Project Charter

0. Table of Content

1. Project Name

Proactive Shield: Smart Maintenance Platform for Aero Engine Industrial Equipment

2. Project Stakeholders

Name	Role	Position	Contact Information
Chunyan Duan	Sponsor	CEO	duanchunyan77@163.com
Xuanhe Yang	Project Manager	Manager	2252709@tongji.edu.cn
Jingxiao Han	Team Member	Programmer	2251548@tongji.edu.cn
Tong Li	Team Member	Programmer	2251653@tongji.edu.cn
Fubin Chen	Team Member	Programmer	1958440015@qq.com

3. Project Introduction

3.1 Background

In modern industrial production, equipment maintenance is a key factor in ensuring production efficiency, reducing downtime, and lowering operating costs. Traditional maintenance strategies (such as post-maintenance or regular maintenance) are often inefficient and may lead to excessive maintenance or sudden failures. Predictive maintenance (PdM) based on artificial intelligence (AI) and Machine Learning (ML) can predict equipment failures in advance and become a better solution.

However, existing predictive maintenance systems still face problems such as insufficient prediction accuracy, poor real-time adaptability, and difficult system integration. To solve these problems, this project proposes a neural networks-based equipment remaining life (RUL) prediction web system, which uses artificial neural networks and long short-term memory network (LSTM) models, combined with real-time sensor data (such as temperature, vibration, pressure, etc.) to achieve early warning of faults and optimize maintenance plans. It also provides users with an intuitive visual interface to help enterprises transform from traditional maintenance mode to intelligent Facility Management.

3.2 Description of the challenge or opportunity

3.2.1 Challenge

- **High Technical Implementation Difficulty**
  - **Accuracy Achievement:** The accuracy of the AI model is key to realizing the platform's core value. However, the complexity of equipment failure data (e.g., imbalanced datasets, noise interference) may lead to prediction accuracy falling short of expectations.
  - **System Integration:** Technical obstacles exist in integrating with legacy systems (e.g., maintenance work order systems, SCADA), such as missing interface documentation or outdated protocols, which could increase development effort.
  - **Real-Time Requirements:** The high demands for real-time performance and consistency in IoT sensor data require resolving performance bottlenecks in data transmission, storage, and processing.
- **Stringent Resource and Time Constraints**

  Budget limitations force the team to prioritize cost-effective technical solutions, potentially sacrificing some customized features. With less than a year remaining until the delivery deadline (by June 10, 2025) and a fixed team size, any technical risks or requirement changes could jeopardize the final deadline.
- **Stakeholder Management Risks**

  Industrial clients' expectations of AI technology may be unrealistic (e.g., demanding "zero false alarms"), requiring a balance between functional promises and feasibility. If data quality assumptions prove invalid (e.g., historical maintenance records missing critical fields), project rework may be necessary.

3.2.2 Potential Opportunities

a. **Market Expansion & Commercialization Potential**
  - **Industry Benchmarking Effect:** Achieving over 95% anomaly detection accuracy in high-value equipment (e.g., aircraft engines) could establish this as a benchmark case for industrial AI predictive maintenance, attracting more manufacturing clients. Successful pilot validations (in 2 scenarios) can serve as marketing material, accelerating replication in industries like aviation, energy, and rail transportation.
  - **Subscription-Based Business Model:** The platform could evolve into a SaaS service, generating recurring revenue through real-time monitoring and AI diagnostics. Expansion into other high-value equipment (e.g., gas turbines, marine engines) could unlock broader market opportunities.

b. **Data Assets & AI Technology Barriers**
  - **Industrial Data Accumulation:** Equipment operation data collected via the platform can continuously refine models, creating exclusive industry datasets

that enhance competitive advantage. Accumulated data could enable new features (e.g., lifespan prediction, energy efficiency optimization), improving customer retention.

- **Technology Reusability:** The developed deep learning frameworks (e.g., LSTM, Transformer) can be adapted to other industrial scenarios, reducing R&D costs for future projects. Integration experience with legacy systems (e.g., SCADA, work order systems) can be reused in other digital transformation initiatives.

c. **Cost Savings & Operational Efficiency Gains**

- **Direct Economic Benefits:** Upon success, clients could see a 25% reduction in unplanned downtime and fault resolution time cut from 72 to 30 hours, minimizing production losses. A 25% drop in maintenance costs (target) would significantly improve customer ROI, boosting renewal rates.

- **Automation Replacing Manual Labor:** The platform’s automated diagnostics and alerts reduce the need for manual inspections, helping clients optimize maintenance team allocation.

d. **Policy & Industry Trend Advantages**

- **Industry 4.0 & Smart Manufacturing:** Global policies promoting manufacturing digitalization (e.g., China’s "14th Five-Year Plan" for smart manufacturing) align with this project, potentially qualifying for subsidies or partnerships.

- **ESG & Sustainability:** Reducing unplanned downtime lowers energy waste and equipment wear, supporting clients’ carbon neutrality goals and improving their ESG ratings.

3.3 Overview of the desired impact

- **Technical Aspect:** Realize the transformation of aircraft engine maintenance from "post-failure repair" to "predictive maintenance", using AI algorithms to increase fault warning accuracy to industry-leading levels (>95%), reduce real-time monitoring response time to within 500ms, and establish a full lifecycle equipment health record database.

- **Business Aspect:** In terms of operational cost optimization, the system is expected to reduce unplanned downtime by over 40%, decrease annual maintenance costs by 25-30%, and improve Overall Equipment Effectiveness (OEE) by 15 percentage points. For decision support enhancement, the system will provide data-driven maintenance dashboards and generate quantifiable maintenance KPI reporting frameworks.

- **Organizational Aspect:** The solution enables restructuring of traditional maintenance department workflows and fosters cross-functional maintenance teams with data analytics capabilities. Additionally, it establishes an aircraft engine fault characteristic database and develops reusable intelligent maintenance methodologies.

- **Industry Aspect:** The initiative aims to create a benchmark smart maintenance case in aviation manufacturing, developing replicable solutions applicable to other high-value industrial equipment. This promotes deeper adoption of industrial internet technologies in high-end equipment sectors and accelerates digital transformation across the aviation industry chain.

4. Measurable Organizational Value (MOV)

Objective	Target Value
Improve anomaly detection accuracy	Achieve >95% accuracy in engine part anomaly detection within 6 months of deployment
Reduce unplanned maintenance events	Decrease unplanned maintenance incidents by ≥25% within the first year
Shorten average downtime per incident	Reduce average downtime from 72 hours to ≤30 hours within 12 months
Enhance operational efficiency	Increase equipment uptime by ≥15% within the first year
Optimize maintenance costs	Reduce annual maintenance costs by ≥20% within 18 months

5. Project Scope

5.1 In-Scope

a. **Cloud-based Predictive Maintenance Platform Development**

- **Develop centralized web monitoring platform:** Build a browser-based unified monitoring portal supporting centralized management of multi-site equipment, featuring responsive design for cross-device compatibility with integrated access control and single sign-on functionality.
- **Implement real-time data visualization functionality:** Develop dynamic dashboards with millisecond-level refresh for equipment parameters (vibration/temperature/pressure etc.), offering multiple visualization modes (trend charts etc.) with customizable monitoring thresholds.
- **Build AI analytics engine:** Deploy LSTM neural networks for failure pattern recognition and remaining useful life prediction, supporting online model updates with auto-evaluation (target accuracy >95%)

b. **System Integration**

- **Interface with existing IoT devices:** Develop a standardized device communication protocol adaptation layer supporting industrial protocols, compatible with over 90% of existing sensor equipment in plant areas.
- **Enable data collection and transmission:** Establishes reliable data pipelines from industrial equipment to the analytics platform. It handles multi-protocol connectivity, performs edge preprocessing (filtering, compression), and ensures secure, fault-tolerant transmission with buffering and auto-recovery capabilities. The system validates data quality while supporting both real-time streaming and batch transfers to backend systems.

c. **Functional Modules**

- **Real-time monitoring dashboard:** Includes equipment operation status, device detail cards, overall health trend charts, real-time energy consumption curves, and real-time parameter cards.
- **Flexible alert notification system:** Includes alarm device information, alert timestamp, severity level, and detailed alert descriptions, while also providing statistical monitoring data for the current week.
- **Predictive maintenance analysis tools:** Select models and equipment for simulation to obtain the equipment's health index and remaining lifespan prediction.

5.2 Out-of-Scope

a. **Hardware Related**

- Excludes new hardware procurement
- Does not involve equipment retrofitting

b. **System Integration**

- Excludes deep ERP system integration

c. **Mobile Development**

- No standalone mobile application development
- Web browser access only supported

6. Project Schedule Summary

6.1 Project start date

March 24, 2024

6.2 Project end date

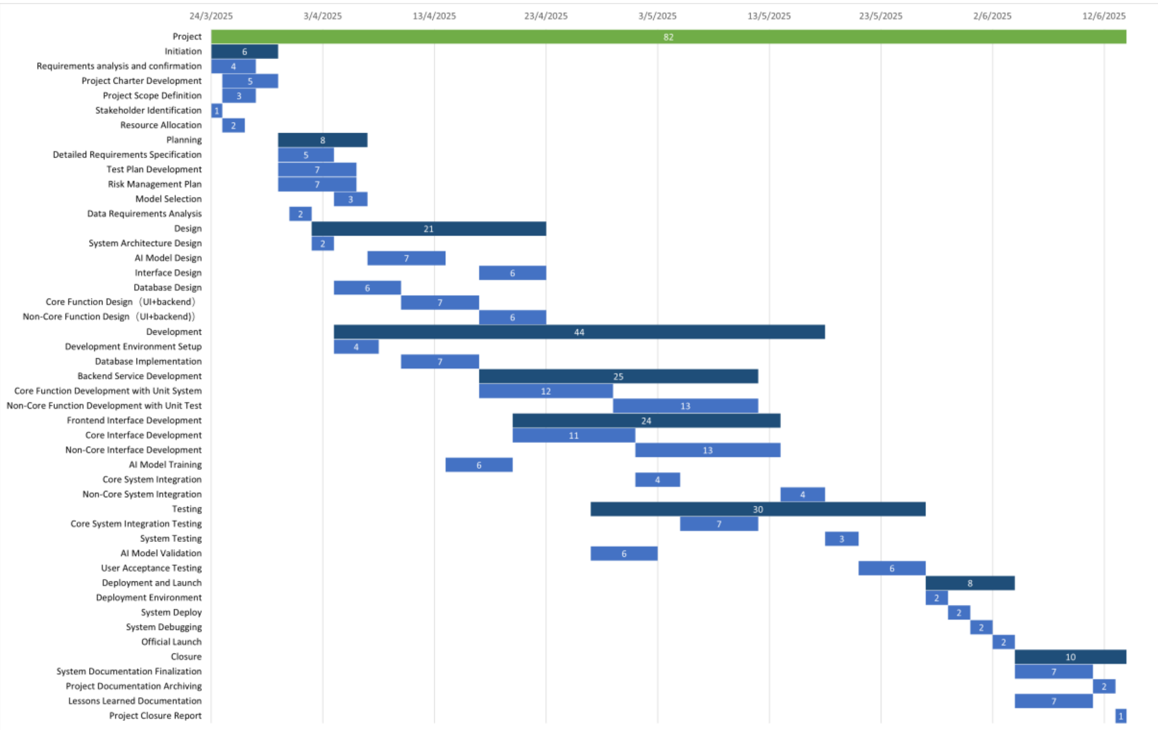
June 15, 2024

6.3 Gantt Chart

任务编码	任务名称	开始时间	结束时间	工期(天)	前置任务	负责人	预计工作时长(小时)
/	Project	24/3/2025	13/6/2025	82			549
1	Initiation	24/3/2025	29/3/2025	6		Xuanhe Yang	40
1.1	Requirements analysis and confirmation	24/3/2025	27/3/2025	4		Xuanhe Yang	16
1.2	Project Charter Development	25/3/2025	29/3/2025	5		Fubin Chen	15
1.3	Project Scope Definition	25/3/2025	27/3/2025	3		Xuanhe Yang	4
1.4	Stakeholder Identification	24/3/2025	24/3/2025	1		Xuanhe Yang	2
1.5	Resource Allocation	25/3/2025	26/3/2025	2		Tong Li	3
2	Planning	30/3/2025	6/4/2025	8	2		55
2.1	Detailed Requirements Specification	30/3/2025	3/4/2025	5	4	Tong Li	18
2.2	Test Plan Development	30/3/2025	5/4/2025	7		Jingxiao Han	12
2.3	Risk Management Plan	30/3/2025	5/4/2025	7		Fubin Chen	12
2.4	Model Selection	4/4/2025	6/4/2025	3	9,13	Xuanhe Yang	8
2.5	Data Requirements Analysis	31/3/2025	1/4/2025	2		Xuanhe Yang	5
3	Design	2/4/2025	22/4/2025	21			140
3.1	System Architecture Design	2/4/2025	3/4/2025	2		Fubin Chen	8
3.2	AI Model Design	7/4/2025	13/4/2025	7	12	Xuanhe Yang	30
3.3	Interface Design	17/4/2025	22/4/2025	6		Jingxiao Han	24
3.4	Database Design	4/4/2025	9/4/2025	6		Tong Li	24
3.5	Core Function Design （UI+backend）	10/4/2025	16/4/2025	7		Xuanhe Yang	30
3.6	Non-Core Function Design （UI+backend))	17/4/2025	22/4/2025	6		Xuanhe Yang	24
4	Development	4/4/2025	17/5/2025	44			170
4.1	Development Environment Setup	4/4/2025	7/4/2025	4	15	Jingxiao Han	4
4.2	Database Implementation	10/4/2025	16/4/2025	7	18	Tong Li	8

4.3	Backend Service Development	17/4/2025	11/5/2025	25			68
4.3.1	Core Function Development with Unit System	17/4/2025	28/4/2025	12		Tong Li	40
4.3.2	Non-Core Function Development with Unit Test	29/4/2025	11/5/2025	13		Jingxiao Han	28
4.4	Frontend Interface Development	20/4/2025	13/5/2025	24			54
4.4.1	Core Interface Development	20/4/2025	30/4/2025	11		Fubin Chen	28
4.4.2	Non-Core Interface Development	1/5/2025	13/5/2025	13		Fubin Chen	26
4.5	AI Model Training	14/4/2025	20/4/2025	6	16	Xuanhe Yang	12
4.6	Core System Integration	1/5/2025	4/5/2025	4	28	Fubin Chen	12
4.7	Non-Core System Integration	14/5/2025	17/5/2025	4	29	Jingxiao Han	12
5	Testing	27/4/2025	26/5/2025	30			62
5.1	Core System Integration Testing	5/5/2025	11/5/2025	7	31	Jingxiao Han	28
5.2	System Testing	18/5/2025	20/5/2025	3	32	Tong Li	12
5.3	AI Model Validation	27/4/2025	2/5/2025	6	30	Xuanhe Yang	10
5.4	User Acceptance Testing	21/5/2025	26/5/2025	6	35	Fubin Chen	12
6	Deployment and Launch	27/5/2025	3/6/2025	8			26
6.1	Deployment Environment	27/5/2025	28/5/2025	2		Jingxiao Han	2
6.2	System Deploy	29/5/2025	30/5/2025	2	39	Jingxiao Han	4
6.3	System Debugging	31/5/2025	1/6/2025	2	40	Xuanhe Yang	16
6.4	Official Launch	2/6/2025	3/6/2025	2	41	Xuanhe Yang	4
7	Closure	4/6/2025	13/6/2025	10	38		56
7.1	System Documentation Finalization	4/6/2025	10/6/2025	7		Xuanhe Yang	30
7.2	Project Documentation Archiving	11/6/2025	12/6/2025	2	44	Fubin Chen	4
7.3	Lessons Learned Documentation	4/6/2025	10/6/2025	7		Tong Li	20
7.4	Project Closure Report	13/6/2025	14/6/2025	1	45	Jingxiao Han	2

任务名称	预计工作时长(小时)
Core Function Development with Unit System	40
Model Management System Development	12
User Management Module	12
Forecast Analysis System Development	8
Device Management Module	8
Non-Core Function Development with Unit Test	28
Monitoring Center	12
Alert System	8
Report Generation Module	8
Core Interface Development	28
Model Management System Development	6
User Management Module	8
Forecast Analysis System Development	8
Device Management Module	6
Non-Core Interface Development	26
Monitoring Center	10
Alert System	8
Report Generation Module	8



- AI model training and validation is a critical component of the project, lasting from April 14 to May 2

- Core and non-core functions are developed separately, with core functions completed first
- System integration is conducted in two phases, first integrating the core system, then the complete system

6.4 Project reviews and dates

Review Type	Date	Key Objectives
Initiation Review	2025/3/29	Confirm project scope, stakeholders, and resource allocation
Planning Review	2025/4/6	Approve detailed requirements, test plan, risk management plan, and model selection
Design Review	2025/4/22	Finalize system architecture, AI model design, and interface designs
Development Phase Review	2025/5/17	Assess progress of all development components and integration status
Testing Review	2025/5/26	Evaluate test results and obtain approval for deployment
Pre-Deployment Review	2025/5/28	Verify deployment readiness and final system checks
Post-Launch Review	2025/6/3	Assess system performance after launch and address immediate issues
Project Closure Review	2025/6/13	Finalize all documentation, archive project materials, and document lessons learned

6.5 Milestones

Design Phase Completed: April 22, 2025 (All system and functional design documents finalized)

Core Development and Integration Completed: May 4, 2025 (Core system operational)

Testing and Acceptance Completed: May 26, 2025 (System passes comprehensive testing and user acceptance)

Official System Launch: June 3, 2025 (Deployment and debugging completed with official release)

7. Project Budget Summary

7.1 Total budget

The project budget primarily consists of internal labor costs, mainly allocated for a 5-member team's 3-month development period. Hardware and software costs are strictly controlled within RMB 500, utilizing minimum viable solutions (such as student edition development tools and basic cloud service packages). Additional budgets may be requested as needed for testing/validation and unforeseen expenses during project execution, with the total supplementary amount not exceeding 100% of the initial budget.

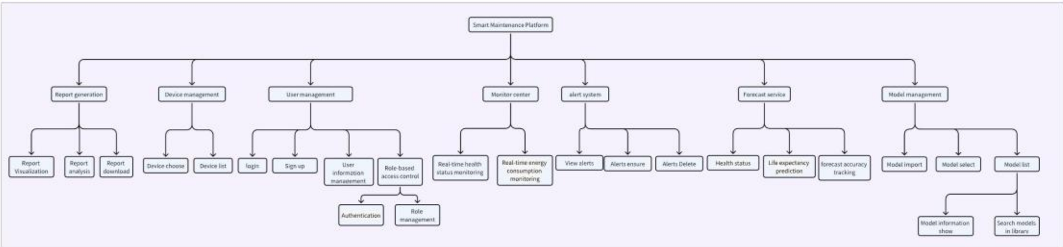
7.2 Breakdown by phase

Phase	Budget Allocation	Key Cost Components
Initiation	5% of total budget	• Stakeholder workshops • Project charter development • Initial risk assessment
Planning	10% of total budget	• Requirements analysis • Technical specifications • Resource planning
Design	15% of total budget	• System architecture design • Data modeling • UI/UX prototyping
Development	40% of total budget	• Backend/frontend coding • AI model training • Unit testing
Testing	20% of total budget	• Integration/system testing • Model validation • UAT support

Deployment & Launch	5% of total budget	• Production rollout • User training
Closing	5% of total budget	• Documentation • Lessons learned • Project archiving

8. Requirements Breakdown Structure(RBS)

RBS



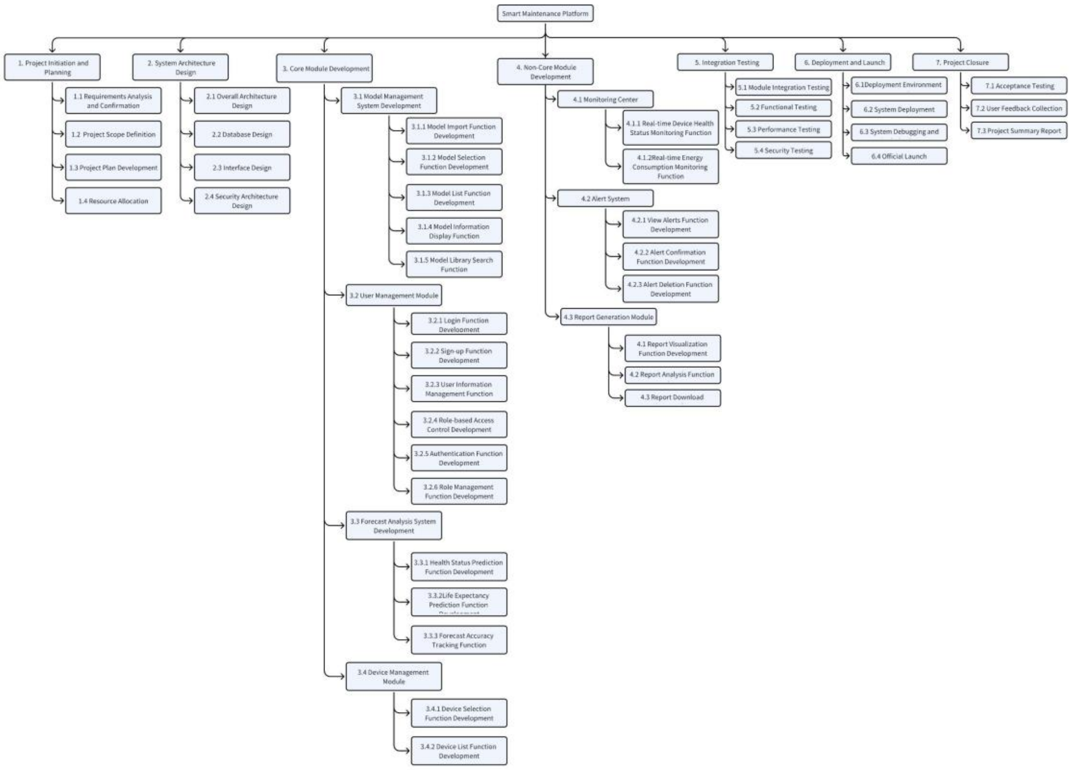
This Requirements Breakdown Structure(RBS) shows the requirements in tree structrue. At the top level is the "Smart Maintenance Platform," which branches into seven main modules:

1. **Report generation**
   - Report Visualization
   - Report analysis
   - Report download
2. **Device management**
   - Device choose
   - Device list
3. **User management**
   - login
   - Sign up
   - User information management
     - Authentication
     - Role management
   - Role-based access control
4. **Monitor center**

- Real-time health status monitoring
- Real-time energy consumption monitoring

5. **alert system**
   - View alerts
   - Alerts ensure
   - Alerts delete
6. **Forecast service**
   - Health status prediction
   - Life expectancy prediction
   - forecast accuracy tracking
7. **Model management**
   - Model import
   - Model select
   - Model list
   - Model information show
   - Search models in library

9. Work Breakdown Structure(WBS)



Here is the Work Breakdown Structure (WBS) for the Smart Maintenance Platform:

1. Project Initiation and Planning

- **1.1 Requirements Analysis and Confirmation:** Identify user and system requirements, establishing the foundation for the entire project
- **1.2 Project Scope Definition:** Define project boundaries, determining what is included and excluded
- **1.3 Project Plan Development:** Develop detailed project schedule and milestones
- **1.4 Resource Allocation:** Allocate human, material, and financial resources

2. System Architecture Design

- **2.1 Overall Architecture Design:** Design the overall system structure, including module division and interactions
- **2.2 Database Design:** Design data storage structures and relationships
- **2.3 Interface Design:** Design user interfaces and system interfaces
- **2.4 Security Architecture Design:** Design system security protection measures

3. Core Module Development

- **3.1 Model Management System Development**
    - *3.1.1 Model Import Function Development*
    - *3.1.2 Model Selection Function Development*
    - *3.1.3 Model List Function Development*

    - *3.1.4 Model Information Display Function*
    - *3.1.5 Model Library Search Function*
- **3.2 User Management Module**
    - *3.2.1 Login Function Development*
    - *3.2.2 Sign-up Function Development*
    - *3.2.3 User Information Management Function*
    - *3.2.4 Role-based Access Control Development*
    - *3.2.5 Authentication Function Development*
    - *3.2.6 Role Management Function Development*
- **3.3 Forecast Analysis System Development**
    - *3.3.1 Health Status Prediction Function Development*
    - *3.3.2 Life Expectancy Prediction Function Development*
    - *3.3.3 Forecast Accuracy Tracking Function*
- **3.4 Device Management Module**
    - *3.4.1 Device Selection Function Development*
    - *3.4.2 Device List Function Development*

4. Non-Core Module Development

- **4.1 Monitoring Center**
    - *4.1.1 Real-time Device Health Status Monitoring Function*
    - *4.1.2 Real-time Energy Consumption Monitoring Function*
- **4.2 Alert System**
    - *4.2.1 View Alerts Function Development*
    - *4.2.2 Alert Confirmation Function Development*
    - *4.2.3 Alert Deletion Function Development*
- **4.3 Report Generation Module**
    - *4.3.1 Report Visualization Function Development*
    - *4.3.2 Report Analysis Function Development*
    - *4.3.3 Report Download Function Development*

5. Integration Testing

- **5.1 Module Integration Testing:** Test interfaces and interactions between modules
- **5.2 Functional Testing:** Verify system functions meet requirements
- **5.3 Performance Testing:** Test system performance under different loads
- **5.4 Security Testing:** Detect system security vulnerabilities and risks

6. Deployment and Launch

- **6.1 Deployment Environment Preparation:** Prepare servers, networks, and other necessary infrastructure
- **6.2 System Deployment:** Install the system in the production environment
- **6.3 System Debugging and Optimization:** Resolve issues discovered after deployment and optimize performance
- **6.4 Official Launch:** Formally put the system into use

7. Project Closure

- **7.1 Acceptance Testing:** User confirmation that the system meets requirements
- **7.2 User Feedback Collection:** Collect user experience and suggestions
- **7.3 Project Summary Report:** Summarize project experiences and lessons learned

This WBS demonstrates a complete development process for a Smart Maintenance Platform, covering all phases from project initiation to closure. The platform primarily focuses on equipment health monitoring, predictive maintenance, energy consumption monitoring, and report analysis functionalities, suitable for intelligent management and maintenance of industrial equipment. Core functionalities include model management, user management, predictive analysis, and device management, while non-core functionalities include monitoring center, alert system, and report generation. The entire project follows a standard software development lifecycle, including requirement analysis, design, development, testing, deployment, and maintenance phases.

10. Quality Issues

10.1 Specific quality requirements

Feature	**Metric**	**Expectation**
Device Lifespan Prediction	Prediction Accuracy	• ≥ 90%
	Response Time	• ≤ 3 seconds
	Model Update	• At least once per week

	Frequency	
Real-time Device Health Monitoring	Data Refresh Rate	• ≤ 5 minutes/refresh
	Monitoring Accuracy	• ≥ 95%
	System Availability	• ≥ 99.5%
Device Anomaly Alerts	Alert Timeliness	• ≤ 10 minutes
	False Positive Rate	• ≤ 5%
	False Negative Rate	• ≤ 2%
Alert Notification List	Notification Delivery Rate	• ≥ 99%
	Notification Delay	• ≤ 1 minute
	List Refresh Speed	• ≤ 2 seconds
Device Details View	Page Loading Time	• ≤ 2 seconds
	Data Completeness	• 100%
	Historical Data Query Speed	• 100%
Overall System	Concurrent User Support	• ≥ 1000 users
	System Reliability	• ≥ 720 hours
	System Security	• Compliance with industry standard

		security protocols

11. Resources Required

11.1 People

11.1.1 Artificial Intelligence Engineer

- **Responsibilities**

1. Design, develop, and implement AI algorithms and machine learning models
2. Analyze and preprocess data for model training and testing
3. Optimize models for performance, accuracy, and scalability
4. Develop predictive maintenance solutions using anomaly detection
5. Integrate AI capabilities with existing platform infrastructure
6. Stay updated with the latest AI research and technologies
7. Collaborate with data engineers to ensure proper data pipelines
8. Document AI systems and models for future maintenance

- **Importance**

The AI Engineer is critical to the platform's competitive advantage. They enable predictive maintenance capabilities that identify potential system failures before they occur, reducing downtime and maintenance costs. Their work directly impacts the platform's ability to automatically detect patterns, optimize resource allocation, and provide intelligent insights to clients. As the core technical specialist for our predictive maintenance features, this role ensures we can deliver on our promise of enhanced reliability and operational efficiency.

11.1.2 Front-end Development Engineer

- **Responsibilities**

1. Design and implement user interfaces for web applications
2. Develop responsive and accessible web components
3. Optimize application performance and load times
4. Ensure cross-browser compatibility and responsive design
5. Implement user interaction features and animations
6. Collaborate with UX designers to implement designs accurately
7. Integrate front-end with back-end services via APIs
8. Conduct unit testing for UI components

- **Importance**

The Front-end Engineer creates the direct touchpoint between users and our platform. Their work determines how users perceive, interact with, and ultimately derive value from our services. A well-designed, intuitive interface significantly reduces training time and support costs while increasing user satisfaction and adoption rates. This role ensures our complex system capabilities are presented in an accessible, efficient manner that enhances rather than hinders productivity, directly impacting customer retention and satisfaction.

11.1.3 Back-end Development Engineer

- **Responsibilities**

1. Design and develop server-side logic and APIs
2. Implement database schemas and data storage solutions
3. Develop microservices architecture for scalability
4. Create and maintain RESTful services
5. Implement authentication and authorization mechanisms
6. Optimize application for performance and reliability
7. Handle server deployment and infrastructure integration
8. Monitor and troubleshoot system performance issues

- **Importance**

The Back-end Engineer builds the foundation and infrastructure that powers our entire platform. Their work ensures the system can handle increasing loads, process data efficiently, and maintain high availability. This role is fundamental to delivering reliable services that perform consistently under varying conditions. The back-end architecture directly impacts scalability, maintenance costs, and system resilience. As the architect of our core services, this engineer enables all other platform capabilities and ensures we can deliver on service level agreements.

11.1.4 Data Engineer

- **Responsibilities**

1. Design and implement data pipelines for collection and processing
2. Develop and maintain database systems and data warehouses
3. Create ETL (Extract, Transform, Load) processes
4. Ensure data quality, integrity, and security
5. Optimize database performance and scalability
6. Implement data governance policies
7. Collaborate with AI engineers on data requirements

8. Build data monitoring and logging systems

- **Importance**

The Data Engineer establishes the framework for all data-driven functionalities in our platform. They ensure data flows efficiently throughout the system, is properly structured, and remains accessible when needed. This role is crucial for enabling real-time analytics and AI-based predictive maintenance. Without robust data engineering, our platform would lack the insights that provide value to clients. As organizations increasingly rely on data-driven decision making, the Data Engineer becomes essential to delivering actionable intelligence and maintaining competitive advantage through proper data utilization.

11.1.5 Test Engineer

- **Responsibilities**

1. Design and implement comprehensive test strategies
2. Develop automated testing frameworks and scripts
3. Perform functional, integration, and regression testing
4. Identify and document bugs and issues
5. Validate fixes and improvements
6. Conduct performance and load testing
7. Test for security vulnerabilities
8. Ensure quality standards are met before deployment

- **Importance**

The Test Engineer is the guardian of system quality and reliability. They systematically identify issues before they impact users, ensuring a stable and trustworthy platform. This role directly impacts customer satisfaction by preventing service disruptions and functionality problems. In the context of critical infrastructure monitoring and maintenance, reliability is paramount—a single undetected bug could lead to significant downtime or data loss. The Test Engineer's work reduces maintenance costs, improves system reputation, and builds customer trust through consistent performance and reliability.

11.1.6 Project Manager

- **Responsibilities**

1. Develop and maintain project plans and schedules
2. Allocate resources and manage team workloads
3. Track progress against milestones and deliverables
4. Identify and mitigate project risks

5. Facilitate communication between team members and stakeholders
6. Manage scope changes and requirement adjustments
7. Coordinate with clients and external partners
8. Ensure project deliverables meet quality standards
9. Report on project status and metrics to leadership

- **Importance**

The Project Manager orchestrates all aspects of development and implementation, ensuring on-time, on-budget delivery of platform capabilities. They balance technical considerations with business objectives, resource constraints, and client needs. This role is essential for transforming plans into executable actions and coordinating interdependent work streams. The Project Manager's effectiveness directly impacts development efficiency, resource utilization, and stakeholder satisfaction. By maintaining clear focus on priorities and facilitating collaboration, they enable the entire team to work cohesively toward shared goals, reducing waste and maximizing productivity.

11.2 Technology

11.2.1 Front-end Technologies

- **Vue.js Framework**: Core framework for building the reactive user interface
- **Element Plus**: UI component library for consistent design and rapid development
- **Vue Router**: Client-side routing for single-page application navigation
- **Vuex**: State management for complex application data flow
- **Node.js**: JavaScript runtime environment for development and build processes
- **npm/Yarn**: Package management for front-end dependencies
- **ECharts**: Data visualization library for interactive charts and graphs
- **Axios**: HTTP client for API requests to the backend services
- **SCSS/SASS**: CSS preprocessor for advanced styling capabilities

11.2.2 Back-end Technologies

- **Spring Boot**: Java-based framework for building microservices and RESTful APIs
- **Spring Security**: Authentication and authorization framework
- **Spring Data JPA**: Data persistence layer for database operations
- **Maven**: Dependency management and build automation tool
- **JWT (JSON Web Tokens)**: Authentication mechanism for securing API endpoints

11.2.3 Database Technologies

- **MySQL**: Primary relational database for persistent data storage
- **Redis**: High-performance caching and temporary data storage
- **Flyway**: Database migration tool for version control of database schemas
- **MyBatis**: SQL mapping framework for database operations
- **MySQL Workbench**: Database design and administration tool

11.2.4 Data Processing

- **Apache Spark**: Big data processing framework
- **Python**: Programming language for data processing and ML model implementation
- **Pandas**: Data manipulation and analysis library
- **NumPy**: Numerical computing library
- **Scikit-learn**: Machine learning library for preprocessing and basic models

11.2.5 Machine Learning Frameworks

- **TensorFlow/Keras**: Deep learning frameworks for complex model development
- **PyTorch**: Alternative deep learning framework
- **XGBoost**: Gradient boosting framework for predictive models
- **LSTM Networks**: Specialized neural networks for time series analysis
- **Prophet**: Time series forecasting library

11.3 Facilities

- **Development environment**

  Windows PC
- **Conference**

  Tencent Meeting
- **Development Space**

  Collaborative space

11.4 Other

- **Data Collection Resources**

Historical equipment performance data、Sensor data collection systems、Time series data repositories......

- **Third-Party API Support**

Maintenance management system connectors、Notification services......

- **External Consultation**

Industrial equipment specialists、Data privacy consultants、Predictive maintenance experts、Security and compliance experts......

11.5 Resources to be Provided

11.5.1 Resources

Resource	Description
NASA N-CMAPSS Dataset	Commercial Modular Aero-Propulsion System Simulation dataset for training RUL prediction models
Skip-GANomaly Model	Pre-trained model for anomaly detection in turbine blade surfaces
CNN-LSTM Model	Pre-trained model for remaining useful life prediction
GPU Computing Resources	High-performance computing infrastructure for model training and inference
Development Environment	Configured environment with required frameworks and libraries
API Documentation	Complete documentation of system APIs and integration points
User Interface Prototypes	Figma Design

11.5.2 Name of Resource Provider

Resource	Provider
NASA N-CMAPSS Dataset	NASA Prognostics Center of Excellence
Skip-GANomaly Model	Internal AI Research Team
CNN-LSTM Model	Internal AI Research Team
GPU Computing	Leader's server

Resources	
Development Environment	DevOps Team
API Documentation	Back-end Team
User Interface Prototypes	Front-end Team

11.5.3 Date to be Provided

Resource	**Availability Date**
NASA N-CMAPSS Dataset	Immediately Available
Skip-GANomaly Model	Immediately Available
CNN-LSTM Model	Immediately Available
GPU Computing Resources	Immediately Available
Development Environment	Already Available in the second week
API Documentation	Expected in April
User Interface Prototypes	Already Available in the second week

12. Assumptions and Risks

12.1 Assumptions used to develop estimates

This project planning and estimation is based on the following key assumptions: we assume high-quality historical device failure data will be available for training the neural network model; team members possess adequate expertise in neural networks and machine learning; sufficient computing resources will be available for model training and deployment; third-party systems will provide reliable APIs for data collection; end users will actively engage with the system and follow recommended actions; no significant regulatory changes will occur during development; key personnel will be available throughout the project lifecycle; and core technologies will remain stable during the development period.

12.2 Key risks, probability of occurrence, and impact

Risk	Probability	Impact	Mitigation Strategy
Insufficient training data	Medium	High	Implement data augmentation techniques; use synthetic data generation
Model accuracy below target	Medium	High	Develop multiple model architectures; implement ensemble methods
System performance issues	Medium	Medium	Conduct regular performance testing; implement scalable infrastructure
Data security breaches	Low	High	Implement robust security measures; conduct regular security audits
Integration failures with existing systems	Medium	Medium	Develop comprehensive API specifications; conduct thorough integration testing
User resistance to adoption	Medium	High	Involve end users in design process; provide comprehensive training
Budget overruns	Medium	Medium	Implement strict budget monitoring; establish contingency reserves
Schedule delays	Medium	Medium	Use agile methodology; build buffer time into project schedule
Scalability issues	Low	Medium	Design for scalability from the start; implement load testing
Regulatory compliance issues	Low	High	Regular compliance reviews; consultation with legal experts

12.3 Constraints

Constraint	Description
Budget	Project must be completed within the allocated budget of budget amount
Schedule	The system must be deployed by [target date] to meet business requirements
Resources	The development team is limited to a number of full-time engineers
Technology	Solution must utilize existing technology infrastructure where possible
Data Privacy	System must comply with all relevant data protection regulations
Backward Compatibility	New system must maintain compatibility with existing device monitoring systems
Backend Developers	API development, data processing and cloud service deployment
User Interface	UI must follow organizational design standards
Performance	The system must meet all performance metrics outlined in quality requirements
Security	Solution must adhere to organizational security protocols and industry standards

12.4 Dependencies on other projects or areas within or outside the organization

Dependency	Description	Impact
Data Engineering Team	Provision of clean, structured historical device data	High - Critical for model training

IT Infrastructure	Server allocation and network configuration	Medium - Required for deployment
Device Manufacturers	API access to device telemetry data	High - Essential for real-time monitoring
Security Team	Security review and approval	Medium - Required for deployment
UI Team	Monitoring Accuracy	Medium - Important for user adoption
Maintenance Teams	Feedback on alert mechanisms	Low - Valuable for usability
Existing Monitoring Systems	Integration points for data exchange	High - Required for comprehensive monitoring

12.5 Assessment project's impact on organization

Area	Impact	Description
Operational Efficiency	High Positive	Reduction in unexpected device failures by 30-40%
Maintenance Costs	High Positive	Estimated 20-25% reduction in maintenance costs
Resource Allocation	Medium Positive	More efficient scheduling of maintenance personnel
Decision Making	Medium Positive	Data-driven decisions on equipment replacement
IT Infrastructure	Low Negative	Additional load on existing IT systems
Work Processes	Medium Transitional	Temporary disruption during adoption phase

Customer Satisfaction	High Positive	Fewer service interruptions due to equipment failure
Environmental Impact	Medium Positive	Extended equipment lifecycle and reduced waste
Competitive Position	Medium Positive	Enhanced service reliability compared to competitors

12.6 Outstanding issues

Issue	**Description**	**Resolution Plan**	**Priority**
Data Quality	Historical data contains gaps and inconsistencies	Work with data engineering to clean and normalize data	High
Model Selection	Optimal neural network architecture not yet determined	Conduct comparative testing of multiple architectures	High
Performance Benchmarks	Specific thresholds for alert generation not established	Collaborate with domain experts to define thresholds	Medium
Integration Protocol	Communication protocols with existing systems undefined	Document API requirements and develop interfaces	High
User Training	Training plan for system users not developed	Create training materials and schedule sessions	Medium
Alerting Mechanisms	Alert prioritization logic not finalized	Define alert categories and escalation procedures	Medium
Recovery Procedures	System failure recovery processes not documented	Develop disaster recovery procedures	Medium
Feedback	Mechanism for user	Implement feedback	Low

Collection	feedback not established	collection within the application	
Maintenance Plan	Long-term system maintenance plan not defined	Develop maintenance schedule and procedures	Mediu m

13. Project Administration

13.1 Communications plan

- **Team Meeting : weekly offline**
- **Instant Communication : Wechat**
- **Code Management : GitHub**
- **Document Management : Feishu Docs**

13.2 Scope management plan

Current Core Feature:

- **Login/Register**
- **Device Center**
- **Device Moniter**
- **Data Simulation**
- **Alert System**
- **Alert Reporter**

If there are any changes, we will discuss together and update.

13.3 Quality management plan

- Establish quality metrics including 90%+ prediction accuracy, 99.5%+ system availability, and <5% false positive rate
- Implement quality assurance through code reviews, automated testing, and security validation
- Define quality control processes including CI/CD pipeline and pre-release quality gates
- Assign quality responsibilities across project roles
- Maintain quality documentation via test plans, metrics dashboards, and validation reports
- Conduct regular quality improvement through retrospectives and root cause analysis

13.4 Change management plan

- Form Change Control Board (CCB) with key stakeholders meeting weekly to evaluate changes
- Follow structured change request process: submission, analysis, review, implementation, verification
- Prioritize changes as Critical, High, Medium, or Low based on business impact
- Assess all changes for impacts on schedule, budget, scope, quality, and risk
- Document all changes in change register with complete traceability
- Establish expedited process for emergency changes requiring immediate action

13.5 Human resources plan

	YXH	HJX	LT	CFB
scope	Y	N	N	Y
Architecture	Y	Y	Y	Y
Front-end	N	Y	Y	Y
Back-end	Y	N	N	N
AI	Y	Y	Y	Y
Test	Y	N	N	N
Deliver	Y	N	N	N

14. Acceptance and Approval

Name	Signature	Approval Date
YXH	杨烜赫	2025.3.30
HJX	韩敬霄	2025.3.30
LT	李彤	2025.3.30
CFB	陈甫彬	2025.3.30

15. References

- Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.
- Project Management Institute. (2021). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)* (7th ed.).
- Sullivan, W. G., Wicks, E. M., & Koelling, C. P. (2023). *Engineering Economy* (18th ed.). Pearson.

16. Terminology or Glossary

Term	Definition
AI (Artificial Intelligence)	The simulation of human intelligence processes by computer systems, particularly those involving learning, reasoning, and self-correction.
API (Application Programming Interface)	A set of protocols, routines, and tools for building software applications that specifies how software components should interact.
Anomaly Detection	The identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data, used in predictive maintenance.
CDN (Content Delivery Network)	A geographically distributed network of proxy servers that helps minimize latency by delivering content to users from nearby servers.
DevOps	A set of practices that combines software development (Dev) and IT operations (Ops) to shorten the development lifecycle and provide continuous delivery.
Downtime	The period during which a system is unavailable or offline, often referring to periods when a system fails to provide its primary function.
GDPR (General Data Protection Regulation)	A regulation in EU law on data protection and privacy for all individuals within the European Union and the European Economic Area.

High Availability	System design approach and associated service implementation that ensures a prearranged level of operational performance will be met during a contractual measurement period.
IoT (Internet of Things)	The network of physical devices embedded with electronics, software, sensors, and connectivity which enables these objects to collect and exchange data.
KPI (Key Performance Indicator)	A measurable value that demonstrates how effectively a company is achieving key business objectives.
Machine Learning	A type of AI that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
Neural Network	A computing system inspired by biological neural networks that can learn tasks by considering examples.
Predictive Maintenance	Techniques that use data analysis tools and techniques to detect anomalies in operation and possible defects in equipment before they occur.
QA (Quality Assurance)	A way of preventing mistakes and defects in manufactured products and avoiding problems when delivering products or services to customers.
REST API (Representational State Transfer)	An architectural style for an application program interface that uses HTTP requests to access and manipulate data.
SaaS (Software as a Service)	A software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted.
SDLC (Software Development Life Cycle)	A process that produces software with the highest quality and lowest cost in the shortest time possible.
Version Control	The management of changes to documents, computer programs, large web sites, and other

	collections of information.
UI (User Interface)	The means by which the user and a computer system interact, in particular the use of input devices and software.
UX (User Experience)	A person's emotions and attitudes about using a particular product, system or service.