Project Architecture

📊 System Overview

```
┌─────────────────────────────────────────────────────────────┐
│                Traffic Light Detection System               │
│                                                             │
│  ┌──────────────┐      ┌──────────────┐      ┌────────────┐ │
│  │   Upload     │─────▶│   YOLOv5     │─────▶│   Color    │ │
│  │   Image      │      │  Detection   │      │  Analysis  │ │
│  └──────────────┘      └──────────────┘      └────────────┘ │
│                              │                      │       │
│                              ▼                      ▼       │
│                        ┌──────────────────────────────┐     │
│                        │  Classification Results      │     │
│                        │  (Red/Yellow/Green)          │     │
│                        └──────────────────────────────┘     │
│                                                             │
└─────────────────────────────────────────────────────────────┘
```

🏗️ Module Architecture

Core Components

```
┌───────────────────────────────────────────────────┐
│                      app.py                       │
│                (Main Application)                 │
│  ┌──────────────────────────────────────────┐     │
│  │ - Page configuration                     │     │
│  │ - Model initialization                   │     │
│  │ - Application flow orchestration         │     │
│  └──────────────────────────────────────────┘     │
└───────────────────────────────────────────────────┘
             │          │          │
    ┌────────▼──┐  ┌────▼────┐  ┌▼─────────┐
    │  Models   │  │  Utils  │  │    UI    │
    └───────────┘  └─┬───────┘  └┬─────────┘
             │          │          │
             ▼          ▼          ▼
```

1. Models Package (`src/models/`)

```
YOLOModelHandler
    ├── __init__()
    ├── _load_model()        # Load YOLOv5 with caching
    ├── detect(image)        # Perform detection
    ├── is_loaded()          # Check model status
    └── get_model_info()     # Model metadata
```

Purpose: Manages YOLO model lifecycle and inference

2. Utils Package (`src/utils/`)

detection.py

```
# Color Detection Functions
├── detect_traffic_light_color(image, box)
├── _count_red_pixels(hsv_image)
├── _count_green_pixels(hsv_image)
├── _count_yellow_pixels(hsv_image)
├── _determine_dominant_color(r, g, y)
└── validate_color_ranges()
```

image_processing.py

```
# Detection Processing
├── DetectionResult (class)
├── process_detection_results(output, image)
├── get_detection_summary(results)
├── filter_results_by_color(results, color)
└── get_highest_confidence_detection(results)
```

Purpose: Core detection algorithms and result processing

3. UI Package (`src/ui/`)

components.py

```
# UI Components
├── render_header()
├── render_about_section()
├── render_sidebar()
├── render_upload_section()
├── render_detection_result(result)
├── render_detection_results(results)
├── render_summary_statistics(summary)
└── render_annotated_image(image, width)
```

styles.py

```
# Styling
└── apply_custom_styles()
    ├── Traffic light color cards
    ├── Button styling
    └── Layout styling
```

Purpose: User interface rendering and styling

4. Config Package (`config/`)

```
# config.py
├── Application Settings
│   ├── APP_TITLE
│   ├── APP_ICON
│   └── PAGE_LAYOUT
│
├── Model Configuration
│   ├── MODEL_NAME
│   ├── MODEL_REPO
│   └── CONFIDENCE_THRESHOLD
│
├── Color Ranges (HSV)
│   ├── COLOR_RANGES["red"]
│   ├── COLOR_RANGES["green"]
│   └── COLOR_RANGES["yellow"]
│
└── Display Settings
    ├── COLOR_EMOJIS
    └── COLOR_MESSAGES
```

Purpose: Centralized configuration and constants

🔄 Data Flow

Detection Pipeline

```
1. User Upload
   │
   ├─▶ Image File (PNG/JPG)
   │
   ▼
2. Image Loading
   │
   ├─▶ PIL.Image → NumPy Array
   │
   ▼
3. YOLO Detection
   │
```

```
       ├─▶ YOLOModelHandler.detect()
       │
       ├─▶ Bounding boxes for traffic lights
       │
       ▼
    4. Color Analysis
       │
       ├─▶ For each detected traffic light:
       │   ├─▶ Crop region
       │   ├─▶ RGB → HSV conversion
       │   ├─▶ Count red/yellow/green pixels
       │   └─▶ Determine dominant color
       │
       ▼
    5. Result Processing
       │
       ├─▶ Create DetectionResult objects
       │
       ├─▶ Generate summary statistics
       │
       ▼
    6. UI Rendering
       │
       ├─▶ Display color-coded results
       │
       ├─▶ Show annotated image
       │
       └─▶ Present summary metrics
```

🔐 Design Patterns

1. **Separation of Concerns**

- Models: AI/ML logic
- Utils: Business logic
- UI: Presentation logic
- Config: Configuration data

2. **Caching Strategy**

```
@st.cache_resource
def _load_model():
    # Model loaded once and cached
    # Improves performance significantly
```

3. **Error Handling**

```python
try:
    # Attempt operation
except Exception as e:
    logger.error(f"Error: {e}")
    st.error("User-friendly message")
```

4. **Modular Functions**

- Single responsibility
- Clear inputs/outputs
- Well documented
- Testable

📦 Dependencies Graph

```
Streamlit
    ├─▶ UI Rendering
    └─▶ File Upload

PyTorch
    ├─▶ YOLOv5 Model
    └─▶ GPU Acceleration

OpenCV
    ├─▶ Image Processing
    ├─▶ Color Space Conversion
    └─▶ Pixel Masking

NumPy
    ├─▶ Array Operations
    └─▶ Numerical Computing

Pillow
    └─▶ Image Loading
```

🎯 Extension Points

Adding New Features

1. **New Detection Algorithm**
   - Add function to `src/utils/detection.py`
   - Update `config/config.py` if needed
   - Create tests in `tests/`
2. **New UI Component**
   - Add function to `src/ui/components.py`

   - Update styles in `src/ui/styles.py`
   - Call from `app.py`

3. **Different Model**

   - Modify `src/models/yolo_model.py`
   - Update `config/config.py`
   - Adjust detection parameters

4. **Additional Colors**

   - Add ranges to `config/config.py`
   - Extend `detection.py` functions
   - Update UI components

📝 Best Practices Used

✅ **Modular architecture** ✅ **Type hints and docstrings** ✅ **Logging for debugging** ✅ **Configuration management** ✅ **Error handling** ✅ **Code reusability** ✅ **Clear naming conventions** ✅ **Professional structure**