Test Document

1. Document Information

1.1 Document Control Information

- **Document Title:** Smart Maintenance Platform for Aero Engine Industrial Equipment - Test Document
- **Version:** 1.0 (Initial Draft)
- **Date:** April 6, 2025
- **Status:** In Progress
- **Document Owner:** Testing Team

1.2 Revision History

Version	Date	Description	Author
0.1	March 30, 2025	Initial structure	Jingxiao Han
0.5	April 3, 2025	Test strategy draft	Jingxiao Han
1.0	April 6, 2025	Completed test plan framework	Jingxiao Han

1.3 Approvals

Name Role Signature Date

1.4 Document Purpose

This document defines the testing strategy, methodology, and plan for the Smart Maintenance Platform for Aero Engine Industrial Equipment. It provides a comprehensive testing methodology to ensure system quality and serves as a guide for test execution and verification.

1.5 Audience

- Testing Team
- Development Team
- Project Management
- Quality Assurance Team
- Project Stakeholders

2. Introduction

2.1 Test Overview

This test document outlines a comprehensive testing approach for the Smart Maintenance Platform for Aero Engine Industrial Equipment. Testing activities will span the entire development lifecycle, from unit testing through system integration testing to user acceptance testing. Testing aims to validate functional and non-functional requirements, ensure system reliability, performance, and security, and verify that the system meets all stakeholder expectations.

2.2 Test Scope

Testing will cover all key components and functionalities of the system, including:

- Data acquisition and processing modules
- Machine learning models (anomaly detection and remaining useful life prediction)
- User interface and visualization components
- Alerting and notification system
- Reporting generation functionality
- System integration and data flow
- Security and access control
- System performance and scalability

2.3 Test Objectives

- Validate correct implementation of all functional requirements
- Ensure system meets performance and scalability requirements
- Verify accuracy and reliability of machine learning models
- Detect and eliminate defects and vulnerabilities in the system
- Confirm system usability and user experience
- Validate system integration with existing maintenance systems
- Ensure compliance with relevant aviation industry safety and regulatory standards

2.4 Test Methodology

The system will employ an agile testing methodology, synchronized with development iterations. This includes:

- Continuous integration and continuous testing
- Combination of automated and manual testing
- Risk-based test prioritization
- Test-driven development (TDD) principles for critical components
- Regression testing to ensure changes don't impact existing functionality
- Model validation and periodic re-evaluation

2.5 Test Environment Requirements

Testing will be conducted in the following environments:

- Development environment: for unit testing and preliminary integration testing
- Test environment: an isolated environment similar to production for functional and integration testing
- Performance testing environment: dedicated environment configured to simulate actual load
- Pre-production environment: configuration closest to production for final system testing and user acceptance testing

Specific hardware and software configurations for each environment will be detailed in subsequent sections.

3. Test Strategy

3.1 Test Levels

Based on the test strategy developed by Jingxiao Han, the system testing will be divided into four levels:

3.1.1 Unit Testing

- **Objective**: Verify correct functioning of individual software components
- **Scope**: All key services, classes, and methods
- **Approach**: Automated unit tests with appropriate mocks to isolate dependencies
- **Tools**: Unit testing frameworks (e.g., PyTest, JUnit, etc., depending on implementation language)
- **Responsible**: Development team members
- **Timing**: Performed concurrently with component development

3.1.2 Integration Testing

- **Objective**: Verify interfaces and data flow between components
- **Scope**: All interfaces between components and services
- **Approach**: Combined bottom-up and top-down integration testing approaches
- **Tools**: Integration testing frameworks and API testing tools
- **Responsible**: Development and testing teams
- **Timing**: Performed after unit tests have passed

3.1.3 System Testing

- **Objective**: Verify functional and non-functional requirements of the entire system
- **Scope**: End-to-end system functionality, including anomaly detection, life prediction, alerts, and reporting
- **Approach**: Combination of black-box testing, exploratory testing, and scripted testing
- **Tools**: Automated testing tools, performance testing tools, and security testing tools
- **Responsible**: Testing team
- **Timing**: Performed after successful integration testing

3.1.4 User Acceptance Testing

- **Objective**: Confirm the system meets business requirements and user expectations
- **Scope**: Key business scenarios and user workflows
- **Approach**: Scenario-based testing and user experience evaluation
- **Tools**: User acceptance test scripts and feedback collection tools
- **Responsible**: Testing team and business stakeholders
- **Timing**: Performed after successful system testing

3.2 Test Types

3.2.1 Functional Testing

- Verify all system functions work according to specifications
- Check accuracy of data acquisition and processing

- Validate correctness of anomaly detection and life prediction
- Test alert generation and delivery mechanisms
- Verify report generation and query functionality

3.2.2 Performance Testing

- Load testing to evaluate system performance under expected user numbers
- Stress testing to determine system limits
- Endurance testing to verify long-term stability
- Response time testing to ensure system responses meet requirements
- Data processing speed testing, particularly for large volumes of sensor data

3.2.3 Security Testing

- Authentication and authorization testing
- Data protection and privacy testing
- Input validation and injection prevention testing
- API security testing
- Security compliance verification

3.2.4 Usability Testing

- User interface navigation and workflow testing
- Accessibility checks
- Error handling and user feedback evaluation
- Responsive design verification
- User experience assessment

3.2.5 Reliability Testing

- Fault recovery testing
- Data consistency verification
- System availability measurement
- Failover testing (if applicable)
- Backup and recovery testing

3.3 Entry and Exit Criteria

3.3.1 Test Entry Criteria

- Requirements documentation has been reviewed and approved
- Test environment has been configured and is ready
- Test data has been prepared
- Test plan and test cases have been written
- Development team has completed the relevant functionality and is ready for testing

3.3.2 Test Exit Criteria

- All planned test cases have been executed
- All high and medium priority defects have been resolved
- All critical paths and functionalities have been verified
- Performance metrics meet requirements
- Stakeholder approval of test results has been obtained

3.4 Test Tools and Frameworks

Based on project requirements, the testing team plans to use the following tools and frameworks:

- **Unit testing tools**: PyTest (Python), JUnit (Java), Jest (JavaScript)
- **API testing tools**: Postman, REST-assured
- **Performance testing tools**: JMeter, Locust
- **Automation testing frameworks**: Selenium, Cypress
- **Continuous integration tools**: Jenkins, GitHub Actions
- **Test management tools**: Jira, TestRail
- **Security testing tools**: OWASP ZAP, SonarQube
- **Model evaluation tools**: Scikit-learn, TensorFlow evaluation tools

Specific tool selection will be finalized in the next phase.

3.5 Test Data Requirements

Testing will require the following types of data:

- Historical engine operational data (normal and abnormal states)
- Synthetic sensor data to simulate various operating conditions
- Boundary condition datasets for limit testing
- Labeled datasets with known anomalies for validating anomaly detection models
- Labeled datasets with known remaining useful life for validating prediction models
- Data streams simulating large numbers of concurrent users and devices for performance testing

Test data preparation strategy will include anonymization, synthetic data generation, and data augmentation techniques.

3.6 Defect Management Process

Defects discovered during testing will be managed through the following process:

1. **Defect identification and recording**: Testers record discovered defects in the test management system
2. **Defect classification**: Defects are classified by severity and priority
3. **Defect assignment**: Defects are assigned to appropriate developers
4. **Defect fixing**: Developers fix defects and submit updates
5. **Defect verification**: Testers verify defect fixes
6. **Defect closure**: Defects are closed after successful verification

Defect severity will be classified into four levels: Critical, High, Medium, and Low, with detailed definitions to be provided in the complete test document.

4. Test Plan

4.1 Test Schedule

Testing activities will align with the project development plan. The preliminary schedule is as follows:

Phase	Testing Activity	Start Date	End Date	Responsible Party
Design Phase	Test Plan Development	April 7, 2025	April 15, 2025	Jingxiao Han

Design Phase	Test Case Design	April 15, 2025	April 30, 2025	Testing Team
Development Phase	Unit Testing	April 29, 2025	Ongoing	Development Team
Development Phase	Integration Testing	May 5, 2025	May 20, 2025	Testing Team
System Integration	System Testing	May 21, 2025	June 3, 2025	Testing Team
System Acceptance	User Acceptance Testing	June 4, 2025	June 15, 2025	Testing Team/Business Representatives

This schedule will be adjusted based on project progress.

4.2 Resource Allocation

Preliminary resource allocation for testing activities is as follows:

- **Testing team**: 3 test engineers, including 1 test lead (Jingxiao Han)
- **Development support**: Development team members required to assist with issues discovered during testing
- **Environment support**: Infrastructure team required to configure and maintain test environments
- **Business experts**: Domain experts required for user acceptance testing
- **Testing tools**: Licenses and support for all necessary testing tools

Detailed resource allocation plans will be completed in the next phase.

4.3 Risk Assessment and Mitigation

Key risks and mitigation strategies for the testing process include:

Risk	Impact	Likelihood	Mitigation Strategy
Insufficient test data	High	Medium	Develop synthetic data generation tools, identify data requirements early
Performance test environment cannot	High	Medium	Configure performance test environment as close as possible to

simulate actual load			production, use simulation tools
Test automation script maintenance challenges	Medium	High	Adopt modular testing frameworks, establish automation testing standards
Tight testing schedule	High	Medium	Prioritize testing, automate critical paths, parallel testing activities
Model validation complexity	High	High	Develop specialized model testing frameworks, ensure soundness of model validation methodology

Risks will be continuously assessed and updated throughout the project.

4.4 Responsibilities

Responsibilities of the testing team and related participants are as follows:

- **Test Lead (Jingxiao Han)**:
    - Develop test strategy and plans
    - Coordinate testing activities
    - Oversee test execution
    - Report test status to project management
- **Test Engineers**:
    - Design and execute test cases
    - Report and track defects
    - Execute automated test scripts
    - Participate in test result analysis
- **Development Team**:
    - Execute unit tests
    - Fix reported defects
    - Support integration testing activities
    - Participate in technical issue resolution
- **Business Analysts**:
    - Clarify functional requirements

    - Participate in acceptance test criteria definition
    - Support user acceptance testing
    - Provide business domain knowledge

5. Test Cases (Preliminary Framework)

5.1 Core Functionality Test Cases

Based on the current project phase, the following are preliminary core functionality testing areas. Detailed test cases will be developed in the next phase:

5.1.1 Device Center Module Test Cases

- Device registration and configuration
- Device information query and update
- Device categorization and grouping functionality
- Device history data access
- Device status monitoring

5.1.2 Monitoring Center Module Test Cases

- Real-time data display
- Historical trend charts
- Threshold setting and monitoring
- Custom dashboard creation
- Data export and report generation

5.1.3 Data Simulation Module Test Cases

- Simulation data generation
- Scenario configuration and management
- Simulation run control
- Result analysis and visualization
- Comparison of simulated data with actual data

5.1.4 Alert System Module Test Cases

- Alert rule configuration

- Alert triggering mechanisms
- Alert notification methods
- Alert acknowledgment and management
- Alert history and analysis

5.1.5 Reporting System Module Test Cases

- Standard report generation
- Custom report creation
- Report scheduling and distribution
- Report export functionality
- Report data accuracy

5.2 Non-Functional Test Cases

5.2.1 Performance Test Cases

- System response time
- Concurrent user handling capacity
- Data processing throughput
- Long-term stability testing
- Resource usage monitoring

5.2.2 Security Test Cases

- User authentication and authorization
- Data encryption and protection
- Security vulnerability scanning
- API security testing
- Audit log verification

5.2.3 Usability Test Cases

- User interface navigation
- Error message clarity
- Responsive design adaptation

- Help and documentation accessibility
- Overall user experience evaluation

5.3 Machine Learning Model Test Cases

5.3.1 Anomaly Detection Model Test Cases

- Model accuracy validation
- False positive and false negative rate evaluation
- Model robustness testing
- Performance comparison across different datasets
- Boundary condition testing

5.3.2 Life Prediction Model Test Cases

- Prediction accuracy validation
- Prediction stability testing
- Performance under different operating conditions
- Prediction error analysis
- Model iteration improvement validation

6. Test Procedures

6.1 Test Execution Process

Test execution will follow this process:

1. **Test preparation**:
    - Prepare test environment
    - Load test data
    - Prepare test tools and scripts
2. **Test execution**:
    - Execute test cases according to test plan
    - Record test results
    - Report discovered defects
3. **Test analysis**:

    - Analyze test results
    - Evaluate test coverage
    - Determine if additional testing is needed
4. **Test reporting**:
    - Generate test summary reports
    - Provide test metrics and measurements
    - Submit test results to stakeholders

Detailed test execution procedures will be provided in the complete test document.

6.2 Test Data Setup Procedures

Test data will be prepared through the following procedures:

- Collect and anonymize real historical data
- Generate synthetic data to supplement real data
- Create test data for boundary conditions and abnormal situations
- Prepare high-volume data for performance testing
- Establish data validation procedures to ensure data quality

6.3 Test Reporting Procedures

Test reports will include the following:

- Test summary and scope
- Test execution status
- Defects found and their status
- Quality metrics and trends
- Risks and issues
- Recommendations and next actions

6.4 Defect Resolution Workflow

Defect resolution will follow this workflow:

1. Tester discovers and records defect
2. Defect review and classification

3. Defect assignment to developer
4. Developer fixes and updates status
5. Tester verifies fix
6. Defect closure or reopening

7. Test Reports

7.1 Test Summary Report Template

The test summary report will include the following sections:

- Project information and test scope
- Test execution status overview
- Summary of key metrics
- Major findings and issues
- Risks and mitigation measures
- Conclusions and recommendations

7.2 Defect Report Template

Each defect report will include:

- Defect ID and title
- Severity and priority
- Detailed description
- Steps to reproduce
- Expected vs. actual results
- Environment information
- Attachments (screenshots, logs, etc.)

7.3 Progress Report Template

The test progress report will include:

- Reporting period
- Completed testing activities
- Test execution statistics

- Defect statistics and trends
- Blockers and risks
- Next cycle plan

7.4 Final Test Report Template

The final test report will comprehensively summarize testing activities, including:

- Test scope and objectives
- Summary of testing activities
- Test results and quality assessment
- Unresolved issues and risks
- Recommended improvements and lessons learned
- Final testing conclusions

8. Traceability Matrix

8.1 Requirements to Test Cases

The traceability matrix will ensure each requirement has corresponding test coverage:

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Status
[To be filled after requirements refinement]				

8.2 Test Cases to Results

Test execution results tracking matrix:

Test Case ID	Test Name	Execution Date	Executed By	Status	Related Defects
[To be filled during test execution phase]					

9. Appendices

9.1 Test Environment Setup

[Test environment configurations will be detailed in the next phase]

9.2 Test Datasets

[Test dataset specifications will be developed in the next phase]

9.3 Glossary

- **RUL**: Remaining Useful Life
- **PdM**: Predictive Maintenance
- **TDD**: Test-Driven Development
- **UAT**: User Acceptance Testing
- **API**: Application Programming Interface
- **CI/CD**: Continuous Integration/Continuous Deployment
- **QPS**: Queries Per Second
- **MTBF**: Mean Time Between Failures
- **MTTR**: Mean Time To Repair

9.4 References

- Project Requirements Specification
- System Design Document
- Quality Assurance Standards
- Industry Testing Best Practices
- [To be expanded in the next phase]