ReqTrackManager

Project Specifications

Table of contents

Introduction

ReqTrackManager is aiming to fill the market gap for teams (predominatly product development teams) that need a formal engineering requirements management system, but cannot afford propietory systems, and find other requirements software “not quite right” (generally because it’s designed for continuous improvement/agile software teams).

ReqTrackManager aims to have the following features (details in below specifications):

- Primarily OpenSource
- Server hosted

Background

Hardware teams that I have been apart of, have lacked proper engineering requirement systems and formal change request processes.

The main reason for this is cost. IBM DOORS is arguably the industry standard for this type of software, but it comes with a heafty cost. Small to medium size teams simply can’t afford it, and so either try alternatives (such as Jira Requirements), manual process or go without.

However due to the complexity of tracking requirements, most alternatives simply don’t have all the features required by hardware teams (they are developed for agile software). Manual processes seemingly end up in the too hard basket and as such teams generally revert back to a static document which doesn’t change throughout the project and only gets revised at the end (generally changing the requirements to match what was made).

As such, I wanted to develop my own system that I could work on in my spare time, using industry best practices and looking like a modern software package (IBM DOORS looks like it’s from the 90s).

Project Structure

Backend (Server) Specifications

Task	Priority	Status
Task 1	High	In Progress
Task 2	Medium	Not Started
Task 3	Low	Completed
Task 4	High	In Progress
Task 5	Low	Not Started
Task 6	Medium	Completed

Frontend (User Interface) Specifications

Appendix A - Terminology

Standard Fields

- **Requirement Name:** The requirement name serves as a clear and concise identifier for each specific requirement. It allows stakeholders to easily reference and discuss individual requirements without confusion. A well-defined name facilitates communication and understanding throughout the development process.
- **Reasoning:** The reasoning behind a requirement provides context and justification for its inclusion. It helps stakeholders understand why a particular requirement is necessary for the product. This understanding is crucial for making informed decisions, prioritizing requirements, and resolving conflicts or discrepancies during the development process.
- **Requirement Level:** The requirement level indicates the importance or priority of a requirement in relation to others. It helps stakeholders understand the criticality of each requirement and guides decision-making regarding resource allocation, scheduling, and trade-offs. Clear requirement levels ensure that development efforts are focused on fulfilling the most essential needs of the project.
- **Timeline/Horizon:** The timeline or horizon specifies when the requirement needs to be implemented or achieved. It provides a clear understanding of the project's timeline and helps stakeholders coordinate activities, plan resources, and manage expectations. Additionally, it allows for the prioritization of requirements based on their deadlines or dependencies, ensuring that critical milestones are met on time.
- **Success Criteria:** Success criteria define the conditions that must be met for a requirement to be considered successfully implemented or fulfilled. They provide objective benchmarks for evaluating the quality and completeness of deliverables, guiding testing, validation, and acceptance processes. Clear and measurable success criteria ensure alignment between stakeholder expectations and project outcomes, facilitating transparency and accountability throughout the development lifecycle.
- **History of Requirement:** This section documents the changes, updates, or modifications made to the requirement over time. It includes a timeline of change requests, both approved and disapproved, providing a comprehensive view of the requirement's evolution. Tracking the history of requirements helps stakeholders understand the rationale behind changes, assess the impact on project timelines and resources, and maintain transparency and accountability in requirement management. This section is auto-generated based on change requests and updates.

Requirement Levels

We have three standard requirement levels

- **Requirement:** A requirement denotes a feature or functionality that is indispensable for the core functionality of the product. It must be implemented to ensure the product meets its fundamental objectives and serves its intended purpose. Failure to include a requirement may result in the product not functioning as intended or failing to meet essential user needs.
    - Requirements generally use the terms “must” have or “shall” have.
- **Recommended:** Recommended features are enhancements or additions that are deemed beneficial for the product's overall usability, performance, or user experience. While not mandatory for the initial release, recommended features provide valuable insights into the product's future direction and evolution. Integrating recommended features is encouraged to enhance the product's value proposition and competitiveness, but their absence does not compromise the core functionality.
    - Recommendations generally use the term “should” have.
- **Optional:** Optional features represent functionalities that are desirable but not critical for the product's core objectives. These features serve as “nice-to-haves” that can enhance user satisfaction or provide additional value, but their inclusion is not imperative for the product's basic functionality. Optional features may be implemented

based on resource availability, project timeline, or specific user preferences, offering flexibility in tailoring the product to varying needs and preferences.

- Optionals generally use the term “may” have.