

```

1
2
3 //////////////////////////////////////////////////
4 // FILE: analyze_size.py
5 //////////////////////////////////////////////////
6 import os
7 import sys
8 def get_files_with_size(start_path):
9     """递归获取项目文件大小，并排除标准开发目录。"""
10    # 需要排除的目录列表
11    EXCLUDE_DIRS = ['venv', 'node_modules', '.git', '__pycache__', '.pytest_cache', 'frontend/
node_modules', 'docker/']
12    file_list = []
13    for root, dirs, files in os.walk(start_path, topdown=True):
14        # 在 os.walk 运行时排除目录，以提高效率
15        dirs[:] = [d for d in dirs if d not in EXCLUDE_DIRS]
16        for file_name in files:
17            file_path = os.path.join(root, file_name)
18            # 再次检查排除列表中的文件（如 .DS_Store 等）
19            if any(exclude in file_path for exclude in ['.git', '.vscode', '.idea', 'thumbs.db']):
20                continue
21            try:
22                if os.path.exists(file_path):
23                    size_bytes = os.path.getsize(file_path)
24                    file_list.append((file_path, size_bytes))
25            except OSError:
26                # 忽略权限不足的文件
27                continue
28    return file_list
29 def main():
30    # 从当前目录开始扫描
31    start_path = '.'
32    print("--- 正在扫描项目文件大小（已排除 venv, node_modules, .git 等） ---")
33    # 获取文件并按大小降序排序
34    files_data = get_files_with_size(start_path)
35    files_data.sort(key=lambda item: item[1], reverse=True)
36    print(f"\n项目总文件数（不含排除项）: {len(files_data)}")
37    print("\n--- 最大的 10 个文件（Largest 10 Files） ---")
38    print
39    print(f"| {'Size (MB)':<10} | {'Size (Bytes)':<15} | {'Path':<50}")
40    print
41    # 打印前 10 个结果
42    for path, size in files_data[:10]:
43        size_mb = size / (1024 * 1024)
44        print(f"| {size_mb:10.2f} | {size:15} | {path}")
45    print
46    print("\n请根据列表检查 yolov8n.pt 和 drone.db 是否出现在其中。")
47    if __name__ == "__main__":
48        main()
49
50 //////////////////////////////////////////////////

```

```

51 // FILE: app\main.py
52 ///////////////////////////////////////////////////
53 from fastapi import FastAPI
54 from fastapi.middleware.cors import CORSMiddleware
55 from contextlib import asynccontextmanager
56 import asyncio
57 from app.infrastructure.mavsdk.connection import mavsdk_manager
58 from app.api.routers import missions, telemetry, vision
59 from app.infrastructure.database.db import engine, Base
60 from app.infrastructure.database import models
61 @asynccontextmanager
62 async def lifespan(app: FastAPI):
63     print(">>> System Starting...")
64     # Initialize DB Tables
65     async with engine.begin() as conn:
66         await conn.run_sync(Base.metadata.create_all)
67     # Initialize MAVSDK connection (non blocking)
68     asyncio.create_task(mavsdk_manager.connect())
69     yield
70     print(">>> System Shutting Down...")
71 app = FastAPI(title="Drone Control System", lifespan=lifespan)
72 # CORS Configuration
73 app.add_middleware(
74     CORSMiddleware,
75     allow_origins=["*"],
76     allow_credentials=True,
77     allow_methods=["*"],
78     allow_headers=["*"],
79 )
80 # Mount Routers
81 app.include_router(missions.router, prefix="/api/v1/missions", tags=["Missions"])
82 app.include_router(telemetry.router, tags=["Telemetry"])
83 app.include_router(vision.router, prefix="/vision", tags=["Vision"])
84 @app.get("/")
85 async def root():
86     return {"message": "System Online"}
87
88 ///////////////////////////////////////////////////
89 // FILE: app\api\routers\missions.py
90 ///////////////////////////////////////////////////
91 from fastapi import APIRouter, HTTPException, Depends
92 from pydantic import BaseModel
93 from typing import List
94 import uuid
95 from datetime import datetime
96 import json
97 from sqlalchemy.ext.asyncio import AsyncSession
98 from sqlalchemy.future import select
99 from app.infrastructure.mavsdk.mission_service import mavsdk_mission_service
100 from app.infrastructure.mavsdk.connection import mavsdk_manager
101 from app.infrastructure.database.db import get_db
102 from app.infrastructure.database.models import MissionModel
103 from app.domain.mission import Mission, Waypoint
104 from app.core.drone_state import drone_state
105 router = APIRouter()
106 class WaypointDTO(BaseModel):
107     latitude: float
108     longitude: float

```

```

109     relative_altitude: float
110     speed_m_s: float
111 class MissionDTO(BaseModel):
112     name: str
113     waypoints: List[WaypointDTO]
114 @router.post("/upload")
115 async def upload_mission(mission_data: MissionDTO, db: AsyncSession = Depends(get_db)):
116     try:
117         # Convert DTO to Domain Entities
118         domain_waypoints = [
119             Waypoint(
120                 latitude=wp.latitude,
121                 longitude=wp.longitude,
122                 relative_altitude=wp.relative_altitude,
123                 speed_m_s=wp.speed_m_s
124             ) for wp in mission_data.waypoints
125         ]
126         domain_mission = Mission(
127             id=uuid.uuid4(),
128             name=mission_data.name,
129             waypoints=domain_waypoints,
130             created_at=datetime.utcnow(),
131             status="UPLOADED"
132         )
133         # 1. Upload to Drone (Hardware/Sim)
134         if mavsdk_manager.system:
135             await mavsdk_mission_service.upload_mission(mavsdk_manager.system, domain_mission)
136         else:
137             print(" Drone system not connected. Skipping hardware upload.")
138         # 2. Update Physics Engine Target
139         if mission_data.waypoints:
140             last_wp = mission_data.waypoints[-1]
141             drone_state.target_lat = last_wp.latitude
142             drone_state.target_lon = last_wp.longitude
143             print(f">>> New Target Set: {drone_state.target_lat}, {drone_state.target_lon}")
144         # 3. Persist to Database
145         mission_entry = MissionModel(
146             name=mission_data.name,
147             status="UPLOADED",
148             waypoints_json=json.dumps([wp.dict() for wp in mission_data.waypoints])
149         )
150         db.add(mission_entry)
151         await db.commit()
152         return {"message": f"Mission '{mission_data.name}' uploaded and saved."}
153     except Exception as e:
154         print(f"Error uploading mission: {e}")
155         raise HTTPException(status_code=500, detail=str(e))
156 @router.get("/history")
157 async def get_mission_history(db: AsyncSession = Depends(get_db)):
158     result = await db.execute(select(MissionModel).order_by(MissionModel.timestamp.desc()))
159     missions = result.scalars().all()
160     return missions
161
162 //////////////////////////////////////
163 // FILE: app\api\routers\telemetry.py
164 //////////////////////////////////////
165 from fastapi import APIRouter, WebSocket, WebSocketDisconnect
166 import asyncio

```

```

167 from app.core.drone_state import drone_state
168 from pydantic import BaseModel
169 class SpeedRequest(BaseModel):
170     speed: float
171 router = APIRouter()
172 @router.websocket("/ws/telemetry")
173 async def websocket_endpoint(websocket: WebSocket):
174     await websocket.accept()
175     try:
176         while True:
177             # Update Physics
178             drone_state.update_position()
179             data = {
180                 "lat": drone_state.lat,
181                 "lon": drone_state.lon,
182                 "heading": drone_state.heading,
183                 "alt": drone_state.alt
184             }
185             await websocket.send_json(data)
186             await asyncio.sleep(0.05) # 20Hz update rate
187     except WebSocketDisconnect:
188         print("Telemetry client disconnected")
189 @router.post("/api/v1/telemetry/speed")
190 async def update_speed(request: SpeedRequest):
191     drone_state.set_speed(request.speed)
192     return {"message": f"Speed set to {drone_state.speed}"}
193
194 //////////////////////////////////////
195 // FILE: app\api\routers\vision.py
196 //////////////////////////////////////
197 from fastapi import APIRouter, UploadFile, File, HTTPException
198 from app.infrastructure.vision.yolo_service import yolo_service
199 router = APIRouter()
200 @router.post("/analyze")
201 async def analyze_image(file: UploadFile = File(...)):
202     if not file.content_type.startswith("image/"):
203         raise HTTPException(status_code=400, detail="File must be an image")
204     try:
205         contents = await file.read()
206         detections = await yolo_service.analyze_image(contents)
207         return {
208             "filename": file.filename,
209             "detections": detections
210         }
211     except Exception as e:
212         raise HTTPException(status_code=500, detail=str(e))
213
214 //////////////////////////////////////
215 // FILE: app\core\drone_state.py
216 //////////////////////////////////////
217 import math
218 class DroneState:
219     def __init__(self):
220         # Initial Position (Chengdu)
221         self.lat = 30.598
222         self.lon = 103.991
223         self.alt = 100.0
224         self.heading = 0.0

```

```

225     # Navigation Target
226     self.target_lat = None
227     self.target_lon = None
228     # Physics Constants
229     self.speed = 0.00005      # Speed per tick
230     self.max_radius = 0.005   # Start spiraling from ~550m
231     self.min_radius = 0.001   # Final hold distance ~110m
232     self.spiral_decay = 0.00002 # How fast the circle tightens
233     # Dynamic State
234     self.current_radius = self.max_radius
235     self.tick = 0
236     def update_position(self):
237         self.tick += 0.1
238         # Mode 1: No Target -> Idle (Hover in place or circle locally)
239         if self.target_lat is None:
240             self.heading = (self.heading + 1) % 360
241             return
242         # Calculate distance to target
243         lat_diff = self.target_lat - self.lat
244         lon_diff = self.target_lon - self.lon
245         distance_to_center = math.sqrt(lat_diff**2 + lon_diff**2)
246         # Mode 2: Transit (Fly to the outer edge of the spiral)
247         # We fly until we hit the max_radius edge
248         if distance_to_center > self.max_radius:
249             # Reset spiral state for next arrival
250             self.current_radius = self.max_radius
251             # Move linearly towards target
252             angle = math.atan2(lon_diff, lat_diff)
253             self.lat += self.speed * math.cos(angle)
254             self.lon += self.speed * math.sin(angle)
255             self.heading = math.degrees(angle) % 360
256         else:
257             # Mode 3: Spiral-In Loiter (The Fun Part)
258             # 1. Decay the radius until it hits minimum
259             if self.current_radius > self.min_radius:
260                 self.current_radius -= self.spiral_decay
261             else:
262                 self.current_radius = self.min_radius # Hold at 100m
263             # 2. Calculate position on the circle
264             # We use offset from target based on current dynamic radius
265             self.lat = self.target_lat + (self.current_radius * math.sin(self.tick))
266             self.lon = self.target_lon + (self.current_radius * math.cos(self.tick))
267             # 3. Update heading (tangent to circle)
268             self.heading = (self.heading + 5) % 360
269         def set_speed(self, factor: float):
270             """
271             Update speed based on a factor (1-100).
272             Base speed is approx 0.00001.
273             """
274             # Clamp factor between 1 and 100
275             factor = max(1.0, min(100.0, factor))
276             self.speed = factor * 0.00001
277     # Global Singleton Instance
278     drone_state = DroneState()
279
280     //////////////////////////////////////
281     // FILE: app\domain\mission.py
282     //////////////////////////////////////

```

```

283 from dataclasses import dataclass
284 from datetime import datetime
285 from typing import List
286 from uuid import UUID
287 @dataclass
288 class Waypoint:
289     latitude: float
290     longitude: float
291     relative_altitude: float
292     speed_m_s: float
293 @dataclass
294 class Mission:
295     id: UUID
296     name: str
297     waypoints: List[Waypoint]
298     created_at: datetime
299     status: str # "DRAFT", "EXECUTING", "COMPLETED"
300
301 //////////////////////////////////////////////////
302 // FILE: app\domain\interfaces\mission_repository.py
303 //////////////////////////////////////////////////
304 from abc import ABC, abstractmethod
305 from typing import List, Optional
306 from uuid import UUID
307 from app.domain.mission import Mission
308 class IMissionRepository(ABC):
309     @abstractmethod
310     async def save(self, mission: Mission) -> Mission:
311         pass
312     @abstractmethod
313     async def get_by_id(self, mission_id: UUID) -> Optional[Mission]:
314         pass
315     @abstractmethod
316     async def get_all(self) -> List[Mission]:
317         pass
318
319 //////////////////////////////////////////////////
320 // FILE: app\infrastructure\database\db.py
321 //////////////////////////////////////////////////
322 from sqlalchemy.ext.asyncio import create_async_engine, AsyncSession
323 from sqlalchemy.orm import sessionmaker, declarative_base
324 DATABASE_URL = "sqlite+aiosqlite:///./drone.db"
325 engine = create_async_engine(DATABASE_URL, echo=True)
326 AsyncSessionLocal = sessionmaker(
327     engine, class_=AsyncSession, expire_on_commit=False
328 )
329 Base = declarative_base()
330 async def get_db():
331     async with AsyncSessionLocal() as session:
332         yield session
333
334 //////////////////////////////////////////////////
335 // FILE: app\infrastructure\database\models.py
336 //////////////////////////////////////////////////
337 from sqlalchemy import Column, Integer, String, DateTime, Text
338 from datetime import datetime
339 from app.infrastructure.database.db import Base
340 class MissionModel(Base):

```

```

341     __tablename__ = "missions"
342     id = Column(Integer, primary_key=True, index=True)
343     name = Column(String, index=True)
344     timestamp = Column(DateTime, default=datetime.utcnow)
345     status = Column(String, default="UPLOADED")
346     waypoints_json = Column(Text) # Storing JSON string of waypoints
347
348     //////////////////////////////////
349     // FILE: app\infrastructure\mavsdk\connection.py
350     //////////////////////////////////
351     import asyncio
352     from mavsdk import System
353     class MavsdkConnectionManager:
354         _instance = None
355         def __new__(cls):
356             if cls._instance is None:
357                 cls._instance = super(MavsdkConnectionManager, cls).__new__(cls)
358                 cls._instance.system = None
359             return cls._instance
360         async def connect(self, system_address: str = "udp://:14540"):
361             self.system = System()
362             await self.system.connect(system_address=system_address)
363             print(f"Waiting for drone to connect on {system_address}...")
364             # In a real app, we might wait for state, but for init we just start
365             # async for state in self.system.core.connection_state():
366             #     if state.is_connected:
367             #         print("Drone connected!")
368             #         break
369     mavsdk_manager = MavsdkConnectionManager()
370
371     //////////////////////////////////
372     // FILE: app\infrastructure\mavsdk\mission_service.py
373     //////////////////////////////////
374     import math
375     from app.domain.mission import Mission
376     from mavsdk import System
377     from mavsdk.mission import MissionItem, MissionPlan
378     class MavsdkMissionService:
379         """
380         Infrastructure service to convert Domain Missions to MAVSDK Mission Plans
381         and upload them to the drone.
382         """
383         async def upload_mission(self, system: System, mission: Mission):
384             mission_items = []
385             for wp in mission.waypoints:
386                 # STRICT COMPLIANCE: Passing float('nan') for optional parameters
387                 # to avoid MAVSDK v2.0 validation errors.
388                 item = MissionItem(
389                     latitude_deg=wp.latitude,
390                     longitude_deg=wp.longitude,
391                     relative_altitude_m=wp.relative_altitude,
392                     speed_m_s=wp.speed_m_s,
393                     is_fly_through=True,
394                     gimbal_pitch_deg=float('nan'),
395                     gimbal_yaw_deg=float('nan'),
396                     camera_action=MissionItem.CameraAction.NONE,
397                     loiter_time_s=float('nan'),
398                     camera_photo_interval_s=float('nan'),

```

```

399         acceptance_radius_m=float('nan'),
400         yaw_deg=float('nan'),
401         camera_photo_distance_m=float('nan'),
402         vehicle_action=MissionItem.VehicleAction.NONE
403     )
404     mission_items.append(item)
405     mission_plan = MissionPlan(mission_items)
406     print(f"Uploading mission '{mission.name}' with {len(mission_items)} waypoints...")
407     try:
408         await system.mission.upload_mission(mission_plan)
409         print("Mission uploaded to hardware.")
410     except Exception as e:
411         print(f" [SIMULATION MODE] Hardware upload failed: {e}")
412         print(" Mocking success response for UI testing.")
413         # Do NOT raise the exception to simulate success
414     mavsdk_mission_service = MavsdkMissionService()
415
416     //////////////////////////////////////
417     // FILE: app\infrastructure\vision\geo_math.py
418     //////////////////////////////////////
419     import math
420     class GeoLocator:
421         def __init__(self, camera_fov_h=80.0, camera_fov_v=60.0, image_width=640, image_height=480):
422             self.camera_fov_h = camera_fov_h
423             self.camera_fov_v = camera_fov_v
424             self.image_width = image_width
425             self.image_height = image_height
426             self.earth_radius = 6371000.0 # Meters
427         def pixel_to_angle(self, u, v):
428             """
429             Calculate angular offsets (alpha_x, alpha_y) from image center.
430             u, v: Pixel coordinates (top-left origin)
431             Returns: (alpha_x_deg, alpha_y_deg)
432             """
433             center_u = self.image_width / 2.0
434             center_v = self.image_height / 2.0
435             # Horizontal angle (positive right)
436             alpha_x = (u - center_u) / center_u * (self.camera_fov_h / 2.0)
437             # Vertical angle (positive down? Assuming standard convention where y increases down)
438             # If pitch=0 means looking down (Nadir), then +y in image is +angle (forward/up from
439             nadir?)
440             # Let's stick to the prompt's likely intent:
441             # alpha_y is offset from the center ray.
442             alpha_y = (center_v - v) / center_v * (self.camera_fov_v / 2.0)
443             return alpha_x, alpha_y
444         def calculate_gps_location(self, drone_lat, drone_lon, drone_alt, drone_heading, object_u,
445             object_v):
446             """
447             Calculate the GPS location of an object in the image.
448             Assumes Flat Earth projection for short distances.
449             """
450             # Step 1: Get angles
451             alpha_x, alpha_y = self.pixel_to_angle(object_u, object_v)
452             # Step 2: Calculate ground distance
453             # Formula: D = drone_alt * tan(pitch + alpha_y)
454             # Assuming pitch = 0 (Nadir/Down-facing for this formula to make sense with D ~ alt * tan
455             (alpha))
456             # If pitch=0 is horizontal, this formula is weird unless alpha_y is depression.

```



```

454         # We use the prompt's exact formula.
455         pitch = 0.0 # Mock pitch
456         # Convert to radians
457         angle_rad = math.radians(pitch + alpha_y)
458         # Avoid tan(90)
459         if abs(angle_rad - math.pi/2) < 0.001:
460             distance = 10000.0 # Max range clamp
461         else:
462             distance = drone_alt * math.tan(angle_rad)
463         # Clamp distance to avoid crazy values if looking at horizon
464         if distance < 0: distance = 0 # Should not happen if looking down
465         if distance > 1000: distance = 1000 # Max 1km range
466         # Step 3: Calculate bearing
467         # Target_Bearing = drone_heading + alpha_x
468         bearing_deg = drone_heading + alpha_x
469         bearing_rad = math.radians(bearing_deg)
470         # Step 4: Calculate new Lat/Lon using Haversine destination formula
471         lat_rad = math.radians(drone_lat)
472         lon_rad = math.radians(drone_lon)
473         angular_distance = distance / self.earth_radius
474         new_lat_rad = math.asin(
475             math.sin(lat_rad) * math.cos(angular_distance) +
476             math.cos(lat_rad) * math.sin(angular_distance) * math.cos(bearing_rad)
477         )
478         new_lon_rad = lon_rad + math.atan2(
479             math.sin(bearing_rad) * math.sin(angular_distance) * math.cos(lat_rad),
480             math.cos(angular_distance) - math.sin(lat_rad) * math.sin(new_lat_rad)
481         )
482         return {
483             "lat": math.degrees(new_lat_rad),
484             "lon": math.degrees(new_lon_rad),
485             "distance_m": distance,
486             "bearing_deg": bearing_deg
487         }
488
489     //////////////////////////////////////
490     // FILE: app\infrastructure\vision\yolo_service.py
491     //////////////////////////////////////
492     from ultralytics import YOLO
493     import concurrent.futures
494     import asyncio
495     import io
496     from PIL import Image
497     from app.infrastructure.vision.geo_math import GeoLocator
498     class YoloService:
499         def __init__(self):
500             print("Initializing YOLOv8 model...")
501             # Initialize YOLOv8 nano model (auto-downloads if needed)
502             self.model = YOLO('yolov8n.pt')
503             # Thread pool for CPU-bound inference tasks
504             self.executor = concurrent.futures.ThreadPoolExecutor(max_workers=1)
505             self.geo_locator = GeoLocator()
506             print("YOLOv8 model initialized.")
507         def _predict_sync(self, image_bytes: bytes):
508             """
509             Synchronous helper to run inference on image bytes.
510             """
511             try:

```

```

512         image = Image.open(io.BytesIO(image_bytes))
513         # Run prediction
514         results = self.model.predict(image, verbose=False)
515         detections = []
516         for result in results:
517             for box in result.bboxes:
518                 detections.append({
519                     "label": result.names[int(box.cls)],
520                     "confidence": float(box.conf),
521                     "bbox": box.xyxy[0].tolist() # [x1, y1, x2, y2]
522                 })
523         return detections
524     except Exception as e:
525         print(f"Error in YOLO prediction: {e}")
526         return []
527     async def analyze_image(self, image_bytes: bytes):
528         """
529         Asynchronous wrapper to run inference in a separate thread.
530         """
531         loop = asyncio.get_running_loop()
532         detections = await loop.run_in_executor(
533             self.executor,
534             self._predict_sync,
535             image_bytes
536         )
537         # Mock Drone State (Chengdu)
538         drone_lat = 30.598
539         drone_lon = 103.991
540         drone_alt = 100.0 # Meters
541         drone_heading = 0.0 # North
542         # Enrich with Geolocation
543         for d in detections:
544             bbox = d["bbox"] # [x1, y1, x2, y2]
545             center_u = (bbox[0] + bbox[2]) / 2
546             center_v = (bbox[1] + bbox[3]) / 2
547             geo = self.geo_locator.calculate_gps_location(
548                 drone_lat, drone_lon, drone_alt, drone_heading,
549                 center_u, center_v
550             )
551             d["geo_location"] = geo
552         return detections
553 # Global instance
554 yolo_service = YoloService()
555
556 ///////////////////////////////////////////////////
557 // FILE: frontend\vite.config.js
558 ///////////////////////////////////////////////////
559 import { fileURLToPath, URL } from 'node:url'
560 import { defineConfig } from 'vite'
561 import vue from '@vitejs/plugin-vue'
562 import vueDevTools from 'vite-plugin-vue-devtools'
563 // https://vite.dev/config/
564 export default defineConfig({
565     plugins: [
566         vue(),
567         vueDevTools(),
568     ],
569     resolve: {

```

```

570     alias: {
571       '@': fileURLToPath(new URL('./src', import.meta.url))
572     },
573   },
574 })
575
576 ///////////////////////////////////////////////////
577 // FILE: frontend\src\App.vue
578 ///////////////////////////////////////////////////
579 <script setup>
580 import { RouterView } from 'vue-router'
581 </script>
582 <template>
583   <RouterView />
584 </template>
585 <style>
586   /* Force full screen layout */
587   html, body, #app {
588     margin: 0;
589     padding: 0;
590     height: 100%;
591     width: 100%;
592     overflow: hidden;
593   }
594 </style>
595
596 ///////////////////////////////////////////////////
597 // FILE: frontend\src\main.js
598 ///////////////////////////////////////////////////
599 import { createApp } from 'vue'
600 import { createPinia } from 'pinia'
601 import App from './App.vue'
602 import router from './router'
603 // Note: Removed CSS imports to prevent errors if files are missing
604 const app = createApp(App)
605 app.use(createPinia())
606 app.use(router)
607 app.mount('#app')
608
609 ///////////////////////////////////////////////////
610 // FILE: frontend\src\components\DroneMap.vue
611 ///////////////////////////////////////////////////
612 <template>
613   <div style="height: 100vh; width: 100%; position: relative;">
614     <l-map
615       ref="map"
616       v-model:zoom="zoom"
617       :center="[30.598, 103.991]"
618       :use-global-leaflet="false"
619       @click="onMapClick"
620       @zoomend="updateZoom"
621     >
622       <l-tile-layer
623         url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
624         layer-type="base"
625         name="OpenStreetMap"
626       ></l-tile-layer>
627       <!-- 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(() => {

```

```
739     return waypoints.value.map(wp => [wp.latitude, wp.longitude]);
740   });
741   const updateZoom = (e) => {
742     currentZoom.value = e.target.getZoom();
743   };
744   // Handle map clicks to add waypoints
745   const onMapClick = (e) => {
746     waypoints.value.push({
747       latitude: e.latlng.lat,
748       longitude: e.latlng.lng,
749       relative_altitude: 20.0, // Default altitude 20m
750       speed_m_s: 5.0 // Default speed 5m/s
751     });
752   };
753   const clearMission = () => {
754     waypoints.value = [];
755   };
756   const uploadMission = async () => {
757     const missionData = {
758       name: `Mission ${new Date().toLocaleTimeString()}`,
759       waypoints: waypoints.value
760     };
761     try {
762       const response = await fetch('http://127.0.0.1:8080/api/v1/missions/upload', {
763         method: 'POST',
764         headers: {
765           'Content-Type': 'application/json'
766         },
767         body: JSON.stringify(missionData)
768       });
769       if (response.ok) {
770         const result = await response.json();
771         alert(`Success: ${result.message}`);
772       } else {
773         const error = await response.json();
774         alert(`Error: ${error.detail || 'Upload failed'}`);
775       }
776     } catch (err) {
777       console.error(err);
778       alert("Network Error: Check console.");
779     }
780   };
781   const changeSpeed = async () => {
782     try {
783       await fetch('http://127.0.0.1:8080/api/v1/telemetry/speed', {
784         method: 'POST',
785         headers: {
786           'Content-Type': 'application/json'
787         },
788         body: JSON.stringify({ speed: parseFloat(speedFactor.value) })
789       });
790     } catch (err) {
791       console.error("Failed to update speed:", err);
792     }
793   };
794   const triggerUpload = () => {
795     fileInput.value.click();
796   };

```

```
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(() => {
```

```
854 // Connect to Telemetry WebSocket
855 socket = new WebSocket('ws://127.0.0.1:8080/ws/telemetry');
856 socket.onopen = () => {
857     console.log("Telemetry Connected");
858 };
859 socket.onmessage = (event) => {
860     try {
861         const data = JSON.parse(event.data);
862         dronePos.value = data;
863     } catch (e) {
864         console.error("Error parsing telemetry:", e);
865     }
866 };
867 socket.onclose = () => {
868     console.log("Telemetry Disconnected");
869 };
870 socket.onerror = (error) => {
871     console.error("WebSocket Error:", error);
872 };
873 });
874 onUnmounted(() => {
875     if (socket) {
876         socket.close();
877     }
878 });
879 </script>
880 <style>
881 /* Ensure map tiles render correctly */
882 .leaflet-pane { z-index: 1 !important; }
883 .dashboard-panel {
884     position: absolute;
885     top: 10px;
886     right: 10px;
887     bottom: 10px;
888     width: 250px;
889     z-index: 1000;
890     background: rgba(255, 255, 255, 0.9);
891     backdrop-filter: blur(5px);
892     padding: 15px;
893     border-radius: 8px;
894     box-shadow: 0 4px 12px rgba(0,0,0,0.2);
895     display: flex;
896     flex-direction: column;
897     gap: 15px;
898     overflow-y: auto;
899     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
900 }
901 .dashboard-panel h3 {
902     margin: 0;
903     text-align: center;
904     color: #333;
905     border-bottom: 2px solid #007bff;
906     padding-bottom: 10px;
907 }
908 .panel-section {
909     background: rgba(255, 255, 255, 0.5);
910     padding: 10px;
911     border-radius: 6px;
```



```
970 <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor">
971   <path
972     d="M15 4a1 1 0 1 0 0 2V4zm0 11v-1a1 1 0 0 0-1 1h1zm0 4l-.707.707A1 1 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 1.414 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"
973   />
974 </svg>
975 </template>
976
977 //////////////////////////////////////
978 // FILE: frontend\src\components\icons\IconDocumentation.vue
979 //////////////////////////////////////
980 <template>
981   <svg xmlns="http://www.w3.org/2000/svg" width="20" height="17" fill="currentColor">
982     <path
983       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 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 2.253v13h2v-13h-2zm1.553 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"
984     />
985   </svg>
986 </template>
987
988 //////////////////////////////////////
989 // FILE: frontend\src\components\icons\IconEcosystem.vue
990 //////////////////////////////////////
991 <template>
992   <svg xmlns="http://www.w3.org/2000/svg" width="18" height="20" fill="currentColor">
993     <path
994       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 11.447-.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 0H0zm8.553 12.394a1 1 0 1
        0 .894-1.789l-.894 1.789zm-1.106-2.789a1 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
11-2-1-.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"
995 />
996 </svg>
997 </template>
998
999 //////////////////////////////////////
1000 // FILE: frontend\src\components\icons\IconSupport.vue
1001 //////////////////////////////////////
1002 <template>
1003   <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor">
1004     <path
1005       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
1006       5.5 0 0 0-.114-7.665 5.5 5.5 0 0 0-7.666-.105l-.61.61z"
1007     />
1008   </svg>
1009 </template>
1010
1011 //////////////////////////////////////
1012 // FILE: frontend\src\components\icons\IconTooling.vue
1013 //////////////////////////////////////
1014 <!-- This icon is from <https://github.com/Templarian/MaterialDesign>, distributed under Apache
1015       2.0 (https://www.apache.org/licenses/LICENSE-2.0) license-->
1016 <template>
1017   <svg
1018     xmlns="http://www.w3.org/2000/svg"
1019     xmlns:xlink="http://www.w3.org/1999/xlink"
1020     aria-hidden="true"
1021     role="img"
1022     class="iconify iconify--mdi"
1023     width="24"
1024     height="24"
1025     preserveAspectRatio="xMidYMid meet"
1026     viewBox="0 0 24 24"
1027   >
1028     <path
1029       d="M20 18v-4h-3v1h-2v-1H9v1H7v-1H4v4h16M6.33 8l-1.74 4H7v-1h2v1h6v-1h2v1h2.41l-1.74-4H6.33
1030       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
1031       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
1032       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
1033       .67c.88 0 1.49.41 1.83 1.22l2.34 5.39z"
1034       fill="currentColor"
1035     ></path>
1036   </svg>
1037 </template>
1038
1039 //////////////////////////////////////
1040 // FILE: frontend\src\router\index.js
1041 //////////////////////////////////////
1042 import { createRouter, createWebHistory } from 'vue-router'
1043 import HomeView from '../views/HomeView.vue'
1044 const router = createRouter({
```

```

1039   history: createWebHistory(import.meta.env.BASE_URL),
1040   routes: [
1041     {
1042       path: '/',
1043       name: 'home',
1044       component: HomeView
1045     }
1046   ]
1047 })
1048 export default router
1049
1050 //////////////////////////////////////
1051 // FILE: frontend\src\stores\counter.js
1052 //////////////////////////////////////
1053 import { ref, computed } from 'vue'
1054 import { defineStore } from 'pinia'
1055 export const useCounterStore = defineStore('counter', () => {
1056   const count = ref(0)
1057   const doubleCount = computed(() => count.value * 2)
1058   function increment() {
1059     count.value++
1060   }
1061   return { count, doubleCount, increment }
1062 })
1063
1064 //////////////////////////////////////
1065 // FILE: frontend\src\views\AboutView.vue
1066 //////////////////////////////////////
1067 <template>
1068   <div class="about">
1069     <h1>This is an about page</h1>
1070   </div>
1071 </template>
1072 <style>
1073 @media (min-width: 1024px) {
1074   .about {
1075     min-height: 100vh;
1076     display: flex;
1077     align-items: center;
1078   }
1079 }
1080 </style>
1081
1082 //////////////////////////////////////
1083 // FILE: frontend\src\views\HomeView.vue
1084 //////////////////////////////////////
1085 <script setup>
1086 import DroneMap from '../components/DroneMap.vue'
1087 </script>
1088 <template>
1089   <main>
1090     <DroneMap />
1091   </main>
1092 </template>

```