How to Write a Software Requirements

Specification (SRS) Document

Brocoders · Follow

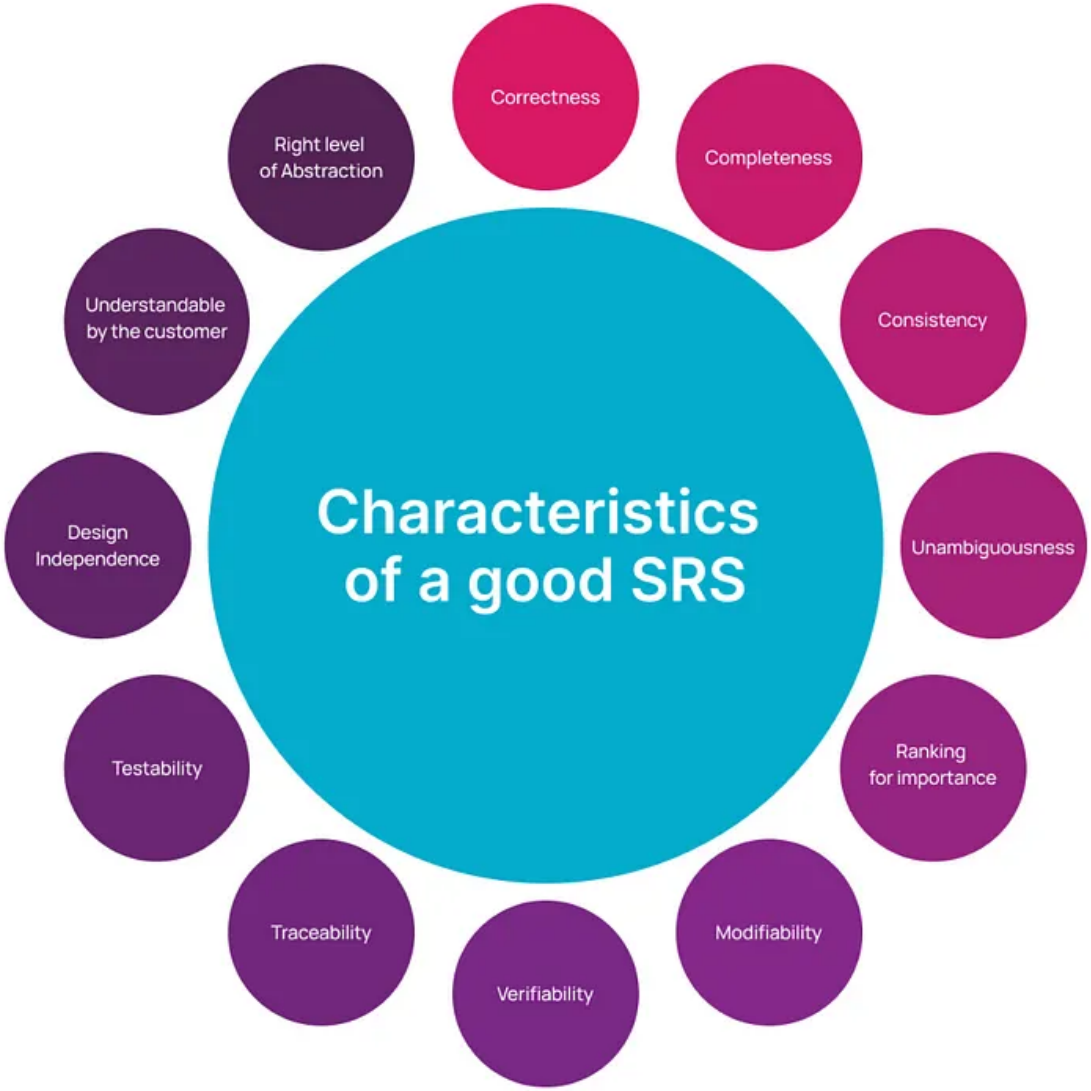
Published in Brocoders Team ·
11 min read · Feb 15, 2022

There are plenty of horror stories about outsource software developers and clients clashing on project requirements, and one of the most common causes of such quarrels is ambiguous language. Vague terms like “urgent,” “pending,” “ASAP,” and so on might mean something different to the client and to the contractor. Or, perhaps, the involved parties have a different impression of the product’s

prioritized functionalities. It's neither party's fault, per se — they are just speaking different languages, in a sense. The customer has an idea for the product, its functions, and what the user interface might look like. But the developers are thinking about the "behind-the-scenes" of the project, how to write a scalable and readable code, ensure security, make a solution performant, minimize bugs, etc. When there is a disconnect between both viewpoints, it can lead to extra work, wasted money, stress, and even a

sub-par final product. But, thankfully, there's a way to avoid such miscommunications from the get-go: Software Requirements Specifications.

What Are SRS and Why Are They Needed?


Characteristics of a good SRS
Correctness
Completeness
Consistency
Unambiguousness
Ranking for importance
Modifiability
Verifiability
Traceability
Testability
Design Independence
Understandable by the customer
Right level of Abstraction

An SRS is a document that describes a software system that will be developed. It's accessible to all parties, including the stakeholders, investors, a project manager, programmers, designers, and so on. There are three types of requirements that the SRS contains:

1. **Top-tier:** These are the high-level business requirements. They outline the business'

measurable goals, define the purpose behind the project, and align the project goals with stakeholder goals.

2. **Middle-tier:** These are the user requirements. They reflect specific user needs and expectations, describe who is using the software, and highlight user interactions.

3. **Bottom-tier:** These specify the product's functionality in tech terms. They identify

functions, features, use cases, and non-functional requirements, as well as describe the project as functional modules + non-functional attributes.

By combining specifications from all three tiers, you're giving all involved parties a clear understanding of the project and measurable deliverables. This is especially important if you're outsourcing the project; a thorough SRS document enables an outsourcing agency to

communicate your project's requirements clearly and accurately to their own team. For a deeper illustration of how SRS improve communication and collaboration between parties, check out the table below:

PARTY	BENEFIT OF SRS
Software Developers	They can accurately estimate the scope of the project.
Project Managers	They can use the documentation to plan their work.
Designers	Through an SRS document, they gain insight into a project and adapt the design to the use case.
Testers	They receive guidance from the SRS on how to create test cases that meet business needs.
Stakeholders	They use the SRS to understand the software — for instance, how their idea is translated into architecture, functionality, and tasks, which are then all evaluated.
Investors	The SRS provides them with an overview of the system's functions, enabling them to make more informed investment decisions.

Summed up, an SRS document is important

because it provides stakeholders, PMS, programmers, and so on with a single source of information and expectations. By compiling all necessary information in one place, you can greatly reduce the frequency of misunderstandings.

Basic Elements of an SRS

In order for an SRS to be useful to all involved parties, it must have several basic elements, as

outlined below:

- Business Drivers: Include the reasons why the customer wants you to build the software. It's important to include the rationale or the driving force behind the new system; this information guides the decisions that the business analysts, system architects, and developers will make.
- Business Model: Which basic customer

business model will the system need to support? Include organizational and business context, key business functions, current and future state diagrams, and process flow diagrams.

- Business and System Use Cases: Typically, this section will contain a UML — Use Case Diagram that shows which main external objects the system will interact with.
- Technical Requirements: In this section, the

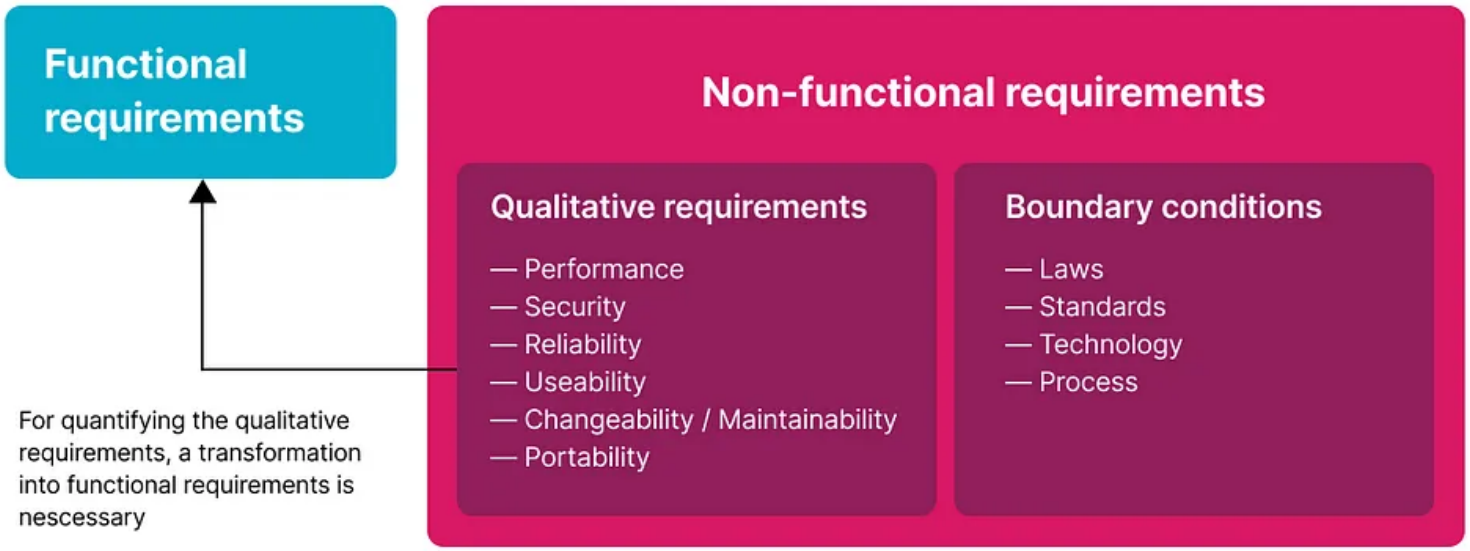
document will list any “non-functional” requirements that compose the technical environment.

- System Qualities: You will list any non-functional requirements that determine the system’s quality.
- Constraints and Assumptions: This section includes software constraints that the customer requires, which can be used to exclude options from development

consideration.

- Acceptance Criteria: Lastly, this element illustrates how the customer will sign off on the final system.

Functional vs. Non-functional Requirements



Within the SRS, you'll need to address functional and non-functional requirements. What do these

terms imply?

- **Functional Requirements** — The functions a software must perform.
- **Non-Functional Requirements** — Standards used to judge the system's operating quality

And why are both requirements important?

In essence, functional requirements help

software engineers capture the intended behavior of the application — and said behavior may be expressed as services, tasks, or functions that the system must perform. On the other hand, a non-functional requirement ensures that the entire system is usable and effective. If all functional requirements are met but non-functional requirements are lacking, the system would fail to meet the user's needs.

Some examples of functional requirements

include:

- If/then behaviors
- A description of the system's workflow
- Data inputs and their corresponding outputs
- The logic behind data handling

Non-functional requirements might look like:

- Performance and scalability
- Compatibility and portability
- Maintainability, availability, and reliability
- Security
- Usability
- Localization

How to Write Documentation

You now know the main elements of an SRS, but how do you put them in order to create one cohesive document? Not to worry: we've created a template that addresses all requirements relevant to all parties. Requirements engineering is based on:

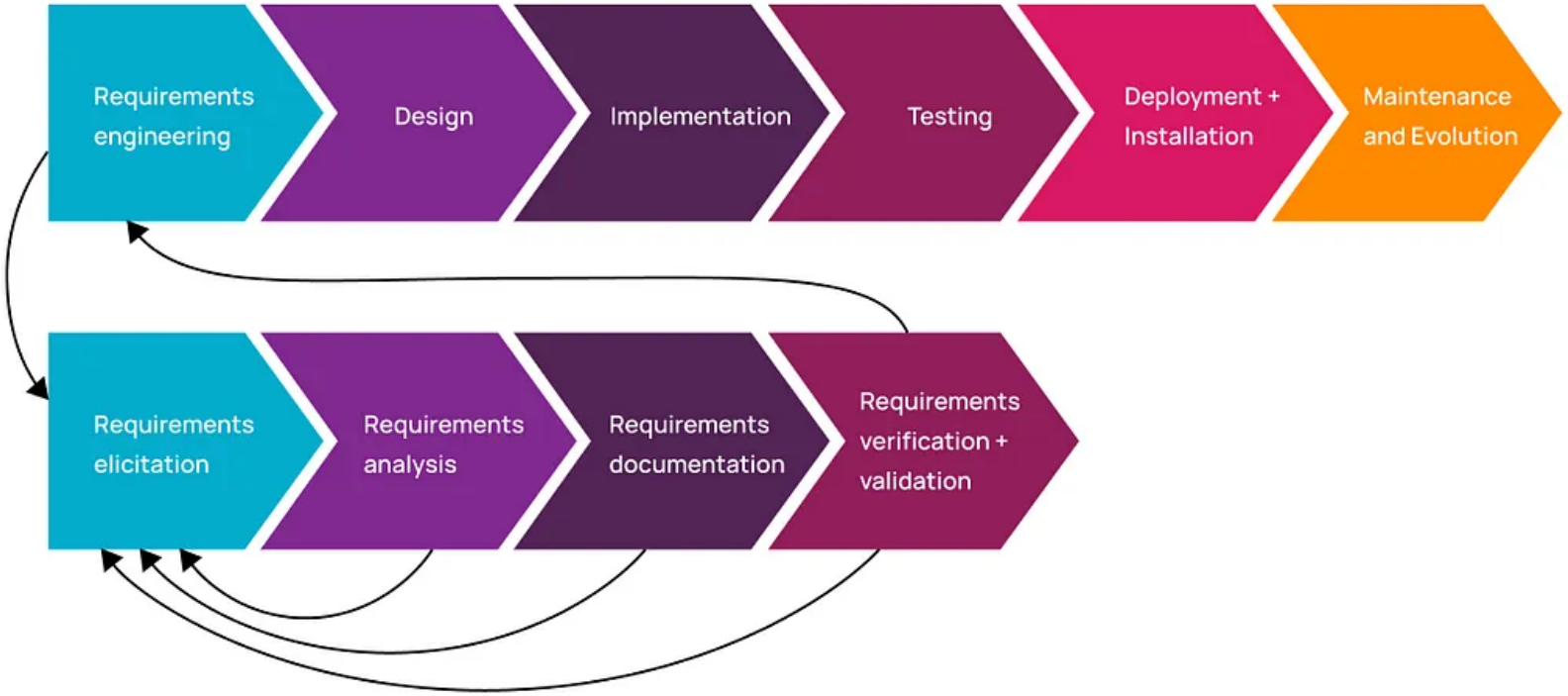
- requirements elicitation (interview with

stakeholders and decision makers)

- requirements analysis (are they consistent)
- requirements documentation
- requirements verification and validation (am I building the system right and am I building the right system)

Requirements engineering in SDLC

Software Engineering



To write SRS, you need a framework, and there are plenty of templates to streamline the process. There are also standards to adhere to when writing technical requirements. For instance, there is IEEE830 — which is outdated but still regularly used.

SRS contents IEEE Std 830-1998

Introduction

- Purpose
- Scope
- Definitions, acronyms and abbreviations
- References
- Overview

Overall description

- Product perspective
- Product functions
- User characteristics
- General constraints
- Assumptions and dependencies
- Lower ambition levels

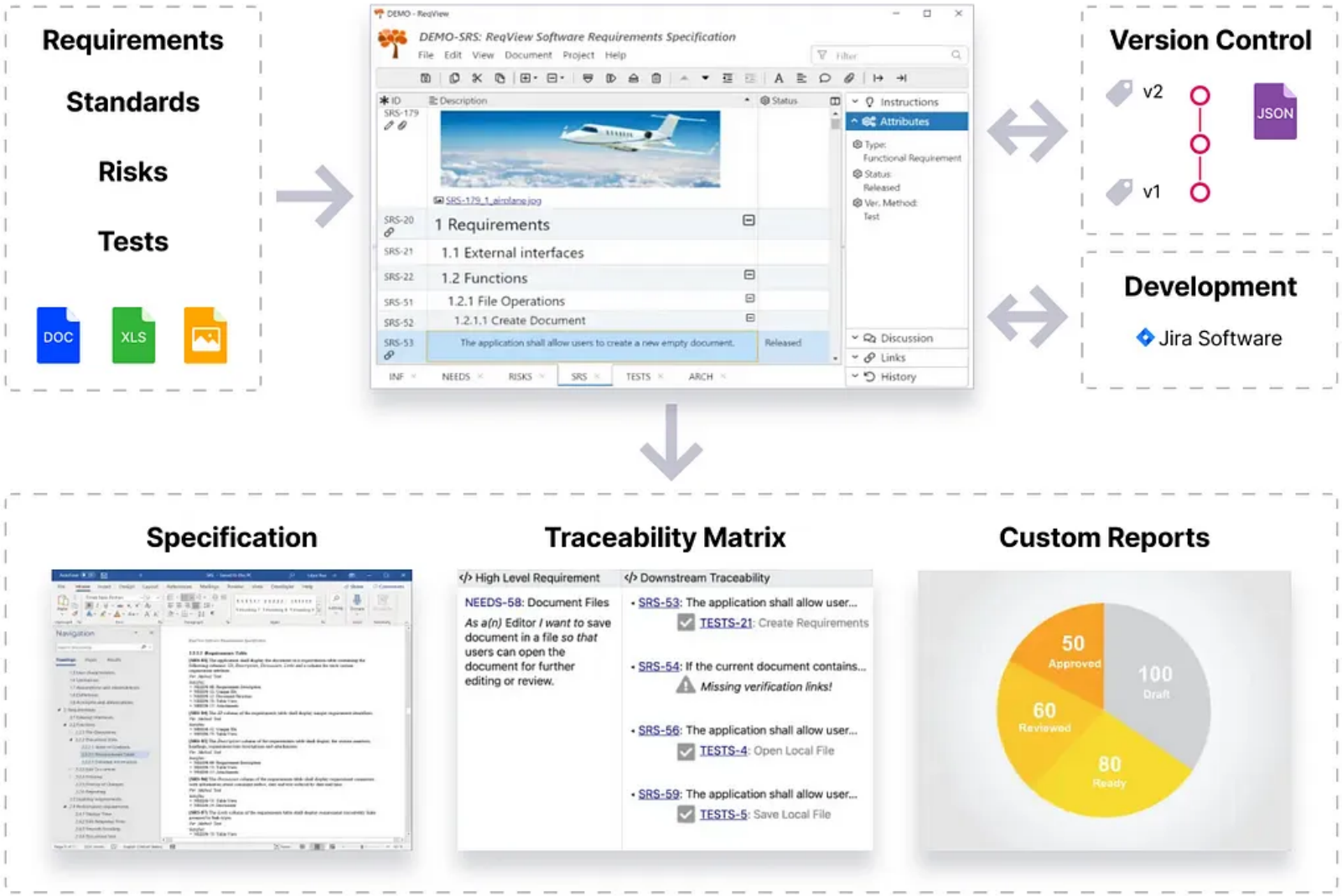
Specific requirements

- Interface requirements
  - User interfaces
  - Hardware interfaces
  - Software interfaces
  - Communication interfaces
- Functional requirements
- Performance requirements
- Design constraints
- Software system attributes
- Other requirements

Supporting information

- Index
- Appendices

There is also an international standard called IEEE/ISO/IEC 29148–2018, which we have adapted for our needs. Below we'll show examples from Reqview tool that is used to organize requirements documentation in accordance with standard mentioned above.


Requirements
Standards
Risks
Tests
DOC
XLS
DEMO - ReqView
DEMO-SRS: ReqView Software Requirements Specification
File Edit View Document Project Help
SRS-179_1_airplane.jpg
SRS-20 1 Requirements
SRS-21 1.1 External interfaces
SRS-22 1.2 Functions
SRS-51 1.2.1 File Operations
SRS-52 1.2.1.1 Create Document
SRS-53 The application shall allow users to create a new empty document. Released
Instructions
Attributes
Type: Functional Requirement
Status: Released
Ver. Method: Test
Discussion
Links
History
INF NEEDS RISKS SRS TESTS ARCH
Version Control
v2
v1
JSON
Development
Jira Software
Specification
Traceability Matrix
High Level Requirement
Downstream Traceability
NEEDS-58: Document Files
As a(n) Editor I want to save document in a file so that users can open the document for further editing or review.
SRS-53: The application shall allow user...
TESTS-21: Create Requirements
SRS-54: If the current document contains...
Missing verification links!
SRS-56: The application shall allow user...
TESTS-4: Open Local File
SRS-59: The application shall allow user...
TESTS-5: Save Local File
Custom Reports
50 Approved
100 Draft
60 Reviewed
80 Ready

One more tool that is used in our company for requirements management is Confluence by Atlassian.

Product documentation can be separated into:

1. Needs Document (It describes high-level user needs via user stories)
2. SRS (Software Requirements Specification)

3. Tests (This document describes test cases verifying functionality described in Requirements)

4. Architecture (Describes the architecture of the application)

In the examples below, we'll show you how to create Needs, Architecture, and SRS documents. At the very least, we suggest preparing a Needs document so it can be sent to potential vendors.

Needs Document

1. Introduction

This document describes the high-level functional requirements of software.

1.1 Purpose. This is informational section that is used to describe the purpose of this document. Usually we can put there “The purpose of this

document is description of software that is going to be build for…”

1.2 Project Scope. Briefly describe the scope of the product.

For example for development software that is used for creating SRS (such as Reqview tool) it can be described in a such way

Users should be able to:

- Record requirements specifications

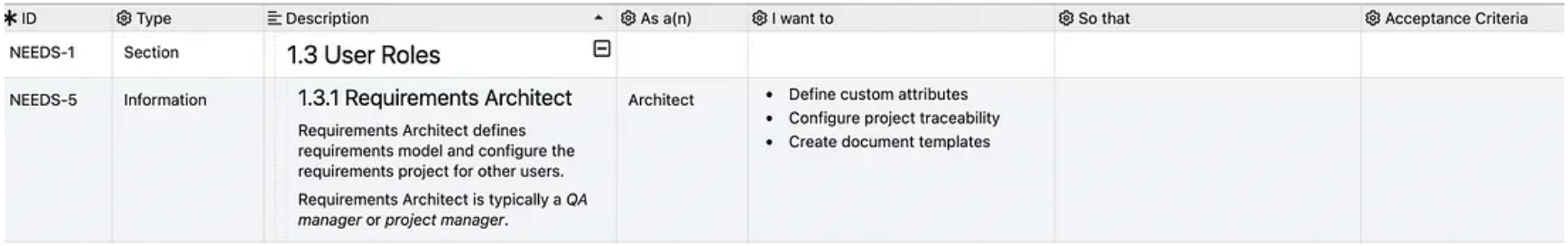
- Use custom attributes to manage requirements
- Set up requirements traceability & browse the matrix
- Review and comment upon requirements
- Filter and search for requirements
- Import requirements from Excel and MS Word

- Export requirements to PDF, HTML, DOCX, XLSX, or CSV
- Print requirements specifications
- Analyze requirements coverage and change impact

1.3 User Roles. Identify user roles for the product; for each role, describe the characteristics of a typical user. Consider creating personals with a real name — each one

should be described sufficiently, so all team members feel like they know the persona.

This is an example of user roles definition from Reqview



ID	Type	Description	As a(n)	I want to	So that	Acceptance Criteria
NEEDS-1	Section	1.3 User Roles				
NEEDS-5	Information	1.3.1 Requirements Architect Requirements Architect defines requirements model and configure the requirements project for other users. Requirements Architect is typically a *QA manager* or *project manager*.	Architect	• Define custom attributes • Configure project traceability • Create document templates		

Below is example from our product

documentation

2. Product Functions

2.1 User stories. We use user stories to describe a product and its functionality, and for easier navigation, they can be grouped into “epics.” To write a user story, set the following attributes:

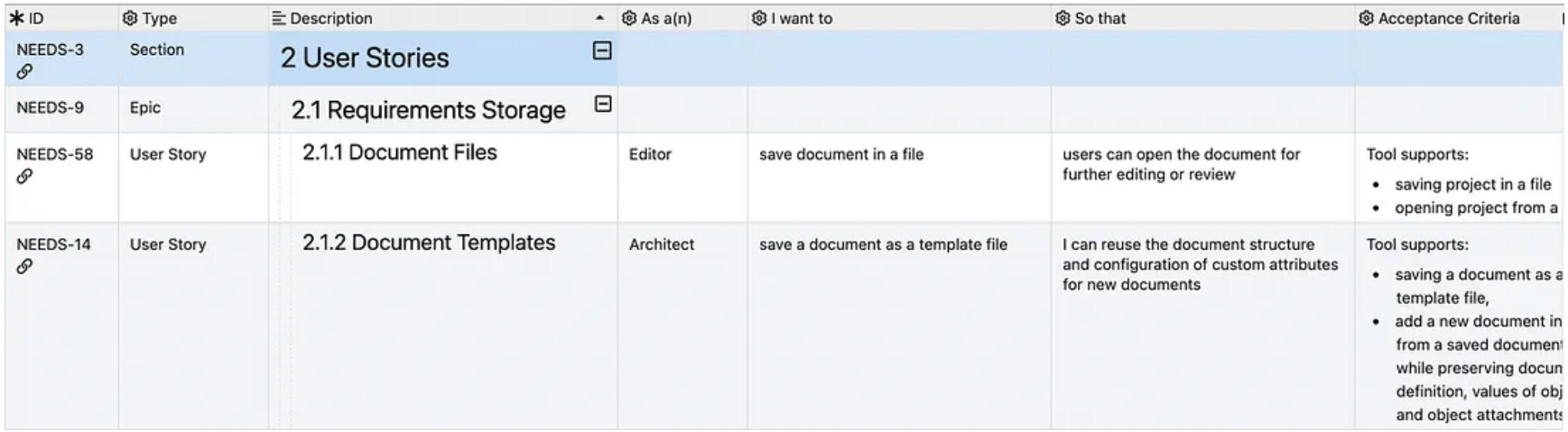
- Heading: A short story name
- As a(n): The type of user

- I Want: A goal
- So That: A reason
- Acceptance Criteria: Details on how the story should be verified; basic criteria to determine if the story is fully implemented
- Priority: “High” for fundamental features, “Medium” for important features with a short-term workaround, and “Low” for features that will be needed in a later release

or that can be omitted if time runs out.

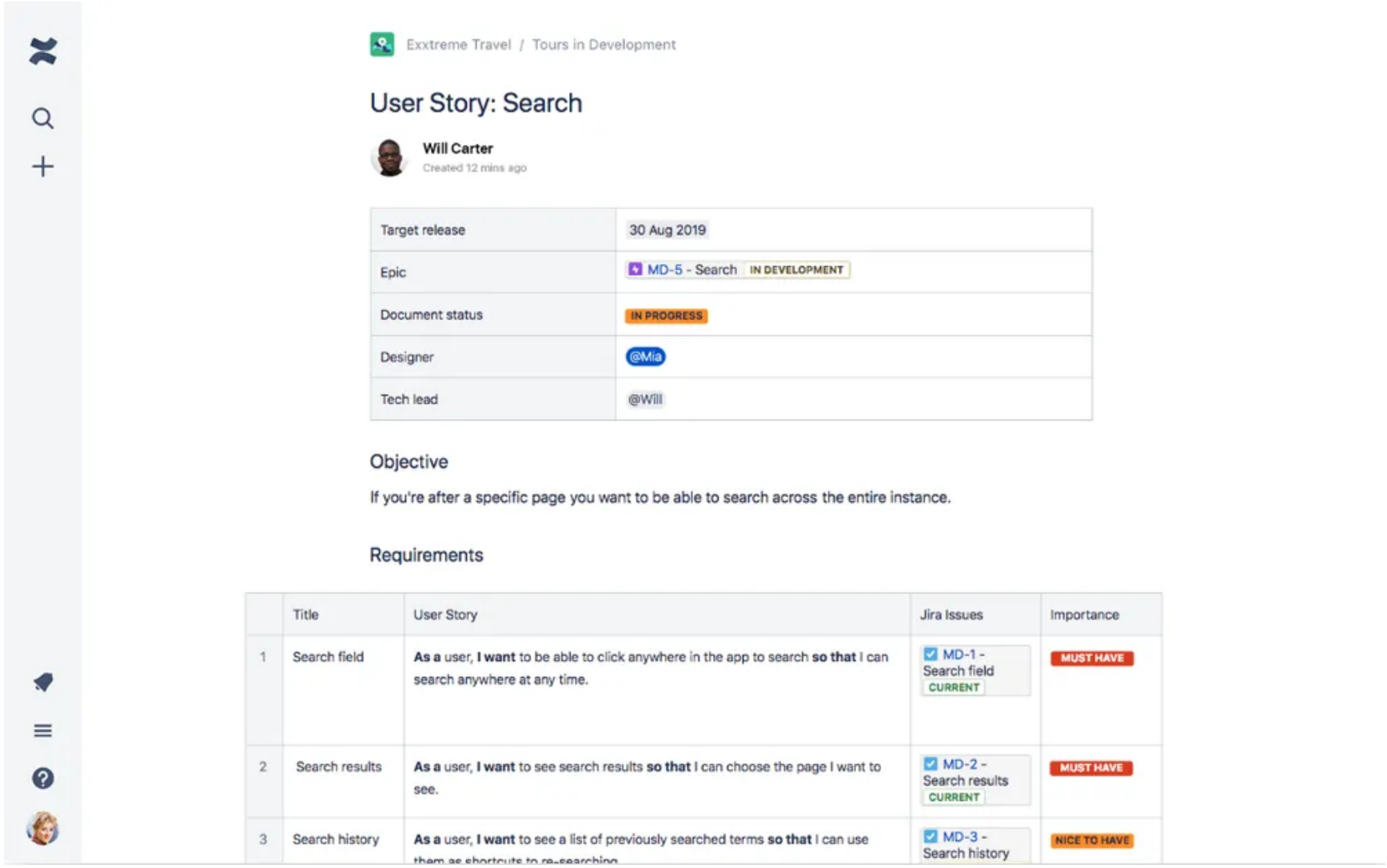
- Risk: “High,” Medium,” or “Low”
- Estimation: Story points that offer relative estimates of the complexity, effort, or duration of a story
- Status: Workflow information that classifies stories as “New,” “Planned,” “Implemented,” or “Verified.”

User Stories example from Reqview:



ID	Type	Description	As a(n)	I want to	So that	Acceptance Criteria
NEEDS-3	Section	2 User Stories				
NEEDS-9	Epic	2.1 Requirements Storage				
NEEDS-58	User Story	2.1.1 Document Files	Editor	save document in a file	users can open the document for further editing or review	Tool supports: • saving project in a file • opening project from a
NEEDS-14	User Story	2.1.2 Document Templates	Architect	save a document as a template file	I can reuse the document structure and configuration of custom attributes for new documents	Tool supports: • saving a document as a template file, • add a new document in from a saved document while preserving docun definition, values of obj and object attachments

User Stories example from Confluence. Source: Atlassian Documentation



Exxtreme Travel / Tours in Development

User Story: Search

Will Carter
Created 12 mins ago

Target release	30 Aug 2019
Epic	MD-5 - Search IN DEVELOPMENT
Document status	IN PROGRESS
Designer	@Mia
Tech lead	@Will

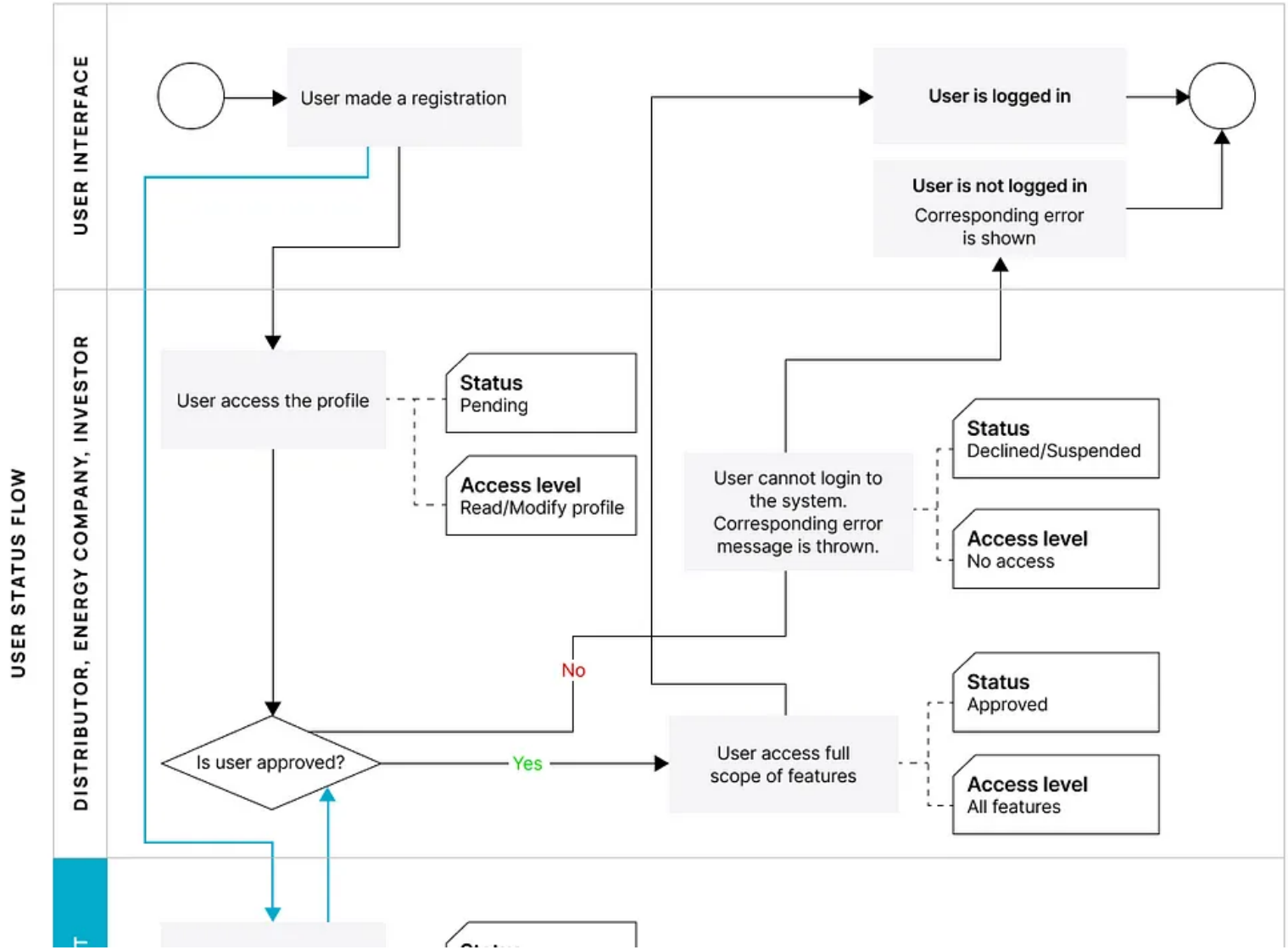
Objective

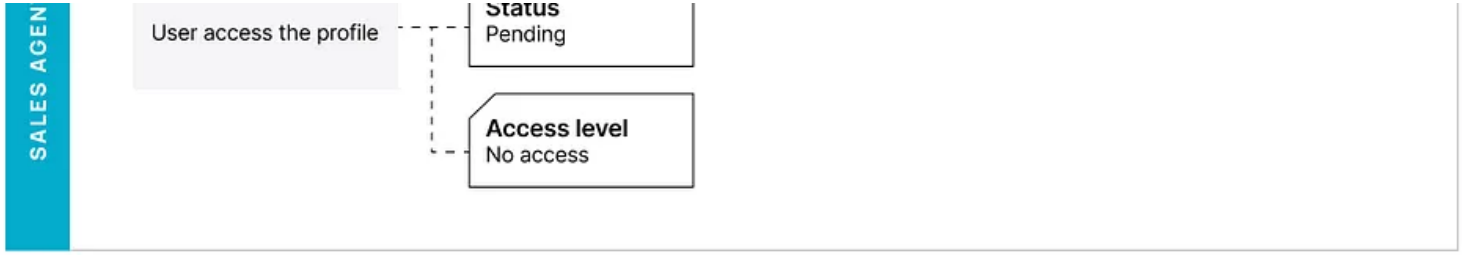
If you're after a specific page you want to be able to search across the entire instance.

Requirements

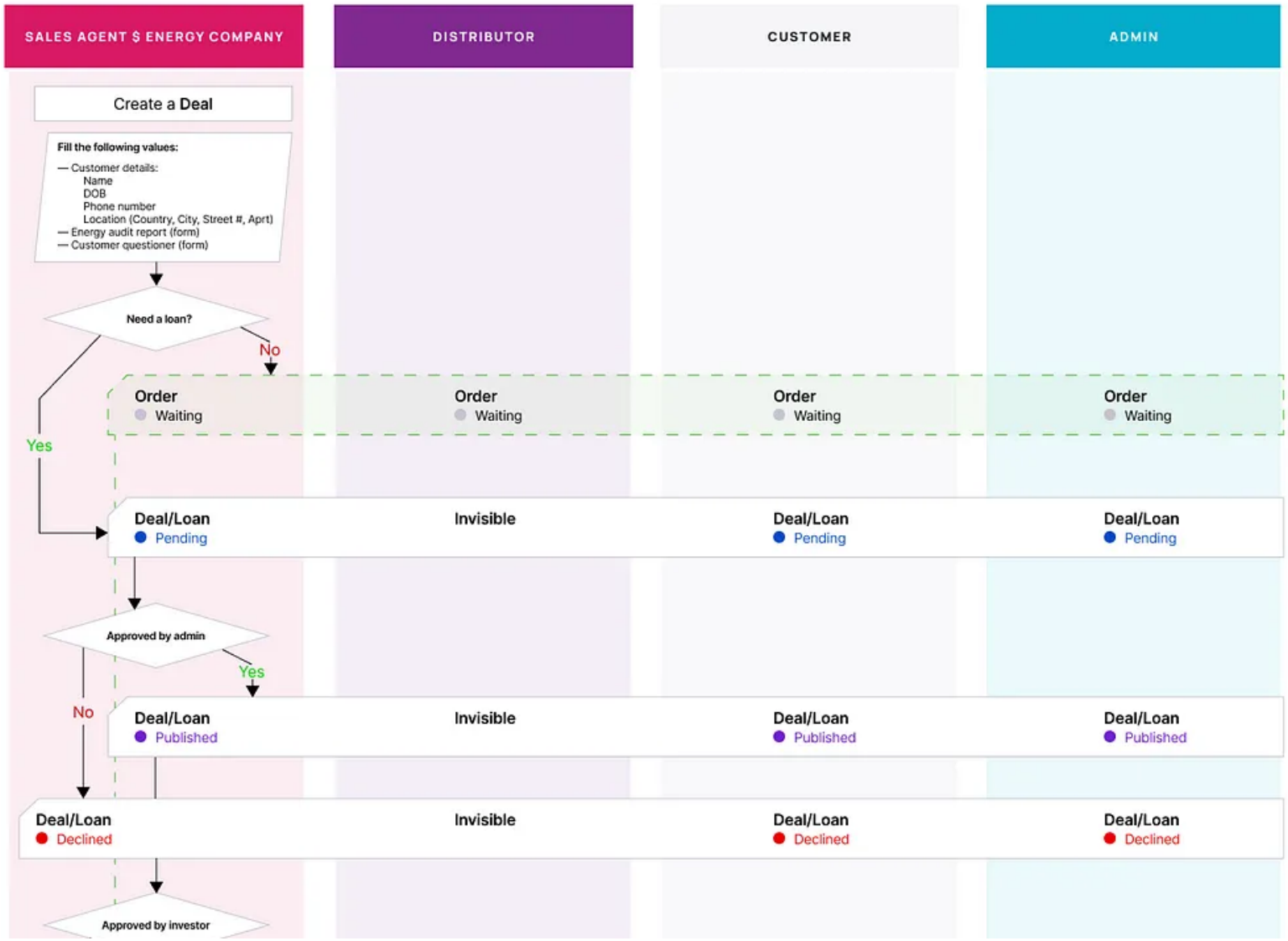
	Title	User Story	Jira Issues	Importance
1	Search field	**As a** user, **I want** to be able to click anywhere in the app to search **so that** I can search anywhere at any time.	MD-1 - Search field CURRENT	MUST HAVE
2	Search results	**As a** user, **I want** to see search results **so that** I can choose the page I want to see.	MD-2 - Search results CURRENT	MUST HAVE
3	Search history	**As a** user, **I want** to see a list of previously searched terms **so that** I can use them as shortcuts to re-searching	MD-3 - Search history	NICE TO HAVE

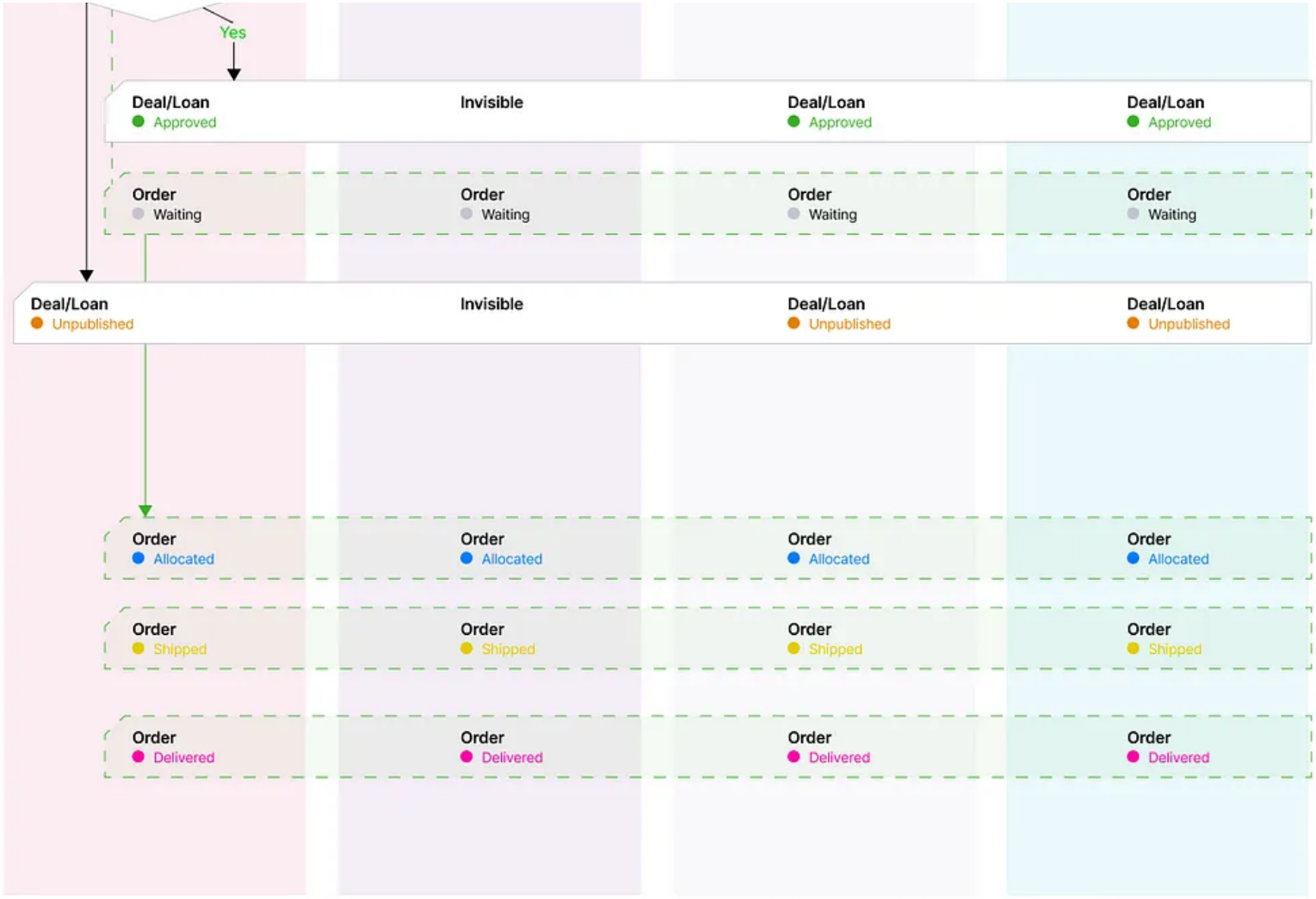
2.2 User statuses flow. Sometimes, you may need to show flow using diagrams — like, for example, in this User Status flow.


USER STATUS FLOW
USER INTERFACE
DISTRIBUTOR, ENERGY COMPANY, INVESTOR
User made a registration
User is logged in
User is not logged in
Corresponding error is shown
User access the profile
Status
Pending
Access level
Read/Modify profile
User cannot login to the system. Corresponding error message is thrown.
Status
Declined/Suspended
Access level
No access
No
Is user approved?
Yes
User access full scope of features
Status
Approved
Access level
All features



As another example, here's a Deal/Order Creation and Status flow:


SALES AGENT $ ENERGY COMPANY
DISTRIBUTOR
CUSTOMER
ADMIN
Create a Deal
Fill the following values:
— Customer details:
Name
DOB
Phone number
Location (Country, City, Street #, Aprt)
— Energy audit report (form)
— Customer questioner (form)
Need a loan?
No
Yes
Order
Waiting
Order
Waiting
Order
Waiting
Order
Waiting
Deal/Loan
Pending
Invisible
Deal/Loan
Pending
Deal/Loan
Pending
Approved by admin
Yes
No
Deal/Loan
Published
Invisible
Deal/Loan
Published
Deal/Loan
Published
Deal/Loan
Declined
Invisible
Deal/Loan
Declined
Deal/Loan
Declined
Approved by investor



3. Constraints

In this section we should describe constraints that are effectively global requirements, such as speed of application, limited development resources.

Example from Reqview

NEEDS-4	Section	3 Constraints				
NEEDS-39	Constraint	3.1 Startup Time	Editor	quickly start the tool	I do not need to wait long time for editing a project	Tool starts in less than 10s projects.

Architecture Document

1. Introduction

2. Architecture

It can be described this way.

Overall architecture

Operates with 6 types of users, which are allowed to access the system via 3 interfaces:

- Web application

- Admin panel — web application
- Mobile application — both Android and iOS platforms

Admin panel

Admin Moderator (sub-admin)

Web application

Distributor Investor Company Customer (currently — all the activities within the UI are blocked)

Mobile application

Sales Agents Sub Agents

Staging Environment

Is hosted at Brocoders AWS resources and can be handled by the local team. The overall infrastructure includes Redis instance for delayed jobs handling, Amazon s3 bucket storage for system files, and general instances for the admin panel and web app.

Production Environment

Is hosted on the client's AWS resources. Overall infrastructure is set up in the same manner.

3.References

Lists the SRS, as well as any documents or websites that the document refers to.

SRS Document

1. Introduction

1.1 Purpose. The main purpose of this document is to provide a working example of a Software Requirements Specification (SRS)

based on the ISO/IEC/IEEE 29148:2018 standard.

1.2 Scope. As with the Needs Document, you should describe what problem the software will solve and how it will be used.

Users should be able to:

- Record requirements specifications
- Use custom attributes to manage requirements
- Set up requirements traceability & browse

the matrix

- Review and comment upon requirements
- Filter and search for requirements
- Import requirements from Excel and MS Word
- Export requirements to PDF, HTML, DOCX, XLSX, or CSV
- Print requirements specifications

- Analyze requirements coverage and change impact

1.3 Product Perspective. In this section you describe how this product relates to the others elements of the system, for example if it is an element of the larger system. You can put here requirements for User interfaces, System interfaces, Hardware interfaces, Software interfaces, Communications interfaces, Memory constraints etc.

1.3.1 System Interfaces. List each system interface and identify the functionality of the software to accomplish the system requirement. For example: The application runs on the latest version of Chrome or Firefox browser on Windows, Linux, and Mac.

1.3.2 User Interfaces. List the logical characteristics of all interfaces between the users and the software product — for instance, GUI standards and sample screen images

1.4 Product Functions. Link this section to

Section 2 of the Needs document (User Stories).

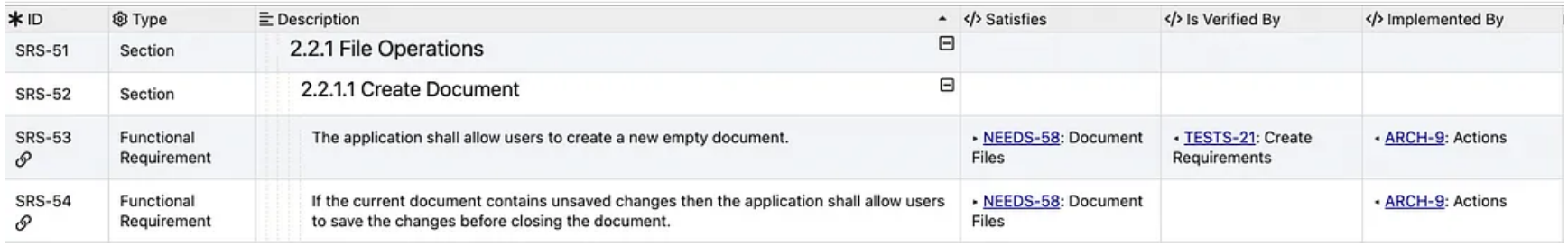
2. Requirements

Product functions are summarized in Section 2 of the Needs document (User Stories).

2.1 Functions. This section describes functional requirements. Every function should be linked to Needs, Tests, and Architecture documentation and sometimes it is convenient to linked it with

the task in Jira.

This is how it looks like in Reqview tool



ID	Type	Description	Satisfies	Is Verified By	Implemented By
SRS-51	Section	2.2.1 File Operations			
SRS-52	Section	2.2.1.1 Create Document			
SRS-53	Functional Requirement	The application shall allow users to create a new empty document.	▸ NEEDS-58: Document Files	◂ TESTS-21: Create Requirements	◂ ARCH-9: Actions
SRS-54	Functional Requirement	If the current document contains unsaved changes then the application shall allow users to save the changes before closing the document.	▸ NEEDS-58: Document Files		◂ ARCH-9: Actions

2.2 Usability Requirements. Define usability and quality in use requirements and objectives for the software system, which can include measurable effectiveness, efficiency, satisfaction

criteria, and avoidance of harm that could arise from specific contexts of use.

An interface design's usability requirements should support the following for its primary users:

- Efficiency: Little to no setbacks for user and quickly-accomplishable goals.
- Intuitiveness: Easy-to-learn interface with

simple navigation. All headings, buttons, and error messages are easy to understand.

- Low perceived workload: The interface appears simple to use and isn't intimidating, frustrating, or demanding.

2.3 Performance Requirements. Performance requirements define how well the software system accomplishes certain functions under specific conditions. Examples include the

software's speed of response, throughput, execution time, and storage capacity. The service levels comprising performance requirements are often based on supporting end-user tasks. Like most quality attributes, performance requirements are key elements in the design and testing of a software product. For example, some performance requirements include:

- Database requirements

- Design constraints
- Standards compliance
- Software system attributes

3. Verification

Verification tests are specified in the [TESTS] document.

4. Supporting Information

Provide any information and define any terms necessary to understand the document. Define abbreviations and acronyms, provide diagrams and models, and create a numbered list of any TBD occurrences.

5. References

Include the following information regarding references:

- Provide a complete list of all documents referenced elsewhere
- Identify each document by title, report number (if applicable), date, and publishing information
- Specify the sources from which the references can be obtained

Once you've completed all sections of the SRS, you'll need to get it approved by the project's key stakeholders. Make sure everybody is reviewing the document's most recent version!

SRS Design Guidelines

Now that you know the general structure of an SRS document, you're well on your way! We'd just like to give you a few design guidelines, so

you can be sure your SRS is as clear as possible.

- Include plenty of diagrams and models, as these contribute to a better understanding of processes.
- Avoid ambiguity. The SRS should be clear, so the involved parties can avoid endless discussion and doubt. Pro tip: to identify ambiguity, carry out a formal peer review.
- Pay attention to sequence. One section of the

SRS shouldn't conflict with another. Make sure to use the same terminology and ensure the document is in a consistent format.

- Use accurate, precise language. Avoid saying things like "the app needs to be released as soon as possible." Use specific dates, figures, and goals.
- Create a glossary, especially if you're using "internal" definitions.

The following five tips will help you to work with SRS efficiently:

- Save the history of changes, so anybody can trace changes back to the first version.
- Assign each requirement an identifier so it can be easily traced in the documentation.
- Use critical thinking to prioritize the

product's main functions.

- Focus on the customer. There should be a portrait of the average end-user.
- The SRS document should be flexible enough to be updated.

Summing Up

SRS documents can be lengthy, and they might seem complicated at first. But by following the

template above, you'll see that creating the document really isn't that bad. The benefits that come from SRS (improved communication and aligned goals) greatly outweigh the effort of creating and approving the form. At Brocoders, we create SRS for all of our projects, and our clients are big fans! Reach out today if you want to learn more about our software development process.

Originally published at https://brocoders.com.

Software Documentation

Software Requirements

Specifications

Documentation

Published in Brocoders Team

Follow

65 Followers · Last published Jun 27, 2024

Technical guides and useful articles for startup' and business' owners

Written by Brocoders

Follow

214 Followers · 84 Following

Technical partner for well-funded startups and small-medium businesses. Our full-cycle development team bring product from MVP to a successful product stage

No responses yet

What are your thoughts?

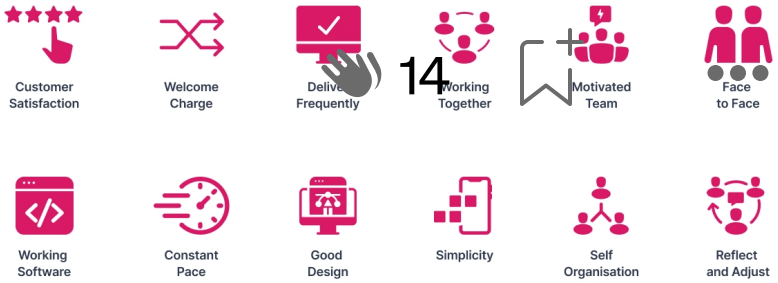
Respond

More from Brocoders and Brocoders Team

In Brocoders T... by Brocod...

Functional vs. Non-Functional...

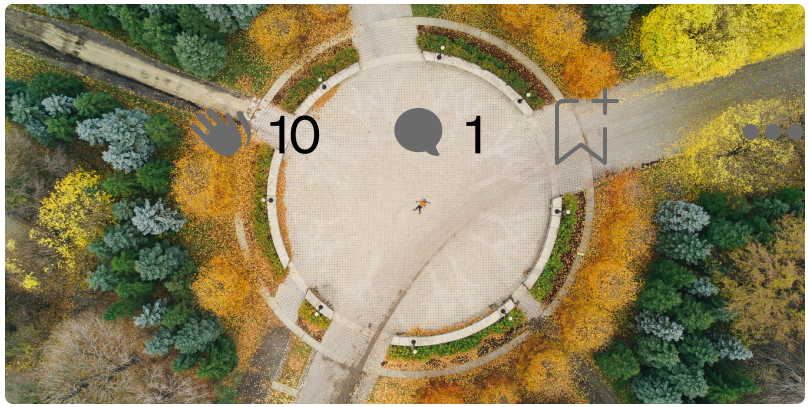
If you’re ordering custom software, you and your...



In Brocoders T... by Brocod...

Top Alternatives to OpenAI

AI can not only boost our analytic and decision-...



In Brocoders T... by Brocod...

The 12 Key Principles of Agile Methodology

The Agile Manifesto is the approach to software...

Mar 23, 2022 2

See all from Brocoders

In Brocoders T... by Brocod...

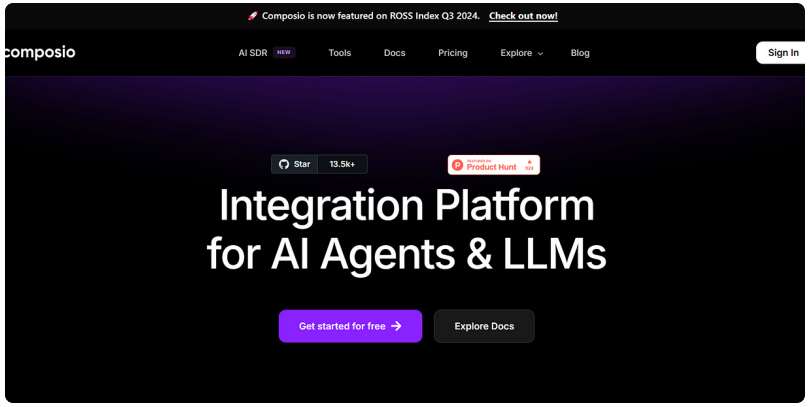
A Guide to the Agile Software...

A comprehensive guide with clear explanations an...

Nov 9, 2021 7

See all from Brocoders Team

Recommended from Medium



In Level Up C... by Jacob Be...

The 5 paid subscriptions I...

Tools I use that are cheaper than Netflix

Jan 7 · 4K · 97

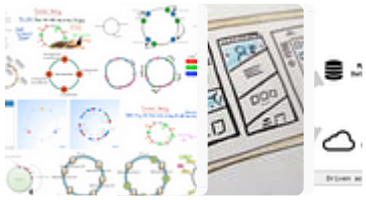
Let's Code Future

15 Modern Open-Source Tools for...

13 open-source tools to supercharge your next big...

5d ago · 210

Lists



General Coding...

20 stories · 1874 saves

ChatGPT

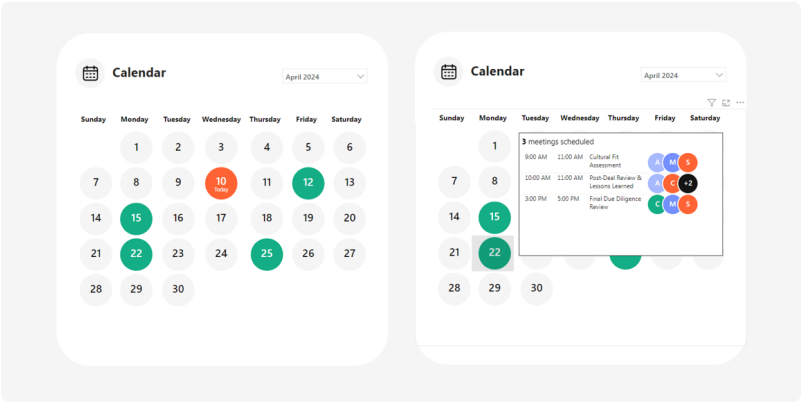
21 stories · 943 saves

stories saves

Leadership

62 · 512

stories saves



In Towards ... by Thuwarak...

How to Build a Knowledge Graph i...

I tried and failed creating one—but it was when LLM...

3d ago · 451 · 5

In The BI Co... by Isabelle Bi...

How to Build a Dynamic Calendar...

A Step-by-Step Tutorial to Create an Interactive...

Apr 11, 2024 · 378 · 8

In JavaScript i... by Digvijay...

The Popover API: Building Modals Ha...

How this HTML API is replacing hundreds of line...

Jan 6 · 589 · 11

Mehdi BAFDIL

Angular's Game-Changing Feature...

Last week, I stumbled upon something in Angular that...

3d ago · 60 · 7

See more recommendations