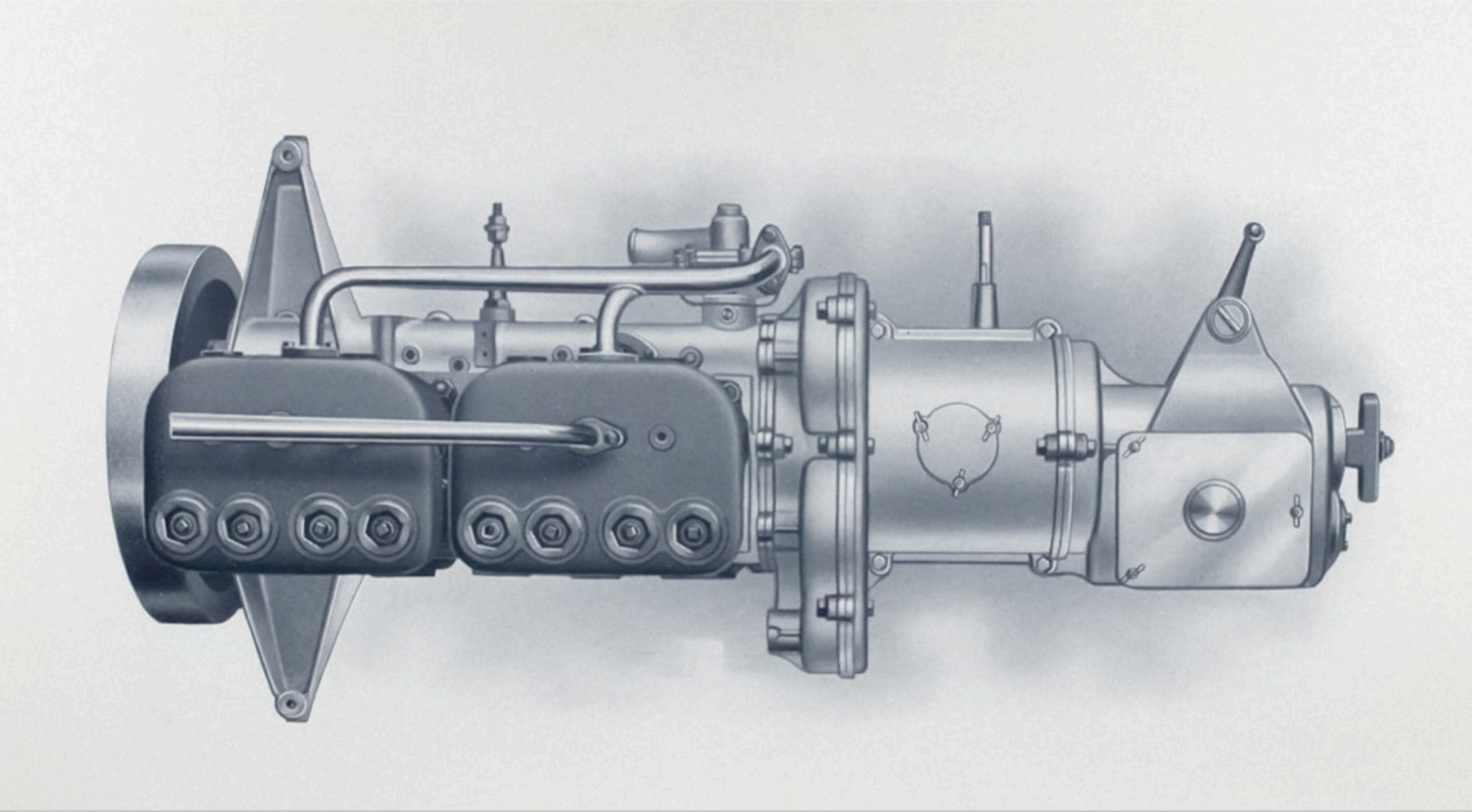


SMART MAINTENANCE

Salleha

Systems Analysis and Design
First Semester
2025/2026

University of Jordan

Systems Analysis and Design
Supervised by: Dr.Hamad Alsawalqah
First Semester 2025-10-15

Student Name	ID
Anas AL-Jallad	0225343
Orjoan Aldabaibah	0224933
Mosa Daradkah	0222634
Haneen Alajaleen	0226320
Shaima Hasan	0227646

Version Control

Version	Description
Version 1.0	Initial version for the software documentation. Added Project Initiation and Project Management Plan

Excutive Summary

Contents

Tables

Figures

1. Project initiation

1.1. Project Overview

SALLEHA is a platform designed for managing maintenance requests in facilities like offices or residential buildings, making maintenance and reporting more efficient and easier. Users can report issues, track progress, and get updates. Admins and technicians can assign, prioritize, and resolve tasks effectively.

1.2. Problem Definition

In many residential buildings, offices, and shared facilities people often face significant challenges in reaching authority of those In charge of maintenance managers and staff. In traditional methods such as : emails, paper forms, or phone calls, are typically inefficient, lack transparency and lead to delays. This often creates a communication gap between users and the authorities responsible, which results in frustration, unaddressed issues, and potential safety hazards.

1.2.1. Issues

Issue	weight
Users often struggle to reach the right maintenance personnel, resulting in delays or ignored requests. Without a centralized and accessible system, reporting issues becomes time-consuming and unreliable.	10
Maintenance teams often work without proper tools to prioritize, assign, and track tasks. This leads to missed or delayed repairs, no clear ownership of responsibilities, and no data to measure performance or improve operations.	9
Users rarely receive updates on the status of their maintenance requests. This lack of visibility creates frustration and reduces trust in the system, while maintenance teams struggle to keep everyone informed.	7
Users lack Privacy through and through with traditional reporting methods, this can lead to an upset and distrust to some people. Users care for their own privacy hence why some reports have never been sent before because of their own worry about the system.	6

1.2.2. Objectives

1. Simplify and centralize issue reporting through a user-friendly web/mobile interface that allows users to easily report maintenance problems and is available 24/7.
2. Enhance communication and transparency by providing real-time updates and notifications on request statuses
3. Create an analytics dashboard to provide administrators with insights and help them to identify trends and areas needing improvement.
4. Create a confidential system that ensures users anonymity and keeping their data secure and private from intruders.

1.2.3. Requirements

1. The system must ensure data security and protect the privacy of all users.
2. The system must be intuitive and user-friendly, allowing non-technical users to navigate and interact with it easily.

3. The analytics dashboard must be restricted to administrators only.
4. Maintenance reports must be submitted anonymously to ensure user comfort and honesty.

1.2.4. Constraints

1. Development costs must not exceed 45,000 JD
2. The project should be done by Sunday 4, Jan 2026

1.2.5. Vision Document

1.2.5.1. Problem Description

Ever since the digitalization of almost everything, people's expectations rose. People are in constant demand of systems that fulfills there needs. Current methods are almost obsolete they are ineffecient and insuffecient because the older methods have lack of transparency, increased delays, and unavaliable hence the overall user frustration, not to mention the diffuculty of managing the reports.

Without a system to hold everything together it require a lot of effort to pull through the maintenance tasks. To satisfy users, they need a system to adapt to their needs. Providing a smooth, painless experience through an easy to use interface. A system is needed such that it enables feedback submission, tracks administrative responses, and provides data-driven insights for continuous service improvement. Delaying this solution risks further dissatisfaction and missed opportunities for institutional growth.

1.2.5.2. System Capabilities

1. Ticket Submission Capabilities.
   - Users are able to submit maintenance tickets through a user-friendly interface.
   - Tickets include:
     - Text description of the issue.
     - Image attachments to provide visual context.
     - Location tagging to help technicians identify where the issue is.

2. Role-Based Dashboards.
   - The system provides separate dashboards based on user roles.
   - Roles include:
     - Users: Can submit and track tickets, view status updates, and provide feedback.
     - Technicians: Can view assigned tickets, update task statuses, and log maintenance work.
     - Administrators: Can assign tasks, monitor performance, and access analytics dashboards (restricted access).

3. Task Assignment and Scheduling.
   - Administrators can assign tasks to technicians based on priority and availability.
   - System supports:
     - Priority-based task distribution depending on if its urgent or normal.
     - Scheduling of tasks to optimize technician workload and response time.

4. Push and Email Notifications.
   - The system provides real time updates on ticket statuses.
   - Notification types include:
     - Push notifications via web or mobile.
     - Email alerts for important status changes.

5. Maintenance History and Analytics.
   - System keeps a log of all past maintenance activities.
   - Analytics dashboard features:
     - Insights into frequent issues by area or equipment.
     - Performance tracking for continuous improvement.
     - Access restricted to administrators only.

1.2.5.3. Business Benefits

- Providing a better quality of life.
- Overall increase of the user satisfaction.
- Increasing speed of maintenance tasks.
- Eliminating delays and lessening risks.
- Providing better communication channels.

1.3. Feasbility Studies

1.3.1. Techinical Feasibility

The technical feasibility assesses the technological components necessary to develop and operate the SALLEHA platform. This includes evaluating the required hardware, software tools, and the technical skills essential for building and maintaining the system.

Technology: The SALLEHA website is built using basic and easy-to-use Web tools like HTML, CSS, JavaScript, Bootstrap, and jQuery. These Tools help create a clean and responsive design that works well on Different devices. We also use Canva to design simple and clear images and graphics, making the website easy for Seniors users to understand and use .

Cloud Hosting: We are using GitHub to store and manage the project online. It helps us work together, keep track of changes, and easily share the project with others.

All the required resources including hardware, software tools, and hosting services are already available and accessible, ensuring smooth development and operation of the SALLEHA platform.

Since all these technologies are part of what we have learned at university and practiced in various projects, we are fully capable of developing and maintaining the SALLEHA platform using them. Our academic background and hands-on experience give us the technical foundation and confidence to build this system successfully.

1.3.2. Operational Feasibility

The proposed web and mobile application is operationally feasible. It is designed to receive maintenance requests in facilities such as universities, offices, or residential buildings, enabling users to report issues, track progress, and receive updates. Since it is both a web and mobile application, users can access it from anywhere. We expect that our system will gain wide acceptance from users, admins, and technicians because it addresses an essential need and saves time and effort. It will have clear privacy guidelines and mechanisms to ensure that our users' data will be secured, and it complies with the policies set by the country's laws and institutions. Additionally, the system is well-suited to the local culture and environment. The end users are capable of using it smoothly and effectively without requiring extensive training, due to its simple and user-friendly design.

1.3.3. Economic Feasibility

Development Costs:

Expense Category	Amount
Salaries	20,000 JD
Equipment and installations	8,000 JD
Training	1,500 JD
Facilities	2,000 JD
Utilities	1,000 JD
Travel\Miscellaneous	2,000 JD
Total	**39,500 JD**

Table 1: Devlopment Costs

Operational Costs:

Service	Annual Cost(Per year)
Operational maintenance	7,000 JD
Total Cost	**7,000 JD**

Table 2: Operational Costs

Intangible Benefits
Enhanced Institutional Trust and reputation
Increasing users satisfaction
Saving time and effort for both users and Institutions

Table 3: Intangible Benefits

Benefit and Payback Analysis:

Category	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
Value of benefits	0	16,000 JD	17,000 JD	18,000 JD	19,000 JD	20,000 JD
Development costs	−39,500 JD	0	0	0	0	0
Annual expenses	0	−7,000 JD	−7,000 JD	−7,000 JD	−7,000 JD	−7,000 JD
Net Benefit / Costs	−39,500 JD	9,000 JD	10,000 JD	11,000 JD	12,000 JD	13,000 JD
Discount Rate (7%)	1	0.934	0.873	0.813	0.763	0.713
Net Present Value (NPV)	−39,500 JD	8,406 JD	8,730 JD	8,943 JD	9,156 JD	9,269 JD
Cumulative NPV	−39,500 JD	−31,094 JD	−22,364 JD	−13,421 JD	−4,265 JD	5,004 JD
Payback Period	**4 years+**					**4.6 years**

Table 4: Benefit and Payback Analysis

Lifetime ROI = $\frac{90{,}000-74{,}500}{74{,}500} = 0.208 \vee 20.8\%$

Annual ROI = $\frac{20.8\%}{5} = 4.16\%$

1.3.4. Schedule Feasibility

Phase	Task	Estimated Time
Planning	Define Project Scope & Objectives	1 week
Analysis	Requirements Gathering, Process Analysis, and Document Delivery	2 weeks
Design	System Architecture and Interface Design	2 weeks
Implementation	Development of Core Features	4 weeks
Testing	System Testing and Quality Assurance	2 weeks
Deployment	System Deployment	1 week

Table 5: Project Development Schedule

1.3.5. Legal Feasibility

The proposed platform fully aligns with Jordanian laws, university policies, and institutional standards. All required approvals will be obtained from the University of Jordan's relevant departments before deployment. The system does not infringe upon any legal frameworks or intellectual property rights.

Licensing Compliance: All development tools, frameworks, and libraries used in the platform will be properly licensed. Open-source components will be used in accordance with their respective licenses, while any proprietary technologies will be incorporated only after acquiring valid usage rights.

Copyright and Intellectual Property Protection: The platform will comply with the Jordanian Copyright Law No. 22 of 1992 and its amendments. Any third-party content whether text, images, or software will be original, licensed, or used under fair use conditions with full attribution.

Data Privacy and Confidentiality: To comply with the Jordanian Personal Data Protection Law No. 24 of 2023, the system will:

1. Obtain explicit user consent prior to collecting or processing personal information.
2. Employ encryption and secure storage for sensitive data.
3. Ensure that personal data is used strictly for its intended purpose and accessed only by authorized personnel.

Electronic Communication and Records Compliance: Under the Electronic Transactions Law No. 15 of 2015, all digital communications and transactions carried out through the platform will be handled as legally recognized records, protected through appropriate technical and procedural safeguards.

Terms of Service and Legal Disclosures: Users will be provided with clear Terms of Service and Privacy Policy agreements outlining:

- Data collection and usage practices
- User rights and responsibilities
- Risk disclosures and security provisions

These documents will comply with both university IT regulations and national legal requirements.

1.4. Recommended Solution and Expected Project Deliverables

To manage the maintenance requests of issues as they arise, we can use a great solution: a Maintenance Request software system that allows requesters to report maintenance issues directly to the maintenance team using a web-based form or mobile app. It helps streamline communication, submission, and tracking without wasting time gathering complete and accurate information or delaying repairs. The people who the maintenance teams usually rely on, such as employees and visitors, will be able to submit detailed maintenance forms that include descriptions, images, and location information. Other processes such as workflows for reviewing and approving requests will be managed through a dashboard that allows the team to assign, prioritize, and monitor tasks. Technicians can update the status of each request in real time, and notifications will be sent to users to inform them about progress and completion.

Expected Deliverables: A Maintenance Request Software that will include:

- **Request Submission:**

Maintenance teams will accept requests through the system to ensure they collect all necessary information to proceed with other maintenance processes.
Users will submit it like a post; It'll have a title, location, photo and description.

- **Review and Approve Requests:**

A dashboard and analysis tool will help the organization review all forms and decide which requests will be approved. Each request will be evaluated to ensure that it is valid, not redundant, and not already being addressed.

Status Updates to Deliver Better Customer Service: To build trust between managers and customers, there will be a communication tool that responds back to requesters to inform them that their requests are accepted and to update them about the status of their issues until completion. These updates will be automated through real-time notifications.

Database Design and Documentation: For storing and tracking requests, there will be request records that provide a clear view of all issues, and a database that documents what issues were reported, how they were resolved, and when. This will help with future planning and decision-making, as well as provide tools for summarizing the analysis, design, and implementation process.

Performance Tracking: Updates on team work status for measuring team performance will help ensure that the original problem has been addressed and will identify bottlenecks in the request process. This will lead to providing excellent customer service and demonstrate that the maintenance team is responsive, works well, and continues improving the organization's overall reputation. This will be done by tracking average response time and turnaround time.

1.5. Local and Global Impact of the Proposed Solution

Locally: The maintenance request system will ensure accuracy and enable the maintenance team to begin planning and scheduling maintenance work more quickly through automatically generated work orders from approved requests. Following best practices will also provide better customer service in this area. Automation means better experiences! It will reduce delays in handling requests, minimize manual paperwork, and ensure maintenance work is managed from start to finish through automated tools. This can lead to better resource management, higher efficiency, and greater satisfaction not just among staff but also among customers by keeping them updated about the status of their requests without delaying feedback.

Globally: It contributes to digital transformation and sustainability efforts. It is essential for businesses of all sizes to rely on such systems in their operations to efficiently allocate resources and maximize the performance of their assets. A well-designed maintenance request system demonstrates how technology enables organizations to make data-driven decisions to achieve better operational efficiency by centralizing all maintenance requests in one platform.

2. Project Management plan

2.1. Project Organization

The project organization shows how the team is structured and who is responsible for each part of the project. It helps make sure everyone knows their role and who to report to, keeping the work organized and efficient. The structure for our team also supports good communication between team members during development, as shown in Figure 1.

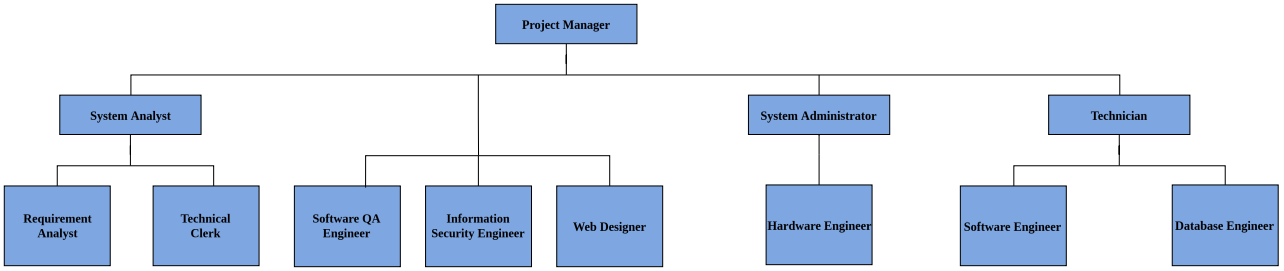


Figure 1: Project Organizational Structure

Assigned Membe	Roles
Anas	Project Manager, System Analyst, Requirement Analyst
Orjoan	Technical Clerk, Technician
Mosa	System Administrator, Hardware Engineer,Database Engineer
Haneen	Software Engineer, Web Designer,
Shaima	Software Engineer, Web Designer, Technical Clerk
External	Information Security Engineer, Software QA Engineer

Table 6: Team Roles Assignments and Responsibilities

2.2. Roles and Responsibilities

In this section, we will explain each role with its responsibilities in perspective. see Table 7.

Role	Responsibility
Project Manager	Responsible for planning, organizing, and overseeing projects to ensure they are completed on time, within budget, and meet quality standards. They coordinate team efforts, manage resources, and communicate with stakeholders throughout the project lifecycle.
System Analyst	Systematically assesses how businesses function by examining the inputting and processing of data and the outputting of information with the intent of improving organizational processes.
Requirement Analyst	Gathers and documents user requirements, ensuring alignment between business needs and system design.

Technical Clerk	Assists in maintaining documentation, schedules, and techinical records. Supports techinical team with admin tasks.
Software QA Engineer	Ensuring that software products meet the highest standards of quality and functionality.
Information Security Engineer	Rresponsible for designing, implementing, and maintaining security systems to protect an organization's data and networks from cyber threats. They also monitor security measures, respond to incidents, and ensure compliance with security policies and regulations.
Web Designer	Responsible for creating the visual layout and user experience of websites, ensuring they are both attractive and functional. Their tasks include designing page layouts, coding navigation, and collaborating with clients to meet their needs.
System Administrator	Responsible for managing and maintaining an organization's computer systems and networks, ensuring they operate efficiently and securely. This role often involves troubleshooting issues, installing software, and managing user accounts.
Hardware Engineer	Responsible for analyzing blueprints and technical drawings, reviewing system tests and performing updates as needed, implementing the latest systems and processes and ensuring everyone follows them, monitoring the manufacturing and assembly of hardware equipment, acting as the technical leader in product development
Technician	Responsible for installing, maintaining, and repairing equipment and systems across various industries. Their key duties include troubleshooting issues, performing routine maintenance, and ensuring compliance with safety standards
Software Engineer	Responsible for designing, developing, testing, and maintaining software applications and systems to meet user needs. Their responsibilities include analyzing user requirements, writing and testing code, and collaborating with other team members to ensure software functionality and performance.
Database Engineer	Rponsible for designing, implementing, and maintaining database systems to ensure efficient data storage and retrieval. Their key responsibilities include database design, data security, performance optimization, backup and recovery, and data migration.

Table 7: Roles and Responsibilities

2.3. Software Process Model

We are going to use the Waterfall Software Process Model. This model is suitable because our project goals and requirements are clearly defined from the beginning, and we have a strict timeline to follow. Since this model is based on a highly structured approach, it will help us maintain organization and ensure that all deliverables are completed on time. See Figure 2.

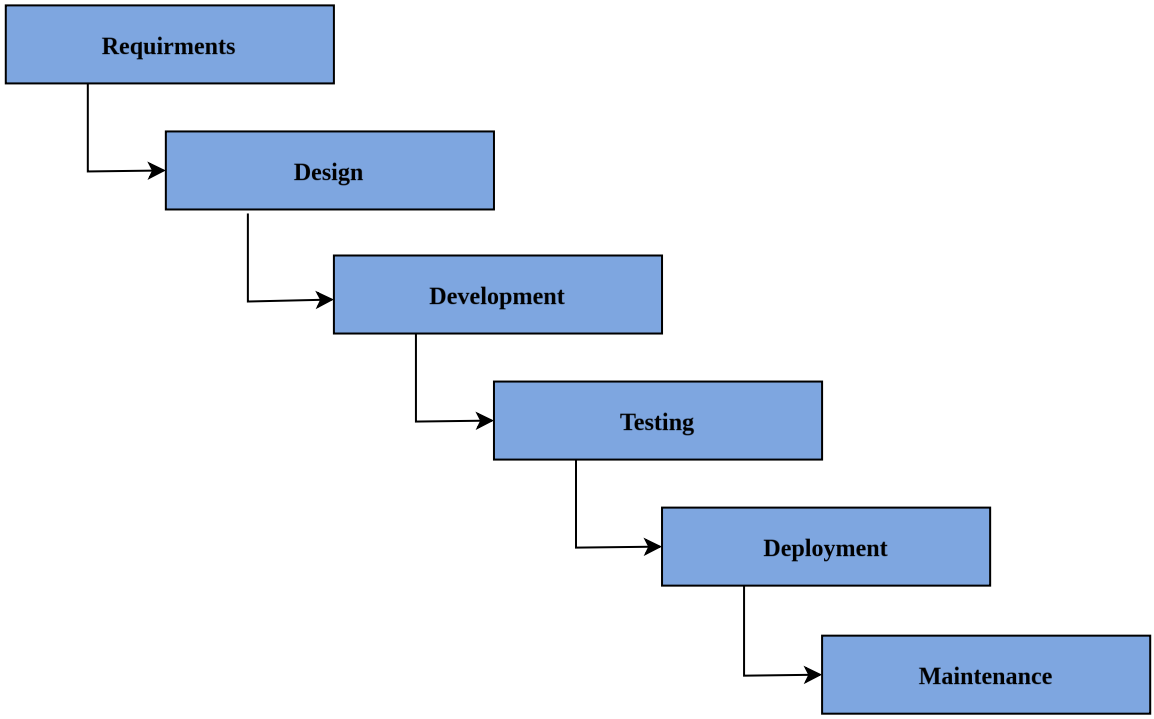


Figure 2: Waterfall Software Process Model

2.3.1. Main Phases :

1. **Requirements Analysis and Specification:**
   - Requirements Analysis: Gathering and understanding all the requirements of the client, then documenting and analyzing them.
   - Requirement Specification: Documenting the analyzed requirements in a software requirement specification document that serves as a reference for the next phases.
2. **System Design:**
   - Translating the requirements from the requirement specification document into a detailed system design as well as creating the overall architecture.
3. **Development:**
   - Developing the web and mobile applications according to the designs created earlier, using a suitable programming languages and frameworks.
4. **Testing and Deployment:**
   - Testing the whole software and verifying that all components work correctly and satisfy user expectations. After testing, the software is ready and available for use.
5. **Maintenance:**
   - The final, ongoing phase. It ensures that the software remains functional, secure, and up-to-date throughout its operational life.

2.4. Project Environment

2.4.1. Procedures

- **Initiation:**
  - Establish the project team, define the requirements goals, and potential risks.
- **Planning:**
  - Create a detailed project plan using the Waterfall methodology, this includes outlining all necessary steps, allocating resources, and developing cost, schedule, and communication plans to achieve our outcomes.
- **Execution and Testing:**
  - Implement the planned activities and test them carefully to ensure quality and functionality.
- **Monitoring:**
  - Track progress and compare it with planned goals to ensure timely delivery and quality control.
- **Documentation:**
  - All design diagrams, reports, and testing results are documented using Typst and stored in a shared repository to ensure collaboration among team members.

2.4.2. Tools

Tool	Purpose
Development Tools	Visual Studio Code for developing the web application, and Flutter for building a responsive mobile application using one shared codebase.
Documentation	Typst for creating our PDF documentation.
Version Control	GitHub for sharing and tracking project progress and managing team collaboration.
Design & Prototyping	Figma for creqting UI/UX design, and Draw.io for diagrams.
Database Management	MySQL, chosen for its reliability, speed, and support for relational data.
Testing	JUnit, an open-source testing framework, to verify our code's correctness and performance.
Communication Tools	Google Meet, WhatsApp, and GitHub for coordination.

Table 8: Tools

2.4.3. Hardware (HW) Resources:

Category	Description
Backend Server	Personal laptops and PCs for developing both the web and mobile applications.
Database Server	A server for hosting MySQL database, optimized for security and data backup.
Testing Devices	PCs for testing the website, and Android smartphones for mobile testing

Table 9: Hardware (HW) Resources

2.4.4. Software (SW) Resources:

Category	Description
Frontend Technologies	HTML, Tailwind CSS, and JavaScript for dynamic and responsive designs.
Mobile Development	Flutter for building the mobile app.
Frameworks and Libraries	React for web development, and Flutter for mobile.
Backend Technologies	Java and Node.js for handling server-side operations and building a secure and scalable backend.
Database	MySQL for storing and managing maintenance request data efficiently.
Operating System	Windows

Table 10: Software (SW) Resources

2.5. Project tasks

Phase	Task	Detailed task	EstimatedTime
• Resource & Schedule Planning (T1) • Requirements Gathering (T2)	• Develop Work Breakdown Structure for web and mobile app development • Create risk register and define quality standards for the application	• Define project phases and deliverables • Assign WBS elements to team members • Identify risks and quality objectives • Define acceptance criteria and mitigation strategies	1 weeks
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• **Skills**
	  ▸ A project leader to coordinate meetings, and define a schedule
	  ▸ A team work with technical and technology skills
	• **Hardware**
	  ▸ Team members laptops for development and documentation
	  ▸ A Shared test device for quick mobile and web checks
	• **Software**
	  ▸ GitHub is used to store both the Typst documentation files and the platform's source code
		• **Dependencies**
		  ▸ Approved project charter
		  ▸ Availability of key stakeholders for interviews such as the requesters and the facility managers
		• **Constraints**
		  ▸ To complete this phase in 1 week
		  ▸ The availability of stakeholders, so that the approvals and reviews will not be delayed
		  ▸ Legacy constraints
		  ▸ Budget constraints
			• **Deliverables**
			  ▸ Work breakdown structure diagram
			  ▸ Table with risk, impact, probability, priority, and mitigation
			  ▸ Quality standards document (acceptance criteria checklist)
			• **Milestones**
			  ▸ M1.1: Charter Approved and communicated to the team.
			  ▸ M1.2: Work Breakdown Structure completed and reviewed by all members.

	▸ Draw.io for diagrams and Figma for design ▸ Typst for documentation ▸ Risk register template, and a simple quality checklist template ▸ Google Sheets for schedule and resource tables and the risk register		▸ M1.3: Resource plan and schedule baseline approved. ▸ M1.4: Risk register and quality standards finalized.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Analysis	• Requirements Elicitation (T3) • Requirements Modeling & Documentation (T4) • Requirements Validation (T5)	• Conduct stakeholder interviews • Distribute surveys to end-users • Observe existing maintenance request processes • Write Software Requirements Specification (SRS) • Create use-case diagrams for maintenance workflows • Develop entity-relationship diagrams (ERDs) for database • Document non-functional requirements (performance, security, compatibility) • Facilitate requirements review sessions	2 weeks

		• Build wireframes for React web and Flutter mobile interfaces • Resolve conflicts and ambiguities • Obtain formal sign-off on SRS	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• **Skills**
	  ‣ Business Analyst to lead stakeholder interviews and surveys
	  ‣ Designers to create wireframes and prototypes for the React web and the Flutter mobile.
	  ‣ DB Designer to build ERDs and map entities to DB structure
	  ‣ Developers
	  ‣ Project manager to be responsible for the overall execution of the project
	  ‣ Quality assurance specialists
	• **Hardware**
	  ‣ Team laptops
	• **Software**
	  ‣ Elicitation tools: Google Forms for surveys, Google Meet for interviews, and audio notes.
		• **Dependencies**
		  ‣ Stakeholder availability
		  ‣ Access to collaboration tools so the informations can be shared and reviewed.
		• **Constraints**
		  ‣ 2 weeks for full elicitation, modeling and validation.
		  ‣ Stakeholder time; limited availability may restrict depth of interviews
		  ‣ Data privacy when handaling the facility data
			• **Deliverables**
			  ‣ Software Requirements Specification
			  ‣ Use-case diagrams for the main maintenance workflows
			  ‣ Entity-Relationship Diagram
			  ‣ Wireframes
			  ‣ Survey results and interview notes
			  ‣ Non-functional requirements list
			  ‣ Requirements validation report
			  ‣ Requirements Traceability Matrix
			• **Milestones**
			  ‣ M2.1: Complete stakeholder interviews, surveys, and observations.
			  ‣ M2.2: Draft Software Requirements Specification (SRS) prepared.

	▸ Draw.io for diagrams, MySQL Workbench, and a UML tool for use-case diagrams. ▸ Figma for early UI/UX validation by creating wireframes and low-fidelity prototypes ▸ GitHub for collaboration and tracking ▸ Requirements traceability matrix spreadsheet		▸ M2.3: Software Requirements Specification (SRS) reviewed and approved by stakeholders. ▸ M2.4: Requirements validation and traceability matrix completed.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Design	• Architectural Design (T6) • High-Level (Logical) Design (T7) • Detailed (Physical) Design (T8) • Design Review & Approval (T9)	• Choose overall system architecture for web and mobile • Define network topology and cloud infrastructure • Break system into modules (frontend, backend, database) • Define REST API contracts and module interfaces • Draft high-level sequence diagrams for maintenance request workflows • Create class diagrams for React and Flutter components • Design database schema with tables,	2 weeks

		indices, and constraints • Specify UI layouts and navigation flows for web and mobile • Define error-handling and logging approaches • Organize design walkthroughs with team and stakeholders	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**
	• **Skills** ‣ Software Architect experienced in web and mobile systems ‣ UI/UX Designer ‣ Backend Developer ‣ Frontend Developers (React and Flutter) ‣ Database Engineer for schema and constraints ‣ QA Engineer • **Hardware** ‣ Developer laptops and PCs ‣ Cloud host for test servers • **Software** ‣ Visual Studio Code, Android Studio ‣ Draw.io and Figma for	• **Dependencies** ‣ Completion and approval of Software Requirements Specification ‣ UI design depends on confirmed user workflows and functional requirements because they show what the user needs to do and how the system should respond • **Constraints** ‣ Limited by team availability ‣ Design must comply with quality standards and security requirements	• **Deliverables** ‣ Web and Mobile System Architecture Document ‣ High-Level Design including system modules and interaction diagrams ‣ Detailed Design including Class diagrams, Sequence diagrams, Database schema and API documentation ‣ UI/UX prototypes for web and mobile • **Milestones** ‣ M3.1: System architecture approved. ‣ M3.2: High-level and detailed

	diagrams and UI design ‣ GitHub for version control ‣ Database tools (MySQL Workbench) ‣ Communication tools	‣ Time constraint: 2 weeks	design documents completed. ‣ M3.3: UI/UX prototypes finalized and validated with stakeholders. ‣ M3.4: Design reviewed and approved by all stakeholders.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Development	• Development Setup (T10) • Front-end Code (T11) • Back-end Code (T12) • Database Physical Design (T13)	• Configure Git repository and version control • Set up web development environment • Set up Flutter development environment with Android/iOS SDKs • Initialize backend framework and dependencies • Configure linters, formatters, and testing frameworks • Implement web frontend components for maintenance request management • Build Flutter screens for mobile app navigation • Develop responsive UI for web and mobile platforms • Create RESTful API endpoints for	4 weeks

		maintenance operations • Implement authentication and authorization logic • Build business logic for request processing and notifications • Create database tables, indices, and relationships • Implement data access layer with ORM • Set up database migrations and seeders	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**
	• **Skills** ‣ Frontend Developers (React for web, and Flutter for mobile) ‣ Backend Developer (we will use Node.js) ‣ Database Engineer (MySQL) ‣ DevOps Engineer for setup ‣ QA Engineer for unit and integration testing • **Hardware** ‣ Developer PCS and laptops	• **Dependencies** ‣ Depends on approved design phase deliverables auch as architecture, and database schema • **Constraints** ‣ We must follow the coding standards, security rules, and framework versions that were set in the earlier phases ‣ The work is limited by team availability and the performance of devices or emulators used for development	• **Deliverables** ‣ Git repository ‣ Working web and mobile front-end modules (React and Flutter) ‣ Implemented database schema ‣ Unit testing and integration tests • **Milestones** ‣ M4.1: Development environment configured and repository initialized. ‣ M4.2: Initial Functional prototype completed. ‣ M4.3: Unit and integration

	▸ Android and iOS testing devices or emulators ▸ Server environment • **Software** ▸ Visual Studio Code, Android Studio, and Xcode for Flutter ▸ Node.js, npm, Flutter SDK, Git ▸ Database tools (MySQL Workbench) ▸ Postman for API testing ▸ GitHub for repository hosting and version control ▸ GitHub as an integration tool for both our documentation (created using Typst) and the project code	▸ The time limit is 4 weeks; any delays may affect the testing and deployment phases ▸ Internet or cloud service interruptions may slow down integration and testing activities	testing completed with 80% code coverage. ▸ M4.4: Quality assurance (QA) verification passed and build approved for testing. ▸ M4.5: Production environment prepared and ready for deployment.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Testing	• Test Planning (T14) • Integration Testing (T15) • System & Acceptance Testing (T16) • Regression & Release Testing (T17)	• Develop comprehensive test plan for web and mobile platforms • Define test environments (development, staging, production) • Prepare test data sets for maintenance request scenarios	2 weeks

		• Test integration between web frontend and backend API • Test integration between Flutter mobile app and backend API • Verify database operations and data integrity • Conduct end-to-end system testing across all platforms • Perform user acceptance testing with stakeholders • Test cross-platform compatibility (iOS, Android, Web browsers) • Execute regression tests after bug fixes • Perform security and performance testing • Conduct final release testing and quality checks	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• **Skills**
	  ▸ QA engineers skilled in web and mobile testing, test automation, and bug tracking
	• **Hardware**
	  ▸ Test devices (Android and iOS)
	• **Software**
		• **Dependencies**
		  ▸ It is dependent on completion of the Development Phase and availability of a stable build.
		  ▸ Test data and environment setup depends on finalized database
		• **Constraints**
			• **Deliverables**
			  ▸ Completed integration and system test reports
			  ▸ Signed-off User Acceptance Test report from stakeholders to be ready for deployment
			• **Milestones**

	▸ Postman for API testing, and JMeter for performance testing	▸ Limited time for testing may constrain full regression coverage ▸ Must follow project's quality assurance and version control procedures	▸ M5.1: Test plan and test cases developed and approved. ▸ M5.2: System and integration tests executed successfully. ▸ M5.3: User Acceptance Testing (UAT) completed and approved. ▸ M5.4: Exit criteria met and testing phase formally closed.
Phase	**Task**	**Detailed task**	**EstimatedTime**
Deployment	• Release Planning (T18) • Environment Provisioning (T19) • Go-Live Execution (T20) • Transition & Support (T21) • Post-Deployment Review (T22)	• Create release plan with rollback strategy • Prepare deployment documentation and checklists • Schedule go-live date with stakeholders • Provision cloud servers and configure network • Set up production database with security settings • Configure CI/CD pipelines for automated deployment • Deploy web application to hosting platform • Publish Flutter mobile app to App	1 weeks

		Store and Google Play • Configure production environment variables and API keys • Conduct user training sessions for web and mobile platforms • Provide technical documentation and user guides • Establish helpdesk and support channels • Monitor system performance and user feedback • Review deployment metrics and KPIs • Document lessons learned and improvement areas	
	Resources Needed	**Dependencies and Constraints**	**Deliverables & Milestones**

	• **Skills**
	  ▸ DevOps engineer
	  ▸ System administrator
	  ▸ Database administrator
	  ▸ Technical support specialist.
	• **Software**
	  ▸ GitHub Actions
	  ▸ Google Cloud (hosting)
	  ▸ Firebase (mobile integration)
		• **Dependencies**
		  ▸ Dependent on successful completion of testing
		• **Constraints**
		  ▸ It must be scheduled
		  ▸ Risk of configuration errors
		  ▸ Security and data compliance must be checked and confirmed before
			• **Deliverables**
			  ▸ Successfully deployed web and mobile applications
			  ▸ User training and support materials completed
			  ▸ Documented post-deployment review and lessons learned
			• **Milestones**
			  ▸ M6.1: Final stakeholder

	▸ Google Analytics as a monitoring tool • **Hardware** ▸ Cloud servers ▸ Mobile and desktop devices	the system goes live	approval for production release obtained. ▸ M6.2: Web and mobile applications deployed to production environment. ▸ M6.3: User training sessions completed. ▸ M6.4: Post-deployment review completed. ▸ M6.5: Project officially closed.

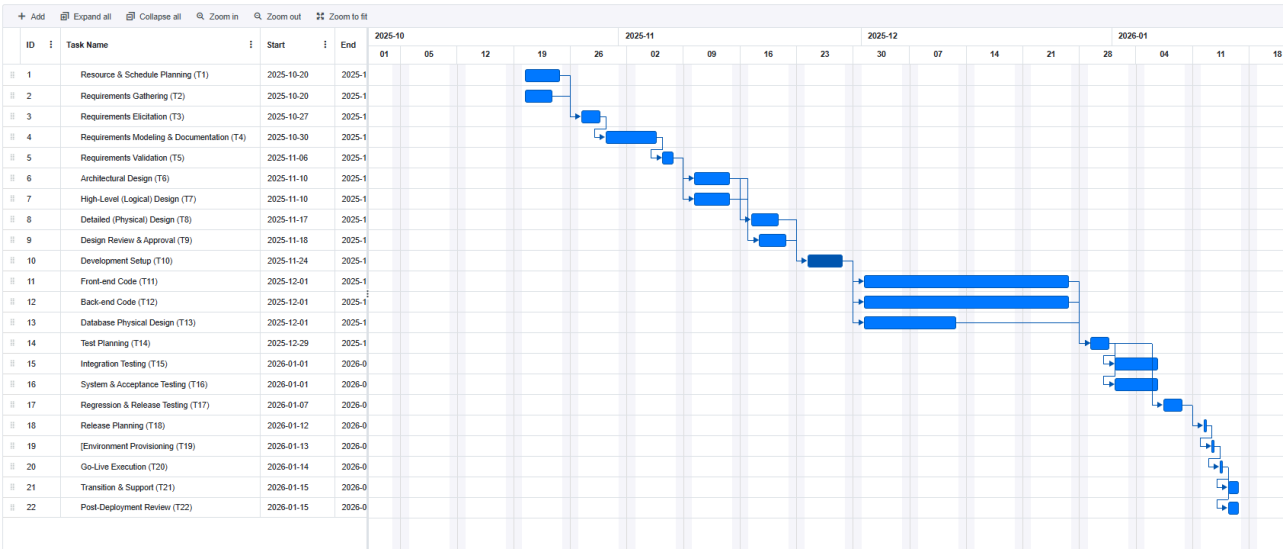
Table 11: Project Tasks

2.6. Project Schedule

2.6.1. Activity Network

Figure 3: Activity Network Diagram

2.6.2. Gantt Chart



ID	Task Name	Start	End
1	Resource & Schedule Planning (T1)	2025-10-20	2025-1
2	Requirements Gathering (T2)	2025-10-20	2025-1
3	Requirements Elicitation (T3)	2025-10-27	2025-1
4	Requirements Modeling & Documentation (T4)	2025-10-30	2025-1
5	Requirements Validation (T5)	2025-11-06	2025-1
6	Architectural Design (T6)	2025-11-10	2025-1
7	High-Level (Logical) Design (T7)	2025-11-10	2025-1
8	Detailed (Physical) Design (T8)	2025-11-17	2025-1
9	Design Review & Approval (T9)	2025-11-18	2025-1
10	Development Setup (T10)	2025-11-24	2025-1
11	Front-end Code (T11)	2025-12-01	2025-1
12	Back-end Code (T12)	2025-12-01	2025-1
13	Database Physical Design (T13)	2025-12-01	2025-1
14	Test Planning (T14)	2025-12-29	2025-1
15	Integration Testing (T15)	2026-01-01	2026-0
16	System & Acceptance Testing (T16)	2026-01-01	2026-0
17	Regression & Release Testing (T17)	2026-01-07	2026-0
18	Release Planning (T18)	2026-01-12	2026-0
19	[Environment Provisioning (T19)	2026-01-13	2026-0
20	Go-Live Execution (T20)	2026-01-14	2026-0
21	Transition & Support (T21)	2026-01-15	2026-0
22	Post-Deployment Review (T22)	2026-01-15	2026-0

Figure 4: Gantt Chart

2.7. Assigning Team Members to Tasks

Task ID	Assigned to
T1	Project Manager (Anas), Technical Clerk (Orjoan)
T2	Project Manager (Anas)
T3	System Analyst (Anas), Requirement Analyst (Anas), Technical Clerk (Orjoan)
T4	System Analyst (Anas), Requirement Analyst (Anas)
T5	System Analyst (Anas), Technical Clerk (Orjoan)
T6	Software Engineer (Haneen & Shaima), Hardware Engineer (Mosa), Web Designer (Haneen & Shaima)
T7	Software Engineer (Haneen & Shaima), Database Engineer (Mosa)
T8	Software Engineer (Haneen & Shaima), Database Engineer (Mosa), Web Designer (Haneen & Shaima)
T9	System Analyst (Anas), Software Engineer (Haneen & Shaima), InfoSec Engineer (Ext.)
T10	Software Engineer (Haneen & Shaima)
T11	Software Engineer (Haneen & Shaima)
T12	Software Engineer (Haneen & Shaima)
T13	Database Engineer (Mosa)
T14	Software Engineer (Haneen & Shaima), Software QA Engineer (External)
T15	Software Engineer (Haneen & Shaima)
T16	Software QA Engineer (External), System Analyst (Anas)
T17	Software QA Engineer (External), Software Engineer (Haneen & Shaima)
T18	Software QA Engineer (External), System Analyst (Anas)
T19	Software QA Engineer (External)
T20	System Administrator (Mosa), Project Manager (Anas), Software Engineer (Haneen & Shaima)
T21	System Administrator (Mosa), Database Engineer (Mosa)
T22	System Administrator (Mosa), Software Engineer (Haneen & Shaima), InfoSec Engineer (Ext.)

Table 12: Task-to-Team Member Assignment

2.8. Monitoring and Controlling Mechanisms

Earned value management

Phase	Estimated cost	Cumulative estimate	Estimated duration	Stage completed	Actual cost of Phase to date	Actual cost of project to date
Planning	4,000 JD	4,000 JD	2 weeks	80%	1,000 JD	1,000 JD
Analysis	6,500 JD	10,500 JD	2 weeks	0%	Not yet begun	Not yet begun
Design	8,000 JD	18,500 JD	2 weeks	0%	Not yet begun	Not yet begun
Implementation	16,000 JD	34,500 JD	4 weeks	0%	Not yet begun	Not yet begun
Testing	8,000 JD	42,500 JD	2 weeks	0%	Not yet begun	Not yet begun
Deployment	5,000 JD	47,500 JD	1 week	0%	Not yet begun	Not yet begun

Table 13: Earned Value Management Progress Tracking

- $P = 80\%$
- **Planned Value (PV) = 4,000 JD**
- **Actual Cost (AC) = 1,000 JD**
- **Earned Value (EV) = 3,200 JD**
- **Cost Variance (CV) = 2,200 JD**
  - We have saved 2,200 JD on the planning phase
- **Schedule Variance (SV) = (-800) JD**
  - This indicates delayed progress in the planning phase.
- **Cost Performance Index (CPI)= (3.2)**
  - We are under budget. The team is spending less than planned to accomplish the work.
- **Schedule Performance Index (SPI) = (0.8)**
  - We are behind schedule. The progress is slower than expected.
- **Estimate to Complete (ETC) = (13,593.75) JD**
- **Estimate at Completion (EAC) = 14843.75 JD**

- Based on information we gained from EVM analysis, we have to expedite our schedule.

Activity	Estimated duration (days)	Crash time (days)	Cost/day (JD)
T1	5	1	200
T2	4	2	200
T3	3	2	300
T4	5	2	400
T5	2	1.5	300
T6	5	1	300
T7	5	2	300
T8	4	3	300
T9	4	2	300
T10	5	2	250
T11	20	9	250
T12	20	10	400
T13	10	18	400
T14	3	2	250
T15	4	2	250
T16	4	3	300
T17	3	2	300
T18	1	0.75	300
T19	1	0.75	250
T20	1	0.75	250
T21	2	1.25	250
T22	2	1.5	300

Table 14: Time and Cost Options

Activity chosen	Expediting time (days)	Cumulative time saved	Cost (JD)	Cumulative cost (JD)
T1	4	4	800	800
T10	3	7	750	1550
T14	1	8	250	1800
T19	0.25	8.25	62.5	1862.5
T5	0.5	8.75	150	2012.5
T9	2	10.75	600	2612.5
T8	1	11.75	300	2912.5
T4	3	14.75	1200	4112.5

Table 15: Implemented Schedule Acceleration Plan (Based on Selected Activities)

2.9. Risk Analysis

2.9.1. Effects and Causes

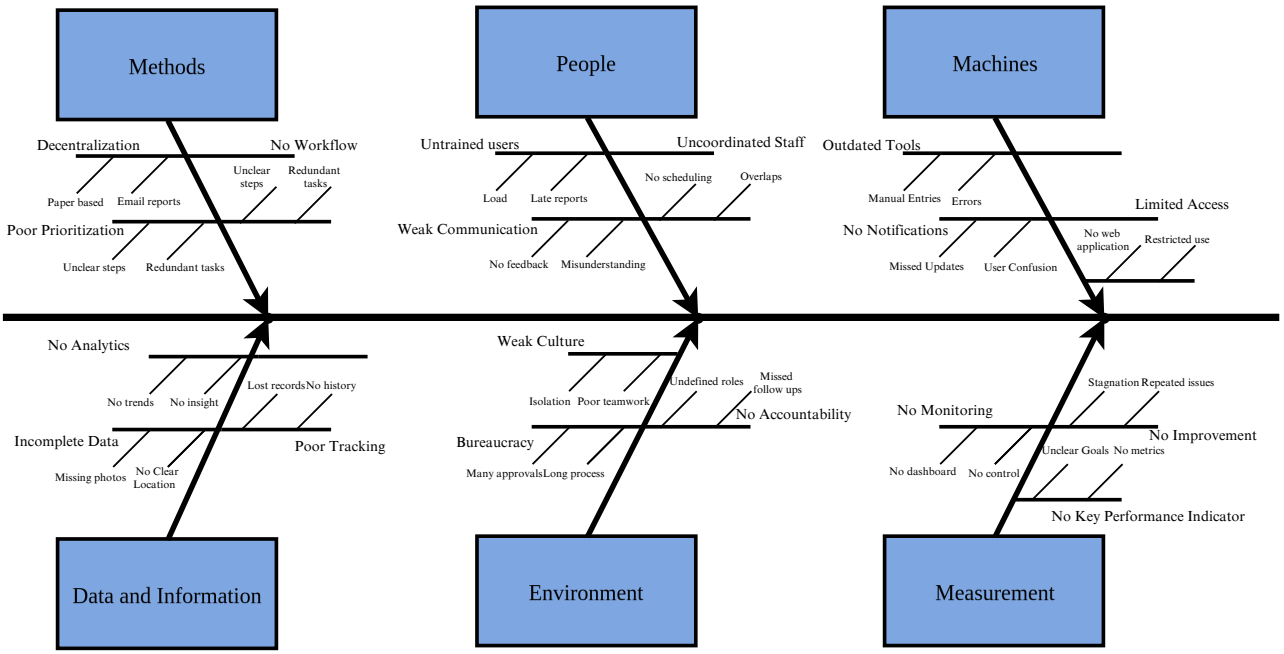


Figure 5: Fishbone Diagaram

2.10. Communication Plan

Our team will maintain communication through online meetings using Google Meet, and instant messaging using a WhatsApp group to assign tasks and ensure that everyone is updated on progress and deadlines. **Communication Methods:**

- In-Class Meetings:
  - Held every Sunday to discuss progress, issues, and next steps.
- Out-of-Class Meetings:
  - Through Google Meet whenever there is a need for rapid discussion.
- Messaging:
  - A WhatsApp group for updates and daily coordination.
- Version Control:
  - GitHub is used for tracking code updates and version history.
  - We also use GitHub to store and organize our documentation created using Typst, ensuring all of us can access the latest versions easily.