Indiana University Southeast

Test Plan Document for *TrafficLouisville*

Capstone Team: **So Much For Subtlety**

John Brown

Mason Napper

Aaron O'Brien

1/30/2024

Table of Contents:

1.1 Introduction

Our project is called TrafficLouisville, a system that aims to give timely information about the density of traffic on the major highways of Louisville, KY through a simple map interface that can be accessed through the Internet. TrafficLouisville will not be a commercial system, although it could be monetized through ad-revenue, and is not intended to be used in critical applications. Traffic Louisville is an academic project that illustrates potential applications of software development and machine learning applications.

1.2 Functionality Context

This project is intended to demonstrate near real-time data acquisition, analysis, and presentation over a long period of time. The system processes live images, analyzes traffic density, and displays the traffic density as accurately as possible with minimal downtime.

1.3 Test Objectives

1. Ensure the accuracy of traffic density calculated from live images to established ranges through programmatic means and periodic human oversight
2. Validate integration of backend API with the SQLite database.
3. Validate integration of backend API with the React frontend.
4. Establish logging, deployment, and performance analysis plans
5. Identify usability, functionality, and security vulnerabilities in the system.

1.4 Scope

Included components:

1. Frontend heatmap rendering
2. Frontend UI elements
3. Frontend request scheduling, format, and security
4. Frontend response handling and security
5. Backend request and response handling, format, and security
6. Backend Main Loop functions
7. Backend Auxiliary Loop functions
8. Backend Yolov8 functions
9. Backend Database creation and initial population
10. Backend Database CRUD operations
11. Backend Database Validation

Excluded components:

1. External API for weather conditions reliability
2. External image acquisition reliability

2.1 Test Types Identified

Functional Testing: Ensuring each component behaves according to requirements and produces valid output.

<u>Components Involved: Main Loop, Auxiliary Loop, Model Image Analysis, Server Endpoint, Database, Frontend</u>

Unit Testing: Ensuring each function within each component behaves according to requirements

<u>Components Involved: Main Loop, Auxiliary Loop, Model Image Analysis, Server Endpoint, Database, Frontend</u>

Integration Testing: Ensuring interactions between each component behaves according to requirements and produces valid output given valid input in previous components.

Component Interactions Involved:

Main Loop -> Database

Auxiliary Loop -> Database

Main Loop -> Model Image Analysis

Database -> Server Endpoint

Server Endpoint -> Frontend

Frontend -> Server Endpoint

Performance Testing: Stress testing at maximum throughput and analyzing issues

Entire system involved running at simulated full operation.

Security Testing: Ensuring security measures are working correctly.

Involved Components: Frontend and Server Endpoint

Usability Testing: Validating interactions with frontend

Involved Components: Frontend

2.1.1 Problems Perceived

1. Potential CPU bottlenecks when running at full operation during specific bursts of activity (main loop and auxiliary loop updating database while Yolov8 is utilizing CPU for image analysis)
2. Managing inconsistent or malformed data from third-party APIs.
3. Handling edge cases for time-based logic.
4. Handling of exceptions and restarting server from downtime.

3.1 Architecture

Our system consists of the following components:

- **Data Source Retrieval (Main Loop and Auxiliary Loop on Backend Local Hardware)** - Downloading an image at a fixed rate from CCTV camera links, retrieving camera metadata from ArcGIS REST API, retrieving weather conditions that affect image processing, responsible for error handling that may arise from external factors
- **Image Processing (Model on Backend Local Hardware)** - Applying appropriate model for detecting and classifying vehicles within retrieved images using the YOLOv8 model, will potentially account for camera orientation and zooming through TBD
- **Image Analysis (Model, Database, and Server Endpoint on Backend Local Hardware)**– Converting object detection results into traffic counts, storing traffic data and calculating density
- **Database Management (Database on Backend Local Hardware)**– Stores information retrieved from data source, current and historical traffic counts to allow for calculating traffic density and supports CRUD operations from other subsystems to keep them independent.
- **Server Endpoint (on Backend Local Hardware)** – Receive GET request from frontend and respond by retrieving information from database, verifies authenticity of request
- **Frontend (on Remote Cloud Hosting Platform)** – Sending request to endpoint, receiving it and displaying it at the correct position on the heat map, displaying other relevant information, encrypts request and verifies authenticity and integrity of information in response

3.2 Environment

Development Environment:

Python 3.10, Flask, SQlite, PyTorch (Yolov8)

React frontend with react-leaflet library

Windows 11 OS

Testing Environment:

Unittest and Pytest libraries for Backend Functionality (including Model Testing)

3.3 Assumptions

1. External APIs are reliable and functional during tests
2. Yolov8 model is trained and model weights are in directory and ready
3. Static frontend map for frontend is accurate and ready

3.4 Security

- Validate endpoints for security effectiveness
- Ensure sensitive environment variables are not present in public facing documentation
- Validate file upload handling for vulnerabilities from APIs

3.5 Schedule

Phase	Date to be completed by	Tasks
Unit Testing	2/11/2025	Complete unit tests for all components, complete setting up local testing environment for frontend<->backend interactions
Integration Testing	2/18/2025	Complete tests that validate communication between components, keep system running locally
Performance Testing	2/25/2025	Implement logs that capture performance including CPU utilization, delays, and missing data
Security Testing	3/4/2025	Validate that system is ready to go into production
Usability Testing	3/11/2025	Finishing up touches on frontend

3.6 Test Team Organization

John Brown – Responsible for testing the main loop, auxiliary loop, database, and server endpoint.

Mason Napper – Responsible for testing involving the frontend.

Aaron O'Brien – Responsible for testing the model image analysis and establishing accuracy limits.

3.7 Defects Classification Mechanism

Critical Defects: Total system crash, database corruption during operation, non-functional image analysis

High Defects: Performance issues related to CPU utilization, analysis cycles longer than 10 seconds.

Medium Defects: Poor accuracy in traffic estimation

Low Defects: Issues in frontend display

3.8 Configuration Management

Github is the public repository for the project, with Vercel deploying from the frontend directory.

Caddyfile config and .env files are NOT public.

3.9 Release Criteria

Backend and frontend pass greater than 95% of unit tests, 100% of functional and integration tests.

Model estimation of traffic density > 85%.

Frontend heatmap updates according to 10 second update cyle.

No critical, high, or medium defects.

Approval from KYTC and sponsor to deploy project.