Shrapnel

Requirements Specification

Dan Jones

Latest Revision: June 23, 2019

Contents

1 Introduction

1.1 Purpose

TBA

1.2 Scope

The Shrapnel system, herein referred to as "The System," shall consist of software intended to facilitate project management activities related to the production of software products. This is achieved through the creation of backlog management features and a virtual project board.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 **Agile**
Explanation
1.3.2 **Backend**
Explanation
1.3.3 **Backlog**
Explanation
1.3.4 **Card**
Explanation
1.3.5 **Endpoint**
Explanation
1.3.6 **Frontend**
Explanation
1.3.7 **Issue**
Explanation
1.3.8 **Kanban**
Explanation
1.3.9 **Lean**
Explanation
1.3.10 **PII: Personally Identifiable Information**
Explanation
1.3.11 **Scrum**
Explanation
1.3.12 **Sprint**
Explanation
1.3.13 **SRS: Software Requirements Specification**
Explanation
1.3.14 **Ticket**
Explanation
1.3.15 **UI: User Interface**
Explanation

1.3.16 Workflow

Explanation

1.4 Overview

The following document outlines the software requirements and specifications for The System, including the functional, nonfunctional, domain, hazard, and system requirements. Requirements are organized into sections based on their applicability to more general goals such as models, controllers, and individual webpages.

2 Overall Description

This section is intended to provide an overview of The System as a whole. It will explain the basic functionality of The System, as well as how its various components are expected to interact with each other. It will also provide some insight regarding potential users and stakeholders, and list the constraints and dependencies involved in the development and use of The System.

2.1 Product Perspective

TBA

2.2 Product Functions

TBA

2.3 User Characteristics

As a software project management product, The System will primarily be designed with the needs of three types of users in mind. These three user categories consist of software engineers, software testers, and project managers.

2.4 Constraints

2.4.1 Browser Support

Old or outdated browsers may not offer the proper support for prerequisite technologies to properly run The System. Given the variety of browsers that are either currently or previously available to the public, providing support for every version of every browser is simply not feasible.

2.4.2 Hardware

Much like the issues that can arise with antiquated browsers, older devices may not be able to meet the demands to adequately run The System due to hardware limitations.

2.4.3 Internet Connection

As a web-based product, restrictions on the user's access to the internet present constraints on The System. Without some form of internet access, the user will be unable to access The System.

2.5 Assumptions and Dependencies

2.5.1 The user's browser is reasonably modern and able to adequately support contemporary web-based applications.

2.5.1.1 The browser has been updated at least once within the past year.

2.5.1.2 The browser is able to support React components.

2.5.2 The user's browser is not actively blocking scripts that may interfere with the operation of The System via an extension.

2.5.3 The user's device is equipped with hardware to adequately support temporary software applications.

2.5.3.1 The device has a CPU containing at least two cores.

2.5.3.2 The device has at least 1GB of RAM.

2.5.4 The user has an internet connection stable enough to support basic REST operations within a reasonable amount of time.

2.5.4.1 A "reasonable amont of time" shall be defined as a time period less than or equal to 2 seconds.

3 User/Stakeholder Profiles

TBA

4 Core System Requirements

This section lists all of the core functional requirements pertaining to the system.

5 Backend Requirements

5.1 Models

5.1.1 The System shall verify the number of attributes present in each model as an additional safeguard to ensure every model is thoroughly tested in the event that attributes are added or removed.

5.1.2 The System shall include an Issue model.

5.1.2.1 The Issue model shall include an identifier obtained by concatenating the parent project's ID with the issue's generated ID, separated by a hyphen (e.g., SHRP-1 for Shrapnel's first issue).

5.1.2.1.1 The issue ID shall be immutable.

5.1.2.2 The Issue model shall include an `issue_type` attribute.

5.1.2.2.1 The attribute shall be of the enum type IssueType.

5.1.2.2.2 The attribute shall be required.

5.1.2.3 The Issue model shall include a `subject` attribute.

5.1.2.3.1 The attribute shall be of type string.
5.1.2.3.2 The attribute shall be required.
5.1.2.4 The Issue model shall include a `description` attribute.
5.1.2.4.1 The attribute shall be of type string (text for migration).
5.1.2.4.2 The attribute shall be optional.
5.1.2.5 The Issue model shall include an `issue_status` attribute
5.1.2.5.1 The attribute shall be of the enum type IssueStatus.
5.1.2.5.2 The attribute shall be required.
5.1.2.6 The Issue model shall include a `due_date` attribute.
5.1.2.6.1 The attribute shall be of type date.
5.1.2.6.2 The attribute shall be optional.
5.1.2.6.3 The attribute shall point to a future date relative to the time of the issue's creation.
5.1.2.7 The Issue model shall include an `assignee` attribute.
5.1.2.7.1 The attribute shall be of type string.
5.1.2.7.2 The attribute shall be optional.
5.1.2.8 The Issue model shall include the default `timestamps` attributes offered by Rails.

5.2 Controllers

5.2.1 The System shall include an Issue Controller.
5.2.1.1 The Issue Controller shall implement a `GET` endpoint.
5.2.1.1.1 The endpoint's request shall not require a request body.
5.2.1.1.2 The endpoint's response shall return a `200` status code when successful.
5.2.1.1.3 The endpoint's response shall return a `403` status code when the user is not authorized to access the project.
5.2.1.1.4 The endpoint's response shall return a `404` status code when the specified project ID is not found.
5.2.1.2 The Issue Controller shall implement a `GET:id` endpoint.
5.2.1.2.1 The endpoint's request shall not require a request body.
5.2.1.2.2 The endpoint's request shall include a URL containing an issue ID as an argument.
5.2.1.2.3 The endpoint's response shall return a `200` status code when successful.
5.2.1.2.4 The endpoint's response shall return a `403` status code when the user is not authorized to access the issue.
5.2.1.2.5 The endpoint's response shall return a `404` status code when the specified issue ID is not found.
5.2.1.3 The Issue Controller shall implement a `POST` endpoint.
5.2.1.3.1 The endpoint's request shall require a request body containing, at a minimum, the required attributes for the Issue model.
5.2.1.3.2 The endpoint's response shall return a `201` status code when successful.

5.2.1.3.3 The endpoint's response shall return a `400` status code when required attributes are not included in the request.

5.2.1.3.4 The endpoint's response shall return a `403` status code when the user is not authorized to create the issue.

5.2.1.3.5 The endpoint's response shall return a `409` status code when the issue ID already exists within the project.

5.2.1.4 The Issue Controller shall implement a `PATCH` endpoint.

5.2.1.4.1 The endpoint's request shall require a request body containing attributes to be updated.

5.2.1.4.2 The endpoint's request shall include a URL containing an issue ID as an argument.

5.2.1.4.3 The endpoint's response shall return a `202` status code when successful.

5.2.1.4.4 The endpoint's response shall return a `204` status code when the request body is empty.

5.2.1.4.5 The endpoint's response shall return a `403` status code when the user is not authorized to modify the issue.

5.2.1.4.6 The endpoint's response shall return a `404` status code when the specified issue ID is not found.

5.2.1.5 The Issue Controller shall implement a `DELETE` endpoint.

5.2.1.5.1 The endpoint's request shall not require a request body.

5.2.1.5.2 The endpoint's request shall include a URL containing an issue ID as an argument.

5.2.1.5.3 The endpoint's response shall return a `200` status code when successful.

5.2.1.5.4 The endpoint's response shall return a `403` status code when the user is not authorized to modify the issue.

5.2.1.5.5 The endpoint's response shall return a `404` status code when the specified issue ID is not found.

5.3 Enumerations

Note: Due to the absence of a built-in enum type in Ruby, the exact details of these implementations are not yet finalized.

5.3.1 The System shall include an IssueType enum to categorize issues.

5.3.1.1 The IssueType enum shall have a "Project" value.

5.3.1.2 The IssueType enum shall have an "Epic" value.

5.3.1.3 The IssueType enum shall have a "Story" value.

5.3.1.4 The IssueType enum shall have a "Task" value.

5.3.1.5 The IssueType enum shall have a "Subtask" value.

5.3.1.6 The IssueType enum shall have a "Defect" value.

5.3.1.7 The IssueType enum shall have a "Spike" value.

5.3.1.8 The IssueType enum shall have a "Release" value.

5.3.2 The System shall include an IssueStatus enum to differentiate issues by development phase.

5.3.2.1 The IssueStatus enum shall have a “Backlog” value.
5.3.2.2 The IssueStatus enum shall have an “In Progress” value.
5.3.2.3 The IssueStatus enum shall have an “In Review” value.
5.3.2.4 The IssueStatus enum shall have a “Testing” value.
5.3.2.5 The IssueStatus enum shall have a “Done” value.

6 Frontend Requirements

6.1 Issue View

6.1.1 The Issue View shall display all attributes of the specified issue.
6.1.2 The Issue View shall allow the user to assign an issue to any member of a project.
6.1.3 The Issue View shall allow the user to move the issue to the next phase of the project workflow.
6.1.4 The Issue View shall allow the user to mark an issue as blocked.

6.2 List View

6.2.1 The List View shall display a list of all issues contained within the backlog, including both closed and open issues.
6.2.2 The List View shall contain a search box allowing the user to filter issues displayed in the list.
6.2.2.1 The search box shall allow the user to filter issues by issue ID.
6.2.2.2 The search box shall allow the user to filter issues by subject.
6.2.2.3 The search box shall allow the user to filter issues by description.
6.2.2.4 The search box shall allow the user to filter issues by issue type.
6.2.2.5 The search box shall allow the user to filter issues by issue status.
6.2.2.6 The search box shall allow the user to filter issues by assignee.
6.2.2.7 The search box shall allow the user to filter issues by all of the above criteria at once.

6.3 Board View

6.3.1 The Board View shall display columns corresponding to each step of the project’s workflow.
6.3.1.1 Each column shall contain a number of cards corresponding to issues in the project’s backlog.
6.3.1.1.1 Each card shall display the corresponding issue’s ID.
6.3.1.1.2 Each card shall display the corresponding issue’s subject.
6.3.1.1.3 Each card shall display the corresponding issue’s assignee (if any).
6.3.1.1.4 Each card shall contain a small, circular, red button in the top-right corner to mark the corresponding issue as blocked.
6.3.1.2 Each column shall alter an issue’s status when its corresponding card is dragged and dropped into it from a different column.
6.3.2 The Board View shall have the option to organize issues by sprint ID.
6.3.2.1 Issues shall include an optional tag to represent a sprint ID.

6.3.2.2 The board shall present the option to only display issues included in a specific sprint.

6.4 Issue Editor

TBA

7 Accessibility Requirements

7.1 Navigation

7.1.1 Hyperlinks shall be visually distinguishable from all other text within the application.

7.1.1.1 Hyperlinks shall be emphasized with an underline.

7.1.1.2 Hyperlinks shall be the only form of text emphasized with an underline.

7.1.1.3 Hyperlinks shall be styled with a color that is not used by any other text within the application.

7.1.1.4 Hyperlinks shall be visually analyzed to ensure they are appropriately contrasted with their surroundings to allow a color vision impaired individual to differentiate them from other text within the application.

7.1.2 Webpages shall facilitate keyboard-only navigation.

7.1.2.1 The tab key shall provide the user with a logical navigation flow, generally from top-to-bottom and left-to-right on the page.

7.1.2.2 Navigation based on the tab key shall prioritize the most important or frequently used features so they are revealed more efficiently.

7.1.2.3 Modal windows shall navigate to the close button/link after the first tab key press.

8 Globalization Requirements

TBA

9 Nonfunctional Requirements

This section lists the nonfunctional requirements pertaining to the system.

9.1 Programming Languages, Tools, and Dependencies

9.1.1 The System shall consist of a backend implemented in Ruby.

9.1.1.1 The System shall use Rails as a web application framework.

9.1.1.2 The System shall use Bundler for dependency management.

9.1.1.3 The System shall use RSpec for testing purposes.

9.1.1.4 The System shall use RuboCop for linting purposes

9.1.2 The System shall consist of a frontend implemented in JavaScript.

9.1.2.1 The System shall use the React library for UI building.
9.1.2.2 The System shall use Webpack for dependency management.
9.1.2.3 The System shall use Jest/Enzyme for testing purposes.
9.1.2.4 The System shall use ESLint for linting purposes.
9.1.2.5 The System shall use TypeScript to enforce strong typing.

9.2 Regulatory

9.2.1 No regulatory requirements have been identified at this time.