except WebSocketDisconnect:
        print("Telemetry client disconnected")
@router.post("/api/v1/telemetry/speed")
async def update_speed(request: SpeedRequest):
    drone_state.set_speed(request.speed)
    return {"message": f"Speed set to {drone_state.speed}"}

////////////////////////////////////////////////
// FILE: app\api\routers\vision.py
////////////////////////////////////////////////
from fastapi import APIRouter, UploadFile, File, HTTPException
from app.infrastructure.vision.yolo_service import yolo_service
router = APIRouter()
@router.post("/analyze")
async def analyze_image(file: UploadFile = File(...)):
    if not file.content_type.startswith("image/"):
        raise HTTPException(status_code=400, detail="File must be an image")
    try:
        contents = await file.read()
        detections = await yolo_service.analyze_image(contents)
        return {
            "filename": file.filename,
            "detections": detections
        }
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

////////////////////////////////////////////////
// FILE: app\core\drone_state.py
////////////////////////////////////////////////
import math
class DroneState:
    def __init__(self):
        # Initial Position (Chengdu)
        self.lat = 30.598
        self.lon = 103.991
        self.alt = 100.0
        self.heading = 0.0
```

```
        # Navigation Target
        self.target_lat = None
        self.target_lon = None
        # Physics Constants
        self.speed = 0.00005       # Speed per tick
        self.max_radius = 0.005    # Start spiraling from ~550m
        self.min_radius = 0.001    # Final hold distance ~110m
        self.spiral_decay = 0.00002 # How fast the circle tightens
        # Dynamic State
        self.current_radius = self.max_radius
        self.tick = 0
    def update_position(self):
        self.tick += 0.1
        # Mode 1: No Target -> Idle (Hover in place or circle locally)
        if self.target_lat is None:
            self.heading = (self.heading + 1) % 360
            return
        # Calculate distance to target
        lat_diff = self.target_lat - self.lat
        lon_diff = self.target_lon - self.lon
        distance_to_center = math.sqrt(lat_diff**2 + lon_diff**2)
        # Mode 2: Transit (Fly to the outer edge of the spiral)
        # We fly until we hit the max_radius edge
        if distance_to_center > self.max_radius:
            # Reset spiral state for next arrival
            self.current_radius = self.max_radius
            # Move linearly towards target
            angle = math.atan2(lon_diff, lat_diff)
            self.lat += self.speed * math.cos(angle)
            self.lon += self.speed * math.sin(angle)
            self.heading = math.degrees(angle) % 360
        else:
            # Mode 3: Spiral-In Loiter (The Fun Part)
            # 1. Decay the radius until it hits minimum
            if self.current_radius > self.min_radius:
                self.current_radius -= self.spiral_decay
            else:
                self.current_radius = self.min_radius # Hold at 100m
            # 2. Calculate position on the circle
            # We use offset from target based on current dynamic radius
            self.lat = self.target_lat + (self.current_radius * math.sin(self.tick))
            self.lon = self.target_lon + (self.current_radius * math.cos(self.tick))
            # 3. Update heading (tangent to circle)
            self.heading = (self.heading + 5) % 360
    def set_speed(self, factor: float):
        """
        Update speed based on a factor (1-100).
        Base speed is approx 0.00001.
        """
        # Clamp factor between 1 and 100
        factor = max(1.0, min(100.0, factor))
        self.speed = factor * 0.00001
# Global Singleton Instance
drone_state = DroneState()

////////////////////////////////////////////////
// FILE: app\domain\mission.py
////////////////////////////////////////////////
```

```
from dataclasses import dataclass
from datetime import datetime
from typing import List
from uuid import UUID
@dataclass
class Waypoint:
    latitude: float
    longitude: float
    relative_altitude: float
    speed_m_s: float
@dataclass
class Mission:
    id: UUID
    name: str
    waypoints: List[Waypoint]
    created_at: datetime
    status: str  # "DRAFT", "EXECUTING", "COMPLETED"

////////////////////////////////////////////////
// FILE: app\domain\interfaces\mission_repository.py
////////////////////////////////////////////////
from abc import ABC, abstractmethod
from typing import List, Optional
from uuid import UUID
from app.domain.mission import Mission
class IMissionRepository(ABC):
    @abstractmethod
    async def save(self, mission: Mission) -> Mission:
        pass
    @abstractmethod
    async def get_by_id(self, mission_id: UUID) -> Optional[Mission]:
        pass
    @abstractmethod
    async def get_all(self) -> List[Mission]:
        pass

////////////////////////////////////////////////
// FILE: app\infrastructure\database\db.py
////////////////////////////////////////////////
from sqlalchemy.ext.asyncio import create_async_engine, AsyncSession
from sqlalchemy.orm import sessionmaker, declarative_base
DATABASE_URL = "sqlite+aiosqlite:///./drone.db"
engine = create_async_engine(DATABASE_URL, echo=True)
AsyncSessionLocal = sessionmaker(
    engine, class_=AsyncSession, expire_on_commit=False
)
Base = declarative_base()
async def get_db():
    async with AsyncSessionLocal() as session:
        yield session

////////////////////////////////////////////////
// FILE: app\infrastructure\database\models.py
////////////////////////////////////////////////
from sqlalchemy import Column, Integer, String, DateTime, Text
from datetime import datetime
from app.infrastructure.database.db import Base
class MissionModel(Base):
```

```
    __tablename__ = "missions"
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, index=True)
    timestamp = Column(DateTime, default=datetime.utcnow)
    status = Column(String, default="UPLOADED")
    waypoints_json = Column(Text) # Storing JSON string of waypoints

////////////////////////////////////////////////
// FILE: app\infrastructure\mavsdk\connection.py
////////////////////////////////////////////////
import asyncio
from mavsdk import System
class MavsdkConnectionManager:
    _instance = None
    def __new__(cls):
        if cls._instance is None:
            cls._instance = super(MavsdkConnectionManager, cls).__new__(cls)
            cls._instance.system = None
        return cls._instance
    async def connect(self, system_address: str = "udp://:14540"):
        self.system = System()
        await self.system.connect(system_address=system_address)
        print(f"Waiting for drone to connect on {system_address}...")
        # In a real app, we might wait for state, but for init we just start
        # async for state in self.system.core.connection_state():
        #     if state.is_connected:
        #         print("Drone connected!")
        #         break
mavsdk_manager = MavsdkConnectionManager()

////////////////////////////////////////////////
// FILE: app\infrastructure\mavsdk\mission_service.py
////////////////////////////////////////////////
import math
from app.domain.mission import Mission
from mavsdk import System
from mavsdk.mission import MissionItem, MissionPlan
class MavsdkMissionService:
    """
    Infrastructure service to convert Domain Missions to MAVSDK Mission Plans
    and upload them to the drone.
    """
    async def upload_mission(self, system: System, mission: Mission):
        mission_items = []
        for wp in mission.waypoints:
            # STRICT COMPLIANCE: Passing float('nan') for optional parameters
            # to avoid MAVSDK v2.0 validation errors.
            item = MissionItem(
                latitude_deg=wp.latitude,
                longitude_deg=wp.longitude,
                relative_altitude_m=wp.relative_altitude,
                speed_m_s=wp.speed_m_s,
                is_fly_through=True,
                gimbal_pitch_deg=float('nan'),
                gimbal_yaw_deg=float('nan'),
                camera_action=MissionItem.CameraAction.NONE,
                loiter_time_s=float('nan'),
                camera_photo_interval_s=float('nan'),
```

```
                acceptance_radius_m=float('nan'),
                yaw_deg=float('nan'),
                camera_photo_distance_m=float('nan'),
                vehicle_action=MissionItem.VehicleAction.NONE
            )
            mission_items.append(item)
        mission_plan = MissionPlan(mission_items)
        print(f"Uploading mission '{mission.name}' with {len(mission_items)} waypoints...")
        try:
            await system.mission.upload_mission(mission_plan)
            print("Mission uploaded to hardware.")
        except Exception as e:
            print(f"  [SIMULATION MODE] Hardware upload failed: {e}")
            print("  Mocking success response for UI testing.")
            # Do NOT raise the exception to simulate success
mavsdk_mission_service = MavsdkMissionService()

////////////////////////////////////////////////
// FILE: app\infrastructure\vision\geo_math.py
////////////////////////////////////////////////
import math
class GeoLocator:
    def __init__(self, camera_fov_h=80.0, camera_fov_v=60.0, image_width=640, image_height=480):
        self.camera_fov_h = camera_fov_h
        self.camera_fov_v = camera_fov_v
        self.image_width = image_width
        self.image_height = image_height
        self.earth_radius = 6371000.0 # Meters
    def pixel_to_angle(self, u, v):
        """
        Calculate angular offsets (alpha_x, alpha_y) from image center.
        u, v: Pixel coordinates (top-left origin)
        Returns: (alpha_x_deg, alpha_y_deg)
        """
        center_u = self.image_width / 2.0
        center_v = self.image_height / 2.0
        # Horizontal angle (positive right)
        alpha_x = (u - center_u) / center_u * (self.camera_fov_h / 2.0)
        # Vertical angle (positive down? Assuming standard convention where y increases down)
        # If pitch=0 means looking down (Nadir), then +y in image is +angle (forward/up from
    nadir?)
        # Let's stick to the prompt's likely intent:
        # alpha_y is offset from the center ray.
        alpha_y = (center_v - v) / center_v * (self.camera_fov_v / 2.0)
        return alpha_x, alpha_y
    def calculate_gps_location(self, drone_lat, drone_lon, drone_alt, drone_heading, object_u,
    object_v):
        """
        Calculate the GPS location of an object in the image.
        Assumes Flat Earth projection for short distances.
        """
        # Step 1: Get angles
        alpha_x, alpha_y = self.pixel_to_angle(object_u, object_v)
        # Step 2: Calculate ground distance
        # Formula: D = drone_alt * tan(pitch + alpha_y)
        # Assuming pitch = 0 (Nadir/Down-facing for this formula to make sense with D ~ alt * tan
    (alpha))
        # If pitch=0 is horizontal, this formula is weird unless alpha_y is depression.
```

```
        # We use the prompt's exact formula.
        pitch = 0.0 # Mock pitch
        # Convert to radians
        angle_rad = math.radians(pitch + alpha_y)
        # Avoid tan(90)
        if abs(angle_rad - math.pi/2) < 0.001:
            distance = 10000.0 # Max range clamp
        else:
            distance = drone_alt * math.tan(angle_rad)
        # Clamp distance to avoid crazy values if looking at horizon
        if distance < 0: distance = 0 # Should not happen if looking down
        if distance > 1000: distance = 1000 # Max 1km range
        # Step 3: Calculate bearing
        # Target_Bearing = drone_heading + alpha_x
        bearing_deg = drone_heading + alpha_x
        bearing_rad = math.radians(bearing_deg)
        # Step 4: Calculate new Lat/Lon using Haversine destination formula
        lat_rad = math.radians(drone_lat)
        lon_rad = math.radians(drone_lon)
        angular_distance = distance / self.earth_radius
        new_lat_rad = math.asin(
            math.sin(lat_rad) * math.cos(angular_distance) +
            math.cos(lat_rad) * math.sin(angular_distance) * math.cos(bearing_rad)
        )
        new_lon_rad = lon_rad + math.atan2(
            math.sin(bearing_rad) * math.sin(angular_distance) * math.cos(lat_rad),
            math.cos(angular_distance) - math.sin(lat_rad) * math.sin(new_lat_rad)
        )
        return {
            "lat": math.degrees(new_lat_rad),
            "lon": math.degrees(new_lon_rad),
            "distance_m": distance,
            "bearing_deg": bearing_deg
        }

////////////////////////////////////////////////
// FILE: app\infrastructure\vision\yolo_service.py
////////////////////////////////////////////////
from ultralytics import YOLO
import concurrent.futures
import asyncio
import io
from PIL import Image
from app.infrastructure.vision.geo_math import GeoLocator
class YoloService:
    def __init__(self):
        print("Initializing YOLOv8 model...")
        # Initialize YOLOv8 nano model (auto-downloads if needed)
        self.model = YOLO('yolov8n.pt')
        # Thread pool for CPU-bound inference tasks
        self.executor = concurrent.futures.ThreadPoolExecutor(max_workers=1)
        self.geo_locator = GeoLocator()
        print("YOLOv8 model initialized.")
    def _predict_sync(self, image_bytes: bytes):
        """
        Synchronous helper to run inference on image bytes.
        """
        try:
```

```
            image = Image.open(io.BytesIO(image_bytes))
            # Run prediction
            results = self.model.predict(image, verbose=False)
            detections = []
            for result in results:
                for box in result.boxes:
                    detections.append({
                        "label": result.names[int(box.cls)],
                        "confidence": float(box.conf),
                        "bbox": box.xyxy[0].tolist() # [x1, y1, x2, y2]
                    })
            return detections
        except Exception as e:
            print(f"Error in YOLO prediction: {e}")
            return []
    async def analyze_image(self, image_bytes: bytes):
        """
        Asynchronous wrapper to run inference in a separate thread.
        """
        loop = asyncio.get_running_loop()
        detections = await loop.run_in_executor(
            self.executor,
            self._predict_sync,
            image_bytes
        )
        # Mock Drone State (Chengdu)
        drone_lat = 30.598
        drone_lon = 103.991
        drone_alt = 100.0 # Meters
        drone_heading = 0.0 # North
        # Enrich with Geolocation
        for d in detections:
            bbox = d["bbox"] # [x1, y1, x2, y2]
            center_u = (bbox[0] + bbox[2]) / 2
            center_v = (bbox[1] + bbox[3]) / 2
            geo = self.geo_locator.calculate_gps_location(
                drone_lat, drone_lon, drone_alt, drone_heading,
                center_u, center_v
            )
            d["geo_location"] = geo
        return detections
# Global instance
yolo_service = YoloService()

////////////////////////////////////////////////
// FILE: frontend\vite.config.js
////////////////////////////////////////////////
import { fileURLToPath, URL } from 'node:url'
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import vueDevTools from 'vite-plugin-vue-devtools'
// https://vite.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    vueDevTools(),
  ],
  resolve: {
```

```
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    },
  },
})

////////////////////////////////////////////////
// FILE: frontend\src\App.vue
////////////////////////////////////////////////
<script setup>
import { RouterView } from 'vue-router'
</script>
<template>
  <RouterView />
</template>
<style>
/* Force full screen layout */
html, body, #app {
  margin: 0;
  padding: 0;
  height: 100%;
  width: 100%;
  overflow: hidden;
}
</style>

////////////////////////////////////////////////
// FILE: frontend\src\main.js
////////////////////////////////////////////////
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import App from './App.vue'
import router from './router'
// Note: Removed CSS imports to prevent errors if files are missing
const app = createApp(App)
app.use(createPinia())
app.use(router)
app.mount('#app')

////////////////////////////////////////////////
// FILE: frontend\src\components\DroneMap.vue
////////////////////////////////////////////////
<template>
  <div style="height: 100vh; width: 100%; position: relative;">
    <l-map
      ref="map"
      v-model:zoom="zoom"
      :center="[30.598, 103.991]"
      :use-global-leaflet="false"
      @click="onMapClick"
      @zoomend="updateZoom"
    >
      <l-tile-layer
        url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
        layer-type="base"
        name="OpenStreetMap"
      ></l-tile-layer>
      <!-- Waypoint Markers (Blue) -->
```

```
628         <l-marker
629           v-for="(wp, index) in waypoints"
630           :key="'wp-'+index"
631           :lat-lng="[wp.latitude, wp.longitude]"
632         ></l-marker>
633         <!-- Flight Path -->
634         <l-polyline
635           :lat-lngs="pathCoordinates"
636           color="blue"
637         ></l-polyline>
638         <!-- AI Targets (Orange) -->
639         <l-circle-marker
640           v-for="(obj, index) in detectedObjects"
641           :key="'target-'+index"
642           :lat-lng="[obj.geo_location.lat, obj.geo_location.lon]"
643           :radius="8"
644           color="orange"
645           fill-color="#ff9800"
646           :fill-opacity="0.9"
647         >
648           <l-popup>
649             <div style="text-align: center;">
650               <strong>Target: {{ obj.label }}</strong><br/>
651               <small>Conf: {{ (obj.confidence * 100).toFixed(1) }}%</small><br/>
652               <small>Dist: {{ obj.geo_location.distance_m.toFixed(1) }}m</small><br/>
653               <button
654                 @click="setTargetAsWaypoint(obj)"
655                 style="margin-top:5px; background-color: #28a745; color: white; border: none;
    padding: 5px 10px; border-radius: 4px; cursor: pointer;"
656               >
657                 Fly Here
658               </button>
659             </div>
660           </l-popup>
661         </l-circle-marker>
662         <!-- Drone Position Marker (Red) -->
663         <l-circle-marker
664           v-if="dronePos"
665           :lat-lng="[dronePos.lat, dronePos.lon]"
666           :radius="10"
667           color="red"
668           fill-color="#f03"
669           :fill-opacity="0.8"
670         >
671           <l-popup>Drone Live</l-popup>
672         </l-circle-marker>
673       </l-map>
674       <!-- Dashboard Panel (Sidebar) -->
675       <div class="dashboard-panel">
676         <h3>Drone Control</h3>
677         <!-- Map Info -->
678         <div class="panel-section">
679           <strong>Map Info</strong>
680           <div>Zoom Level: {{ currentZoom }}</div>
681         </div>
682         <!-- Telemetry -->
683         <div class="panel-section" v-if="dronePos">
684           <strong>Telemetry</strong>
```

```
685         <div>Lat: {{ dronePos.lat.toFixed(5) }}</div>
686         <div>Lon: {{ dronePos.lon.toFixed(5) }}</div>
687         <div>Hdg: {{ dronePos.heading.toFixed(1) }}°</div>
688         <div>Alt: {{ dronePos.alt.toFixed(1) }}m</div>
689       </div>
690       <!-- Speed Control -->
691       <div class="panel-section">
692         <strong>Sim Speed: {{ speedFactor }}x</strong>
693         <input
694           type="range"
695           min="1"
696           max="100"
697           v-model="speedFactor"
698           @input="changeSpeed"
699           style="width: 100%;"
700         >
701       </div>
702       <!-- Mission Control -->
703       <div class="panel-section">
704         <strong>Mission</strong>
705         <div class="button-group">
706           <button @click="uploadMission" :disabled="waypoints.length === 0">Upload</button>
707           <button @click="clearMission" class="btn-danger" :disabled="waypoints.length === 0">
    Clear</button>
708         </div>
709         <div style="margin-top: 5px;">
710            <input type="file" ref="fileInput" @change="analyzeImage" style="display: none" accept
    ="image/*">
711            <button @click="triggerUpload" class="btn-purple" style="width:  100%;">  Vision Recon
    </button>
712         </div>
713       </div>
714       <!-- Waypoint List -->
715       <div class="panel-section waypoint-list" v-if="waypoints.length > 0">
716         <strong>Waypoints ({{ waypoints.length }})</strong>
717         <div v-for="(wp, i) in waypoints" :key="i" class="waypoint-item">
718           WP {{ i+1 }}: [{{ wp.latitude.toFixed(4) }}, {{ wp.longitude.toFixed(4) }}]
719         </div>
720       </div>
721     </div>
722   </div>
723 </template>
724 <script setup>
725 import "leaflet/dist/leaflet.css";
726 import L from "leaflet";
727 import { LMap, LTileLayer, LMarker, LPolyline, LCircleMarker, LPopup } from "@vue-leaflet/vue-
    leaflet";
728 import { ref, computed, onMounted, onUnmounted, watch } from "vue";
729 const zoom = ref(13);
730 const currentZoom = ref(13);
731 const waypoints = ref([]);
732 const dronePos = ref(null);
733 const detectedObjects = ref([]);
734 const fileInput = ref(null);
735 const speedFactor = ref(5);
736 let socket = null;
737 // Compute path for polyline
738 const pathCoordinates = computed(() => {
```

```
  return waypoints.value.map(wp => [wp.latitude, wp.longitude]);
});
const updateZoom = (e) => {
  currentZoom.value = e.target.getZoom();
};
// Handle map clicks to add waypoints
const onMapClick = (e) => {
  waypoints.value.push({
    latitude: e.latlng.lat,
    longitude: e.latlng.lng,
    relative_altitude: 20.0, // Default altitude 20m
    speed_m_s: 5.0 // Default speed 5m/s
  });
};
const clearMission = () => {
  waypoints.value = [];
};
const uploadMission = async () => {
  const missionData = {
    name: `Mission ${new Date().toLocaleTimeString()}`,
    waypoints: waypoints.value
  };
  try {
    const response = await fetch('http://127.0.0.1:8080/api/v1/missions/upload', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(missionData)
    });
    if (response.ok) {
      const result = await response.json();
      alert(`Success: ${result.message}`);
    } else {
      const error = await response.json();
      alert(`Error: ${error.detail || 'Upload failed'}`);
    }
  } catch (err) {
    console.error(err);
    alert("Network Error: Check console.");
  }
};
const changeSpeed = async () => {
  try {
    await fetch('http://127.0.0.1:8080/api/v1/telemetry/speed', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ speed: parseFloat(speedFactor.value) })
    });
  } catch (err) {
    console.error("Failed to update speed:", err);
  }
};
const triggerUpload = () => {
  fileInput.value.click();
};
```

```
797 const analyzeImage = async (event) => {
798   const file = event.target.files[0];
799   if (!file) return;
800   const formData = new FormData();
801   formData.append('file', file);
802   try {
803     const response = await fetch('http://127.0.0.1:8080/vision/analyze', {
804       method: 'POST',
805       body: formData
806     });
807     if (response.ok) {
808       const result = await response.json();
809       const detections = result.detections;
810       detectedObjects.value = []; // Clear old detections
811       if (detections.length === 0) {
812         alert("Vision Recon: No objects detected.");
813         return;
814       }
815       // Filter and store detections with geolocation
816       let targetsFound = 0;
817       detections.forEach(d => {
818         if (d.geo_location) {
819           detectedObjects.value.push(d);
820           targetsFound++;
821         }
822       });
823       // Summarize detections
824       const summary = {};
825       detections.forEach(d => {
826         summary[d.label] = (summary[d.label] || 0) + 1;
827       });
828       const summaryStr = Object.entries(summary)
829         .map(([label, count]) => `${count} ${label}`)
830         .join(", ");
831       alert(`Vision Recon Results:\nFound: ${summaryStr}\n\n${targetsFound} Targets plotted on
      map.`);
832     } else {
833       const error = await response.json();
834       alert(`Vision Error: ${error.detail || 'Analysis failed'}`);
835     }
836   } catch (err) {
837     console.error(err);
838     alert("Vision Network Error: Check console.");
839   }
840 };
841 const setTargetAsWaypoint = (target) => {
842   if (!target.geo_location) return;
843   // Clear existing waypoints and set target as the single waypoint
844   waypoints.value = [{
845     latitude: target.geo_location.lat,
846     longitude: target.geo_location.lon,
847     relative_altitude: 20.0,
848     speed_m_s: 5.0
849   }];
850   // Automatically upload mission
851   uploadMission();
852 };
853 onMounted(() => {
```

```
  // Connect to Telemetry WebSocket
  socket = new WebSocket('ws://127.0.0.1:8080/ws/telemetry');
  socket.onopen = () => {
    console.log("Telemetry Connected");
  };
  socket.onmessage = (event) => {
    try {
      const data = JSON.parse(event.data);
      dronePos.value = data;
    } catch (e) {
      console.error("Error parsing telemetry:", e);
    }
  };
  socket.onclose = () => {
    console.log("Telemetry Disconnected");
  };
  socket.onerror = (error) => {
    console.error("WebSocket Error:", error);
  };
});
onUnmounted(() => {
  if (socket) {
    socket.close();
  }
});
</script>
<style>
/* Ensure map tiles render correctly */
.leaflet-pane { z-index: 1 !important; }
.dashboard-panel {
  position: absolute;
  top: 10px;
  right: 10px;
  bottom: 10px;
  width: 250px;
  z-index: 1000;
  background: rgba(255, 255, 255, 0.9);
  backdrop-filter: blur(5px);
  padding: 15px;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.2);
  display: flex;
  flex-direction: column;
  gap: 15px;
  overflow-y: auto;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
.dashboard-panel h3 {
  margin: 0;
  text-align: center;
  color: #333;
  border-bottom: 2px solid #007bff;
  padding-bottom: 10px;
}
.panel-section {
  background: rgba(255, 255, 255, 0.5);
  padding: 10px;
  border-radius: 6px;
```

```
  border: 1px solid #eee;
}
.panel-section strong {
  display: block;
  margin-bottom: 5px;
  color: #555;
  font-size: 0.9rem;
}
.button-group {
  display: flex;
  gap: 5px;
}
button {
  flex: 1;
  padding: 8px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-weight: bold;
  transition: background-color 0.2s;
  font-size: 0.85rem;
}
button:disabled {
  background-color: #ccc;
  cursor: not-allowed;
}
button:hover:not(:disabled) {
  background-color: #0056b3;
}
.btn-danger {
  background-color: #dc3545;
}
.btn-danger:hover:not(:disabled) {
  background-color: #bd2130;
}
.btn-purple {
  background-color: #6f42c1;
}
.btn-purple:hover:not(:disabled) {
  background-color: #59359a;
}
.waypoint-list {
  max-height: 150px;
  overflow-y: auto;
}
.waypoint-item {
  font-size: 0.8rem;
  padding: 2px 0;
  border-bottom: 1px solid #eee;
}
</style>

////////////////////////////////////////////////
// FILE: frontend\src\components\icons\IconCommunity.vue
////////////////////////////////////////////////
<template>
```

```
  <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor">
    <path
      d="M15 4a1 1 0 1 0 0 2V4zm0 11v-1a1 1 0 0 0-1 1h1zm0 4l-.707.707A1 1 0 0 0 0 16 19h-1zm-4-4l
    .707-.707A1 1 0 0 0 11 14v1zm-4.707-1.293a1 1 0 0 0-1.414 1.414l1.414-1.414zm-.707.707l
    -.707-.707.707.707zM9 11v-1a1 1 0 0 0-.707.293L9 11zm-4 0h1a1 1 0 0 0-1-1v1zm0 4H4a1 1 0 0 0
    1.707.707L5 15zm10-9h2V4h-2v2zm2 0a1 1 0 0 1 1 1h2a3 3 0 0 0-3-3v2zm1 1v6h2V7h-2zm0 6a1 1 0 0
     1-1 1v2a3 3 0 0 0 3-3h-2zm-1 1h-2v2h2v-2zm-3 1v4h2v-4h-2zm1.707 3.293l-4-4-1.414 1.414 4 4
    1.414-1.414zM11 14H7v2h4v-2zm-4 0c-.276 0-.525-.111-.707-.293l-1.414 1.414C5.42 15.663 6.172
    16 7 16v-2zm-.707 1.121l3.414-3.414-1.414-1.414-3.414 3.414 1.414 1.414zM9 12h4v-2H9v2zm4 0a3
     3 0 0 0 3-3h-2a1 1 0 0 1-1 1v2zm3-3V3h-2v6h2zm0-6a3 3 0 0 0-3-3v2a1 1 0 0 1 1 1h2zm-3-3
    H3v2h10V0zM3 0a3 3 0 0 0-3 3h2a1 1 0 0 1 1-1V0zM0 3v6h2V3H0zm0 6a3 3 0 0 0 3 3v-2a1 1 0 0
    1-1-1H0zm3 3h2v-2H3v2zm1-1v4h2v-4H4zm1.707 4.707l.586-.586-1.414-1.414-.586.586 1.414 1.414z"
    />
  </svg>
</template>

////////////////////////////////////////////////
// FILE: frontend\src\components\icons\IconDocumentation.vue
////////////////////////////////////////////////
<template>
  <svg xmlns="http://www.w3.org/2000/svg" width="20" height="17" fill="currentColor">
    <path
      d="M11 2.253a1 1 0 1 0-2 0h2zm-2 13a1 1 0 1 0 2 0H9zm.447-12.167a1 1 0 1 0 1.107-1.666L9
    .447 3.086zM1 2.253L.447 1.42A1 1 0 0 0 0 2.253h1zm0 13H0a1 1 0 0 0 1.553.833L1 15.253zm8
    .447.833a1 1 0 1 0 1.107-1.666l-1.107 1.666zm0-14.666a1 1 0 1 0 1.107 1.666L9.447 1.42zM19
    2.253h1a1 1 0 0 0-.447-.833L19 2.253zm0 13l-.553.833A1 1 0 0 0 20 15.253h-1zm-9.553-.833a1 1
    0 1 0 1.107 1.666L9.447 14.42zM9 2.253v13h2v-13H9zm1.553-.833C9.203.523 7.42 0 5.5 0v2c1.572
    0 2.961.431 3.947 1.086l1.107-1.666zM5.5 0C3.58 0 1.797.523.447 1.42l1.107 1.666C2.539 2.431
    3.928 2 5.5 2V0zM0 2.253v13h2v-13H0zm1.553 13.833C2.539 15.431 3.928 15 5.5 15v-2c-1.92
    0-3.703.523-5.053 1.42l1.107 1.666zM5.5 15c1.572 0 2.961.431 3.947 1.086l1.107-1.666C9.203
    13.523 7.42 13 5.5 13v2zm5.053-11.914C11.539 2.431 12.928 2 14.5 2V0c-1.92 0-3.703.523-5.053
    1.42l1.107 1.666zM14.5 2c1.573 0 2.961.431 3.947 1.086l1.107-1.666C18.203.523 16.421 0 14.5 0
    v2zm3.5.253v13h2v-13h-2zm.553 12.167C18.203 13.523 16.421 13 14.5 13v2c1.573 0 2.961.431
    3.947 1.086l1.107-1.666zM14.5 13c-1.92 0-3.703.523-5.053 1.42l1.107 1.666C11.539 15.431
    12.928 15 14.5 15v-2z"
    />
  </svg>
</template>

////////////////////////////////////////////////
// FILE: frontend\src\components\icons\IconEcosystem.vue
////////////////////////////////////////////////
<template>
  <svg xmlns="http://www.w3.org/2000/svg" width="18" height="20" fill="currentColor">
    <path
      d="M11.447 8.894a1 1 0 1 0-.894-1.789l.894 1.789zm-2.894-.789a1 1 0 1 0 .894 1.789l
    -.894-1.789zm0 1.789a1 1 0 1 0 .894-1.789l-.894 1.789zM7.447 7.106a1 1 0 1 0-.894 1.789l
    .894-1.789zM10 9a1 1 0 1 0-2 0h2zm-2 2.5a1 1 0 1 0 2 0H8zm9.447-5.606a1 1 0 1 0-.894-1.789l
    .894 1.789zm-2.894-.789a1 1 0 1 0 .894 1.789l-.894-1.789zm2 .789a1 1 0 1 0 .894-1.789l-.894
    1.789zm-1.106-2.789a1 1 0 1 0-.894 1.789l.894-1.789zM18 5a1 1 0 1 0-2 0h2zm-2 2.5a1 1 0 1 0 2
     0h-2zm-5.447-4.606a1 1 0 1 0 .894-1.789l-.894 1.789zM9 1l.447-.894a1 1 0 0 0-.894 0L9 1zm
    -2.447.106a1 1 0 1 0 .894 1.789l-.894-1.789zm-6 3a1 1 0 1 0 .894 1.789L.553 4.106zm2.894.789
    a1 1 0 1 0-.894-1.789l.894 1.789zm-2-.789a1 1 0 1 0-.894 1.789l.894-1.789zm1.106 2.789a1 1 0
    1 0 .894-1.789l-.894 1.789zM2 5a1 1 0 1 0-2 0h2zM0 7.5a1 1 0 1 0 2 0H0zm2.553 5.947a1 1 0 1
    0 .894-1.789l-.894 1.789zm1.106-2.789a1 1 0 1 0-.894 1.789l.894-1.789zm1.106 1a1 1 0 1 0
    .894 1.789l-.894-1.789zm2.894.789a1 1 0 1 0-.894-1.789l.894 1.789zM8 19a1 1 0 1 0 2 0H8zm2
    -2.5a1 1 0 1 0-2 0h2zm-7.447.394a1 1 0 1 0 .894-1.789l-.894 1.789zM1 15H0a1 1 0 0 0 .553.894
    L1 15zm1-2.5a1 1 0 1 0-2 0h2zm12.553 2.606a1 1 0 1 0 .894 1.789l-.894-1.789zM17 15l.447.894A1
```

```
      1 0 0 0 18 15h-1zm1-2.5a1 1 0 1 0-2 0h2zm-7.447-5.394l-2 1 .894 1.789 2-1-.894-1.789zm-1.106
      1l-2-1-.894 1.789 2 1 .894-1.789zM8 9v2.5h2V9H8zm8.553-4.894l-2 1 .894 1.789 2-1-.894-1.789
    zm.894 0l-2-1-.894 1.789 2 1 .894-1.789zM16 5v2.5h2V5h-2zm-4.553-3.894l-2-1-.894 1.789 2 1
    .894-1.789zm-2.894-1l-2 1 .894 1.789 2-1L8.553.106zM1.447 5.894l2-1-.894-1.789-2 1 .894 1.789
    zm-.894 0l2 1 .894-1.789-2-1-.894 1.789zM0 5v2.5h2V5H0zm9.447 13.106l-2-1-.894 1.789 2 1
    .894-1.789zm0 1.789l2-1-.894-1.789-2 1 .894 1.789zM10 19v-2.5H8V19h2zm-6.553-3.894l-2-1-.894
    1.789 2 1 .894-1.789zM2 15v-2.5H0V15h2zm13.447 1.894l2-1-.894-1.789-2 1 .894 1.789zM18 15v
    -2.5h-2V15h2z"
      />
  </svg>
</template>

////////////////////////////////////////////////
// FILE: frontend\src\components\icons\IconSupport.vue
////////////////////////////////////////////////
<template>
  <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor">
    <path
      d="M10 3.22l-.61-.6a5.5 5.5 0 0 0-7.666.105 5.5 5.5 0 0 0-.114 7.665L10 18.78l8.39-8.4a5.5
    5.5 0 0 0-.114-7.665 5.5 5.5 0 0 0-7.666-.105l-.61.61z"
    />
  </svg>
</template>

////////////////////////////////////////////////
// FILE: frontend\src\components\icons\IconTooling.vue
////////////////////////////////////////////////
<!-- This icon is from <https://github.com/Templarian/MaterialDesign>, distributed under Apache
    2.0 (https://www.apache.org/licenses/LICENSE-2.0) license-->
<template>
  <svg
    xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    aria-hidden="true"
    role="img"
    class="iconify iconify--mdi"
    width="24"
    height="24"
    preserveAspectRatio="xMidYMid meet"
    viewBox="0 0 24 24"
  >
    <path
      d="M20 18v-4h-3v1h-2v-1H9v1H7v-1H4v4h16M6.33 8l-1.74 4H7v-1h2v1h6v-1h2v1h2.41l-1.74-4H6.33
    M9 5v1h6V5H9m12.84 7.61c.1.22.16.48.16.8V18c0 .53-.21 1-.6 1.41c-.4.4-.85.59-1.4.59H4c-.55
    0-1-.19-1.4-.59C2.21 19 2 18.53 2 18v-4.59c0-.32.06-.58.16-.8L4.5 7.22C4.84 6.41 5.45 6 6.33
    6H7V5c0-.55.18-1 .57-1.41C7.96 3.2 8.44 3 9 3h6c.56 0 1.04.2 1.43.59c.39.41.57.86.57 1.41v1h
    .67c.88 0 1.49.41 1.83 1.22l2.34 5.39z"
      fill="currentColor"
    ></path>
  </svg>
</template>

////////////////////////////////////////////////
// FILE: frontend\src\router\index.js
////////////////////////////////////////////////
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
const router = createRouter({
```

```
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home',
      component: HomeView
    }
  ]
})
export default router

////////////////////////////////////////////////
// FILE: frontend\src\stores\counter.js
////////////////////////////////////////////////
import { ref, computed } from 'vue'
import { defineStore } from 'pinia'
export const useCounterStore = defineStore('counter', () => {
  const count = ref(0)
  const doubleCount = computed(() => count.value * 2)
  function increment() {
    count.value++
  }
  return { count, doubleCount, increment }
})

////////////////////////////////////////////////
// FILE: frontend\src\views\AboutView.vue
////////////////////////////////////////////////
<template>
  <div class="about">
    <h1>This is an about page</h1>
  </div>
</template>
<style>
@media (min-width: 1024px) {
  .about {
    min-height: 100vh;
    display: flex;
    align-items: center;
  }
}
</style>

////////////////////////////////////////////////
// FILE: frontend\src\views\HomeView.vue
////////////////////////////////////////////////
<script setup>
import DroneMap from '../components/DroneMap.vue'
</script>
<template>
  <main>
    <DroneMap />
  </main>
</template>
```