**Test Plan Template:**

website Vehicle Detection

**Prepared by:**

Budnik Sergiy

18.03.24

**TABLE OF CONTENTS**

1.0 INTRODUCTION

2.0 OBJECTIVES AND TASKS

2.1 Objectives

2.2 Tasks

3.0 SCOPE

4.0 Testing Strategy

4.1 Alpha Testing (Unit Testing)

4.2 System and Integration Testing

4.3 Performance and Stress Testing

4.4 User Acceptance Testing

4.5 Batch Testing

4.6 Automated Regression Testing

4.7 Beta Testing

5.0 Hardware Requirements

6.0 Environment Requirements

6.1 Main Frame

6.2 Workstation

7.0 Test Schedule

8.0 Control Procedures

9.0 Features to Be Tested

10.0 Features Not to Be Tested

11.0 Resources/Roles & Responsibilities

12.0 Schedules

13.0 Significantly Impacted Departments (SIDs)

14.0 Dependencies

15.0 Risks/Assumptions

16.0 Tools

17.0 Approvals

**1.0 INTRODUCTION**

The Test Plan for the Site Vehicle Detection from Video project outlines the strategy and methodologies for ensuring the robustness and accuracy of the vehicle detection system. This system aims to detect vehicles within video footage captured by surveillance cameras deployed at various sites. By leveraging advanced computer vision algorithms, the system identifies and tracks vehicles in real-time, providing valuable insights for traffic management, security monitoring, and data analysis purposes.

**2.0 OBJECTIVES AND TASKS**

**2.1 Objectives**

The objectives supported by the Master Test Plan for the Site Vehicle Detection from Video project include:

* **Defining Tasks and Responsibilities:** Clearly outline the roles and responsibilities of team members involved in testing activities, including testers, developers, project managers, and stakeholders.
* **Vehicle for Communication:** Establish effective channels of communication to facilitate collaboration and information sharing among team members, ensuring smooth progress of testing efforts.
* **Document as a Service Level Agreement (SLA):** Use the Test Plan document as a formal agreement between the development and testing teams, outlining the quality standards, testing procedures, and acceptance criteria for the vehicle detection system.

**2.2 Tasks**

**Test Planning:** Define the scope and objectives of testing. Identify test resources, including personnel, tools, and equipment. Develop the Test Plan document outlining the testing approach, methodologies, and schedules.

**Test Case Development:** Create test cases covering all functional and non-functional requirements of the vehicle detection system. Ensure test cases are comprehensive, clear, and traceable to requirements.

**Test Execution:** Conduct testing according to the predefined test cases and procedures. Verify the functionality, performance, and reliability of the system under various scenarios and conditions.

**Defect Reporting and Tracking:** Document any defects or issues encountered during testing using a designated defect tracking system. Assign priorities and severities to reported defects for efficient resolution.

**Regression Testing:** Perform regression testing to ensure that changes or fixes do not adversely affect existing functionality. Re-run previously executed test cases to validate the stability of the system after modifications.

**Documentation and Reporting:** Maintain comprehensive documentation of test results, including test logs, defect reports, and test coverage metrics. Generate test reports summarizing the outcomes of testing activities and highlighting any critical findings or observations.

**Post-Testing Activities:** Conduct post-testing reviews to evaluate the effectiveness of testing processes and identify areas for improvement.

**3.0 SCOPE**

**General:**

The scope of the Test Plan covers testing all functions of the vehicle detection system, including:

* Functionality testing of the vehicle detection algorithm.
* Performance evaluation of object tracking mechanisms.
* Verification of alert generation and data logging/reporting functionalities.
* Integration testing to ensure seamless operation.

**Tactics:**

1. **Stakeholder Engagement:** Schedule meetings to discuss testing requirements and gather feedback.
2. **Test Environment Setup:** Allocate resources for setting up a test environment mirroring production.
3. **Test Case Development:** Collaborate to create comprehensive test cases.
4. **Test Execution Planning:** Develop a detailed test execution plan.
5. **Communication and Collaboration:** Establish clear communication channels for reporting progress.
6. **Documentation and Reporting:** Maintain thorough documentation and generate regular test reports.

**4.0 TESTING STRATEGY**

The testing strategy for the Site Vehicle Detection from Video project follows a systematic approach aimed at ensuring comprehensive coverage of all major feature groups. Each feature group will undergo rigorous testing to verify its functionality, performance, and reliability. The following outlines the overall approach to testing along with the major activities, techniques, and tools employed for testing designated feature groups:

1. Vehicle Detection Algorithm:

Approach: Utilize a combination of manual and automated testing techniques to evaluate the accuracy and robustness of the vehicle detection algorithm.

Major Activities: Develop test cases to simulate various scenarios, including different lighting conditions, vehicle sizes, and angles. Use sample video footage with known ground truth annotations for validation.

Techniques and Tools: Manual inspection of detected vehicles in video frames. Automated testing using computer vision libraries such as OpenCV. Performance evaluation using precision, recall, and F1-score metrics.

2. Object Tracking Mechanisms:

Approach: Employ a combination of synthetic and real-world data to assess the effectiveness of object tracking mechanisms.

Major Activities: Create test scenarios to evaluate tracking accuracy under conditions like occlusion, cluttered scenes, and varying object speeds. Assess the system's ability to handle sudden changes in object trajectories.

Techniques and Tools: Use synthetic datasets generated with ground truth annotations for controlled testing. Utilize real-world video footage captured in different environments for validation. Evaluate tracking performance using metrics such as average tracking error and tracking success rate.

3. Alert Generation and Data Logging/Reporting:

Approach: Conduct functional and stress testing to verify the reliability and responsiveness of alert generation and data logging/reporting functionalities.

Major Activities: Generate test cases to simulate various triggering conditions for alerts, such as vehicle intrusion or abnormal traffic patterns. Stress test the system by simulating high-volume traffic scenarios to assess data logging and reporting capabilities.

Techniques and Tools: Automated test scripts to simulate triggering events and verify alert generation. Load testing tools to assess system performance under high load conditions. Manual inspection of generated alerts and logged data for accuracy and completeness.

4. Integration Testing:

Approach: Perform end-to-end testing to validate the seamless integration of all system components.

Major Activities: Execute test scenarios covering the entire workflow of the vehicle detection system, from video input processing to alert generation and reporting.Validate interoperability between different modules and subsystems.

Techniques and Tools: Use test automation frameworks to orchestrate end-to-end test scenarios. Conduct system integration testing in a controlled environment that mirrors production.

**4.1 Unit Testing**

Definition:

Unit testing involves testing individual units or components of the software in isolation to ensure they function correctly. The minimum degree of comprehensiveness desired for unit testing is to achieve high code coverage, aiming to execute all statements, branches, and paths within the code. Techniques such as code coverage analysis will be used to judge the comprehensiveness of the testing effort. Additionally, completion criteria will include achieving a low error frequency in unit tests.

Participants: Developers responsible for implementing specific units or components of the vehicle detection system.

Methodology: Unit testing will be conducted by the developers who are responsible for writing the code. Test scripts for unit testing will be written by the developers themselves, following the Test-Driven Development (TDD) approach where applicable. The sequence of events for unit testing will typically involve the following steps:

Developers write unit test cases based on the specifications and requirements of the component.

Unit tests are executed against the component in isolation to verify its functionality.

Test results are analyzed to identify any failures or deviations from expected behavior.

Developers debug and fix issues identified during testing.

Unit tests are rerun to ensure that fixes have been successful.

Code coverage analysis is performed to assess the comprehensiveness of the testing effort.

**4.2 System and Integration Testing**

Definition:

System testing involves testing the entire system as a whole to ensure that all components work together correctly to fulfill the requirements and objectives of the project. Integration testing focuses on verifying the interactions and interfaces between individual components to ensure seamless integration.

Participants:

Testers responsible for verifying the overall functionality and integration of the vehicle detection system.

Developers involved in resolving integration issues and ensuring component interoperability.

Methodology:

System and integration testing will be conducted by a dedicated testing team in collaboration with developers responsible for different components of the system. The methodology for system and integration testing will involve the following steps:

Testers develop test scripts based on system requirements and use cases.

Integration testing is performed to verify the interactions between individual components and subsystems.

System testing is conducted to validate the end-to-end functionality of the entire vehicle detection system.

Test results are analyzed to identify any integration issues, functionality gaps, or deviations from expected behavior.

Issues identified during testing are logged, prioritized, and communicated to developers for resolution.

Regression testing is conducted to ensure that fixes and changes do not introduce new issues.

Final acceptance testing is performed to verify that the system meets the specified requirements and objectives before deployment.

**4.3 Performance and Stress Testing**

Definition:

Performance testing evaluates the responsiveness, throughput, reliability, and scalability of the system under normal load conditions, while stress testing assesses the system's behavior and performance under extreme conditions beyond its normal operational limits.

Participants:

Testing team members responsible for performance and stress testing.

System architects and developers involved in optimizing system performance.

Methodology:

Performance and stress testing will be conducted by the testing team with support from system architects and developers. The methodology involves the following steps:

Test Planning: Define performance metrics and objectives based on system requirements and user expectations.

Test Environment Setup: Configure a test environment that mirrors the production environment in terms of hardware, software, and network infrastructure.

Test Scenario Design: Develop test scenarios to simulate typical usage patterns, including varying numbers of concurrent users and traffic loads.

Test Script Development: Write test scripts to automate the execution of performance and stress tests, including load generation, transaction monitoring, and resource utilization measurement.

Test Execution: Execute performance tests using load testing tools to simulate expected user loads and stress tests to push the system beyond its limits.

Performance Monitoring: Monitor system performance metrics such as response time, throughput, CPU utilization, memory usage, and network bandwidth during test execution.

Analysis and Optimization: Analyze test results to identify performance bottlenecks, scalability issues, and areas for optimization. Collaborate with developers to address identified issues and optimize system performance.

Reporting: Document test results, including performance metrics, observations, and recommendations for improvement.

**4.4 User Acceptance Testing**

Definition:

User acceptance testing (UAT) validates that the system meets the requirements and expectations of end-users (customers) and is ready for operational use.

Participants:

End-users (customers) who will be using the system in real-world scenarios.

Testing team members responsible for facilitating and coordinating UAT activities.

Methodology:

User acceptance testing will be conducted in collaboration with end-users and the testing team. The methodology involves the following steps:

Test Planning: Define UAT objectives, acceptance criteria, and test scenarios based on system requirements and user needs.

Test Environment Setup: Configure a UAT environment with representative data and configurations.

Test Script Development: Write test scripts or scenarios to guide end-users through the validation of system functionality and usability.

User Training: Provide training and guidance to end-users on how to perform UAT and provide feedback effectively.

Test Execution: End-users execute test scenarios and validate system functionality against predefined acceptance criteria.

Issue Reporting: End-users report any issues, defects, or deviations from expected behavior encountered during UAT.

Issue Resolution: Developers address reported issues and make necessary adjustments to the system based on user feedback.

Final Acceptance: End-users review and validate the system changes made in response to reported issues. Upon successful validation, the system is considered ready for operational use.

Documentation: Document UAT results, including identified issues, resolutions, and any changes made to the system based on user feedback.

**4.5 Batch Testing**

Validate bulk operations such as batch processing orders.

**4.6 Automated Regression Testing**

Implement automated tests to ensure new updates do not affect existing functionality negatively.

**4.7 Beta Testing**

Release a beta version to a limited audience for real-world feedback.

**5.0 HARDWARE REQUIREMENTS**

**Computers:**

A high-performance server or workstation with multi-core CPU, ample RAM, and dedicated GPU for efficient video processing.

Client devices such as desktops or laptops with modern CPUs and sufficient RAM for accessing the system's user interface.

**Modems:**

Modems may be necessary to establish network connectivity between surveillance cameras and the central server. The type of modem required depends on the network infrastructure in use.

**Additional Considerations:**

Redundancy: Implementing redundant hardware components minimizes downtime.

Scalability: Planning for scalability accommodates future growth and increased processing demands.

Environmental factors: Choose hardware suitable for deployment environments, considering factors like temperature and humidity.

**6.0 ENVIRONMENT REQUIREMENTS**

**6.0 Environment Requirements**

**6.1 Main Frame**

Necessary Properties:

The test environment should include a dedicated mainframe or server capable of handling video processing tasks efficiently.

Hardware specifications should include a multi-core CPU, ample RAM, and a powerful GPU for accelerated processing of computer vision algorithms.

Adequate storage space, preferably SSD, is necessary for storing video data and intermediate results.

The system software should include the necessary operating system and middleware required for running the vehicle detection application.

Communications infrastructure, including network switches and routers, should be in place to facilitate communication between surveillance cameras and the mainframe.

The test environment may be stand-alone or integrated with existing IT infrastructure, depending on project requirements.

Security measures should be implemented to safeguard proprietary components such as software, data, and hardware. This includes access control mechanisms, encryption protocols, and intrusion detection systems.

Special test tools such as load testing software, debugging tools, and performance monitoring utilities may be required to conduct comprehensive testing.

Desired Properties:

High availability and fault tolerance features to minimize downtime during testing.

Scalability to accommodate increasing data volumes and processing demands.

Redundant power supplies and backup systems to ensure continuous operation.

Environmental controls to maintain optimal operating conditions, including temperature and humidity regulation.

**6.2 Workstation**

The workstation requirements for testing may include:

Standard desktop or laptop computers for test engineers to develop and execute test cases.

These workstations should meet minimum hardware specifications, including a modern CPU, sufficient RAM, and adequate storage space.

Test engineers may require access to software development tools, testing frameworks, and collaboration platforms for efficient test execution.

**7.0 Test Schedule**

The test schedule will align with the milestones identified in the Software Project Schedule. Additional test milestones may be defined based on the specific testing needs of the project.

Testing Tasks and Milestones:

Test Planning: Estimate time required - 1 week

Test Case Development: Estimate time required - 2 weeks

Unit Testing: Estimate time required - 2 weeks

System and Integration Testing: Estimate time required - 3 weeks

Performance and Stress Testing: Estimate time required - 2 weeks

User Acceptance Testing: Estimate time required - 2 weeks

Final Review and Acceptance: Estimate time required - 1 week

Schedule for Testing Tasks:

Test Planning: Weeks 1-2

Test Case Development: Weeks 3-5

Unit Testing: Weeks 6-7

System and Integration Testing: Weeks 8-10

Performance and Stress Testing: Weeks 11-12

User Acceptance Testing: Weeks 13-14

Final Review and Acceptance: Week 15

Resource Allocation:

Mainframe/Server: Continuous throughout testing phase

Workstations: Continuous throughout testing phase

Test Tools: Available as needed during respective testing phases

Staff: Test engineers and developers allocated as per testing schedule, with dedicated periods for testing activities.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Duration** | **Start Date** | **End Date** |
| Test Planning | 1 week | Week 1 | Week 2 |
| Test Case Development | 2 weeks | Week 3 | Week 5 |
| Unit Testing | 2 weeks | Week 6 | Week 7 |
| System and Integration Testing | 3 weeks | Week 8 | Week 10 |
| Performance and Stress Testing | 2 weeks | Week 11 | Week 12 |
| User Acceptance Testing | 2 weeks | Week 13 | Week 14 |
| Final Review and Acceptance | 1 week | Week 15 | Week 5 |

**8.0 CONTROL PROCEDURES**

Version control for test cases and documentation.

Change management process for handling updates.

**9.0 FEATURES TO BE TESTED**

All features outlined in the project requirements document.

**10.0 FEATURES NOT TO BE TESTED**

Features explicitly excluded from the project scope.

**11.0 RESOURCES/ROLES & RESPONSIBILITIES**

Testers: Execute tests and report defects.

Developers: Fix reported defects and implement changes.

Project Manager: Oversee testing process and coordinate with stakeholders.

**12.0 SCHEDULES**

Test execution schedule outlined in section 7.0.

**13.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)**

Development team

Customer support

**14.0 DEPENDENCIES**

Availability of development resources for fixing defects.

Timely feedback from stakeholders during user acceptance testing.

**15.0 RISKS/ASSUMPTIONS**

Risks: Potential delays due to technical issues or changes in requirements.

Assumptions: Availability of required resources.

**16.0 TOOLS**

Test management tools (e.g., Jira, TestRail)

Automation testing tools (e.g., Selenium, Cypress)

**17.0 APPROVALS**

Specify the names and titles of all persons who must approve this plan. Provide space for the signatures and dates.

Vasylchuk Oleksandr + 18.04.24

Bukevich Ilya + 18.04.24

Budnik Sergiy + 18.04.24

Yednak Ivan + 18.04.24