Dormitory Cleaning Materials Management System

SRS STUDY

Members :

Mutoni Uwingeneye Denyse

Izere Uwonkunda Marie Claire Kerie

Ntezirizaza Ernest

Cyebukayire Peace

Doc Version: 1.0

Table of Contents:

Contents

List of Abbreviations:

DCMMS: Dormitory cleaning material management system

HIPO: Hierarchical Input Process Output

DFD: Data Flow Diagram

List of Tables:

List of Figures

1. INTRODUCTION

A dormitory cleaning material management system is a system that is going to ease how patrons/matrons/students manage the dormitory cleaning material.

1.2. Background of the Project

DCMMS is a web-based project that helps in the management of the RCA dormitory Cleaning Materials Management System. It provides interfaces for various stakeholders like matrons and students.

Materials can be added to the system by their associated parameters (name, quantity, type, size, date, status, etc.). The matrons and patrons can track the status of the cleaning materials by knowing their status thus if they are damaged or new and their report in general.

1.3 Objectives of the Project

The objectives of this project are to:

- ★ Develop a central database for each cleaning material
- ★ Provide daily, weekly, monthly and annual reports of cleaning materials to matrons and patrons.
- ★ Provide a data entry for cleaning materials.
- ★ Provide a way to change the status of the material.

1.4 The need for a project

Managing Cleaning materials in RCA is a cumbersome task when it comes to tracking the total number of working materials in RCA. This project will ease the way of managing resources in

RCA by providing a statistical report to the end-users. This report will be informative enough to ease the management of cleaning materials.

1.5 Overview of existing technologies

Main technologies associated with DCMMS

- ★ Web programming technologies (React js(Next js), Nodejs (Nest js),
- ★ Design tools(Figma)).
- ★ Database technologies (PostgreSQL)
- ★ Diagram and design tools(Lucid Chart, Figma)

1.6 Deliverables

S. NO	Deliverables
1	Feasibility study (FS).
2	Non-functional HTML Prototype.
3	Working and Tested Software with source code
4	User and Administrator Manuals for the system
5	Setup and Release notes for each new release.
6	Test Cases and Reports.
7	All database scripts.
8	Handover training.
9	Any other relevant documents, supporting software

Table 1: Deliverables

1.7 Risks and Assumptions

The risks identified in the ToRs are lack of availability and collaborations between all players due to other responsibilities; and lack of knowledge of beneficiaries and end-users on what really should be developed to produce the needed system.

1.7.1 Assumptions underlying the project

- Access to all excel templates which are recently in use.
- Frank collaboration between all stakeholders, partners, and beneficiaries of the system.

1.7.2 Risks

- Lack of basic knowledge (IT skills) of end-users.
- Absence of collaboration with stakeholders and beneficiaries.

1.7.3 Approach towards identified risks

- CDMM provides training and knowledge to end-users.
- Earlier involvement of stakeholders and beneficiaries in each phase of this system development and implementation.

2. Requirements Specifications

2.1 Functional requirements

- The system will allow users to login
- The system will allow users to logout
- The system will allow the admin to create a cleaning tool
- The system will allow the user to delete the tool
- The system will allow the user to update the status of the tool(old, new, damaged)
- The system will allow users to request new tools needed
- The system will allow users to report old tools and does need to be fixed
- The system will allow users to view all new and old tools available.
- The system will allow users to search a tool
- The system will allow users to update his/her profile
- The system will allow the user to deactivate his/her account
- The system will allow admin to see new tools requests made
- The system will allow the admin to delete requests
- The system will allow users to update requests
- The system will allow Admin users to mark requests as seen
- The system will allow users to mark requests as completed

2.2 Non-functional requirements

- The system will operate 24/7 days

- The system will grant access to accounts when users enter the correct username and password.

- The system will process 50 requests at the same time without affecting response time.

- User account names will be associated with a username which will be chosen by the user upon first use.

- If the user enters an incorrect password or email 3 times, they will be locked out for 2 min

- The interfaces of the system will be designed in a user-friendly manner such that the user will need less training to perform operations within the system. The interfaces will be easily navigable, clearly labeled, and a help menu will be provided with instructions for performing basic tasks.
- The system should respond to a user's request for information in less than 5 seconds in any case.
- The system should timeout if there is not an activity for 10 minuteS

- System passwords will be case-sensitive, at least 6 characters, hashed, and stored in the database.

- The testers and project managers will be provided access to the system after they are registered into the database
- All history information will be stored in the database.
- The system will need 100MB

2.3 System Development Methodology

Our methodology to develop the DCMMS will be an Agile Development Methodology. We have called Scrum, which employs iterative development cycles, called Sprints. Each two-week Sprint will encompass all phases of the software development life cycle—plan, design, develop, test, and deploy—for a selected body of work. This approach will allow us to deliver functioning software in weeks, rather than months.

2.3.1 Why agile?

- ➢ Truly useful software as the result of continuous collaboration with matron and patron's methodology.
- ➢ Low or no training costs because your users are involved every step of the way from user story development, through design and demonstration, to user acceptance testing.
- ➢ Fewer defects because testers can concentrate on smaller blocks of functionality.
- ➢ No lengthy defect repair cycles can compromise deployment because defects are found and repaired as the system is developed.
- ➢ Functioning software every two weeks means more transparency for the project board and less risk to RCA.
- ➢ Moreover, most importantly…the ability to respond to your changing business needs through continuous backlog prioritization.

2.3.2 How agile methodology will be applied.

Plan	First, we'll collaborate with you to develop "user stories" that define the specific needs of the system being developed, known as the product backlog. Then at the beginning of each Sprint, you'll review and prioritize the backlog to ensure the highest priority items are being developed first. The development team then plans the Sprint by selecting two weeks' worth of work from the top of the backlog.
Design	Collaborate. Collaborate. Collaborate. Whoever said two heads are better than one couldn't have been right. We'll hold joint application design (JAD) sessions to clarify your workflow, pain points, and desires. This allows us to provide you with a truly intuitive and efficient design. Our team then takes all that information and designs a solution that will make you and your users exhale with relief.
Develop	Self-organization at its finest. With Scrum, our team works together to figure out who will do which development work to achieve the highest success. Our teams know where their individual strengths and weaknesses lie, so allowing them to control who does what in the development phase helps the team feel more satisfied. And we've noticed satisfied people produce higher quality work! Then, as the Sprint draws to a close, the team will showcase their work in a product demonstration, allowing you and your users to provide feedback before deployment.

Test	"Quality control" is our middle name, which is why we use state-of-the-art software to help automate certain kinds of testing and to support manual testing efforts. We'll complete unit, functional, system, integration, and regression testing to ensure all new functionality works and all existing functionality still works after new functions are integrated. Once we've tested it all, we'll work with your users in an acceptance test phase to assure them that the system will work just the way they need it to.
Deploy	After your team has completed acceptance testing, the software is deployed to production. So in just two short weeks, we'll be able to give you real functioning software. This process is then repeated to continuously develop the system until it fully satisfies your needs. And since you and your users will be participating every step of the way, they won't need training! (Caution: this phase has been known to lead to extreme excitement and overwhelming joy.)
Maintain	After the deployment of the system, the team will be responsible for the maintenance of the system like system upgrades, repairs, and fixes of the system if it breaks.

Table 2: How agile methodology will be applied

2.3.3 In-Depth look at agile implementation

During the course of this project development we will use Scrum as one of the agile implementations. Normally scrum is used as the simplest and most cost accurate software development methodology based on an iterative and incremental process where both parties : The client and the developer are constantly engaged in the development process. Scrum is an adaptable, fast, flexible, and effective agile methodology that is designed to deliver value to the customer throughout the development of the project. The primary objective of Scrum is to satisfy the customer's need through an environment of transparency in communication, collective responsibility, and continuous progress. The development starts from a general idea of what needs to be built, elaborating a list of characteristics ordered by priority (product backlog) that the owner of the product wants to obtain. Later project backlogs are translated into user stories for developers to write code. Source code takes the MVP that is presented to the customer in a small reasonable time interval called Sprint and this helps both parties to easily monitor the

system development progress, discovering challenges before they grow big. Using this methodology allows the engagement of the customer in the process, preventing the developer from going out of users' needs remaining in the scope, and quickly achieving the customer's satisfaction.

2.3.4 Scrum team composition

The project will be composed of a Project Board: project board composed of a project manager whose role will be to review project progress, approve project change requests, and approve project deliverables. The project will comprise several scrum teams, each focusing on a different workstream, including the DCMMS dashboard.

2.3.4.1 Each scrum team will be comprised of

- ★ **Scrum Master** - who will also be the engineering lead
- ★ **Scrum team:** THA developers, UI designers, Business Analysts, etc.

As specified by Scrum, important key players (main actors) are required for smooth project development: The Product Owner, The Scrum Master, Scrum Team Members. Viewing the nature of this project, considering the time constraints and the size of the company our scrum team composition is as follows:

- ➢ **The product Owner:** We will need the product owner from our team Side to act as a key contact person and Project manager.
    - He/she must have technical knowledge.
- ➢ **One Scrum Master:** He is the project manager from the side of developers. His main tasks include regular meetings with the product owner to plan and evaluate the development process, coordinate the scrum teams, and ensure the safety and health of all team members by availing required resources for the project's success. He will plan and draft the high-level descriptions of milestones and deliverables and determine the size and duration of sprint meetings inside scrum teams.
- ➢ Three Scrum Team Members/development team: This is a team of real code writers, their day-to-day activities are translating users' stories into use cases and implementing use cases by writing source codes and testing them. In this team, we have 1 front-end

developer, 1 back-end developer, 1 UI/UX developer who will in turn act as our Quality assurance personnel.

2.3.5 Scrum implementation

The implementation of scrum is simple for each development phase listed in the life cycle above, we will do the planning, code, test, track, report, and then release. To achieve these, we conduct two scrum meetings in a week and two sprint review meetings in a week. Scrum meetings will happen in the entire scrum team whereas the sprint review meeting will be like small stand-up meetings held in our small development teams like the front end, back end, and UX.

2.4 Scope of the project

- ➔ Main actors of this system
  - ◆ Matrons
  - ◆ Patrons
  - ◆ Students
  - ◆ Accountant
- ➔ Main use cases associated
  - ◆ Matrons, Patrons, and accountants can:
    - View all the materials in the dormitory
    - Assign a cleaning material to a specific dormitory
    - Update the status of a cleaning material
    - Delete a cleaning material
    - Get a daily summary of dormitory cleaning materials

2.7 Technology stack to be used.

At this stage, we are not yet precise on the technology to be used as normally this is something that we determine SRS after some basic data collections, analysis, and designs. But tentatively a technology list is elaborated until when done with data correction, analysis, and design of processes, database, and architecture.

2.7.1 Programming languages

Frontend: Javascript (Using react framework)
Backend: Node js (Using Spring Boot framework.

2.7.2 Database management system

We are to use relational DBMS: PostgreSQL: Powerful, open source object-relational database management system with over 30 years of active development is the most suited for this project, it is reliable, robust, and high performing in a relational environment.

2.7.3 Containers

- ➢ To host node js we will use vercel.

2.7.4 Collaboration tools

For outside communication and inter-team collaboration on this project we will use the following tools:

- ➢ For Inter team collaboration: Slack, Whatsapp
- ➢ For Official communication: Emails.

2.7.5 Revision Control

For easy source codes and project materials sharing, we will use git-based revision controls, Github is chosen for its simplicity and openness.

3. System requirement analysis

3.1 System features

3.1.1 **Strategies**

A plan of action designed to achieve a long-term or overall aim. A strategy groups together different related outcomes that will be called indicators in our system.

3.1.2 **Outcome**

An outcome is the actual indicator that the system should read the user inputs and or computer output values. If an indicator is computable then its inputs will be predefined by a formula whose variables are fed by the data entry clerks at different levels for computation.

3.1.3. **Input Validation**

Our system will take care of data validity in terms of validating inputs. We will ensure that all inputs are validated before being saved in the database so as to achieve data accuracy because some malicious users may enter null values.

Our system is composed of the following features:

- Authentication module
  - ❖ Creating an account(Registration).
  - ❖ Login.
  - ❖ Reset password.
- User management module
  - ❖ Matron or patron may send a request for new material to the accountant.
  - ❖ Accountant may approve or reject a request from a matron or patron.

- Dormitory management module
  - ❖ Add a new dormitory.
  - ❖ Update an existing dormitory.
  - ❖ Delete a dormitory.
- Materials management module
  - ❖ Add a new material.
  - ❖ Update an existing material.
  - ❖ Delete a material.

3.2 Data Flow Diagrams

3.2.1 Accountant DFD

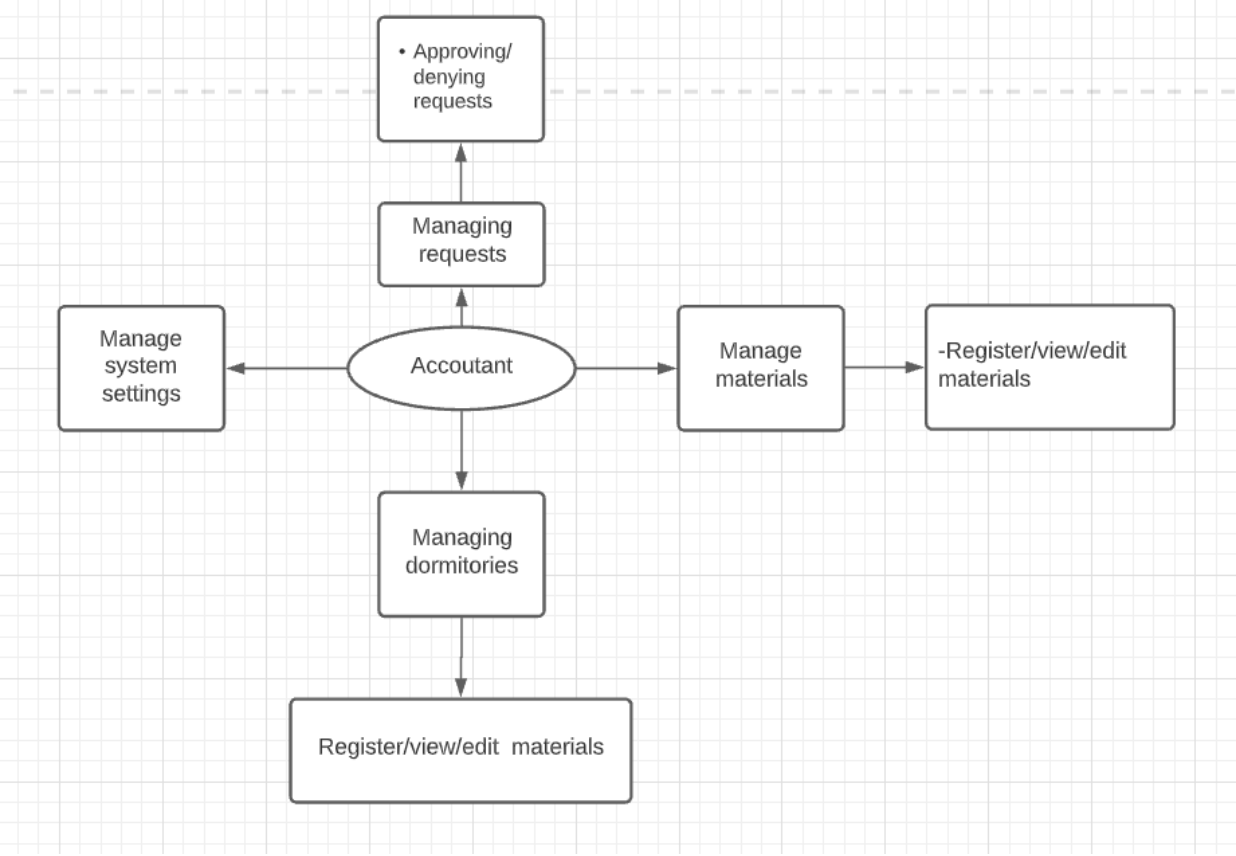

Figure 1: Accountant DFD

3.2.2 Matron’s DFD

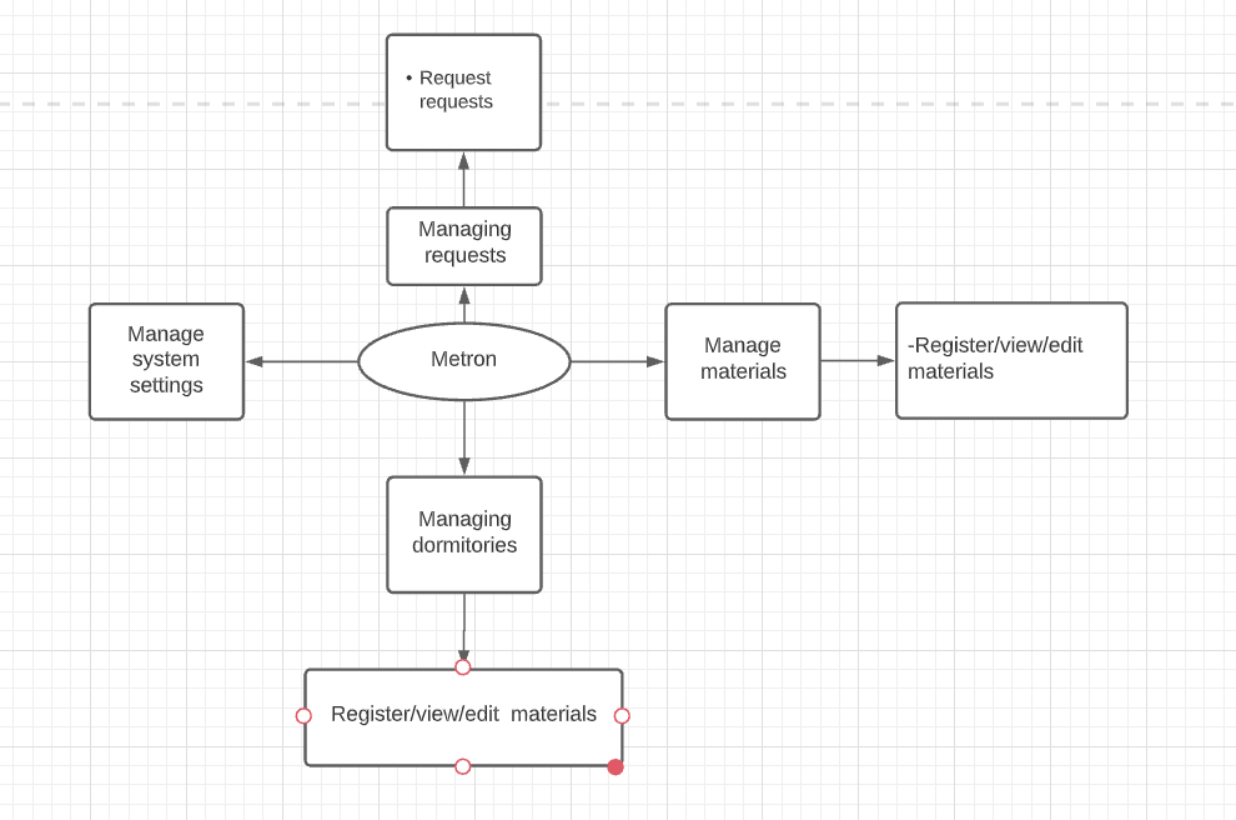

Figure 2: Matron's DFD

3.2.3 Patron’s DFD

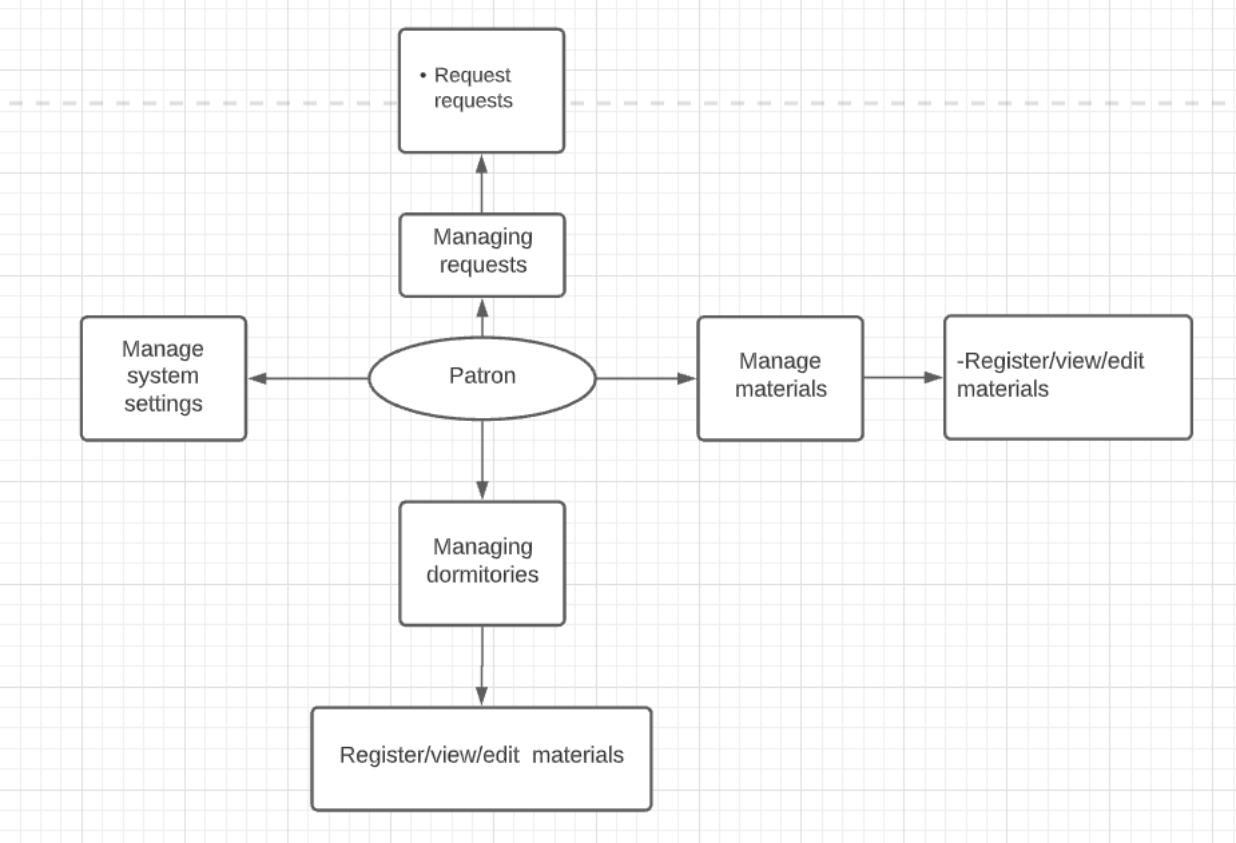

Figure 3: Patron's DFD

3.3 Hierarchical Input Process output (HIPO)

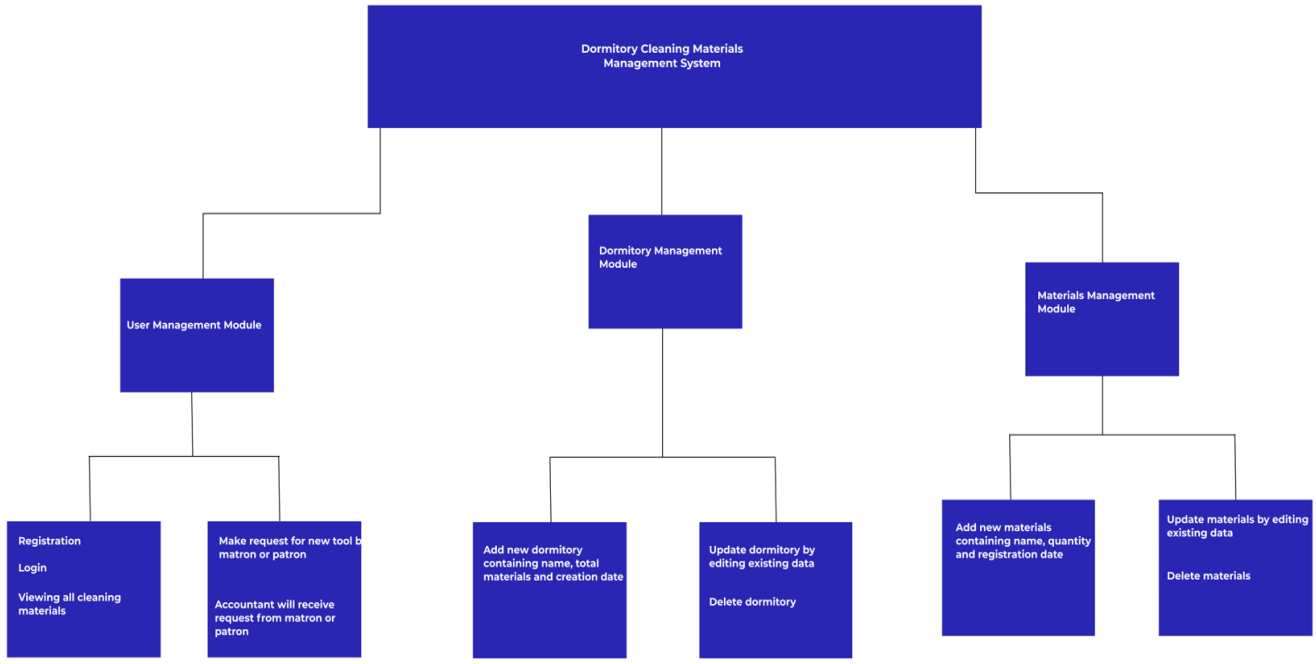

Figure 4: HIPO

3.4 Process flow diagrams

3.4.1 Login process

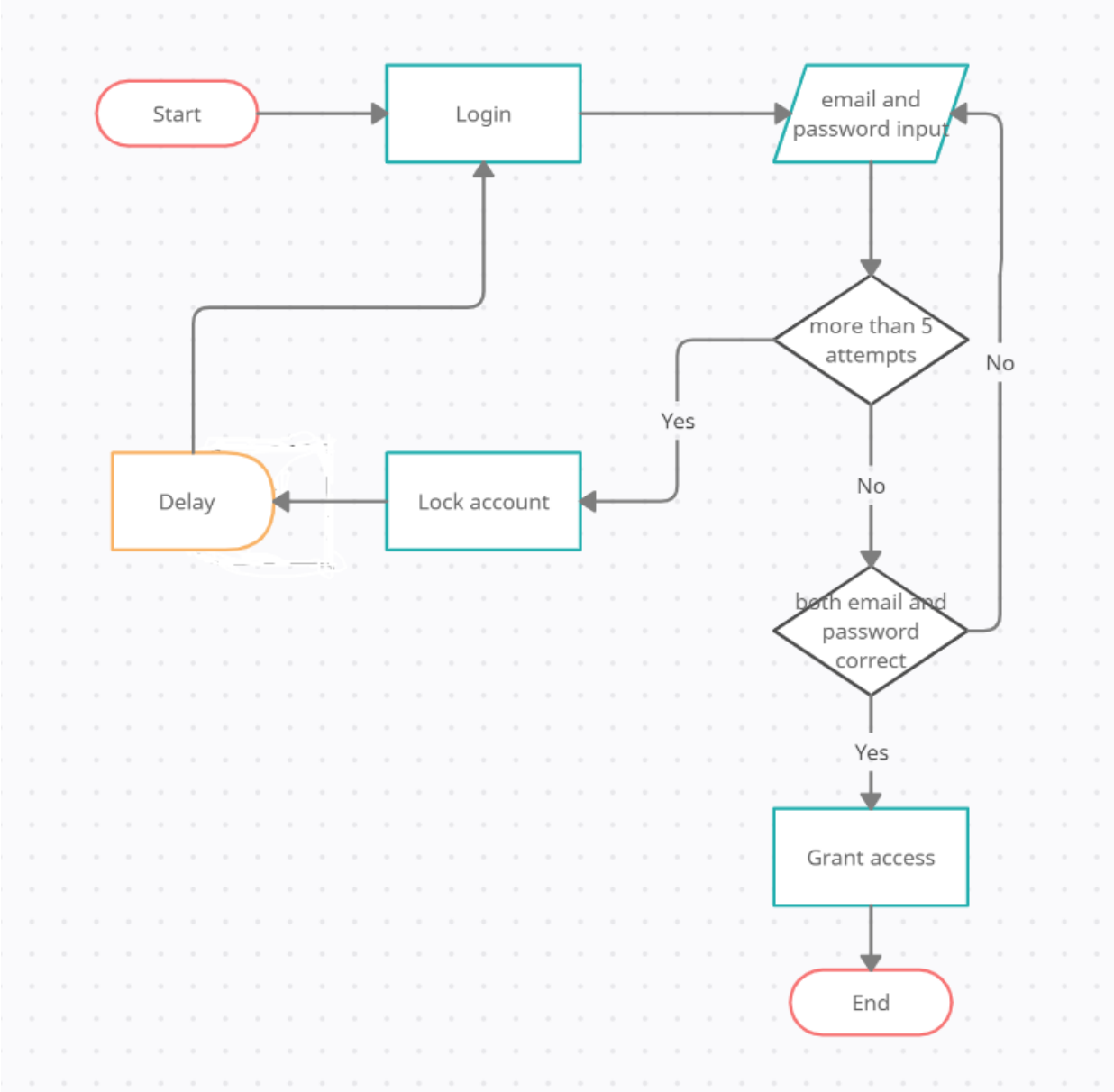

Figure 5: Login process

3.4.2 Register process

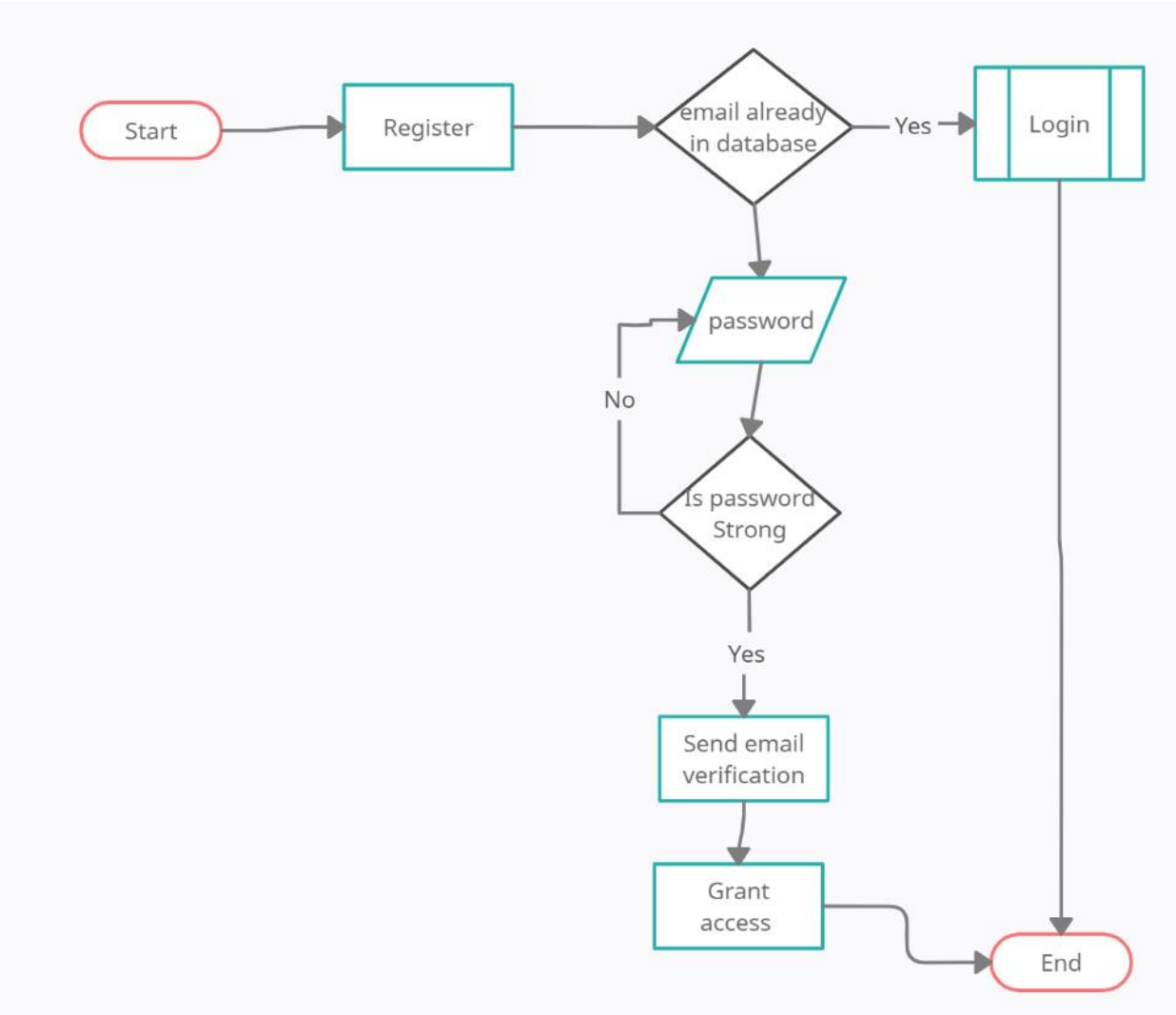

Figure 6: Register process

3.4.3 Create dormitory process

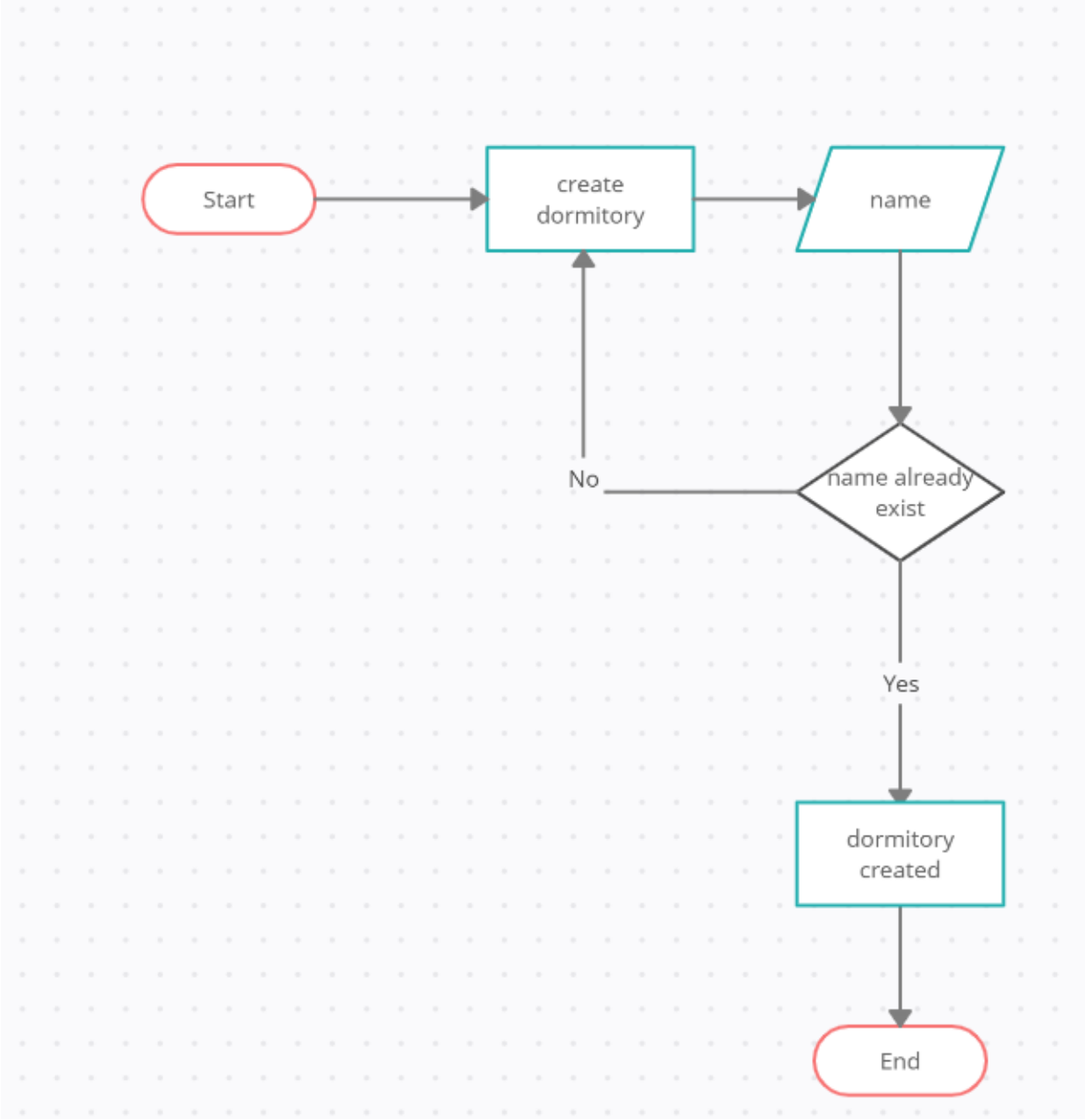

Figure 7: Create dormitory process

3.4.4 Create Category of the tools process

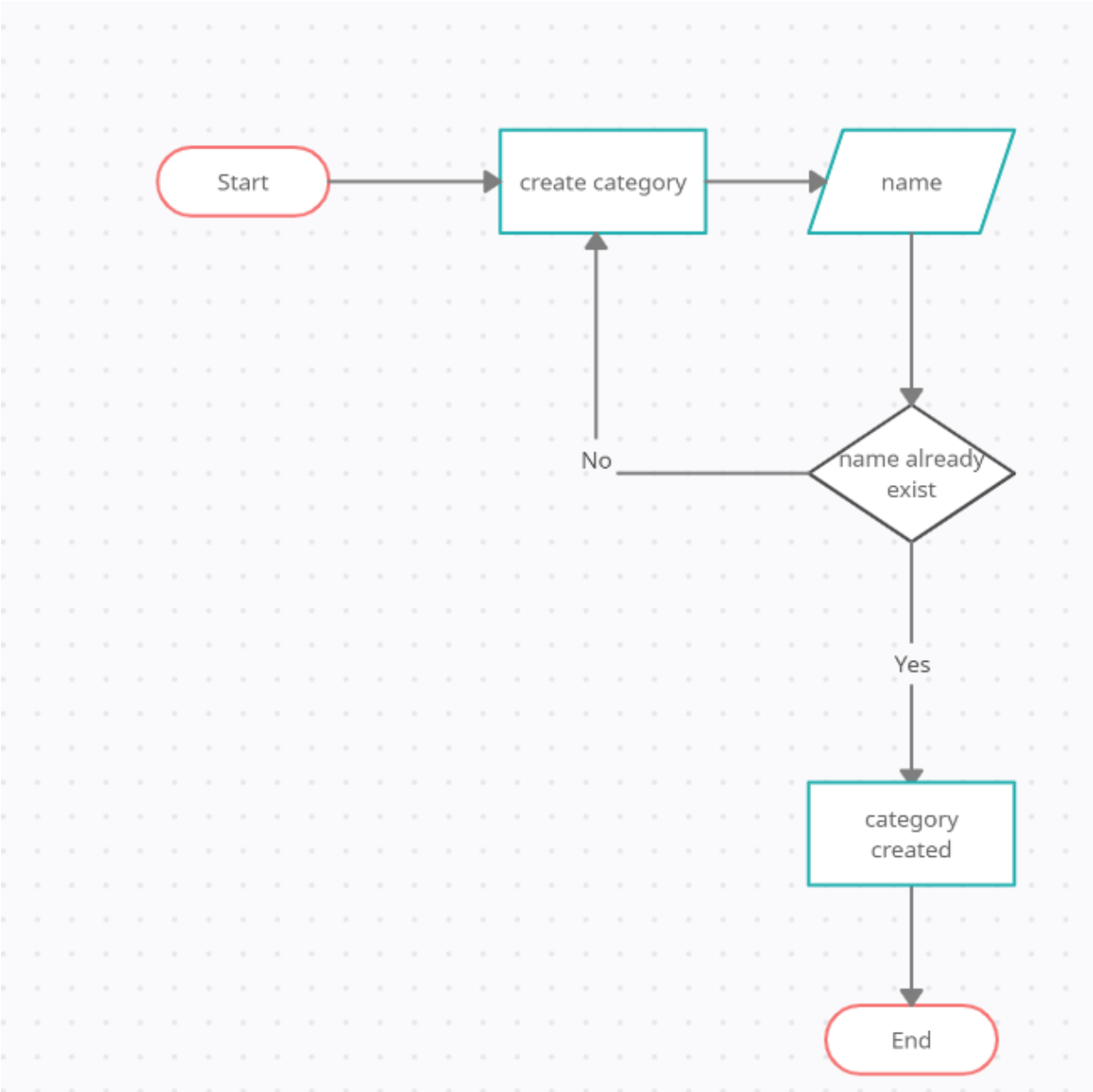

Figure 8: Create category of the tools process

3.4.5 Request new tool process

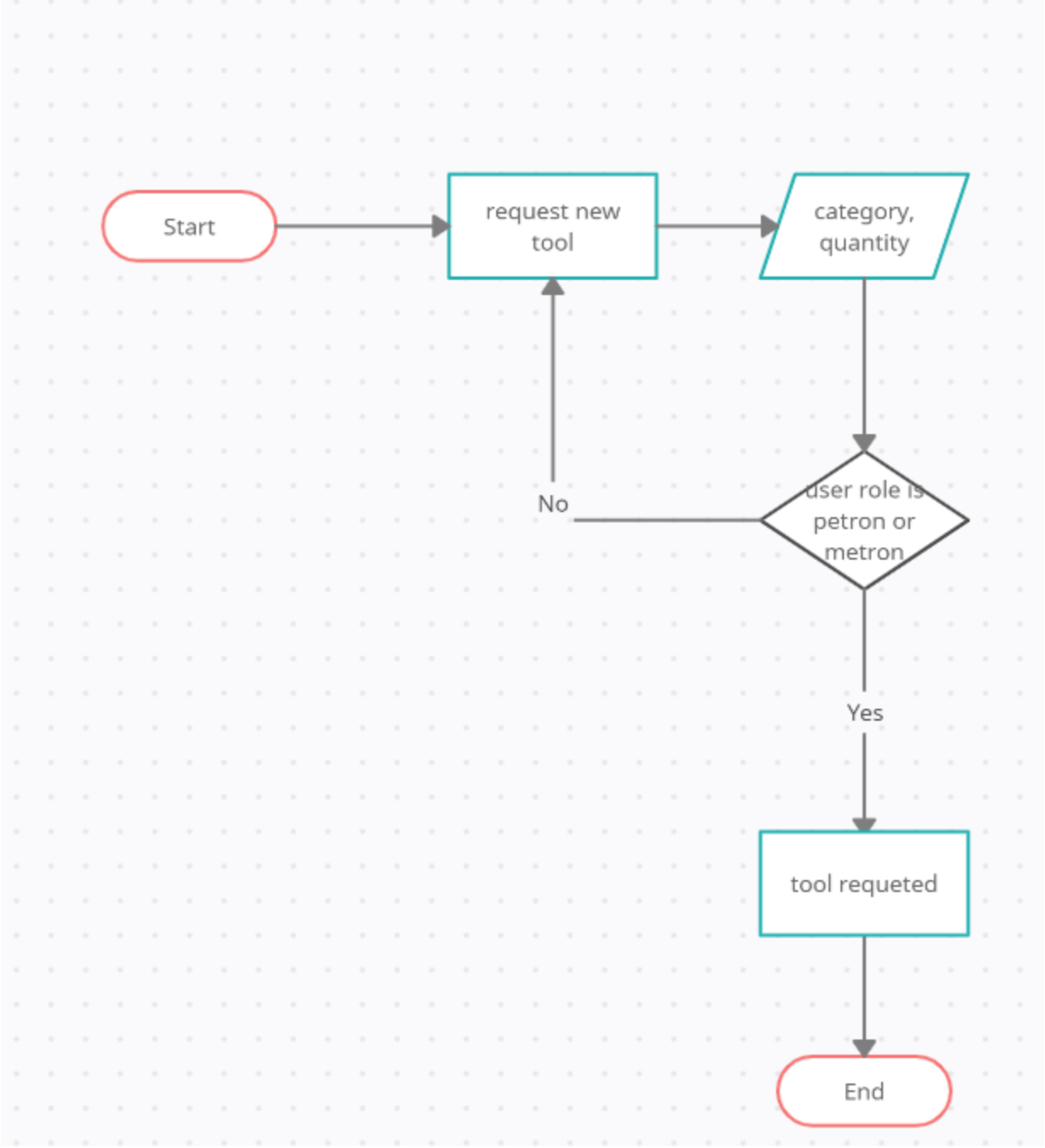

Figure 9: Request new tool process

3.5 Use cases

4. System security

5. Support and maintenance

Our team will progressively ensure the different long-term types of maintenance to the platform during the agreed one year period of free support and maintenance:

5.1. Corrective Maintenance:

The development team is required to develop and deploy the solution to the identified problems; all bugs and errors identified during the first 6 months after the deployment to production are to be fixed according to the design and user need. Meaning corrective software maintenance is guaranteed for that period. Often these fixes permanently solve the problem, but not always. Some fixes act as a temporary solution while working on a more permanent solution and this is not a change request hence not charged.

5.2. Perfective Maintenance:

Programmers may also engage in perfective software maintenance to improve the software's menu layouts and command interfaces. There could also be a need to conduct perfective maintenance on software because of outside influences, such as new government regulations that affect how a business operates. This maintenance request can not be accepted before 6 months and beyond 12 months after deployment. Hence not chargeable between 6 and 12 months.

5.3. Adaptive Maintenance:

Technology constantly evolves in both hardware and software. Adaptive software maintenance can be addressed for these changes. For example, Software can also interact with other software programs on a computer or network, meaning changes in one program can require changes in

other programs. A user can eventually introduce new software to the computer or network, which can also affect how other software already present operates.

5.4. Preventive Maintenance:

This can involve an engagement in preventive software maintenance by trying to prevent problems with software programs before they occur. More often for shared systems in production there is a need to constantly backup data for preventing data loss. For this MIS a replication server is required and our job is to configure a periodic database backup to make sure that no data will be lost in case of disasters.

Also A monitoring tool may be installed to provide system health statistics, on daily or hourly basis and monitor the performance and hence easy and timely decision making before the system failure