FRD:(FUNCTIONAL REQUIREMENT DOCUMENT)

At the core, an FRD or Functional Requirements Document serves as a contract for formal statement, between the business stakeholders and the technology team, on an application’s functional requirements.  The FRD is generally produced by the technical team in response to the business requirements (captured in a BRD - Business Requirements Document, PRD - Product Requirements document, or some other suitable format).

The key purpose of an FRD is to bridge the gap between business and technology.  It’s where project stakeholders and the technical development team meet.  The creation of the FRD forces and ensures collaboration and address both sides of the coin:

* **Business**- it restates the business requirements in terms of functional features and capabilities to be supported by the new system or platform.  This ensures the project team understands the business requirements and are on their way to implement a solution which addresses the business needs or problems.
* **Technology**- it captures key technical constraints and commitments as well key interfaces to external systems

While created by the solution team, the FRD should be solution independent (in general) aka it should express what the application should do and not how it should do it.  The FRD should not commit the technical team to a specific design.

The Functional Requirements Document (FRD) is one way to express functional specifications and define the requirements and functional solution direction of software solution.  The FRD is not the only way - there are other functional specification formats and templates, depending on methodology and organizational needs::

* Systems Requirements Document (SRS)
* Use Cases
* User Stories
* High-Level Impact Assessment

While [FRD templates](https://www.modernanalyst.com/Resources/Templates/tabid/146/ID/4931/Template-for-Writing-Concise-Functional-Requirements-Documents.aspx) vary, some of the more common elements of a functional requirements document are (just a suggestion):

* Business process and workflows - provides a view of the desired target state business process models outlining which steps in the process will stay as-is and which steps will be either automated or impacted by the solution at hand.
* Functional Requirements - these are the traditional “the system shall…” type requirements and can be broken into multiple subsections such as:
  + User Requirements
  + Regulatory Requirements
  + Compliance Requirements
  + User Interface Requirements
* Operational Requirements - while some may be functional in nature, the operational requirements tend to focus more on administrator level capabilities and needs:
  + Role Based Security
  + Audit Trail
  + Configuration
* Data Requirements - describe the data which needs to be supported by the system.  This section may include logical data models, data migration and conversion requirements. It is not uncommon to also see data flow diagrams describing the conceptual flow of data.
* [Non-Functional Requirements](https://requirements.com/Content/What-is/what-are-non-functional-requirements) - these generally describe attributes of the system which, when done right, may never be observed by the users which use the functional capabilities.
  + Security Requirements
  + Performance Requirements
  + Capacity
  + Fault Tolerance
  + Data Retention
  + Recoverability
  + Additional sections or information which may be found in an FRD:
  + Background
  + Scope
  + List of Project Stakeholders
  + Assumptions
  + Constraints

BRD(BUSINESS REQUIREMENT DOCUMENT)

A business requirements document (BRD), is a formal report that details all the objectives or “requirements” for a new project, program or business solution. It describes a business need or objective along with what is expected as the project proceeds. Once the BRD is approved, the company or team can begin finding the best approach to building the solution. In this way, BRD provides clarity, retains focus and removes ambiguity about the project’s needs.

EPIC IS RELATED TO AGILE:

*An agile epic is a body of work that can be broken down into specific tasks (called user stories) based on the needs/requests of customers or end-users. Epics are an important practice for*[*agile*](https://www.atlassian.com/agile/teams)*and [DevOps](https://www.atlassian.com/devops/what-is-devops) teams.*

When adopting agile and DevOps, an epic serves to manage tasks. It's a defined body of work that is segmented into specific tasks (called “stories,” or “user stories”) based on the needs/requests of customers or end-users.

Epics are a helpful way to organize your work and to create a hierarchy. The idea is to break work down into shippable pieces so that large projects can actually get done and you can continue to ship value to your customers on a regular basis. Epics help teams break their work down, while continuing to work towards a bigger goal.

Maintaining agility when organizing large tasks, like epics, is no small task (pun intended).  Learning how epics relate to healthy agile and [DevOps best practices](https://www.atlassian.com/devops/what-is-devops/devops-best-practices) is an essential skill no matter the size of your organization.

What is an agile epic?

An epic is a large body of work that can be broken down into a number of smaller [stories](https://www.atlassian.com/agile/project-management/user-stories), or sometimes called “Issues” in Jira. Epics often encompass multiple teams, on multiple projects, and can even be tracked on multiple boards.

Epics are almost always delivered over a set of sprints. As a team learns more about an epic through development and customer feedback, user stories will be added and removed as necessary. That’s the key with agile epics: Scope is flexible, based on customer feedback and team cadence