WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: COMPUTER SCIENCE IN ENGLISH

BACHELOR THESIS

SUPERVISOR:
Conf. Dr. Cristina Mîndruță

GRADUATE:
Iliuță-Laurențiu Andoni

TIMIŞOARA
2023

WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
BACHELOR STUDY PROGRAM: COMPUTER SCIENCE IN ENGLISH

Application for defining and evaluating the accomplishment of Product Backlog Essentials practice activities

SUPERVISOR:
Conf. Dr. Cristina Mîndruță

GRADUATE:
Iliuță-Laurențiu Andoni

TIMIŞOARA
2023

Abstract

In the past decades, the need for a theory in Software Engineering emerged as a crisis. Hundreds of books have been published in a try to solve the situation. Some of them have been prosperous and motivated individuals, but they eventually fade into an immutable collection of practices.
The Essence standard aims to solve this problem by providing a meta-method. This approach delivers a language and a kernel for describing and creating practices and methods. While other approaches lack the support for designing and overviewing the Product Backlog, Product Backlog in Action is conceived to encapsulate the Product Backlog Essentials components while offering a pleasing user experience. We seek to offer an assistant to team leads and team members that will help them define and check the progress of Product Backlog Essentials practice activities.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Since the debut of the term software engineer, individuals have been striving to come up with an adequate theory. The challenge of addressing literature for this engineering discipline comes up from the diversity and complexity of software systems. Such a theory should be capable of defining the process of planning, designing, building, validating, verifying, and maintaining software systems. This run for finding the substantial theory led to a fragmented collection of methods and practices, something that Ivar Jacobson described as the so-called "software crisis"([1]).

1.2 Problem Statement

The Essence standard arises to offer modularity, thus freeing individuals from following immutable practices and methods. Being a meta-method, it can bridge gaps between uncompleted methods and appropriate practices, which can help successfully achieve future stages of a software system. The Essence language and kernel are used to describe practices. Further, essentialized practices can be combined in methods in a flexible way, adapted to the specific software project development needs. One of the essentialized practices is Product Backlog Essentials ([2]). Having its root in Agile Methodologies, the Product Backlog is the core component that defines a software system. However, more is needed in managing this artifact. Creating a Product Backlog is a complex process.

The primary approach to generating items for the Product Backlog is surveying users to find their needs. This strategy does not solve the lack of theory; it only moves the assignment on the user's shoulder. Sedano, Todd, and Ralph ([3]) published a paper addressing this problem. The paper reveals the need for more research to address the creation and maintenance of the Product Backlog. Moreover, the authors raised the fact that despite having its root in Agile Methodologies, the literature does not assist with the evolution of the Product Backlog.

1.3 Thesis Scope and Objectives

This thesis aims to assist teams in developing software products according to essentialized Product Backlog Practice. Providing an overview of the items in the backlog, prioritization of issues, and roadmaps for advancing the Product Backlog and each item, the application serves as the bridge between Agile Methodologies and the previous missing theory of Product Backlog development. By providing a translation from the Product Backlog Essentials into a concrete application, the creation of a Product Backlog is not only assured, but the quality of it is also measurable. Providing constant feedback on each issue by comparing it against a roadmap is a quick health check for each item.

The rest of the thesis is structured as follows:

- Chapter 2 presents an introduction to the Essence Standard introduced by [1]. Introducing the core concept will offer access to the underlying mechanisms that allow the creation of practices and methods in a method-agnostic form.
- In Chapter 3, Product Backlog Essentials is analyzed in depth to provide a better context for building upon our solution. Chapter 3 presents the information needed to understand the technical requirements and the outlined goal of this thesis.
- Chapter 4 contains all the technical information needed to understand the architectural aspect, requirements, identified use cases, and user interactions with the system. This chapter will introduce the graphical user interface with associated descriptions for each screen present in the application.
- Chapter 5 lists an analysis of the related work to understand the solution proposed in the context of other existing solutions, thus identifying both weak and strong points of the presented solution.
- Chapter 6 concludes the thesis with a summary and final thoughts. Considering the previous chapter, the proposal for feature work is a starting point for overall improvements.

Chapter 2

The Essence Standard

In this chapter, we will look at what Essence is and what it is supposed to accomplish. The Essence standard's basics will be first exposed, along with clarifying the goals that it aims to achieve. We will proceed to understand the components of Essence: the Essence Language and the Essence Kernel. The creation of new methods using already existing practices from already existing methods will demonstrate the modularity of the standard assumed by The Object Management Group as a help for understanding the Essence Standard as a thinking framework rather than a robust methodology.

2.1 Introduction to Essence

The discipline of software engineering raised a need for quality standards. Crafting software systems needs to address these quality standards to provide reliable systems. Many individuals tried to address the need for more guidance when it comes to what reliable systems are. Creation of methodologies inspired engineers to answer the beforehand mentioned standards. A methodology is composed of different practices designed to address the specific activities in the software development process. Methodologies populated the world of software engineering, a few of which provide techniques to merge practices.

Essence is a methodology-agnostic standard designed to bridge the gap between practices and methodologies. Creating a common ground through the Essence Kernel, Essence becomes a platform to describe methods, a common ground focused on the essentials. Having a simple and intuitive visual language, and a concise kernel focused on the things to work with, the things to do, and the competencies practitioners need when developing software, Essence becomes an easy-to-learn and easy-to-use standard.

2.2 The Essence Language and Kernel

A standard aims to be logical to users. One of the two components of the Essence standard, namely the Essence Language, has an important role in achieving this. Introducing visual elements and the relationship between them helps teams to have a better understanding of the context. Every component in the essence language plays a role in describing various aspects of the development process. We use the Essence language to describe only the essentials of practice.

Classification of the terminology falls into two main categories: things practitioners have to do and things practitioners have to work with. For the first category, we have the Activity element of the language. The introduction of the Activity element of the language outlines one or more techniques to accomplish a task and further relates to elements in other categories by recommending approaches to carry out the work. On the other hand, Alphas and Work Products are things to work with when developing software. For example, we can consider artifacts like documentation or the Product Backlog itself to be represented by the Work Product, and we can expect the issues of a Product Backlog to fall under the Alphas elements. Terminology present in the language only highlights the essentials of practice. For example, the Alphas are considered the core elements, just like the Product Backlog Item is the core element of the Product Backlog.

When addressing the elements in the language and the relationship created upon them, we relate to one diagram, as visualized in Figure 2.1, introduced in [1] that illustrates the core idea of the Essence standard. As mentioned before, the Essence standard highlight the essentials used for creating practices further combined to introduce new methods. Besides being intrinsic elements, describing the practice components introduces the other five core elements.

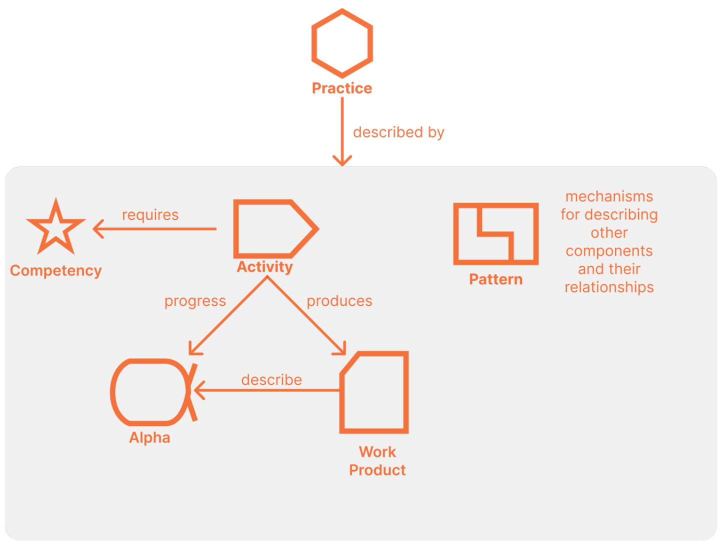


Figure 2.1: The Revised Description of Practice using The Essence Language (source: [1])

Starting with the top, the Activity component, used to propose techniques for things practitioners have to work with, produces the Work Product and progresses the Alpha element. After being assembled, the Work Product, which represents an artifact like documentation or part of the software, describes the Alpha element. To complete different Activities, one must have different competencies: development competencies for implementation purposes, testing competencies for scripting tests, and others. The Pattern element assists in further introducing complex mechanisms for describing other components and their relationships relevant to the context of practices.

Furthermore, the Essence Standard also introduces a common ground for practices to interact with each other in the context of a methodology, namely the Essence Kernel. Seven Alpha elements comprise the kernel, as described in Figure 2.2 introduced in [1], along with Activity Spaces and Competencies. The seven Alphas aim to tackle all the critical aspects of software development: customer, solution, and endeavor. One can link practices regardless of their original methodology by associating each practice within a methodology with one of these Alphas.

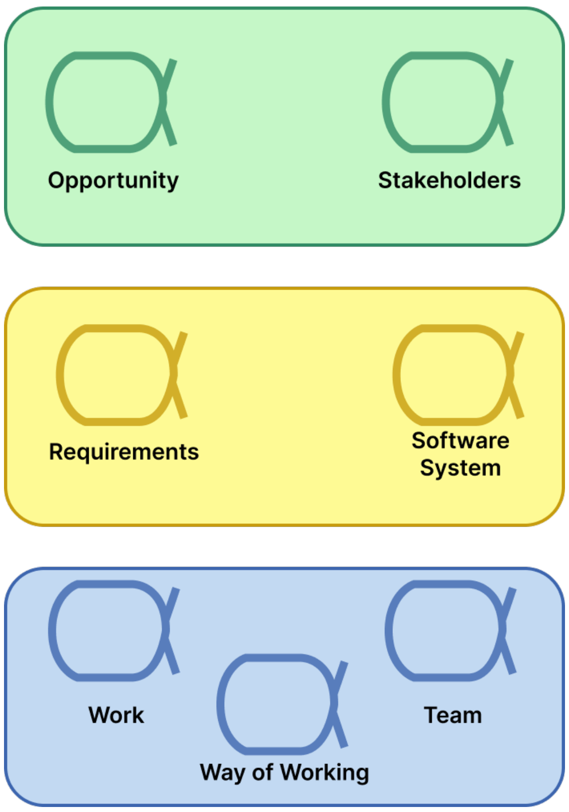


Figure 2.2: The Essence Kernel
(source: [1])

2.3 Practices and Methods

The Essence Standard provides a way to bridge the gaps between methods. Using the elements introduced before, teams can find a way to organize their work with a suitable collection of practices. Methods are built upon a foundation of practices. Practices build upon the kernel compose elements from the Essence Language. A method is a collection of practices. Having practices connected to the Alphas of the Kernel, the common ground, enables teams to create new methods that suit their work the best.

For teams to develop new methods, they can choose one or more practices in the library of practices, like the Product Backlog Essentials, can create a new practice respecting the essence standard or choose another practice that is not present in the library of practices and essentialize it. To create a new method, the team must choose all the practices found to be relevant and make the necessary connection to the kernel. Refer to Figure 2.3 inspired from [1] to visualize this process to get a clearer picture.

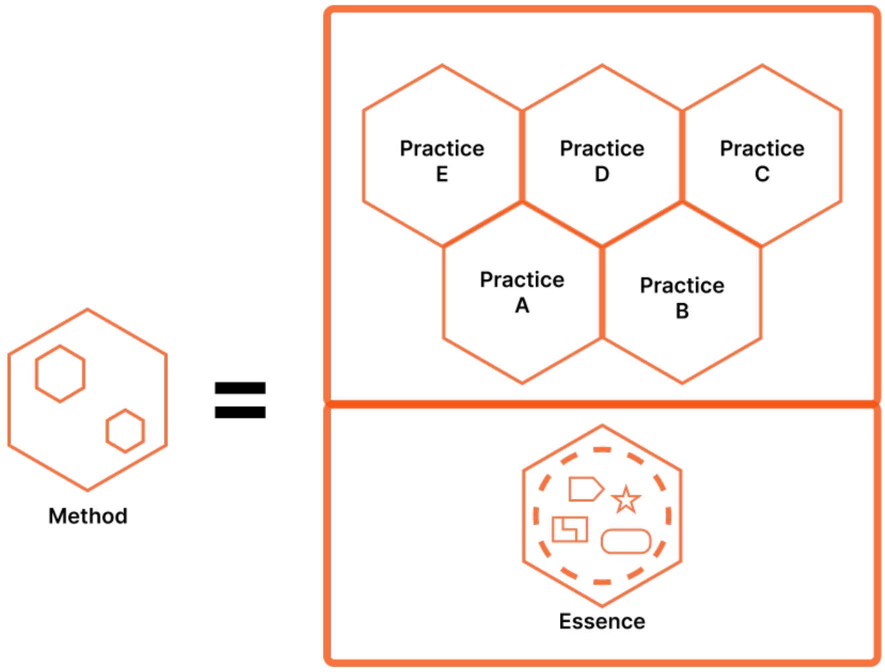


Figure 2.3: Revised Creation of Methods
(source: [1])

Chapter 3

Product Backlog Essentials

This chapter introduces the Product Backlog Essentials from the library of essentialized practices to understand the solution's context better.
In the first part, the main focus will be to explain the relevance and the quality that Product Backlog Essentials provide over the solution. Furthermore, the second part of the chapter introduces each element in the Product Backlog Essentials and the relation between elements.

3.1 Importance of Product Backlog

Schwaber and Sutherland ([4]) clearly explain the elements present in the Scrum Methodology, from the roles in Scrum teams to the events happening during Scrum and artifacts found in a Scrum environment.

Having its roots in Agile Methodologies, the Product Backlog is an artifact in the form of a list containing items responsible for building value in a product. The list must be visible and understood by both the team and stakeholders.
As described by [5], the Product Backlog can contain product features, bug fixes, changes in infrastructure, or any other task that a team considers relevant for delivering quality. Furthermore, the Product Backlog should be a flexible tool for providing value. The content of the Product Backlog should be easily manageable such that teams can effortlessly add new items or eliminate ones that do not provide direct value. The Product Backlog is dynamic and in a permanent evolution.

Owning the Product Backlog, the team typically imposes various conventions on the Product Backlog. One of the most common ones is associating relevant information to each item, such as the title, description, and approximation of the amount of work needed to achieve the value, usually measured in story points. Another commonly used convention is the creation of the definition of done. The definition of done specifies the widely agreed quality standards an item needs to meet to consider the item fully achieved.

It is important to note that the Product Backlog differs from the requirements document. Having an item in the Product Backlog does not necessarily guarantee its inclusion in the final product. The Product Backlog is an essential artifact for the teams using Agile Methodologies. However, there is more guidance needed to have a quality Product Backlog. Although present, both the scrum guide ([4]) and the introduction to the Product Backlog definitions ([5]) do not guide the creation and maintenance of the artifact.

[3] identify the need for more guidance for achieving value in the Product Backlog. The authors specify the need for more research in building a valuable Product Backlog. As stated in the paper, methodologies before Agile have "requirements elicitation," a common pitfall for teams when defining the artifact. The requirement elicitation usually includes potential users crafting system requirements using surveys. The team thoroughly reviews and refines the requirements until they achieve a universally accepted system description. However, this approach does not solve the existing problem but instead moves the effort of designing a system onto the potential users. Relying on users to design the product does not provide an effective and measurable way of providing value. Instead, it raises confusion generated by personal beliefs, as the user will address value according to their perception. Later refinements of the Product Backlog are not quantified and measured either, leading to unclear and incomplete artifacts. There are no guidelines over how the items must be further refined and how to measure when the item has reached the end of refinement.

Considering these problems, the concrete solution is creating and managing the Product Backlog as intended. Product Backlog Essentials capture everything needed to evaluate and improve one team's Product Backlog. Having elements that describe only the essential Activities and components of a Product Backlog keeps the process simple while offering a way to measure value. This essentialized practice provides a complete overview of the Product Backlog, allowing the team to gain complete control over the creation and management of this vital artifact.

3.2 Introducing the elements

This section examines the elements present in the Product Backlog Essentials practice as presented in [2]. Components of the Product Backlog Essentials follow the Essence Standard, described using the Essence Language features as illustrated in Figure 3.1.

The practice contains three Activity elements that produce three Work Products. One Alpha element, described by one of the Work Products, is also present in the representation of the practice according to the Essence Standard. Furthermore, three different Pattern cards outline complex concepts that require consideration when developing the Product Backlog.

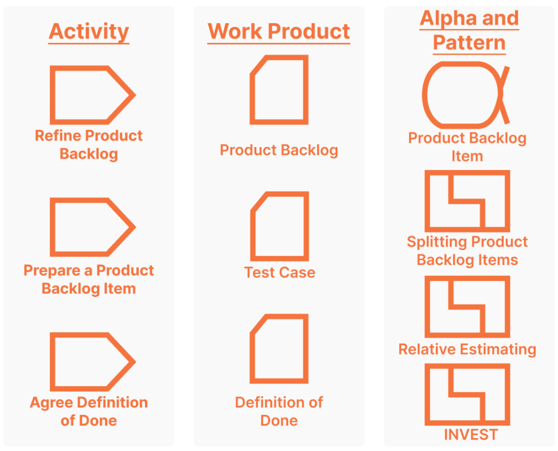


Figure 3.1: Product Backlog Essentials Items

The Product Backlog Item is the most crucial part of the Product Backlog Essentials and the Product Backlog. It is an entry that holds information about the product. Additionally, it has states that indicate the current condition of the product.

The three Activities are: "Refine Product Backlog," "Agree Definition of Done," and "Prepare a Product Backlog Item." The first Activity refers to the main qualities of the Product Backlog, dictating the need to have the artifact well prioritized, visible, and updated to the current version of the product. "Prepare a Product Backlog Item" assures an item is ready for development, underlining testing to validate quality standards. The last Activity ensures the existence and accomplishments of quality criteria that will measure the successful integration of items in the product.

Each Activity produces a Work Product, resulting in three Work Products: The Product Backlog, the Definition of Done, and The Test Case. The first represents the Product Backlog, containing all the items produced. The Product Backlog component also has internal states that provide feedback on the health of the Product Backlog and steps for advancing it. The Definition of Done is usually a piece of documentation listing all the quality criteria. Similarly to the Product Backlog Work Product, the user can analyze its state and get suggestions for the next steps. The third Work Product contains inputs and expected results of tests run by the team to ensure desired functionality.

3.3 Progressing the elements

The elements in the Product Backlog Essentials correlate to one another through well-defined relationships. The general outline is that Activities produce Work Products and progress Alphas; Work Products describe Alphas from the practice or the Kernel, and Patterns guide complex mechanisms found in the presented practice.

The central Alpha - Product Backlog Item, is progressed using ”Refine Product Backlog” and ”Prepare A Product Backlog Item.” The two Activities ensure acquiring both the ”Identified” and ”Ready for Development” states of the Alpha item. The two Activities also produce two Work Products, the Test Case for defining the Product Backlog Item and the refinement of the Product Backlog, which consists of multiple Product Backlog Items. This practice focuses on system specifications, all Work Products describing the Requirements Alpha from the kernel.
The third practice Activity outlines the quality criteria for determining the completeness achieved in integrating a change into the product. Optimizing the outcome’s value involves using the ”INVEST” Pattern. The Definition of Done is a result of this Activity. This practice focuses on system specifications, all Work Products describing the Requirements Alpha from the kernel. To better understand how the elements connect, refer to Figure 3.2.

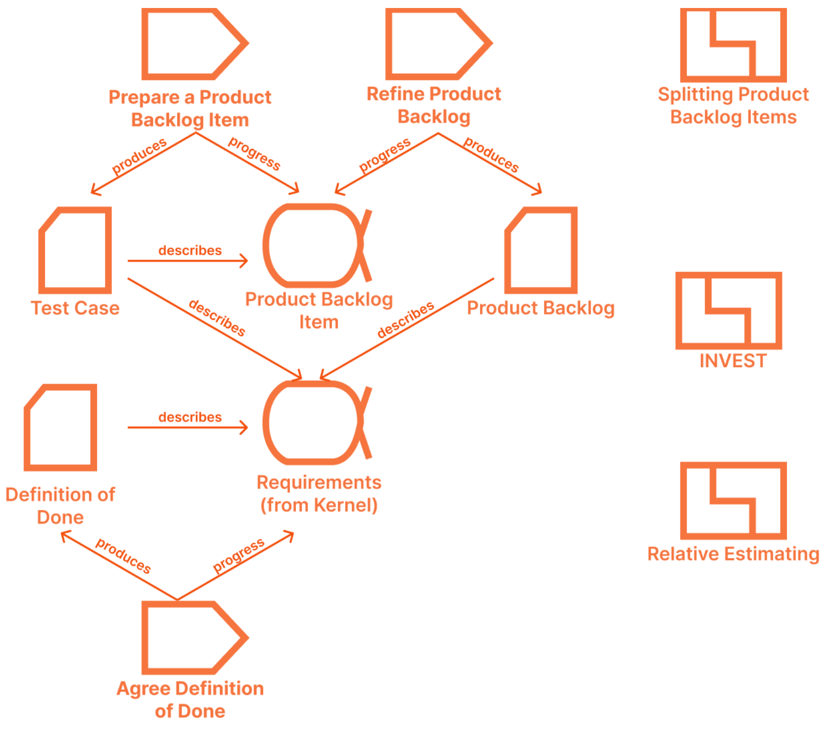


Figure 3.2: Product Backlog Essentials Map

Chapter 4

Product Backlog in Action

The primary objective of this chapter is to provide a comprehensive overview and detailed exposition of the architectural design, application utilization, and graphical user interface of the proposed solution. By elaborating on these critical aspects, we aim to understand the proposed solution's inner workings and core functionalities.

4.1 Author Contribution

The primary focus of the solution is to help teams get a good grasp on the evolution and performance of their Product Backlog. Researching the creation and evolution of the Product Backlog has revealed potential for future improvements over this artifact's guidelines. Conductive research on the Essence Standard and the practices provided by Dr. Ivar Jacobson revealed a solution to the problems showcased in Section 3.1.

Visual equivalence of Product Backlog Essentials

First, an analysis took place to explore how to introduce the critical components of the Product Backlog Essentials as visual elements that enable easy navigation of features. The analysis revealed meaningful connections and dependencies between the elements, which helped to construct a visually appealing and well-organized design that effectively combines all elements.

Modular Data Modeling

Secondly, by capturing data flowing within the application through comprehensive and easily understandable models, it became possible to design relationships between different elements in the system. This design establishes logical connections and associations among components, enabling smooth and straightforward interactions. The modularity brought by the models enables the team to conduct different ways of workings suitable for their needs. For example, the team can choose to have one comprehensive Product Backlog, divide their Product Backlog into multiple ones: one for the frontend development and another for the backend development, or even go out of scope and evaluate and manage items of a period, for example, sprints.

Progression system and Visual support

Furthermore, exploring relationships between components established a system that facilitates the state's progression. The progression of the elements is logical and visually improved, providing tooltips for the reason of unreachable states and arrows showing the direction of the progress. Additionally, the system includes precise classification of states for different alpha items through tables, serving as visual aids.

4.2 Requirements

The following section provides a list of functional requirements fulfilled by the application, along with a brief explanation. A more detailed explanation, supported by visual resources, will be available in the Graphical User Interface section.
When developing the application, the teams were the main actors considered. Therefore, all requirements are an easy-to-use way for teams to develop and maintain the Product Backlog. It is crucial to keep in mind that the Product Backlog in Action does not aim to replace Product Backlog-centered management tools, similar to the ones present in [6], [7] or [8], but rather to accompany those as a tool for measuring value and provide feedback about the health of the Product Backlog. The following subsection will enumerate the functional requirements of the solution.

Project Authentication and Authorization

The application must provide a registration and login feature for projects. Teams may register a new Product Backlog reflecting their Product Backlog using a unique name and a password. After registering their project, they can log in by providing the same credentials provided in the first place. The authentication does not serve only for separating the Product Backlogs of multiple projects but also provides authorization. Each project is authorized using JWT tokens, keeping project-sensitive data protected. All routes are protected using tokens and are not accessible without providing such a token, thus providing security for each project. Furthermore, switching between projects is possible using the logout button available on the main page.

Visualization and Management of the Product Backlog

Visualization and management of the Product Backlog must be easy to use. The refinement of the artifact should be possible as described in the Product Backlog Essentials practice ([2]). Teams can add a new Product Backlog item, delete existing ones or reorder items already present in the Product Backlog using drag-and-drop features. Furthermore, the states of the Work Product representing the Product Backlog are easy to access. These should be carefully advanced. To ensure a valuable Product Backlog progression, users may refer to "Relative Estimating" and "Splitting Product Backlog" Patterns, which are responsible for explaining complex mechanisms that teams must consider for this Activity.

Eachw item has three states: ”Identified,” ”Ready for Development,” and ”Done.” Teams should progress the states as described in the Essence Standard, that is, by checking items present in the checklist of each state. After achieving a state, the items automatically progress to that particular state. Furthermore, the system actively prevents items from being considered ”Done” and progressing to that state unless they fulfill all the required criteria. The system will actively disable the state and present tooltips that explain the essential steps needed to unlock the state.

Previewing Product Backlog Essentials

The Product Backlog Essentials is the core of the Product Backlog in Action. Providing only the essentials of the Product Backlog practice allow teams to focus on what is important. Considering this, the elements present in Product Backlog Essentials and the information provided should be visible. The elements are visible in the shape of poker-like cards, a format also suggested by the Essence Standard ([1]).

Alpha items roadmap and progressing the state

Users can access feedback for each item in the Product Backlog in the form of a roadmap. The roadmap for each item contains the alpha item and its states along with ”Prepare a Product Backlog Item” and ”Accept the Definition of Done” and their respective Work Products. Teams must complete the states of each Work Product before starting to work on finalizing the alpha item. Attempts to mark the item as done before will result in illegal states. The system treats this scenario by disabling the item's final state until all teams meet all required quality criteria. The system addresses possible confusion over the disabled state by providing user-friendly tooltips indicating the required actions to complete the state. Additionally, teams must consult the ”INVEST” Pattern to achieve the most value. Progressing the alpha item through associated states will move the item to the associated tables for a better overview of the size and remaining work over the Product Backlog.

4.3 Architecture and Technologies used

This section aims to present a comprehensive overview of the application's architecture and present the technologies engaged in its development. Moreover, this section will justify choosing technologies and defining mechanisms that require the usage.

Architecture

In terms of architecture, the Product Backlog assistant will be accessible as a web application. Considering this, the application provides two different services, one modeling the front end, responsible for visual aspects and application logic, and another providing mutation to the database as a RESTful API. Moreover, the application implements the Model-View-Controller architectural pattern. As for the Model-View-Controller architecture, the application relies on Mongo Database for creating the models. Once the creation is successful, controllers will handle all the necessary operations on those models. Node runtime environment and the Express framework support controller creation. Node runtime environment allows running Javascript code on the server side. The Express framework provides methods for handling HTTP requests. Data retrieval, verification and validation are the main functionalities of the controllers.

Furthermore, the view element of the Model-View-Controller architecture, implemented through React and various other libraries, fetches available models through HTTP requests. Data provided by the models are mapped to relevant entities allowing the mutation process over the models. Mapped entities visually represent the current data state, allowing visual updates to users. Controllers receive modifications enabled through user input on the view, updating the models accordingly. For a better and more optimized experience, some business logic is present on the view part of the Model-View-Controller, thus avoiding and limiting unnecessary network requests, enabling a better user experience.

Backend Architecture

The core element of the backend service is the Node runtime environment which allows running Javascript code on the server side. Furthermore, the implementation uses the Typescript programming language, a superset of the Javascript programming language, providing types safety. We will use a non-relational database to shape the models introduced in section 4.2 and methods for operating the data to implement the backend service. Choosing Mongo Database as the non-relational database allows effortless modification over time, with data accessible as documents. Furthermore, despite the database's non-relational nature, the application can establish relationships between entities by utilizing unique identifiers.

This approach allows for establishing connections and associations between different data elements, providing relational functionality even within a non-relational database environment. For example, current models associate the project's unique identifier with items specific to that project.

Models will be accessible for retrieval and mutation using a routing system constructed with the Express framework, which allows request handling, and the Mongoose library, a bridge between the server and database through schema modeling, that will provide the user control over the entries stored in the database. The RESTful API architecture enables handling data accordingly to the HTTP methods provided. Thus, the RESTful architecture is easy to use, providing meaningful ways of interacting with the database. Furthermore, all requests on the application's server side will request authorized requests, thus providing better security over the information provided in each project. The application implements the authorization protocol using the JSON Web Token standard, enabling unique token generation and token validation to authenticate projects.

Frontend Architecture

The core element of the frontend architecture is the React library, powered by Vite as the module bundler and Typescript programming language, which provide type safety. The implementation respect the Single-Responsibility design principle organizing the website's structure in well-isolated components. Furthermore, Zustand Library resolves the state management allowing the reduction of passing down data through multiple components. This approach allows the implementation of reusable components, facilitating the capacity to reuse code for implementing new elements.

The React Query library helps fetch the data from the backend side of the application. Offering modern mechanisms like query invalidation when the user modifies data, error handling in a modern manner, or flagging the data as stale after a period provides a better user experience and fast feedback for user actions.

Reordering and prioritizing the items in the Product Backlog is realized using drag-and-drop actions. The React-beautiful-dnd provides a layer of abstraction for easy implementation of this feature. Applying this layer to each table offers independent reordering among the tables, avoiding artificial state progression, for example, forcefully dragging an item from the "Identified" table to the "Ready for Development" table.

Part of the business logic is integrated into the client side to avoid unnecessary network exchanges that might result in slower performance. The business logic migrated includes features like state completions of various elements, computing the ordering of the issues after drag-and-drop actions, and disabling states until elements meet the criteria quality. Actions taken on the front end are further communicated to the backend to update models accordingly.

Moreover, the front end offers error handling through custom components indicating the reason for the errors in a human-readable way. The login and register pages validate data through fallback errors, pointing to faulty credentials or existing projects in the database.

4.4 Use of the application

This section's focus is further exploring the capabilities of Product Backlog in Action. Exploring standard scenarios and providing explanations of internal mechanisms supported by system diagrams will enable a better understanding of the solution and architectural choices.

Main features of Product Backlog in Action

Each team should be able to have personalized content based on project registration, and they should be able to interact with the items they propose by proposing new priorities and new orders among them. Teams can create new Product Backlog Items, visualize existing ones, retrieve health status feedback by using the alpha item roadmaps or by checking the states of the Product Backlog available on the main page, and propose new ones to explore. Teams contribute to the evolution of the items in the Product Backlog as described in the Product Backlog Essentials, using checklists and progressing state of individual items described by the essentialized practice. Additionally, team members can always refer to the information provided by Product Backlog Essentials and check the current states of elements by accessing the card states. Condensing all these described features and scenarios concluded in a simplified use case diagram illustrated in Figure 4.1

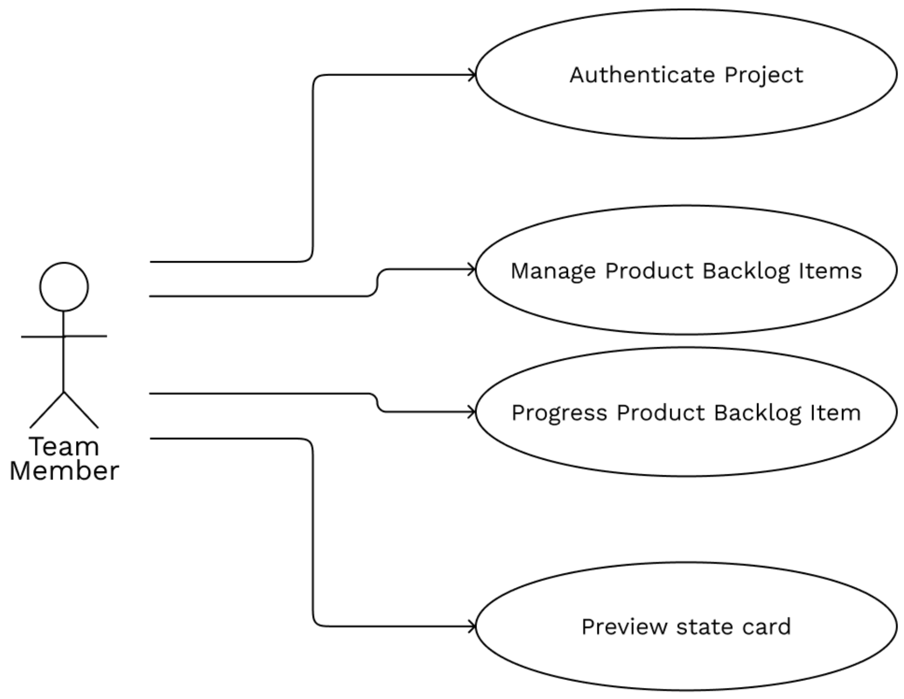


Figure 4.1: Use Case Diagram

Manage Product Backlog Items

The user has numerous features within Product Backlog in Action. The diagram below (Figure 4.2) describes a basic flow of managing the items. After entering the project's credentials, the application's routing system will redirect the user to the Product Backlog Project Board. The Project Board is the starting point for most scenarios users will encounter. One typical application usage is to add a new Product Backlog Item. Adding a new Product Backlog Item is the users' first action when creating a new Product Backlog. Adding a new item is part of the refinement Activity. Product Backlog in Action showcases the adding functionality by providing a pictogram combining the Activity element proposed in the Essence Language and a plus sign labelled "Add Product Backlog Item." Users should provide the name, description and number of story points about the new issues. Once filled, a new entry containing provided information will add to the database after being validated, and the Project Board will update accordingly. The team can start prioritizing the issues as they consider them.

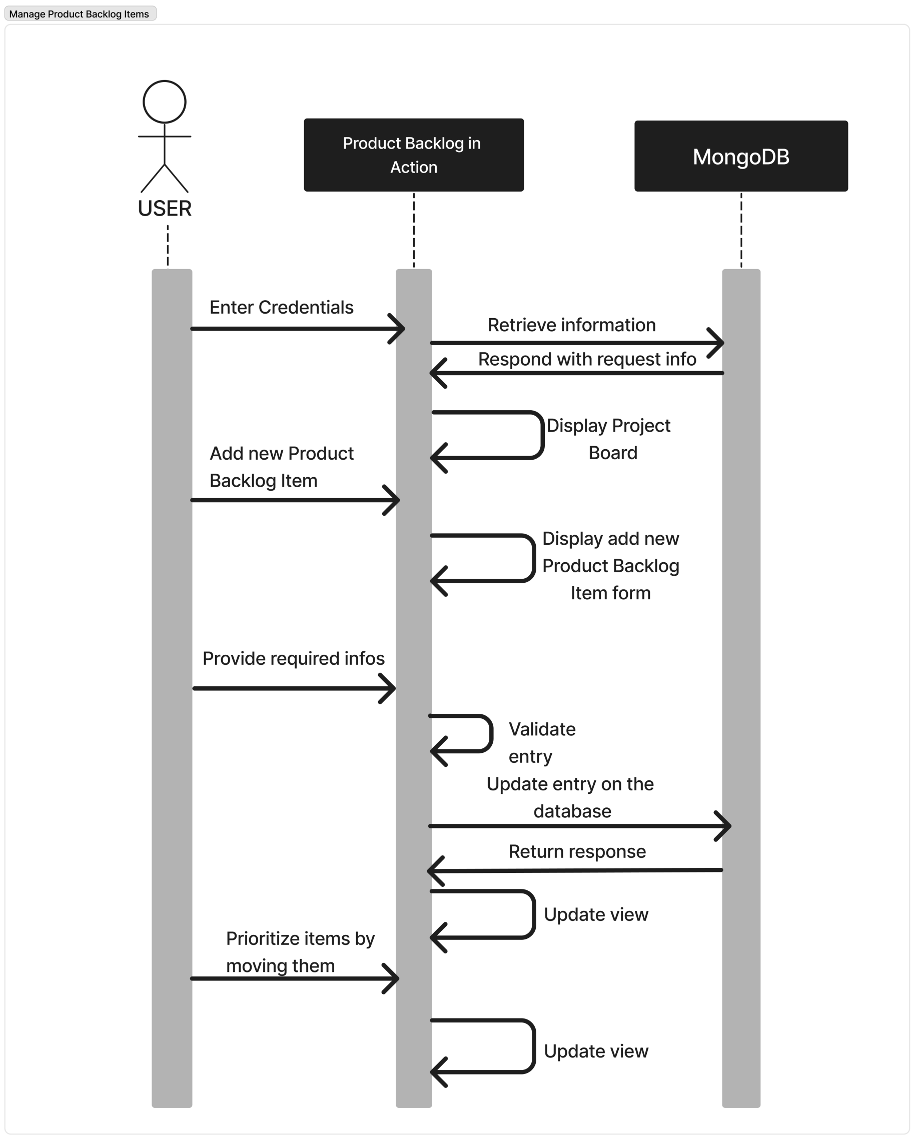

Manage Product Backlog Items
USER
Product Backlog in Action
MongoDB
Enter Credentials
Retrieve information
Respond with request info
Display Project Board
Add new Product Backlog Item
Display add new Product Backlog Item form
Provide required infos
Validate entry
Update entry on the database
Return response
Update view
Prioritize items by moving them
Update view

Figure 4.2: Managing Product Backlog Flow

Progress Product Backlog Item

After adding a new item and reorganizing the Product Backlog, the next step is progressing an existing Product Backlog Item. Double-clicking an item will open the item manager. The item manager consists of the alpha item roadmap, which reunites the two Activity element: ”Prepare a Product Backlog Item” and ”Agree Definition of Done”, their resulting Work Products: ”Test Case” and ”Definition of Done,” the alpha element, ”Product Backlog Item” and the ”INVEST” Pattern, as described in subsection 4.2. The managing interface will allow users to progress the alpha's state by previewing the states' associated cards and ticking the necessary checkboxes. The view updates accordingly once the state validation is over.

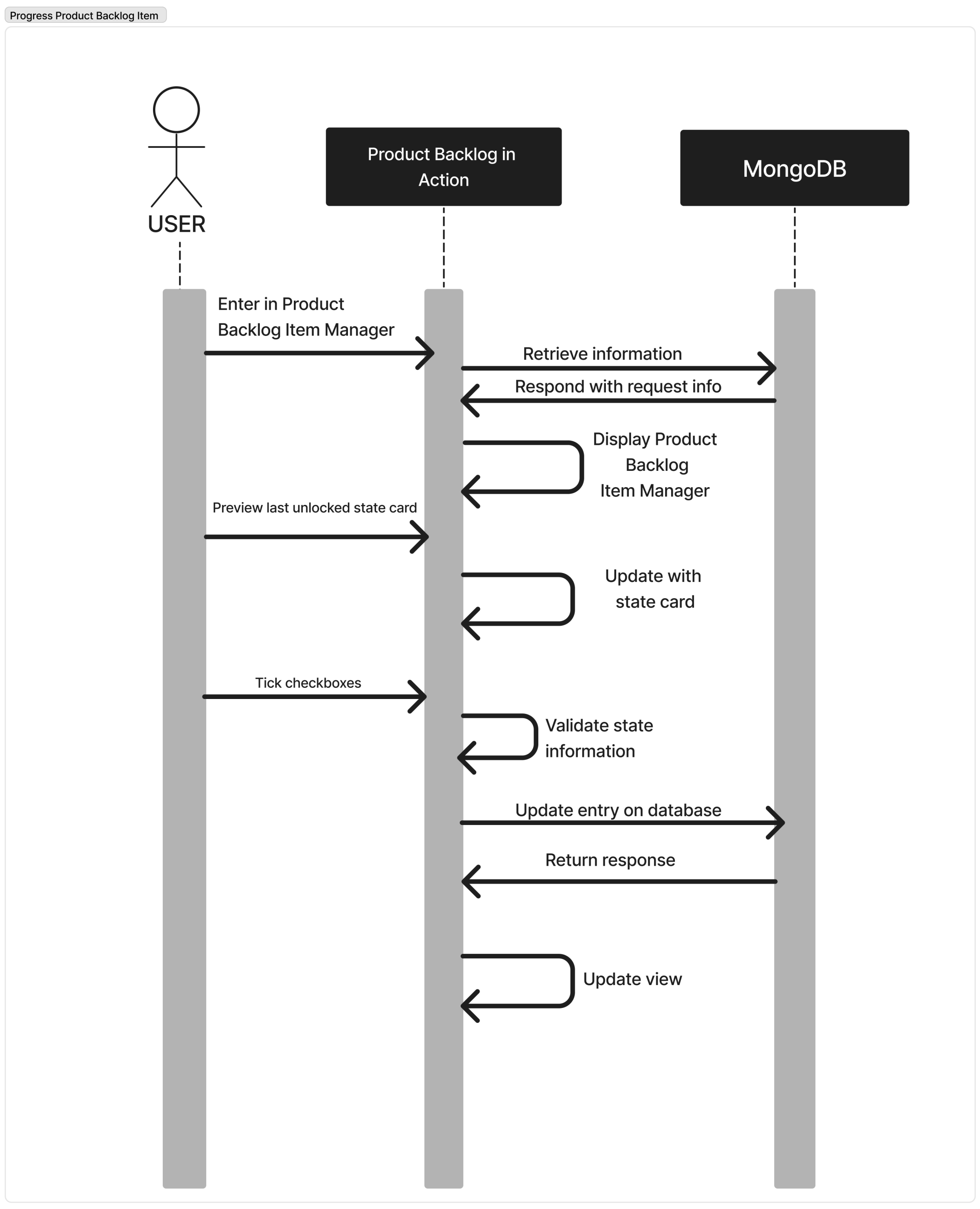

Progress Product Backlog Item
USER
Product Backlog in Action
MongoDB
Enter in Product Backlog Item Manager
Retrieve information
Respond with request info
Display Product Backlog Item Manager
Preview last unlocked state card
Update with state card
Tick checkboxes
Validate state information
Update entry on database
Return response
Update view

Figure 4.3: Progress Product Backlog Item

4.5 Application Testing

As mentioned in previous sections, a part of the business logic is present on the frontend part of the application. This migration of the business logic has emerged the need for testing on the frontend part of the application. The testing aspect of the application uses the Vitest library and react-testing library. Using the two libraries enables the creation of a virtual DOM, simulating the actual DOM and performing various actions to ensure the frontend's correct functioning.

The tests are available as unit tests, ensuring the individual functionality of components on the view part. Testing is done by rendering components with mocked information and performing various actions to test various functional requirements. Furthermore, tests also provide mocked services to avoid unnecessary calls to the backend part.

Unit tests offer multiple advantages. Some of them include better maintainability of the codebase by automatically testing important components after adding new pieces of software, reducing code complexity by having tests describing expected flows of the code, and based on the same principle for reducing code complexity, unit tests also serve as documentation for developers working with already tested components.

4.6 Graphical User Interface

This section will present the different graphical user interfaces available in the application. These interfaces will provide a view of the essential processes in the application and help the users navigate the Product Backlog in Action interface. A visual analysis of the standard scenarios mentioned in the section 4.4 is provided. This section aims to act as a user manual, providing relevant information about the interfaces and their associated functionalities.

The application's Graphical User Interface intends to offer a good user experience by logically correlating the elements in the Product Backlog Essentials. Relations between elements are offered through pleasant interaction and easy-to-understand assumptions, for example, the states of the Product Backlog Work Product being available on the main page, illustrating the connection with all the Product Backlog Item alphas and with the requirements alpha available in the kernel. Additionally, the application uses the same pictograms present in Chapter 2 ([1]), depicting the relevance of the Product Backlog Essentials in defining the solution. Logical relationships between different elements are illustrated by connecting the associated pictograms with lines, suggesting the logical order of phases and connections between the addressed elements.

Login Page

The login page is the first interface any user will encounter (Figure [4.4]). The login page serves as an entry point for the application. The authorization and authentication module is a direct reason for the integration of this page. This interface consists of a relevant logo illustrating the use of [2] and input fields depicting credentials the user must provide for a successful login. The login page has error handling mechanisms, offering relevant information, such as invalid credentials messages, enabling a better user experience.

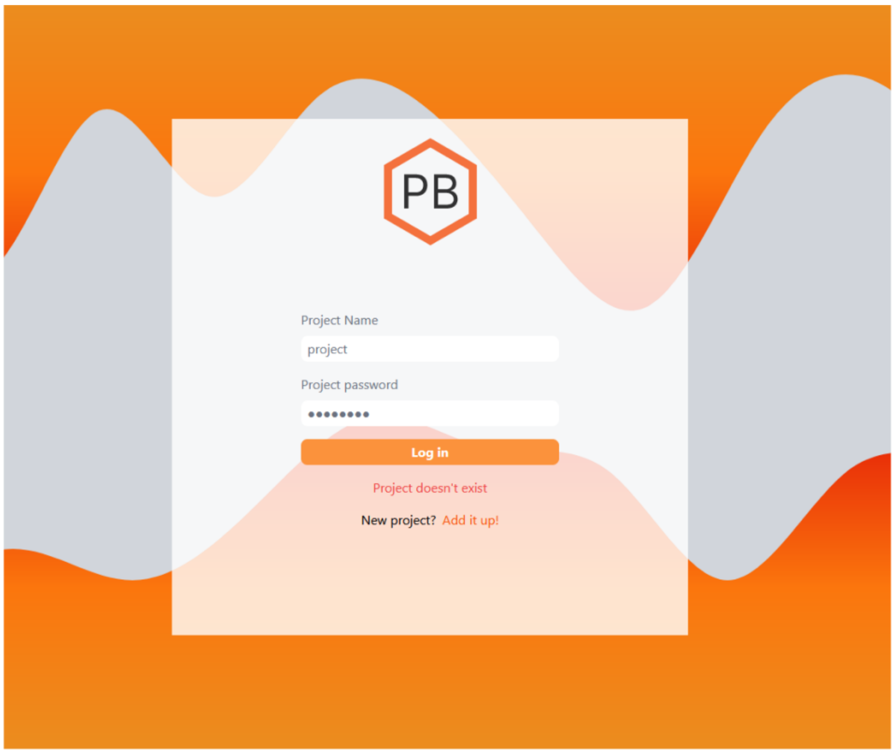


Figure 4.4: Login Page

Moreover, teams can add new projects using the "Add it up!" button at the bottom of the login box. Clicking the register button will navigate the user to a similar page where new projects can be registered (Figure 4.5). Similar to the login page, the registering page also provides error handling mechanisms, able to provide feedback over the validation of data provided in a human-readable way. Additionally, the registration page offers users modularity in organizing the projects. They may choose to register the whole Product Backlog or break it down into several smaller Product Backlogs, like one for the frontend part and another for the backend.

Additionally, if the product support multiple released versions, they can choose to register each version. Another way to work with the issues is to divide projects into several sprints. Dividing the project into multiple sprints will offer a good overview of the team's velocity and efficiently offer sprint segregation.

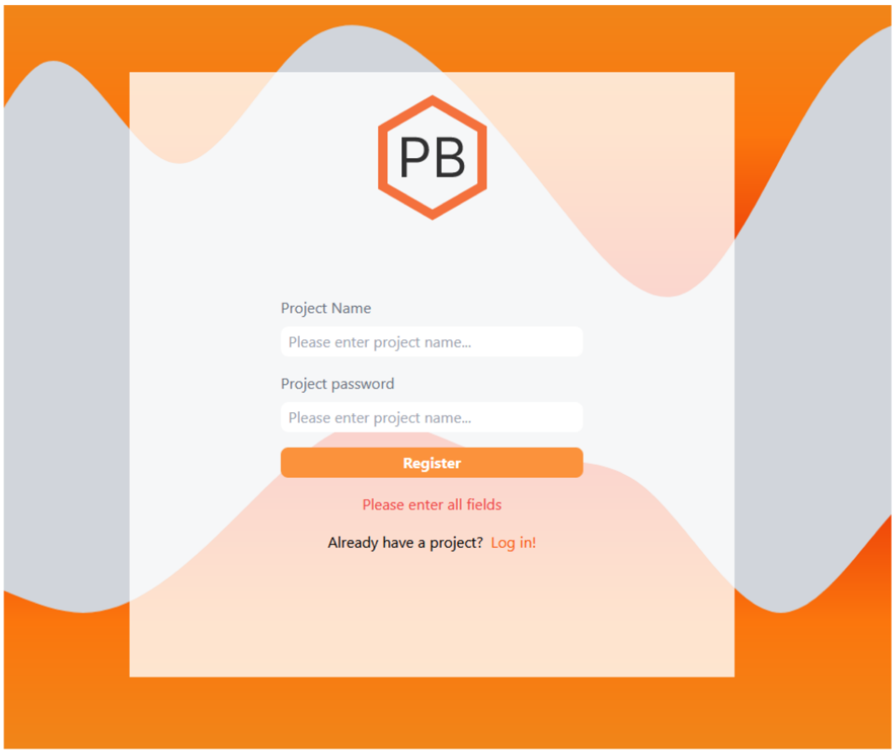
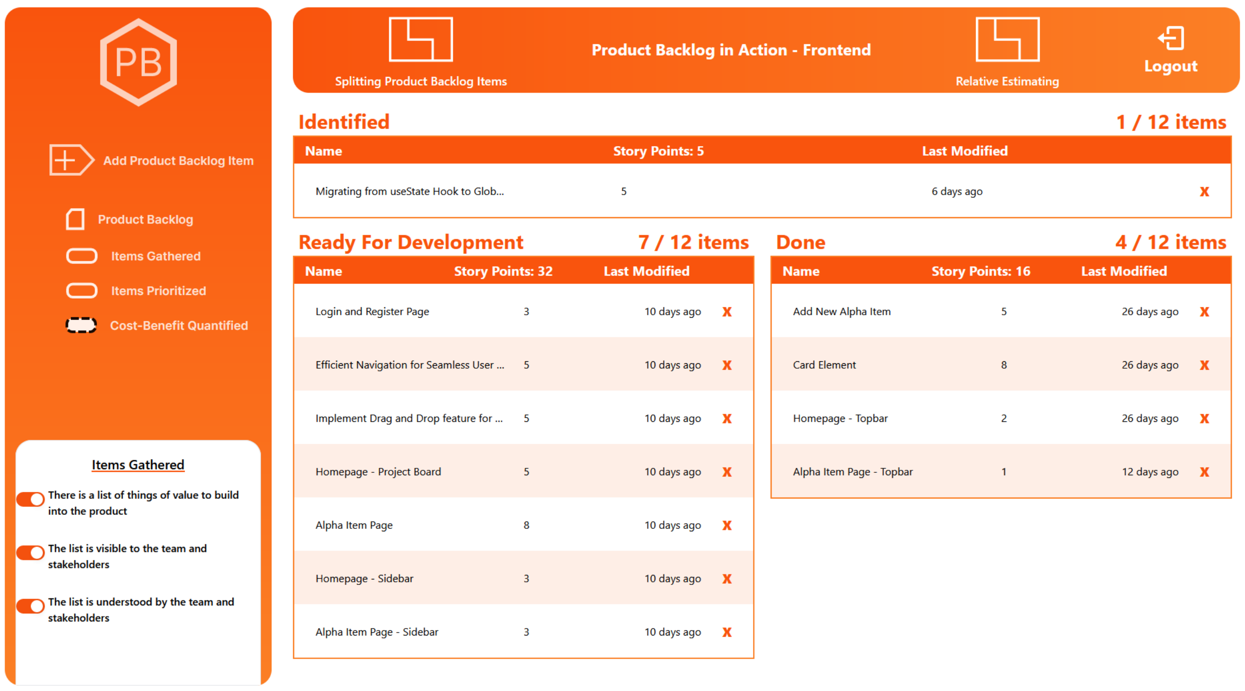

PB
Project Name
Please enter project name...
Project password
Please enter project name...
Register
Please enter all fields
Already have a project? Log in!

Figure 4.5: Register Page

Product Backlog Project Board

The Product Backlog project page acts (Figure 4.6) as the application's main page. The page's design allows a good overview of all items of the Product Backlog, the level of accomplishment over the artifact and a good comprehension of what the team consider as done and what the team need to do next. Additionally, the Product Backlog project page will allow the team to prioritize the items successfully using the drag-and-drop feature available in each table. The menu on the left serves as a way to add new items to the list using the "Add Product Backlog Item". The underlying concept of this button is to serve as one way to represent the Activity "Refine Product Backlog". Teams can preview the current and potential progressing states of the "Product Backlog" Work Product using the side menu. The progression principle relies on checklists. Having items from the checklist unfinished leads to incomplete states. Black-coloured borders and white-filled elements depict incomplete states, while orange-coloured borders highlight complete states. Teams should carefully consider each state and checkbox and progress the state when suitable. Teams can progress multiple states of the elements simultaneously. The sidebar is important when requiring information or progressing the state. Clicking on each meaningful to the Essence Standard will display various cards containing information offered by the Product Backlog Essentials ([2]). Cards can depict essential elements of having an inner checklist, such as the "Items Gathered" state of the Product Backlog Work Product, or purely informational elements, such as the "Relative Estimating" Pattern element. The project name and the two Pattern elements, "Splitting Product Backlog Item" and "Relative Estimating," are displayed on the top menu of the Product Backlog project page. Besides this information, the top bar contains a logout button to switch between projects.



Items Gathered

- There is a list of things of value to build into the product
- The list is visible to the team and stakeholders
- The list is understood by the team and stakeholders

Identified — 1 / 12 items

Name	Story Points: 5	Last Modified	
Migrating from useState Hook to Glob...	5	6 days ago	X

Ready For Development — 7 / 12 items

Name	Story Points: 32	Last Modified	
Login and Register Page	3	10 days ago	X
Efficient Navigation for Seamless User ...	5	10 days ago	X
Implement Drag and Drop feature for ...	5	10 days ago	X
Homepage - Project Board	5	10 days ago	X
Alpha Item Page	8	10 days ago	X
Homepage - Sidebar	3	10 days ago	X
Alpha Item Page - Sidebar	3	10 days ago	X

Done — 4 / 12 items

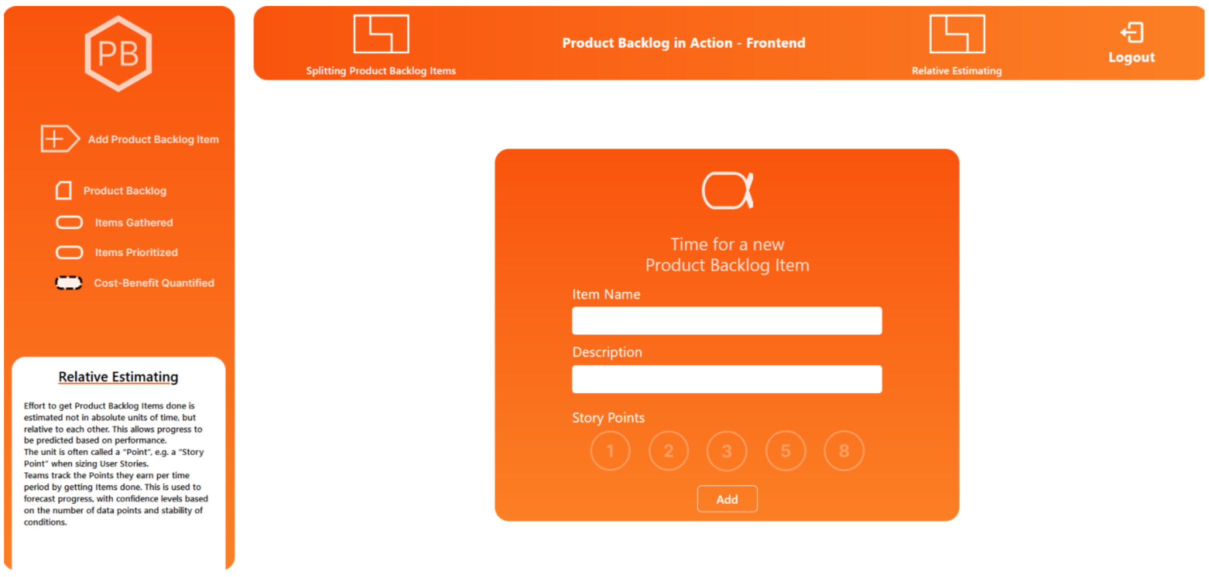
Name	Story Points: 16	Last Modified	
Add New Alpha Item	5	26 days ago	X
Card Element	8	26 days ago	X
Homepage - Topbar	2	26 days ago	X
Alpha Item Page - Topbar	1	12 days ago	X

Figure 4.6: Product Backlog Project Board

Create New Product Backlog Item

A critical aspect of the Product Backlog is that it is entirely customizable. Teams can quickly introduce new items in the Product Backlog by providing only the essential information about an issue: the associated title, description and story points. Using the "Add Product Backlog Item" will display a form, illustrated in Figure 4.7 by which the team can add new items. The design of the form emphasizes once more the elements of the Essence Language (Section 2.2). While the form is displayed, the other elements of the main page are still visible, allowing teams to consult the Pattern cards or reflect on the overall state of the product backlog once more before adding another issue. Once the form is displayed, the team should provide a meaningful name, description and corresponding story points. The name of the issue should be short and unique, illustrating only the main characteristics of the item. The item's description should be concise, providing in a few words a better context for the issue proposed. The Fibonacci sequence is the reference for the proposed story points. The series is limited to a maximum of eight-story points. Eams should only need the proposed number. They should consult the "Splitting the Product Backlog" Pattern card otherwise. The "Add" button creates a new item in the database, and the "Identified" table is updated accordingly.

In case of invalid input for the provided fields of the form, the item's creation will not happen. On the other hand, if the fields provide valid data for the fields, the user will be redirected to the main page, where the view updates accordingly to the mutation provided.



Relative Estimating

Effort to get Product Backlog Items done is estimated not in absolute units of time, but relative to each other. This allows progress to be predicted based on performance.
The unit is often called a "Point", e.g. a "Story Point" when sizing User Stories.
Teams track the Points they earn per time period by getting Items done. This is used to forecast progress, with confidence levels based on the number of data points and stability of conditions.

Time for a new
Product Backlog Item

Item Name:

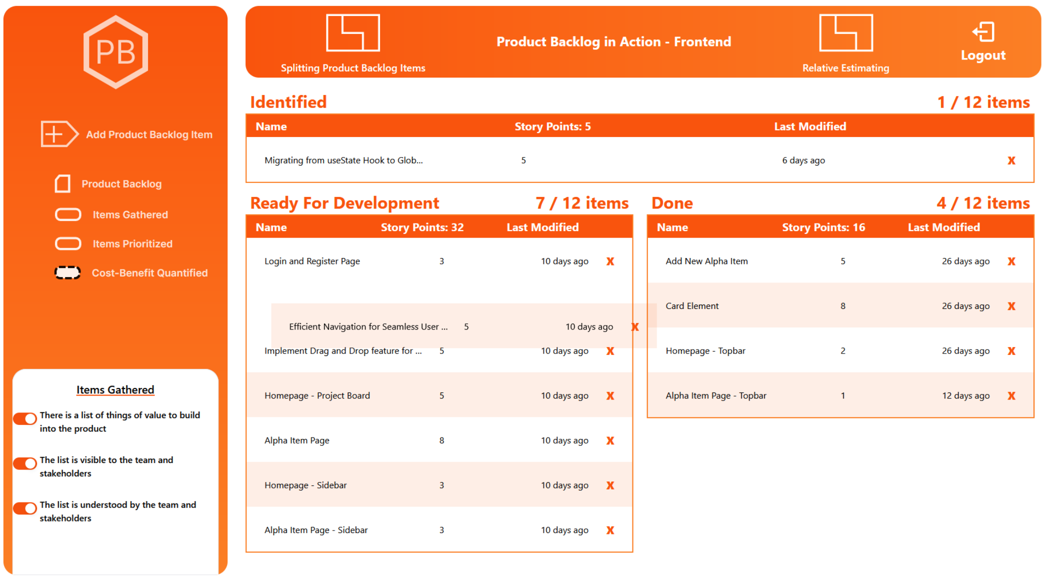
Description:

Story Points: 1 2 3 5 8

Figure 4.7: Add new Product Backlog Item

Managing the Project Board

Teams choosing to have the Product Backlog practice as a part of their methodology should prioritize items as soon as possible. The Product Backlog in Action offers prioritization features through drag-and-drop mechanisms, as illustrated in Figure 4.8. Furthermore, classifying the items by their current state in tables associated with prioritization provides a relevant overview of the Product Backlog. As the project grows, the number of items will also increase. The introduction of tables and reprioritizing mechanisms combat the confusion and help teams to keep visible what is essential for them. Additionally, the application offers a deletion button if teams need to modify one item. The reason behind this mechanism is that modifying one item should impose reconsideration of priorities, especially relating to the other issues, and traversing the alpha roadmap again to validate the correct state of the item. One thing to notice is that users cannot move items using drag and drop from one table to another. The only way of moving between tables is through manual progression or regression. Limiting the drag-and-drop context avoids confusion about how states should be completed and ensure that teams consider all the states' information before progressing, thus ensuring valuable progress of the Product Backlog.



Items Gathered

There is a list of things of value to build into the product

The list is visible to the team and stakeholders

The list is understood by the team and stakeholders

Identified 1 / 12 items

Name	Story Points: 5	Last Modified	
Migrating from useState Hook to Glob...	5	6 days ago	X

Ready For Development 7 / 12 items

Name	Story Points: 32	Last Modified	
Login and Register Page	3	10 days ago	X
Efficient Navigation for Seamless User ...	5	10 days ago	X
Implement Drag and Drop feature for ...	5	10 days ago	X
Homepage - Project Board	5	10 days ago	X
Alpha Item Page	8	10 days ago	X
Homepage - Sidebar	3	10 days ago	X
Alpha Item Page - Sidebar	3	10 days ago	X

Done 4 / 12 items

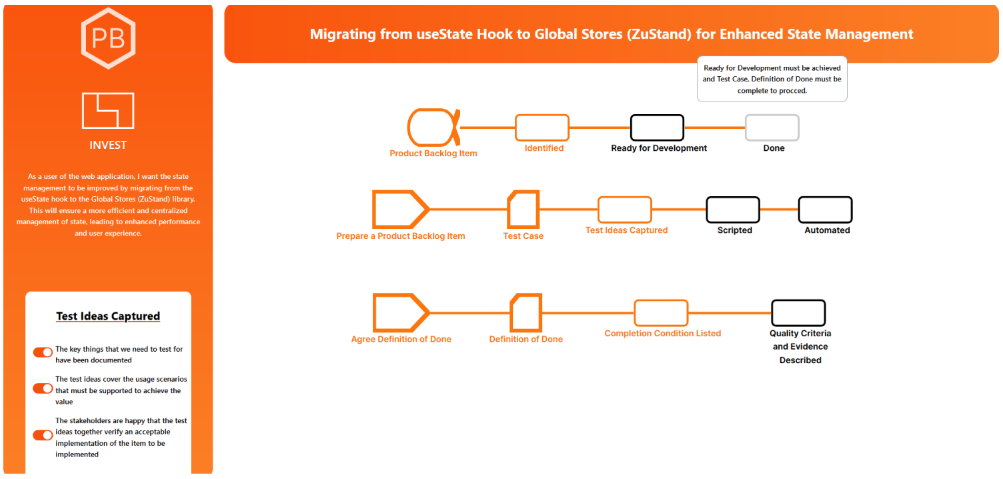
Name	Story Points: 16	Last Modified	
Add New Alpha Item	5	26 days ago	X
Card Element	8	26 days ago	X
Homepage - Topbar	2	26 days ago	X
Alpha Item Page - Topbar	1	12 days ago	X

Figure 4.8: Populated Product Backlog Project Board

Managing a Product Backlog Item

The Product Backlog Item Manager (Figure 4.9) is a crucial interface for ensuring quality over the Product Backlog. The sidebar of this page respects the design imposed previously and is responsible for reflecting the element selected through cards available at the bottom of the menu. The menu also displays the description associated with the selected item from the Product Backlog and the "INVEST" Pattern card, the only Pattern element on this page. The logo icon also serves as a "back" button, allowing users to return to the main page by clicking it. This page's top bar's central role displays the selected issue's title.

This page contains the Product Backlog Item alpha and its three states: "Identified", "Ready for Development", and "Done". Considering that according to the Essence Standard, the alpha elements represent the most important things to work with, the titles of the tables on the main page are inspired by these states' names. Furthermore, each state contains a list of items that must be appreciated to advance the state. Switch buttons are on the state cards, providing a better alternative to checkboxes. As indicated in other screens, a state can be incomplete, thus having black-coloured borders. Besides the alpha element, this page contains two essential activities, "Prepare Product Backlog Item" and "Accept Definition of Done". The Work Products created by those are next to them, connected with a line. As mentioned before, states of the Work Products should be analyzed carefully and advanced when the team considers it appropriate. The associated Work Product for these activities acts as quality standards, respectively, that items meet the quality standards. The Product Backlog in Action provides a different mechanism for ensuring quality and avoiding illegal states: disabling the "Done" state of the alpha item before completing all the necessary states. More specifically, the "Done" state is unreachable until the "Test Case" and "Definition of Done" are fully completed. That is, all their states are marked as done.



INVEST

As a user of the web application, I want the state management to be improved by migrating from the useState hook to the Global Stores (ZuStand) library. This will ensure a more efficient and centralized management of state, leading to enhanced performance and user experience.

Test Ideas Captured

- The key things that we need to test for have been documented
- The test ideas cover the usage scenarios that must be supported to achieve the value
- The stakeholders are happy that the test ideas together verify an acceptable implementation of the item to be implemented

Migrating from useState Hook to Global Stores (ZuStand) for Enhanced State Management

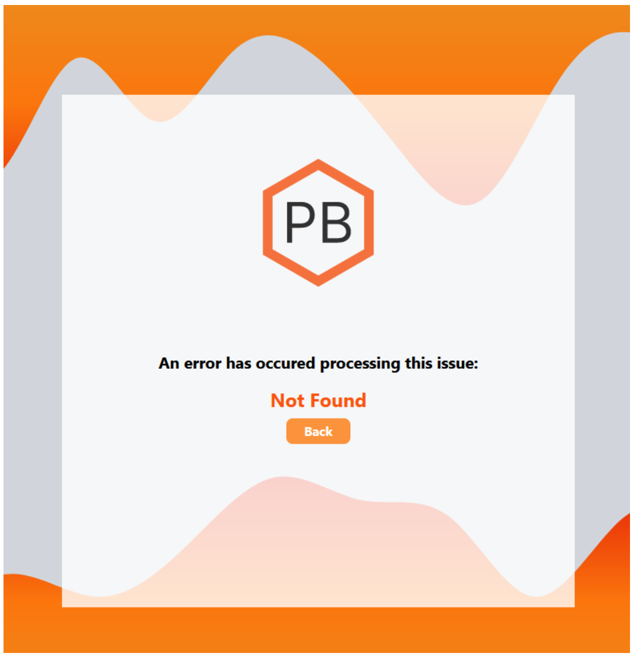
Ready for Development must be achieved and Test Case, Definition of Done must be complete to procced.

Figure 4.9: Product Backlog Item

Moreover, "Ready for Development" must be completed before the item can advance to the last state. The page introduces several visual elements for depicting this: reduced opacity of the visual representation of the state while it is disabled, tooltip elements available on mouse hover indicating the necessary actions for unlocking the state and browser alerts providing information about the necessary action available on click.

Error Handling

The application treats error handling using a fallback page (Figure 4.10) indicating the error in a human-readable form providing concise information about what went wrong. Additionally, a return button is present to ensure the continuity of the flow. Error handling ensures a better user experience.



An error has occured processing this issue:

Not Found

Figure 4.10: Error Page

Chapter 5

Related Work

i The core of this paper is due to work in [1], the premises shared by the authors providing a starting point for deepening the foundations of the Essence standard as a solution for the addressed problem. The authors offered help to the discipline to evolve in a more general and permissive manner. Implementing these guidelines will allow the teams to be more flexible and offer a more personalized experience regarding the way of working.

The Essence standard is the foundation of this work, providing well-defined languages and a common ground, the kernel. Based on those, we may build new practices to make up methodologies for the field of software engineering. Moreover, it allows the combination of methods and practices, thus freeing teams from the prison of methods, described by the authors as a turning point that allows the modelling of preexisting methodologies only with the approval of a so-called methodology guru. This standard thus offers a new phrasing to this discipline, the essential language of software engineering, the language of Essence engineers.

Having the Essence standard as a primary tool for building a solution for the efficient creation and management of the Product Backlog, the application uses a practice offered by the authors of the Essence standard. Present in the practices library ([2]), the Product Backlog Essentials represent the building block for the Product Backlog in Action. The application uses the elements and attached information for each element to offer instructions and progression systems that allow a qualitative evolution of the Product Backlog.

ii As discussed in the article [9], the engineers propose multiple procedures for creating and managing the Product Backlog. The authors of the articles offer beneficial activities for developing and maintaining a Product Backlog.

The procedure for creating this artifact requires careful and often lengthy analysis. Teams must consider multiple steps and aspects on each iteration to achieve maximum quality. Considering this aspect, teams may provide the wrong analysis due to personal beliefs or completely omit some of the proposed procedures due to human error.

Considering these aspects, the Product Backlog in Action provides a way to keep valuable elements that promote techniques proposed by the authors visually to teams. For example, all metrics provided by the [9] are easily visible on the main board or the item management interface (section 4.6). The Product Backlog offers measure systems by indicating the items that are still not in development or done state up to the backlog age based on the last modification using the “Last Modified” column as shown in the section 4.6 or checking the number of story points achieved for each table. [5]

iii As stated in [3], designing a Product Backlog is vital for building complex applications. The authors identified more opportunities to research how the teams should create, evaluate and manage a Product Backlog, as the Agile guides ([4]) omit these crucial aspects, focusing on what the artifact represents for the team.

The ground rules for the practice of Product Backlog are determined. The guides provide theory over what the artifact represents for the team. However, the actual development directions for the Product Backlog are still distant, indicating a need for assistance for the Product Backlog development. The paper’s authors identified twelve typical practices teams usually refer to when creating and managing the Product Backlog. These practices usually vary from creating user stories to building mockups and persona modelling. Although these practices may provide quality over the product, there are no universal measurement scales that can assess the quality.

The solution provided by this paper offers a tangible scale through the inner checklist of elements and logical progression of states residing in the main components of the Product Backlog Essentials. Moreover, the teams can still use the proposed practices proposed in the paper for grooming the Product Backlog. Additionally, they can analyse the results by following the overall progression using the solution proposed, thus enabling a way to measure the accomplishment and efficiency of the teams when applying the practices suggested in the [3].

iv One product backlog management tool has been analyzed in [7]. The authors describe how the National Ignition Facility uses Jira as an issue-tracking system for 63 projects. The papers exemplify how the Jira management tool provides relevant features for tracking issues, from defining the types of issues, creating workflows, defining custom fields and others. The management software offers an overview of the new features, bugs or changes and allows users to add documentation for each item added to the collection. Furthermore, the tool enables workflow, adding custom stages for items from the ”assigned” stage, where different members are associated with an issue, to the ”test passed” stage, where all necessary quality checks can be considered successfully passed. Introducing numerous stages of verification will enable the team to complete the issue in a way that will ensure the quality of the task.

The stages validation is a conclusion of the long lifespan of the projects. National Ignition Facility started using Jira as the main management tool in 2006, and the company performed a major adaptation to the Agile Methodologies in 2013. Jira management tool provides a reliable tool for tracking history and having an overview of the product and its evolution.

However, these aspects do not address the need for guidance on how the team can assess the quality of the issues. Even if the National Ignition Facility managed to create a way to determine the quality of its product over several increments, similar stages might be proven to be inefficient and inconclusive when addressing other products. As discussed in the [1], every project has a different path, and each team have a different way of working. However, each project has some standard metrics that teams need to follow to address the status of work and the quality addressed. These elements are essential in the lifespan of a project. The authors of [1] extracted the essential elements of the Product Backlog practice, thus providing a way to ensure quality metrics over the evolution of a project's Product Backlog. By providing some general qualitative checks, the solution proposed by this paper allows the team to search for adequate qualitative checks over time while assisting in ensuring and evaluating the accomplishment of the Product Backlog. One way the team can choose to work is to use the Jira Atlassian product for listing and associating tasks documentation while using the Product Backlog in Action as a way to ensure accomplishment and evaluation of the Product Backlog artifact. This approach could provide good documentation of the Product Backlog and better creation and evolution.

v An overview of the management tools of Agile Methodologies practices was provided in [8] paper. In the first part of the paper, the authors emphasize the increasing number of projects approaching Agile Methodologies. The paper reveals the increasing percentage of successful projects after converting to Agile Methodologies. The paper reveals that the usage of software tools as means of communication can lead to increased productivity and may shorten the documentation needed.

In the second part, the authors continue the paper by revealing numerous management software agile professionals use to supervise the projects. Most of the tools presented in the paper focus on planning and tracking. Tools like Jira, ActiveCollab and IceScrum, all described in the paper, provide a high overview of the whole project, allowing users to create roadmaps for the projects. All these aspects help in extensive planning and analysis of the evolution of the projects.

Using these tools may provide quality in the longer term by offering teams to identify the quality criteria relevant to their project. Although the usage of software management tools offers a broad overview of the progression of the projects, guidance over the assessment of quality for individual tasks appears only after several increments. Having guidance over the main milestone that issues have to achieve will shorten the trial-and-error phase and provide quality to the development activity of the project.

vi [6] paper reveals many issues regarding the lifespan of a project, bringing to the surface the need for more theory for Product Backlog development. The authors identified many aspects leading to the failure of the projects, predominantly poor planning being a primary factor for this. The second part of the paper introduces the analysis of multiple software management tools for addressing some of the issues identified in the first part. The analyzed tools provide multiple features for tracking and planning the project, communication and collaboration and even the generation of reports and analysis. The tools presented are solutions for project management and overview. The main difference between the proposed tools and the solution that this paper provides is the possibility of tracking the internal states of each issue. Although the Product Backlog in Action is a quality assurance over the Product Backlog accomplishment, the solution does not provide integration with other existing systems. However, teams may use the software proposed in collaboration with other software management tools to achieve a better evolution of the Product Backlog. The main purpose of the Product Backlog in Action is to ensure the quality of the items rather than document the project's life cycle.

Chapter 6

Conclusions and Future Work

The evolution of the discipline of software engineering has come a long way. During the lifespan, multiple approaches and methodologies coexisted to overcome the possibility of the unknown. Many engineers tried to develop an exact formula to predict the potential challenges that teams might encounter. Unfortunately, planning and developing a project is not an exact science. The future is unpredictable, and teams must consider multiple risks in the lifespan of a project. These lead to confusion and the appearance of several methods, leading to what [1] described as a ”software crisis.” The methodologies were created as robust guidelines that can be modified only by the acceptance of the very experienced promoters of the methodology. However, because there are no two identical projects, each team and project requires different sets of practices when producing products. Different teams have different ways of working, inevitably impacting the project’s evolution.

Even if the developing phase of a project can be considered done, the maintenance of the project can imply multiple aspects that can result in inefficient spending of resources. The team’s main goal is to minimize the risk and the unpredictable as much as possible, and this can be achieved only by considering an adequate and rich planning process and the freedom to choose the best practices that suit the team. The planning process must be extensive and dynamic, and teams can only achieve this through multiple reconsiderations of the requirements. Adapting to changes is a crucial competency that teams must accomplish to deliver customer value. For a better transition to future changes, teams must have adequate tools to allow future modelling.

As described in the [8], there is a big transition to the Agile methodologies. This transition promotes the adoption of changes through the planning of the project. The artifact responsible for tracking every change in the project is, as the [4] describe, the Product Backlog. Having a well-defined Product Backlog and keeping it prioritized and visible to the team are two important aspects that teams must consider. However, there are few guidelines over how teams should create or manage a Product Backlog such that it promotes value to the client.

The core of the solution relies on the Essence standard introduced by Ivar Jacobson ([1]), more specifically on the Product Backlog Essentials practice introduced in [2]. By providing only the essential elements of the Product Backlog practice, the thesis aim to allow the user to focus on what is important for this practice, thus ensuring quality over the Product Backlog. Moreover, focusing on the essentials of producing a Product Backlog Item, the solution offers valuable evolution of individual items. Additionally, teams may get an insight into the velocity if they choose to work in sprints by consulting the progress accessible through the metrics provided by the application.

Future Work

There are multiple enchantments to address in the future. The next part of the paper will list some indications for implementing them and describe each.

i One first feature to address in the future should be the possibility of voting over a state or checklist should be considered done. The development and evolution of the Product Backlog should be a collective effort. Any user with access to the project can progress each state and checklist, but the refinement and progress of the artifact have to be a team activity. To emphasize that it should be a team activity, a voting system based on the majority decision has to be implemented such that all members of the team can agree on what is achieved. This feature will promote the inclusion of the majority of the team in the direction of the Product Backlog and will raise discussion over the essential milestones of the issues.

ii Another idea for the future is to provide better support for working in sprints. As mentioned in the section 4.6, teams may organize the Product Backlog in multiple ways. One way is to work by having a project for each sprint. Considering this approach, more extensive support for sprints should be supported. For example, the automatic creation of the next sprint project when one sprint is completed or the possibility to select multiple items from a project and create a sprint project including them.

iii As described in the 5, multiple tools provide the possibility of generating extensive reports over the projects. This feature would enable a comprehensive overview of different aspects of the Product Backlog, having details over different metrics such as the average time for completing an issue or the average number of story points completed during a sprint. This feature can help understand the team's velocity and plan future requirements and sprints.

iv One thing to note is that the current solution is not meant to support documentation. Considering this aspect, an enchantment that should be considered is integrating the solution with already existing software tools that provide this, such as the one described in [8].

v Another aspect that can be further enhanced is the possibility of adding other relevant stages that items should go through before being considered done. Teams may need more extensive testing or a different level of testing to consider an item done, which should be reflected in the solution. Enabling the addition of new stages for items should grant modularity to teams. Following the previous idea, the teams should be able to add new elements that they consider essential to their way of producing the Product Backlog.

vi Furthermore, better deployment of the application should be done. As stated in the paper, testing over the frontend is performed, but there is no automatic pipeline for each deployment except for the successful building. Currently, to ensure quality, test modules must be run before deploying a new build. Considering this, a better DevOps process should be created to ensure multiple verification levels.

Bibliography

[1] I. Jacobson, P.-W. Ng, P. E. McMahon, M. Goedicke, *et al.*, *The essentials of modern software engineering: free the practices from the method prisons!* Morgan & Claypool, 2019.

[2] I. Jacobson, "Product Backlog Essentials." `https://practicelibrary.ivarjacobson.com/content/product-backlog-essentials-publication`.

[3] T. Sedano, P. Ralph, and C. Péraire, "The product backlog," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 200–211, 2019.

[4] K. Schwaber and J. Sutherland, "The scrum guide." `https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf`, 2020.

[5] Anon, "What is a backlog?." `https://rb.gy/58me8`, 2021.

[6] F. Pasarič and M. Pušnik, "Comparison of project management tools," 2022.

[7] J. Fisher, D. Koning, and A. Ludwigsen, "Utilizing atlassian jira for large-scale software development management," tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2013.

[8] D. Özkan and A. Mishra, "Agile project management tools: A brief comprative view," *Cybernetics and Information Technologies*, vol. 19, no. 4, pp. 17–25, 2019.

[9] Anon, "How do you measure and improve the quality of your product backlog in jira?." `https://www.linkedin.com/advice/0/how-do-you-measure-improve-quality-your-product-1c`.