**Comparision of Requirement Formats**

**Jahnavi Bollineni**

**Purdue University Fort Wayne**

**ACS560 Software Engineering**

**Matthew C Parker**

**Introduction**

Requirements are the foundation of any successful project, ensuring that stakeholders' needs are captured, analyzed, and clearly defined. The way requirements are written influences how well they are understood by teams and how effectively they can be implemented. Over the years, several formats have been developed to help teams create comprehensive and unambiguous requirements. This document explores two widely used formats for writing requirements, analyzes the differences between them, and presents a recommendation on which format to choose.

## Formats for Writing Requirements

### 1. ****User Stories****

A user story is a simple, clear description of a feature told from the perspective of the user or stakeholder. Typically used in Agile development, it focuses on what the user needs and the value it provides. User stories are often written in the format:

“As a [type of user], I want [an action] so that [benefit/outcome].”

**Example:** “As a customer, I want to be able to filter products by category so that I can find what I need quickly.”

#### Advantages:

* Focuses on the user and their needs, making it easier for the development team to understand the context.
* Encourages collaboration between teams, as it is less technical and more user-focused.
* Simplifies requirements into manageable chunks that can be prioritized and iterated upon.

#### Disadvantages:

* Lacks detailed information about non-functional requirements like performance or security.
* May require further elaboration, especially in complex systems, to capture all technical details.

### 2. ****Functional Requirements Document (FRD)****

A Functional Requirements Document is a more formalized approach to requirement writing. It provides a detailed description of how a system should operate, listing each function the system must perform. Functional requirements are usually written in technical language and include specifications such as data input/output, system behavior, and dependencies.

**Example:**

* The system shall allow users to filter products based on category, price, and brand.
* The system shall return search results within 2 seconds.

#### Advantages:

* Offers a comprehensive, detailed account of system behavior, making it well-suited for complex systems.
* Captures both functional and non-functional requirements, providing a complete picture for developers and testers.
* Ensures all stakeholders understand exactly what is to be developed, minimizing ambiguities.

#### Disadvantages:

* Can become overly detailed and difficult to maintain, especially for large projects.
* Requires more upfront work compared to user stories, making it less flexible in environments where changes are frequent.

## Differences Between User Stories and FRD

The primary difference between User Stories and FRD is their focus and structure. User stories are short, concise, and centered around the user experience, making them ideal for Agile environments. They focus on capturing the why and what without delving too deeply into technical details. On the other hand, FRDs provide a comprehensive and detailed explanation of both functional and non-functional requirements, making them more suited for large-scale, complex projects where precision is crucial.

### Choosing the Right Format

For projects where agility and user feedback are central, the **User Story format** is recommended due to its simplicity and focus on user value. However, for projects requiring high levels of detail and involving complex systems, the **Functional Requirements Document** is a better choice due to its thoroughness and technical detail. In cases where both user experience and detailed functionality are essential, a combination of both formats may be used.

## **Conclusion**

The format chosen for writing requirements significantly affects the clarity and success of a project. User Stories prioritize the user’s perspective and are suitable for Agile teams, while Functional Requirements Documents provide detailed technical specifications ideal for more complex projects. Depending on the nature of the project, a team may opt for one format or combine both to ensure all aspects of the system are captured effectively.

## **References**

* Cohn, M. (2004). User Stories Applied: For Agile Software Development. Addison-Wesley.
* Wiegers, K., & Beatty, J. (2013). Software Requirements (3rd ed.). Microsoft Press.
* Fowler, M. (2018). Patterns of Agile Practice Adoption: The Technical Cluster. Addison-Wesley.

**Functional Requirements for Existing Functionality**

**1. Retrieve Average Sales for a Specific Month**  
As a sales analyst, I want to retrieve the average sales for a specific month so that I can analyze the performance of sales during that period.

**2. Retrieve Minimum and Maximum Sales for a Specific Month**  
As a sales analyst, I want to retrieve the minimum and maximum sales for a specific month so that I can identify the sales extremes during that period.

**3. Add New Sales Data**  
As a sales admin, I want to add new sales data so that the sales records in the system are up-to-date and accurate.

**Functional Requirements for New Functionality**

**1. Track Most Selling Product**   
As a sales analyst, I want to track the most selling product for a given month or across all months so that I can identify the top-selling products and focus on promoting them.

**2. Update Sales Data by Product Name**  
As a sales admin, I want to update the sales data for a product by specifying the product name, month, and new sales amount so that I can correct or modify sales records accurately.

**3. Generate Sales Report for a Given Year**  
As a sales manager, I want to generate a sales report for a given year so that I can analyze the overall yearly performance and identify trends and top products.

**Non-Functional Requirements for the REST API**

**1. Performance**  
As a system user, I want the API to respond to requests within 2 seconds so that I can retrieve and interact with sales data efficiently without delays.

**2. Scalability**  
As a system architect, I want the API to scale horizontally and support large amounts of sales data so that it can handle future growth without performance degradation.

**3. Security**  
As a security administrator, I want all sales data-related operations (adding, updating, and deleting) to be authenticated and authorized so that only authorized users can make changes to the data.