UNIVERSIDAD TECNOLOGICA DE TIJUANA

SUBJECT:
END-TO-END MOBILE DEVELOPMENT

TITLE:
"SPECIFICATION OF REQUIREMENTS ACCORDING TO THE STANDARD
IEEE 830 IEEE STD. 830-1998”

TEAM MEMBERS:
AMPARO BARBA JOSÉ CARLOS
ARANDA CASTILLO MIGUEL ANGEL
DE LA CRUZ POLANCO JAVIER
MACIEL LEYVA ARIANA LIZETH
LABRADA FARIAS IAN ARMANDO

GROUP:
10A IDGS

TEACHER:
RAY BRUNET PARRA GALAVIZ

DUE DATE:
TUESDAY FEBRUARY 04 2025

INDEX

1 INTRODUCTION

1. Introduction

This section will provide an introduction to the entire Software Requirements Specification (ERS) document. It consists of several subsections: purpose, system scope, definitions, references, and document overview. (THIS ONE IS DONE UNTIL THE END)

1.1. Purpose

The purpose of this document is to develop a detailed proposal for the implementation of a new development system in the industrial sector, the stage of gathering requirements until implementation will be explained, as well as each phase of the system's life cycle will be detailed.

The document is primarily aimed at information technology (IT) professionals, including developers, software engineers, system architects, and industry experts.

1.2. Scope of the System

System Name: MantéFix (Maintenance and Repair)

System Description

MantéFix will be a comprehensive digital solution accessible from both a mobile app and a website. Its main purpose will be to optimize the management of preventive, corrective and predictive maintenance in industries and companies, centralizing information and facilitating the planning of critical tasks.

What the system will do:

- Mobile and web platform: will provide access to the main functionalities from mobile devices and web browsers.
- It will allow industrial assets to be registered with key information such as name, type, location and current status.

- It will offer an interactive calendar to schedule preventive, corrective and predictive maintenance.
- It will integrate a real-time notification system to alert on pending tasks, urgent incidents and preventive recommendations.
- It will facilitate the assignment of technicians to specific tasks according to their availability and specialty.
- It will generate downloadable automated reports in PDF format, showing key metrics and associated costs.
- It will allow you to view the history of incidents, including reported failures, actions taken and resolution times.
- It will incorporate a module to record operating hours of the machines and avoid interruptions during maintenance.
- It will provide a screen for assigning specific machines to maintenance tasks.
- It will implement a system for requesting parts needed for specific maintenance tasks.
- It will provide secure, cross-platform access with authentication for different roles, such as administrators, technicians, and supervisors.

What the system won't:

- It shall not include automated technical diagnostic functions of the machines.
- It will not manage complete inventories of parts or tools beyond specific requests.
- You won't have remote or real-time support features.
- It will not include functionalities for administrative tasks outside the scope of maintenance.

Benefits, Objectives and Goals

Expected Benefits:

- Cross-platform access for flexibility and user convenience.
- Centralization and availability of key information from any device connected to the internet.
- Improved operational efficiency by planning maintenance more accurately and reducing downtime.
- Greater control and traceability in asset and technical management.
- Reduced costs associated with unexpected failures and maintenance delays.
- Increased safety and extended life of critical equipment.

Objectives:

- Design a system accessible from mobile devices and web browsers that centralizes maintenance management.
- Integrate innovative technologies, such as APIs for real-time notifications and automated reporting.
- Provide an optimized user experience for technicians, supervisors, and administrators.

Goals:

- Ensure real-time data synchronization between the mobile and web versions of the system.
- Reduce downtime due to unexpected failures
- Ensure that critical notifications reach the user in real time.
- Increase the use of the platform for preventive maintenance

1.3. Definitions, Acronyms and Abbreviations

API (Application Programming Interface): A set of protocols and tools that allow communication between different systems or applications.

Web platform: An online environment that allows the development and execution of applications accessible through the internet.

DB: Database.

UI: User Interface.

UX: User Experience.

PDF: Portable Document Format.

IoT: Internet of Things.

RAM: Random Access Memory.

CPU: Central Processing Unit.

HTTP: HyperText Transfer Protocol.

HTTPS: HyperText Transfer Protocol Secure.

SMTP: Simple Mail Transfer Protocol.

JSON: JavaScript Object Notation.

CRUD: Create, Read, Update, Delete.

1.4. References

Below is a list of the documents referenced in the System Requirements Specification (ERS) for the MantéFix (Maintenance and Repair) project. These documents provide additional and complementary information that supports the development of the requirements and design of the system:

1. ISO/IEC 9126:2001 - Software Engineering — Product Quality. This standard provides the general criteria for evaluating the quality of software and has been used to define the quality requirements of the MantéFix system.

2. Guide to good practices in software development (PMI, 2017) - Documents that establish best practices for the development of software projects in industrial environments, specifically for maintenance management systems.
3. Real-Time Notifications API (Firebase Cloud Messaging, 2023) - Official Firebase documentation, which will be used for the integration of the real-time notification system.
4. API Integration Manual for Industrial Equipment (Siemens, 2021) – This manual provides details on integrating industrial equipment with software systems for data collection and maintenance management.
5. Documentation on Preventive and Corrective Maintenance Management (MantePro, 2024) - This document provides the basic structure for managing the maintenance of industrial equipment, including best practices in the planning and execution of preventive and corrective maintenance.
6. User Guide for the ERP System (SAP, 2020) - Reference on the future integration of the MantéFix system with existing ERP systems within the company for the optimization of maintenance processes.
7. IEEE 830-1998 - Recommendations on the Specification of Software Requirements. This document has been consulted for the structure and drafting of the system requirements.
8. Inventory Management RESTful API Documentation (Zebra Technologies, 2022) - Provides details on how spare parts inventories will be managed within the system.
9. ISO 55000:2014 - Asset Management. This standard is relevant to the development of system functionality related to industrial asset management.
10. Augmented Reality (AR) System for Industrial Maintenance Documentation (Hololens, 2022) - Reference on how augmented reality could be implemented in the system to aid in industrial maintenance.

1.5. Document Overview

The purpose of this section is to describe in detail the subsections for the development of the system and implementation. Below is a brief description of each of the sections that make up this document:

Section 2: Overview

This section provides an overview of the preventive and corrective maintenance management system, including its purpose, main functions, and the overall outlook of the system. User characteristics, system constraints, assumptions and dependencies, and anticipated future requirements are also addressed.

Subsection 2.1: Product Outlook

The context of the corrective and preventive maintenance system, its relationship with other systems or projects, and the general vision of the functionalities are detailed.

Subsection 2.2: Product Functions

The main functions that the preventive and corrective maintenance management system must perform are listed and described, providing a clear view of its capabilities.

Subsection 2.3: User Characteristics

The different types of users of the corrective and preventive maintenance system are identified and their characteristics, needs and expectations are described.

Subsection 2.4: Restrictions

Limitations that may affect the design and implementation of the system are specified, including technical, business, and regulatory constraints.

Subsection 2.5: Assumptions and Dependencies

Assumptions made during system planning and dependencies on other systems or resources will be detailed.

Subsection 2.6: Future Requirements

Any additional requirements that may arise in the future as the system evolves are outlined.

Section 3: Specific Requirements

This section focuses on specific system requirements, detailing external interfaces, particular functions, performance requirements, design constraints, system attributes, and other relevant requirements.

Subsection 3.1: External Interfaces

The system interfaces are described, including technical and functional details.

Subsection 3.2: Functions

The specific functions of the system are detailed, including descriptions of each functionality and how they should be implemented.

Subsection 3.3: Performance Requirements

This subsection details the essential performance requirements for the system, ensuring efficiency and responsiveness.

Subsection 3.4: Design Constraints

Design constraints that need to be considered are identified, including technical and architectural aspects (CSS).

Subsection 3.5: System Attributes

System quality attributes such as security, usability, reliability, and maintainability are described.

Subsection 3.6: Other Requirements

Other additional requirements that have not been covered in the previous subsections but are relevant to the development of the system are listed.

2 OVERVIEW

This section details the general context of the ManténFix (Maintenance and Repair) system, describing the factors that affect its design, development and operation. The requirements are not specified in detail here, but a general framework is provided that makes them easier to understand in later sections.

2.1. Product Outlook

The MantéFix (Maintenance and Repair) **product perspective** focuses on offering a comprehensive solution for preventive, corrective and predictive maintenance management, with an emphasis on the optimization of industrial and business processes. This system is designed to operate independently and to be integrated into a broader ecosystem of business management, providing connectivity with other relevant systems and services.

The product is framed as a key tool for:

- Automate and organize maintenance tasks.
- Monitor the status of assets and equipment in real time.
- Analyze historical and predictive data to improve decision-making.

Integrated External Services (APIs)

Language API: Google Cloud Translation APIFuntion:
Facilitates the translation of website and mobile app content, allowing you to offer a multilingual experience tailored to users in different regions.
Characteristics:

- Real-time translation of text or content strings.
- Automatic input language detection.
- Supports over 100 languages.
- Easy integration via REST or SDK requests.

Benefit in MantéFix:
Thanks to this API, MantéFix will be able to automatically adapt to the user's linguistic preferences, improving the user experience and expanding the reach of the system globally.

Real-Time Notifications API: Firebase Cloud Messaging (FCM)Function:
Manages the sending of push notifications in real-time to inform technicians of pending maintenance tasks and alert managers to machines that require

intervention.

Characteristics:

- Support for mobile devices (iOS and Android) and web browsers.
- Ability to manage millions of messages daily in a scalable way.
- Sending personalized messages according to the preferences and roles of the users.
- Direct integration with backend systems via SDKs or REST.

Benefit in MantéFix:

This API allows MantéFix to keep technicians and managers informed in real-time, ensuring that critical tasks are addressed quickly and efficiently.

Power BI

Function:

Power BI is a data analysis tool that allows you to create interactive reports and dashboards. Through its platform, you can integrate data from various sources and visualize this data in interactive graphs within your applications or websites.

Characteristics:

1. **Interactive visualizations**: You can create interactive charts, tables, and maps to explore data in real time.
2. **Connecting to multiple data sources**: Power BI can be integrated with databases, Excel files, cloud services, and more.
3. **Power BI Embedded**: Allows you to embed dashboards and reports directly into web or mobile applications, without the need to redirect the user to another page.
4. **Real-time updating**: Dashboards automatically update when data changes, making it easy to view the most up-to-date information.
5. **Interactive filters**: Users can interact with the filters to customize the display according to their needs.
6. **Controlled access**: Allows you to set access permissions to dashboards and reports, ensuring that only authorized users can see certain data.

Benefit in MantéFix:

1. **Real-time monitoring**: Power BI can integrate with maintenance tracking systems and provide real-time visualizations of machine status and maintenance performed. This will allow technicians and managers to have more precise control over operations.
2. **Ease of integration**: Power BI can be easily integrated into MantéFix, using the Power BI Embedded service, without the need to redirect users outside the application. This improves the user experience by keeping navigation within the same system.

2.2. Product Functions

The MantéFix (Maintenance and Repair) system offers a number of fundamental functions to optimize maintenance management in industries and companies. These functions are organized to improve operational efficiency, task planning, resource monitoring, and decision-making. One of the main functions of MantéFix is the management of industrial assets. The system allows assets to be registered with key data, such as their name, type, location and current status. Additionally, users can update or delete assets as needed. The system also provides asset health monitoring, ensuring that up-to-date information is always available.

In terms of maintenance scheduling, the system allows the planning of preventive, corrective and predictive tasks. Users can organize these tasks into an interactive calendar, which makes it easy to view scheduled tasks and allows for the assignment of each task to specific technicians. This assignment is made according to the availability and specialization of the technicians, which optimizes efficiency in the execution of tasks.

Another key component of MantéFix is human resource management, specifically the registration and administration of technician profiles. The system allows technicians to be assigned to maintenance tasks and their performance to be tracked, making it easier to monitor and manage the activities carried out by technical staff.

The system also has real-time alerts and notifications functionality, ensuring that users are immediately informed about pending tasks, critical incidents, and predictive recommendations. These notifications are sent through a messaging API, compatible with mobile devices and emails, ensuring that information reaches users in a timely manner.

An important function of the system is the generation of automated reports. Users can create downloadable reports in formats such as PDF and Excel, which include key metrics such as the number of maintenance performed, equipment downtime, and costs associated with incidents. In addition, the system allows you to customize the report templates according to the needs of the company, which facilitates the generation of reports adapted to the specific requirements of each client.

In terms of inventory management, the system allows you to record the parts needed to perform maintenance tasks. Users can order the required parts for each task and track their availability and status in real-time. This ensures that the necessary resources are available when they are needed, avoiding delays in repairs. Finally, MantéFix includes a functionality for the history of incidents. The system records reported failures, actions taken, and resolution times for each asset. This history can be consulted both by asset and by technician, and allows an analysis of historical data to identify recurring patterns and propose predictive solutions to avoid future problems.

In addition, the system includes a machine operating control function, which allows you to configure the operating hours of each piece of equipment. This ensures that maintenance tasks do not interfere with the normal operation of machinery, and that tasks are assigned at the right times. Maintenance scheduling is integrated with this operating schedule, blocking tasks outside the established schedules for each machine.

2.3. User Characteristics

User	Educational Level	Experience	Technical Expertise
System Administrator	Bachelor's degree in areas such as engineering, computer science, or management	Experience in technology platform management, maintenance software administration, and system operations	Advanced knowledge in database management, user administration, ERP system configuration, and data analysis
Maintenance Technician	Technical or university training in mechanics, electronics, electricity or related areas	Experience in maintenance of industrial machinery, both preventive and corrective	Familiarity with diagnostic tools and maintenance software, intermediate knowledge in the use of technological platforms
Operations Manager	Bachelor's degree in business administration, industrial engineering, or related	Experience in supervising industrial operations, process optimization, and equipment management	Knowledge of maintenance management platforms, reporting and operational efficiency analysis
Machinery Operator	Technical training in industrial machinery operation	Experience operating industrial equipment in different shifts and tasks	Basic knowledge in maintenance scheduling, alert management and control of machinery operation
Maintenance Supervisor	Technical or university training in industrial engineering, mechanics or	Experience supervising maintenance teams, planning and execution of	Advanced knowledge in maintenance scheduling and the use of platforms

	related areas	tasks	for the management of work teams
Quality Manager	University degree in industrial engineering, quality or related	Experience in quality control in maintenance processes and industrial operations	Knowledge in the use of tools for quality assessment and continuous improvement in maintenance processes
Security Manager	University degree in industrial safety, risk prevention or related	Experience in occupational safety supervision, especially in industrial environments	Knowledge of industrial safety regulations and safety protocols in equipment maintenance

2.4. Restrictions

2.4.1 Company Policies

- **Data protection**: Comply with data protection and privacy regulations, such as GDPR (General Data Protection Regulation) and HIPAA (Health Insurance Portability and Accountability Act), to ensure the security and confidentiality of patient and clinic information.
- **Quality standards**: The software must comply with ISO 9001 quality standards to ensure the efficiency and reliability of the system.
- **Sustainability**: To promote sustainable development practices, minimizing the consumption of resources and optimizing the performance of the system.
- **Regulatory Compliance**: Adhere to local and international health and safety regulations, ensuring that the system manages and distributes medicines in accordance with applicable laws and regulations.
- **Regulations and Standards:** Strict compliance with government regulations and pharmaceutical industry standards, such as the FDA, EMA, and GMP.
- **Availability and Scalability:** The system must be highly available and able to scale to handle large volumes of data and users simultaneously.
- **Interoperability:** Need to integrate seamlessly with other inventory management systems and logistics platforms.

2.4.2 Hardware limitations

- **Servers**:
    - CPU: Intel Xeon processor or equivalent.
    - Memory: 16 GB of RAM.
    - Storage: 256 GB SSD.
    - Connectivity: 1 Gbps network interface.
    - Operating System: Compatible with Linux (Ubuntu Server, CentOS) and Windows Server.
- **Workstations**:
    - CPU: Intel Core i5 processor or equivalent.
    - Memory: 8 GB of RAM.
    - Storage: 500 GB HDD or 128 GB SSD.
    - Screen: Minimum resolution of 1280x1024.

2.4.3 Interfaces with other applications

- **Hospital Management Systems (HIS):** It must provide APIs for integration with hospital information systems.
- **Invoicing systems**: It should allow integration with invoicing and accounting applications such as SAP, QuickBooks, etc.
- **Pharmacy management systems**: It must be compatible with pharmacy management software to ensure smooth communication in the supply chain.
- **Medical devices**: It must be compatible with standard protocols for communication with medical devices, such as HL7.

2.4.4 Parallel operations

- **Log locking**: Implementing record locking mechanisms to prevent concurrent access conflicts.
- **Synchronization**: Use of data synchronization techniques to ensure consistency.

2.4.5 Audit functions

- **Activity log**: Detailed record of all actions performed by users, including creating, modifying, and deleting data.
- **Monitoring and alerts**: Real-time monitoring system and generation of alerts for suspicious or unauthorized activities.

2.4.6 Control Functions

- **Access Control**: Permission and role management system to ensure that only authorized personnel can access certain functions and data.
- **Data Validation**: Input validation mechanisms to ensure that the data entered meets the established criteria.

2.4.7 Programming Language(s)

- **Backend**: Django Framework, JavaScript
- **Frontend**: JavaScript, React
- **Databases**: MySQL, PostgreSQL.

2.4.8 Communication Protocols

- **HTTP/HTTPS:** For communication between client and server.
- **RESTful APIs**: For integration with external systems.
- **WebSocket**: For real-time updates.

2.4.9 Skill Requirements

- **Web Development Experience**: Knowledge in frontend and backend technologies.
- **Computer security**: Ability to implement security measures.
- **Database Management**: Experience in the design and management of relational databases and NoSQL.

- **Knowledge in the health sector**: Understanding of the regulations and specific needs of the health sector.

2.4.10 Application criticality

- **Availability**: Must guarantee 99.9% uptime.
- **Fault tolerance**: Implementation of failover and data redundancy mechanisms.

2.4.11 Security considerations

- **Authentication and authorization**: Use of robust authentication methods such as OAuth2 and JWT, and role-based authorization.
- **Data encryption**: Encryption of data in transit (TLS/SSL) and at rest.
- **Regular backups**: Performing regular backups to prevent data loss.
- **Security monitoring**: Implementation of security monitoring tools to detect and respond to threats in real-time.

2.5. Assumptions and Dependencies

Assumptions:

1. **Company Organization:** It is assumed that the organizational structure of the company will maintain its hierarchical form, where maintenance personnel, supervisors, and operational managers have well-defined roles. Any changes in the organization could affect the assignment of tasks and the visualization of data in the system.
2. **Availability of Technology Resources:** It is assumed that the company's devices and technological infrastructure (such as servers, computers, and networks) are available and compatible with the system implementation. Changes in technology resources, such as hardware obsolescence or lack of connectivity in some areas, could affect system performance.
3. **Accessibility to External APIs:** It is assumed that external APIs (for real-time notifications, inventory management, etc.) will be available and will not undergo significant changes that interfere with their integration. If APIs change or become unavailable, dependent modules may not work properly.

4. **Operating System Compatibility:** It is assumed that the system will be compatible with the most commonly used operating systems in the industry, such as Windows, Linux, and macOS. Any changes to operating system support may require additional adjustments.
5. **Legal Standards and Regulations:** It is assumed that the system will comply with the security, privacy and data protection regulations in force in the industry and in the region. A change in legal regulations could require modifications to modules related to information security.

Dependencies:

1. **Database Dependency:** A relational database is relied upon to store information about assets, maintenance, parts, and technicians. If there are changes to the database architecture or format, modifications to the system will need to be made.
2. **Internet dependency:** The system will rely on internet connectivity for some functions, such as real-time notification and integration with external APIs. An unstable connection could affect the user experience and overall functionality of the system.
3. **Reliance on External APIs:** The system relies on integration with external APIs for key functions such as sending real-time notifications, inventory management, and other third-party services. Any changes to the policies or operation of these APIs, or even service interruption, could affect the functionality of the system.
4. **Computers and Development Resources Dependency:** The development of the system will depend on the use of computers compatible with modern development tools (such as Visual Studio Code, database servers, among others). Changes in developers' hardware or software specifications could delay the development process.

2.6. Future Requirements

Below are some possible improvements and expansions that could be considered for the MantéFix system in the future. These functionalities are not necessary for the initial implementation, but could be analyzed and implemented in subsequent

versions of the system, depending on the needs of the company and the advancement of technology.

1. **Integration with IoT (Internet of Things) Devices:** In the future, the system could be integrated with IoT devices for real-time monitoring of equipment. This would allow sensor data (temperature, vibration, etc.) to be obtained to perform predictive maintenance, anticipating possible failures before they occur.
2. **Advanced Predictive Analytics:** Implement artificial intelligence and machine learning algorithms to improve predictive failure analysis. With these algorithms, the system could more accurately predict future equipment failures and schedule preventative maintenance tasks before problems occur.
3. **Integration with the Company's ERP:** In future versions, MantéFix could be integrated with the ERP (Enterprise Resource Planning) system used by the company, facilitating the synchronization of inventory, purchase order and financial data, which would further optimize maintenance management.

3. SPECIFIC REQUIREMENTS

Functional Requirements

Functional requirements	
Requirement identifier	RF01
Name of the requirement	Login.
Guy	■Restriction Requirement □
Description of the requirement	The web and mobile platform should allow users to access the system using a secure and verified login process.
Characteristics of the requirement	Only users registered in the system by the manager will have access to the system with their credentials.

Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF02
Name of the requirement	Registration of technicians.
Guy	■Restriction Requirement ☐
Description of the requirement	It allows users of the system to record relevant information about the technicians who will be in charge of the preventive or corrective maintenance of the machines.
Characteristics of the requirement	The user must enter data such as full name, employee ID, specialty (electrical, mechanical, etc.), role (technician, maintenance manager, etc.), email, phone number, and date of hire.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF03
Name of the requirement	Machine registration.
Guy	■Restriction Requirement ☐
Description of the requirement	It will have the interface that will allow new machines to be registered.
Characteristics of the requirement	Detailed information such as machine name, model, serial number, type (electrical, mechanical, computer, etc.), date of acquisition, and location must be provided for machine registration.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF04
Name of the requirement	Maintenance Inquiry
Guy	■Restriction Requirement ☐
Description of the requirement	The maintenance area manager will have access to consult the complete history of the maintenance carried out on each machine. Such as the technician who carried out the maintenance, the parts that were replaced and the reports generated during the process.
Characteristics of the requirement	Complete overview of the maintenance of each machine.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF05
Name of the requirement	Maintenance schedule
Guy	■Restriction Requirement ☐
Description of the requirement	The system will allow for the visualization of scheduled maintenance within a calendar.
Characteristics of the requirement	● Display of scheduled dates. ● Visualization of schedules. ● Visualization of machine data.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional
Signature (Authorization)	

Functional requirements	
Requirement identifier	RF06
Name of the requirement	Maintenance Assignment
Guy	■ Restriction Requirement ☐
Description of the requirement	It allows you to assign specific maintenance to a machine and a technician. The assignment must be made according to the needs of maintenance, the availability of technicians and the allocation of resources.
Characteristics of the requirement	● The user responsible for maintenance will select a specific machine that needs maintenance. ● The system will display a list of available technicians, based on their specialty, workload, and availability. ● The user must select the appropriate technician and specify the type of maintenance to be assigned. ● The user must enter the scheduled date and time to perform maintenance.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF07
Name of the requirement	Real-time notifications
Guy	■ Restriction Requirement ☐
Description of the requirement	The system includes notifications to send alerts to users about pending tasks, urgent incidents or predictive recommendations related to machine maintenance.
Characteristics of the requirement	Users (technicians, administrators, or maintainers) must be logged into the system and have their notification preferences configured within the system

Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF08
Name of the requirement	Maintenance Report
Guy	■Restriction Requirement ☐
Description of the requirement	System must allow to generate detailed reports on the maintenance activities carried out, including preventive and corrective maintenance
Characteristics of the requirement	• The system must generate the reports in standard formats such as PDF • Reports should contain details such as the machine ID, the type of maintenance performed, the date, the assigned technician, and the maintenance status.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF09
Name of the requirement	Consultation of Technicians' Schedules
Guy	■Restriction Requirement ☐
Description of the requirement	Consult the available schedules of technicians to perform predictive maintenance tasks, adjusting to the maintenance schedule and the availability of technical resources
Characteristics of the requirement	Users should be able to check technicians' schedules based on their previous assignments and predictive maintenance tasks to be performed.

Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF10
Name of the requirement	Check machine operating hours
Guy	■ Restriction Requirement ☐
Description of the requirement	This functional requirement allows users to consult the operating hours of each machine within the company.
Characteristics of the requirement	Users should be able to check the operating hours of the machines, i.e. the periods in which each machine is scheduled to be in operation within the company.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Functional requirements	
Requirement identifier	RF11
Name of the requirement	Predictive maintenance notification
Guy	■ Restriction Requirement ☐
Description of the requirement	This functional requirement establishes the need to send notifications to users (technicians, administrators, or maintenance managers) about predictive maintenance generated by the system, based on machine data.
Characteristics of the requirement	● The system must receive alerts from the predictive maintenance system when anomalies or conditions are detected that indicate a possible failure of a machine. ● Alerts should contain key information about the affected machine, the type of anticipated failure, and maintenance recommendations.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Non-Functional Requirements

Non-functional requirements	
Requirement identifier	RNF01
Name of the requirement	Authentication and authorization
Guy	■Restriction Requirement ☐
Description of the requirement	The system must implement secure authentication based on standard protocols such as OAuth 2.0 and support user roles with differentiated permissions (administrators, technicians and supervisors).
Characteristics of the requirement	● Inclusion of multi-factor authentication (MFA). ● Access and permission auditing.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Non-functional requirements	
Requirement identifier	RNF02
Name of the requirement	Data encryption
Guy	■Restriction Requirement ☐
Description of the requirement	All sensitive information must be encrypted using the AES-256 standard both in transit and at rest.
Characteristics of the requirement	● Use of HTTPS and SSL/TLS certificates for data transmission. ● Automatic encryption of sensitive databases.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Non-functional requirements	
Requirement identifier	RNF 03
Name of the requirement	Information backup
Guy	■ Restriction Requirement ☐
Description of the requirement	System data will need to be automatically backed up every 24 hours, ensuring at least 30 days of available backup history.
Characteristics of the requirement	• Configuring incremental and full copies. • Storage in multiple locations to prevent data loss due to disasters.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

Non-functional requirements	
Requirement identifier	RNF04
Name of the requirement	Multilingual Support
Guy	■ Restriction Requirement ☐
Description of the requirement	The system must be available in at least 2 languages: Spanish, English.
Characteristics of the requirement	• Support for automatic detection of the user's language. • Ability to easily add new languages in the future.
Requirement Priority	☐ High/Essential ■ Medium/Desired ☐ Low/Optional

Non-functional requirements

Requirement identifier	RNF05
Name of the requirement	Intuitive
Guy	■ Restriction Requirement ☐
Description of the requirement	The platform must have intuitive navigation that allows users to perform their tasks in a maximum of 3 clicks.
Characteristics of the requirement	• User-centered design (UX/UI). • Usability testing with end users.
Requirement Priority	■ High/Essential ☐ Medium/Desired ☐ Low/Optional

3.1. External Interfaces

This section describes the requirements related to external system interfaces. These interfaces include interactions with the user, with other systems (hardware and software), and with the communications necessary for the proper functioning of the application.

3.1.1 User Interface

The application must have a friendly and adaptable graphical user interface (GUI), optimized for mobile devices and tablets. This interface should allow users to interact efficiently with the system's functionalities, such as viewing assets, scheduling maintenance tasks, querying reports, and receiving notifications.

Requirements:

- **Home screen**: It should provide quick access to key system functions, such as task logging, maintenance queries, and asset parameter settings.
- **Forms**: Forms should be intuitive, with clearly labeled fields to record asset information and maintenance tasks.

- **Real-time notifications**: The interface should allow the user to receive notifications of pending maintenance or machines that need maintenance.

3.1.2 Interface with Other Systems

The system must be integrated with other systems, both hardware and software, for the correct management of assets and maintenance.

Requirements:

- Notification Systems: Integration with messaging or email platforms should allow alerts and notifications to be sent to users in case of incidents or scheduled tasks.
    - The system should be able to send push notifications to mobile devices, as well as emails to the responsible technicians.

3.1.3 Communications Interface

Communication between system components and with external systems must be based on well-defined communication standards to ensure the security, reliability and scalability of the system.

Requirements:

- Communication Protocols:
    - RESTful APIs for communication between the maintenance system and other systems (ERP, IoT sensors, etc.), allowing the exchange of data in JSON format.
    - Firebase Cloud Messaging (FCM) for real-time notification transmission, ensuring users receive alerts without delay.
- Safety:
    - Communication must be encrypted using security protocols such as HTTPS to protect data in transit.
    - Authentication and authorization should be implemented in APIs to ensure that only authorized users access sensitive data.
- Networks and Connectivity:
    - The system must be able to operate in environments with limited connectivity (e.g. when the mobile device does not have constant access to the internet) and must allow temporary storage of data for later synchronization.

3.1.4 Hardware Requirements

The system must be compatible with the following hardware devices:

- Mobile devices and tablets: The system will be compatible with Android and iOS devices with screens of at least 5 inches.

3.1.5 Software Requirements

Your application must support the following software environments:

- Mobile operating systems: The app must be compatible with the latest versions of Android (from version 9.0) and iOS (from version 12).
- Web browsers: If a web version is deployed, the app must be compatible with the latest browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

3.2. Functions

This section details the main functions of the system, focused on the efficient management of corrective and preventive industrial maintenance. These functions allow users to manage assets, schedule maintenance, generate reports and receive notifications, guaranteeing the operability of the machinery and optimizing resources.

3.2.1 Asset Management

The system must allow the management of all industrial assets subject to maintenance.

Functions:

- Register new assets, specifying characteristics such as machine type, location, acquisition date, and current status.
- Edit and update asset information when necessary.
- View assets using search filters (by name, location, status, type of maintenance required, etc.).
- Manage the maintenance history of each asset, including repairs, inspections, and parts changes.

3.2.2 Preventive Maintenance

The system should facilitate the planning and execution of preventive maintenance to minimize unexpected failures.

Functions:

- Schedule maintenance tasks based on specific dates, hours of operation, or asset conditions.
- Generate automatic reminders for technical staff before the scheduled maintenance date.
- Allow the assignment of responsible technicians for each preventive maintenance task.
- Record the maintenance performed, with details such as replaced parts, adjustments made, and subsequent status of the asset.

3.2.3 Corrective Maintenance

The system must allow for efficient management of corrective maintenance when asset failures occur.

Functions:

- Allow the creation of reports of failures detected by operators or IoT sensors.
- Assign technicians responsible for attending to incidents and monitor their progress.
- Record the diagnosis and corrective actions applied for each failure.
- Generate real-time alerts to inform about machines out of service or at critical risk.

3.2.4 Notifications and Alerts

The system should have a notification system in place to keep users informed about the status of assets and maintenance tasks.

Functions:

- Send push notifications and emails to technicians and supervisors about pending maintenance, reported incidents, and changes in asset status.
- Allow the configuration of custom alerts according to the criticality of the asset or type of maintenance.
- Display an alert dashboard in the main interface with information on the most urgent tasks.

3.2.5 User and Role Management

The system should provide different levels of access and control depending on the user's profile.

Functions:

- Register and manage user accounts with specific roles (administrator, supervisor, maintenance technician, operator).
- Define access permissions for each type of user, restricting certain functions according to their role.
- Enable secure authentication of users using credentials and two-factor authentication (if required).

3.2.6 Reporting and Data Analysis

The system must generate detailed reports on asset maintenance and technical team performance.

Functions:

- Generate periodic reports on the status of assets, maintenance history and associated costs.
- Show statistics on the average response time in corrective maintenance.
- Allow the export of reports in PDF and Excel formats for external analysis.
- Provide interactive charts and visualizations to identify trends and improve decision-making.

3.3. Performance Requirements

The system must meet the following performance requirements to ensure stable and efficient operation within a small company, ensuring adequate response times and scalability in the future.

3.3.1. User Capacity and Concurrency

- A total of 50 to 100 registered users are estimated in the first phase of implementation, with the possibility of scaling up to 200 users in future expansions.
- A maximum of 20 to 30 concurrent users are expected at peak times.
- The latency in the loading of critical modules (maintenance assignment, incident history and reports) must be less than 10 seconds under normal conditions.

3.3.2. Transactions Per Second

- The system must be capable of processing between 5 and 10 transactions per second (TPS) under normal operating conditions.
- In situations of increased demand, the infrastructure must withstand peaks of up to 20 TPS without affecting the stability of the system.

3.3.3. Data Storage and Management

- It is expected to store asset, maintenance and reporting information with the following estimated growth:
    1. Industrial assets: between 100 and 1,000 records in the database.
    2. Scheduled maintenance: approximately 500 records per year.
    3. Incident history: about 1,000 records per year.
    4. Registered technicians: ability to handle between 10 and 50 technicians.

- The database must support efficient queries with optimized indexes to avoid data retrieval delays.

3.3.4. Availability and Scalability

- The system must guarantee a minimum availability of 99%, allowing small windows of planned maintenance.
- A cloud storage scheme for reports and documents is recommended, minimizing the use of local servers.
- The architecture should allow for easy scalability in case of business growth.

3.3.5. Real-time notifications

- The Notifications API should handle an average of 5 to 10 notifications per minute, with peaks of up to 50 notifications per minute on critical events.
- Priority will be given to the delivery of notifications with a maximum delay of 10 seconds in urgent maintenance alerts or equipment failures.

3.3.6. Network and Communication Requirements

- The system should operate optimally with a connection of at least 2 Mbps for standard users and 5 Mbps for administrators with access to advanced reports.
- Support for 4G and WiFi mobile networks will be ensured for technicians accessing from mobile devices.

3.4. Design Constraints

The design of the MantéFix system is subject to various constraints that affect its development and implementation. These restrictions come from industry standards, safety regulations, hardware limitations, and specific technical requirements. The following are the main factors that restrict application design decisions.

1. Compliance with industry standards and regulations

Restriction: MantéFix must comply with maintenance management regulations such as ISO 55000 (Asset Management) and ISO 9001 (Quality Management), as well as occupational safety and data protection regulations.

Design impact: Strict access controls, encryption of sensitive information, and audits must be implemented to ensure regulatory compliance. In addition, the system must generate reports in accordance with the standards established in the industry.

2. Hardware and operating environment limitations

Restriction: The system will be used in industrial environments where devices may have limited or older hardware specifications, and where the internet connection may be unstable or intermittent.

Design impact: The application must be optimized to run on low-performance devices and support offline operation with data synchronization when the connection is restored.

3. Cross-platform compatibility

Restriction: MantéFix should run on modern web browsers and mobile devices with Android and iOS operating systems, without relying on specific proprietary software.

Design impact: The application must be developed with standard web technologies such as React, PWA (Progressive Web Apps) or cross-platform frameworks such as Flutter or React Native to ensure compatibility on different devices and screen resolutions.

4. Scalability and performance

Constraint: The system must be able to handle an increasing number of assets, maintenance orders, and users without affecting its performance.

Design impact: A scalable architecture based on microservices and databases optimized for large volumes of information, such as PostgreSQL or MongoDB,

should be used. It is also necessary to implement caching strategies and query optimization to maintain low response times.

5. Information Security and Privacy

Restriction: Since the system will handle sensitive data about equipment, maintenance, and technical personnel, it is mandatory to comply with security standards such as ISO/IEC 27001 (Information Security Management) and local data protection regulations.

Design impact: Strong authentication (OAuth 2.0, JWT, multi-factor authentication), encryption of data in transit and at rest, and role-based access controls should be included to limit the information that each user can view and modify.

6. Integration with external APIs

Restriction: MantéFix must integrate third-party services such as APIs for real-time notifications (Firebase Cloud Messaging), reporting (Power BI) and content translation (Google Cloud Translation API).

Design impact: The architecture must be modular and flexible, allowing the smooth integration of these APIs without compromising system performance. It is also necessary to handle potential downtime of external services using retry and caching mechanisms.

7. Accessibility requirements

Restriction: The app must be accessible to technicians with disabilities, complying with standards such as WCAG 2.1 (Web Content Accessibility Guidelines).

Design impact: Features such as screen reader support, keyboard navigation, high-contrast color schemes, and adjustable font sizes should be included to improve usability.

3.5. System Attributes

1. **Safety:**
    - Role-based access control (administrators, technicians, supervisors).
    - Multi-factor authentication (MFA).
    - Data encryption in transit (TLS/SSL) and at rest (AES-256).
2. **Usability:**
    - Intuitive and easy-to-use interface.
    - Compatibility with mobile devices and web browsers.
    - WCAG 2.1 accessibility.
3. **Reliability:**
    - 99.9% availability.
    - Automatic data backup every 24 hours.
    - Fault tolerance through redundant servers.
4. **Maintainability:**
    - Modular code for easy updates.
    - Detailed documentation for developers.
    - Automated testing for errors.
5. **Scalability:**
    - Ability to handle more users and data without performance degradation.
    - Microservices-based architecture.
6. **Performance efficiency:**
    - Response time of less than 10 seconds in critical operations.
    - Ability to process 5-10 transactions per second.

3.6. Other Requirements

Any other requirements that do not fit into another section.

4. Selection and justification of methodology

For the development of the **Preventive, Corrective, and Predictive Maintenance System**, the **SCRUM** methodology is proposed, a widely recognized agile framework for software development projects. Below is the selection and justification:

1. Selected Methodology: SCRUM

SCRUM is an agile methodology based on an iterative and incremental approach to manage and control complex projects. It enables continuous adaptation to changes, frequent delivery of functional products, and fosters active collaboration among team members.

2. Justification:

The selection of SCRUM for this project is based on the following reasons:

- **Flexibility to changes:** Since this project involves multiple functionalities (preventive, corrective, and predictive maintenance), SCRUM allows adaptation to changing requirements or priorities, which is essential for developing complex and innovative solutions.
- **Incremental delivery:** SCRUM divides the development process into short sprints or cycles (usually 2 to 4 weeks), ensuring the continuous delivery of functional components of the application. This guarantees visible and consistent progress.
- **Active collaboration:** SCRUM encourages communication between developers, the Product Owner (representing the client or end user), and the Scrum Master. This ensures that the users' needs are always at the center of development.
- **User-focused:** By prioritizing functionalities through the product backlog, SCRUM ensures that the most important features for the client are developed first, optimizing the user experience.
- **Risk reduction:** Short work cycles allow for early identification and resolution of issues, reducing technical and implementation risks.

- **Continuous improvement:** At the end of each sprint, the team conducts a retrospective to identify areas for improvement, allowing processes to be adjusted and workflows optimized.

3. Application of SCRUM in the Project:

The development of the system will be divided into the following key steps:

1. **Creation of the Product Backlog:**
    - Identify all the necessary functionalities of the system, such as asset management, task scheduling, report generation, API integration, and real-time notifications.
    - Prioritize these based on their impact on the general objectives.
2. **Sprint Planning:**
    - Select a set of tasks from the backlog that can be completed within the sprint.
    - Establish clear objectives for each sprint.
3. **Sprint Execution:**
    - Develop and test the selected functionalities.
    - Hold daily meetings (Daily Scrum) to evaluate progress and resolve impediments.
4. **Sprint Review:**
    - Present the developed functionalities to the Product Owner for evaluation and feedback.
5. **Sprint Retrospective:**
    - Analyze the positive and negative aspects of the sprint, identifying improvements for the next cycle.
6. **Product Increment:**
    - At the end of each sprint, a functional and improved product increment will be delivered, incorporating the newly developed features.

5. Architecture selection and justification

Selected Architecture: Microservices Architecture

For the development of the Preventive, Corrective, and Predictive Maintenance System, a microservices-based architecture has been chosen. This modular approach allows the system to be divided into small, independent components that communicate with each other through APIs.

Justification of the Architecture:

1. Scalability:
    - The microservices architecture enables individual scaling of each system component based on specific needs, optimizing resources and performance.
    - For instance, if the real-time notifications module requires more resources due to an increase in incidents, only that service is scaled without affecting others.
2. Modular and Decentralized Development:
    - Each microservice can be developed, deployed, and maintained independently, which facilitates team collaboration and accelerates development.
    - This allows features such as asset management, task scheduling, and report generation to be developed and deployed autonomously.
3. Ease of Integration with APIs:
    - The use of microservices aligns perfectly with the need to integrate at least three APIs into the system. Each API can connect to a specific microservice without adding complexity to the overall structure.
    - Example: An API for failure prediction is integrated only with the predictive maintenance module.
4. Resilience and Fault Tolerance:
    - If one microservice fails, the others continue functioning, ensuring system availability.
    - For example, if the report generation module encounters an issue, the asset tracking and notifications modules will still operate.

5. Adaptability to Future Expansions:
    - The microservices architecture makes it easy to add new functionalities or modernize individual components without impacting the rest of the system.
    - This allows the system to adapt to the changing needs of businesses and industries in the future.
6. Compatibility with Modern Technologies:
    - The architecture is compatible with container technologies like Docker, which simplifies the deployment and management of microservices.
    - It can also integrate with orchestration platforms like Kubernetes for managing services in production environments.

Key Components of the Architecture:

1. Backend:
    - Developed using Django REST Framework, which provides a robust structure for creating secure and scalable APIs.
    - Each key functionality of the system (asset management, tasks, notifications, etc.) will be an independent microservice.
2. Frontend:
    - Developed with React, consuming backend services through REST APIs.
    - This ensures a smooth and intuitive user experience.
3. Database:
    - Distributed databases such as PostgreSQL for structured data persistence and MongoDB for unstructured data, depending on the requirements of each microservice.
4. Asynchronous Messaging:
    - To handle real-time notifications, a messaging system like RabbitMQ or Apache Kafka will be implemented, enabling efficient communication between microservices.
5. Authentication and Security:
    - Use of standards like JWT (JSON Web Tokens) for user authentication and API authorization.

- Implementation of security measures such as data encryption and protection against common attacks (CSRF, XSS, etc.).

6. Workflows and version control tools

Project Workflow

1. Project Planning

- Initial team meeting:
  The scope of the project, the main objectives and responsibilities are distributed among the members:
  - Leader: José Carlos Amparo Barba.
  - Development and testing: Miguel Ángel Aranda Castillo, Javier de la Cruz Polanco, Ariana Lizeth Maciel Leyva, Ian Armando Labrada Farías.

2. Definition of the Software Requirements Specification (ERS):

- Document in detail the functional and non-functional requirements of the system, including:
  - Asset management: registration, cancellation, and modification of information.
  - Task scheduling: frequency options, responsible parties and status.
  - Report generation: templates and required data.
- Include diagrams such as use cases, activity diagrams, and data modeling.

3 Architecture Design

- Create an architecture diagram that integrates the main components: React (frontend), Django (backend), MySQL (database), and Firebase.
- Design RESTful API in Django for communication between frontend and backend.

4 Backend Development

- Configure Django with MySQL and Firebase.
- Create models to manage assets, maintenance tasks, reports, and users.
- Implement APIs to create, read, update and delete (CRUD) maintenance data.

5 Frontend Development

- Set up React environment with reusable components.
- Design views such as the dashboard, asset list, scheduled tasks, and notifications.
- Define a design system based on Material UI, Tailwind CSS or Bootstrap.
- Performance optimization Lazy Loading of components and use of tools like React Profiler.
- Consume Django APIs using Axios or Fetch

6 Testing and Validation

- Perform unit tests to verify each module.
- Perform integrated testing to ensure proper communication between components.
- Validate the functionality with real or simulated data.

7 Deployment and Maintenance

- Configure servers for backend deployment (for example, AWS or DigitalOcean).
- Configure CI/CD for continuous updates.
- Monitor performance and maintain the database.

Version Control Tools

1 GitLab as Core Platform

    - Repository hosted on GitLab.
    - Setting up a branch flow like:
        - main: Stable, production-ready code.
        - develop: active development and integration of new features.

        - feature/*: development of new specific functionalities.
        - bugfix/*: fix bugs found in any branch.
        - hotfix/*: urgent bug fix in production.

Repository link in GitLab: https://gitlab.com/0321101714/mantefix

2 Management Responsibility

- The person in charge of the repository will be Amparo Barba Jose Carlos.
- Supervision and approval of merge requests

3 Automation and CI/CD in GitLab

- Set up pipelines in GitLab CI/CD to:
    - Perform automatic tests when performing a push.
    - Implement automated backend and frontend deployments.

4 Development Environments

- Android Studio: for mobile application development
- Visual Studio Code: for frontend development with React and backend with Django.